Erprobte Ansätze aus dem Softwareumfeld übernehmen

Was Unternehmen und die BWL nutzen können

Prof. Dr. Ulrich Anders, Cologne Business School

Version 1.0.1 · 8. Januar 2017

Was ist GitHub? Was ist Stack Overflow? Was kann man von RUST lernen? Was ist Redux und was verbirgt sich hinter dem State-Action-Model? Und was bedeutet eigentlich SCRUM?

Die Betriebswirtschaftslehre ist voll mit guten Managementbüchern und -konzepten. Erstaunlicherweise haben sich aber Überlegungen, die im Softwareumfeld angestellt wurden, bislang nicht oder nur sehr vereinzelt in der Betriebswirtschaftslehre und Managementliteratur niedergeschlagen. Dementsprechend selten findet auch die Übertragung erfolgreicher Ansätze aus dem Softwareumfeld in die organisatorische Praxis von Unternehmen statt. Das ist umso erstaunlicher, als dass es sich bei den Überlegungen aus dem Softwareumfeld nicht mehr nur um bloße Konzepte handelt, sondern mittlerweile um Verfahrensweisen und Organisationsprozesse, die von vielen Millionen Softwareentwicklern international und sehr erfolgreich angewendet werden.

Nehmen wir als erstes Beispiel GitHub. Kaum jemand außerhalb der Softwareentwickler-Gemeinde kennt GitHub. Das gilt vielfach auch für die Manager von Unternehmen. Dabei ist GitHub riesig und vereint über 20 Millionen Entwickler weltweit. Es stellt damit bestes Anschauungsmaterial für moderne, interaktive, gleichberechtigte und hierarchiefreie Zusammenarbeit zur Verfügung.

1 GitHub, Motivation und Zusammenarbeit

1.1 Was ist also GitHub?

GitHub ist der Dreh- und Angelpunkt der Open-Source-Szene. GitHub ist eine Internet-Plattform, auf der sich jeden Tag Mengen von Programmierern tummeln, die auf die ein oder andere Weise zu einzelnen Softwareprodukten und -projekten beitragen: sie entwickeln neue Anwendungen, Bibliotheken und Rahmenwerke, machen auf Fehler aufmerksam, sie verbessern Code, sie programmieren neue Funktionen, sie ergänzen die Dokumentation, sie stellen Fragen oder sie bekommen sie beantwortet. GitHub hat monatlich circa 30 Millionen Besucher. Diese gehören entweder zu Unternehmen, sind

selbstständig oder arbeiten in ihrer Freizeit mit. 44% der *Fortune 50* Unternehmen nutzen GitHub und 50% der *Fortune 10* Unternehmen.

Die Software-Produkte, die man auf GitHub finden kann, gehören in die unterschiedlichsten Kategorien: Anwendungen, Programmiersprachen, Frameworks, Spezialsoftware, Tools, Code-Schnipsel, Dokumentationen, Tutorials und vieles mehr. Bei den Softwareprodukten handelt es sich in der Regel um Softwareprodukte, die gewisse technische Abläufe vereinfachen oder bestimmte Angebote ermöglichen. Es handelt sich also nicht um kommerzielle Softwareprodukt, die als Kerngeschäft eines Unternehmens entwickelt werden.

Die Programmierer kommen aus allen erdenklichen Ländern und begegnen sich auf GitHub ohne hierarchische Unterschiede. Jeder kann dort seine Software frei zugänglich einstellen. Ist die Software auch für andere interessant, finden sich schnell weitere Programmierer, die beginnen, an der Weiterentwicklung der Software mitzuwirken.

Open Source — also Software, deren Source Code öffentlich ist — ist für viele mehr als nur Software. Open Source geht meist einher mit der Überzeugung, dass Software öffentlich sein sollte. Das ist der Grund dafür, dass viele Unternehmen und Programmierer ihre Leistungen aus Stunden, Tagen und Monaten unentgeltlich in den Dienst der Öffentlichkeit stellen.

Darüber hinaus existiert aber auch ein unschätzbarer Vorteil: die eigene Software wird durch andere Programmierer getestet, verbessert und fehlerreduziert. Das System funktioniert. Initial gibt es natürlich viele Interessenten, die das Software-Angebot von GitHub lediglich nutzen und nicht beitragen. Aber mit der Zeit tritt bei vielen ein Wandel ein. Wer etwas Gutes umsonst bekommt, was Abläufe und Arbeit erleichtert, der fühlt sich meist auch verpflichtet, selbst mitzuhelfen.

Drei wesentliche Gründe sprechen dafür, die eigene Software auf GitHub öffentlich zu machen und Dritten zur Verfügung zu stellen. Microsoft und Facebook sind nur zwei Unternehmen, die so denken:

1. Offene Software führt zu offener Software: viele Firmen verwenden offene Software in ihren internen Prozessen und vielen Unternehmen hat offene Software erst einen Start ermöglicht. In dieser Hinsicht ist es gewissermaßen zwangsläufig, dass dann irgendwann auch die selbst entwickelte Software der Öffentlichkeit zur Verfügung gestellt wird, falls diese auch für Dritte nützlich sein könnte.
2. Offene Software treibt Innovation: die eigene Software wird durch die Nutzergemeinde nicht nur weiterentwickelt, sondern sie bleibt auch auf dem aktuellen Stand und profitiert von der Innovationskraft der Masse.
3. Offene Software stärkt das Geschäft: bessere Software ist nicht nur einfach besser und weniger wartungsintensiv. Sie macht auch die internen Entwickler stolz auf ihre Leistungen, wenn die resultierende Software millionenfach von Dritten eingesetzt wird. Die Produktion und Sichtbarkeit von hoher Qualität ist dementsprechend attraktiv für Bestleister. Auf diese Art lassen sich also auch die besten Softwareingenieure gewinnen.

1.2 Was lässt sich nun von GitHub lernen?

Open Source lehrt uns etwas über die Menschen und Organisation. Menschen wollen beitragen, Menschen sind vom Zweck getrieben.[1] Das gilt nicht nur für Open Source. Die hohe Qualität von Wikipedia oder Millionen von kompetenten Antworten in den entsprechenden Fachforen beweisen das gleiche.

[1] Siehe auch Pink (2010).

Und wie ist es in Unternehmen? Mit Erschrecken liest man in Studien, wie z.B. dem regelmäßig erscheinenden *Engagement Index* von Gallup, dass Engagement und emotionale Bindung von Mitarbeitern auf einem niedrigen Niveau sind. Nur circa 15% der Arbeitnehmer empfinden laut diese Studie eine hohe emotionale Bindung an das Unternehmen.

Eine »geringe emotionale Mitarbeiterbindung«, so die Studie von 2014, lässt »sich in der Regel auf Defizite in der Personalführung zurückführen. Viele Arbeitnehmer steigen hoch motiviert in ein Unternehmen ein, werden dann aber zunehmend desillusioniert, verabschieden sich irgendwann ganz aus dem Unternehmen und kündigen innerlich. Die Hauptrolle in diesem Prozess spielt fast immer der direkte Vorgesetzte.«

Leidglich dem Vorgesetzten die Schuld an der geringen emotionalen Bindung zu geben, ist sicherlich etwas zu vereinfachend. Häufig hat auch der Vorgesetzte in Hierarchien deutlich weniger Entscheidungs- und Handlungsspielraum, als man glauben mag. Möglicherweise muss man also darüber nachdenken, ob die traditionellen hierarchischen Organisationsmodelle in allen Fällen noch zeitgemäß sind. Hierarchien sind notwendig, aber sie machen Unternehmen immer etwas starr und unbeweglich. Außerdem nehmen sie Menschen oft ihre Selbstbestimmtheit in Hinsicht auf das Erreichen gesetzter Ziele. Das kostet Motivation und damit emotionale Bindung.

Dass Menschen motiviert arbeiten, wenn sie selbstbestimmt einen Zweck verfolgen, lehrt uns GitHub. Wahrscheinlich benötigen Unternehmen für die Zukunft ganz neue Organisationsmodelle. Diese müssen viel stärker einen unmittelbaren und greifbaren Zweck in den Mittelpunkt stellen und nicht nur ein schwer zu beeinflussendes übergeordnetes Unternehmensziel, an das dann noch ein Teil der Bezahlung geknüpft wird. Und vielleicht muss man auch über Hierarchien ganz neu nachdenken. Hierarchien in GitHub gibt es nicht. Möglicherweise kommen auch deshalb die Leute da so gerne hin.

Ein zweites Thema lässt sich bei GitHub studieren: Zusammenarbeit. Die Zusammenarbeit findet bei GitHub unbesehen der Person statt. Egal welchen Alters, welcher Herkunft, welchen Geschlechts oder welcher Rasse, jeder begegnet jedem in der Regel vorurteilsfrei. Das Einzige, was in hierarchiefreien Projektorganisationen bei der Zusammenarbeit zählt, ist die Qualität des Ergebnisses und die Kompetenz derjenigen, die dazu beitragen. Und die Kompetenz wird schnell transparent und steuert die Projektorganisation.

Ist das nicht auch eine Blaupause für Unternehmen, die sich unter anderem mit folgenden drängenden Fragen auseinandersetzen müssen:

1. Wie erzeugen wir möglichst hochqualitative Ergebnisse, die nicht durch starre Hierarchien behindert werden?
2. Wie nutzen wir das Wissen von erfahrenen Mitarbeitern in Projekten, die Innovation erzeugen sollen? Ist Frühpensionierung das richtige Mittel, um jüngeren Mitarbeitern mehr Spielraum zu ermöglichen und darüber Innovationen zu ermöglichen?
3. Wie schaffen wir es, Bezahlung und Gehalt an Kompetenz anstatt an hierarchischen Status zu binden?

Ein Teil der Antworten finden sich vielleicht bei GitHub: in kompetenzorientierten Organisationen kommt es immer auf die Kompetenz für eine bestimmte Aufgabe an. Es zählt also nicht, welche sonstigen Kompetenzen eine Person hat, sondern ob sie die Kompetenzen hat, die für die gefragte Aufgabe notwendig sind. Mit dem Prinzip »Kompetenz« bewegt man sich also weg von einer statuszentrierten Welt (es zählt nicht der Status einer Person) und hin zu einer kompetenzzentrierten Welt (es zählt die erforderliche Kompetenz einer Person). Wer so denkt, wird deutlich flexibler.

Ein drittes Thema lässt sich ebenfalls bei GitHub beobachten: Kommunikation. Alle Kommunikation

in Open-Source-Projekten ist ebenfalls öffentlich. Jeder kann sie mitlesen, jeder kann die Diskussionen auch nach Jahren noch nachvollziehen. Das hat folgende Vorteile:

1. In der Regel pflegen die Menschen auf GitHub einen freundlichen Umgangston.
2. Man kann nachvollziehen, welche Entscheidungen auf welche Art und Weise zustande gekommen sind und welche Einwände berücksichtigt wurden.
3. Entscheidungen werden verargumentiert und nicht einfach hierarchisch gesetzt.

Unternehmen, die engagierte Mitarbeiter wünschen, die nach einem gemeinsamen Unternehmensziel streben sollen, müssen ihre Mitarbeiter informieren. Mitarbeiter, die Entscheidungen nachvollziehen können, die verstehen, welche Einwände vorgebracht wurden und die lesen können, auf Basis welcher Argumente eine Entscheidung letzten Endes gefällt wurde, werden mündig. Je besser Mitarbeiter die strategischen Überlegungen verstehen, desto eher können sie ihre eigenen Aktivitäten auf das Interesse des Unternehmens abstimmen. Sicher muss nicht jede Diskussion öffentlich gemacht werden, aber wahrscheinlich brauchen Unternehmen auch ganz neun Möglichkeiten, ihre Mitarbeiter zu informieren und zu kommunizieren.

2 Stack Overflow, Reputation statt Bewertung

Seit Jahren hält sich die Überzeugung, die Mitarbeiter eines Unternehmens müssten mindestens jährlich bewertet werden. Eine seiner stärksten Ausprägungen hat dieser Gedanke durch Jack Welch erfahren. 20% der Mitarbeiter sollten in General Electric jährlich als gute ‘A’-Mitarbeiter, 70% als mittlere ‘B’-Mitarbeiter und 10% als schlechte ‘C’-Mitarbeiter identifiziert werden. Hunderte von Unternehmen haben solche oder ähnliche Bewertungsschemata eingeführt und halten noch heute an dieser Praxis fest.

Die ökonomische Sinnhaftigkeit dieses Ansatzes ist schwer nachvollziehen. Nur eine kleine Minderheit wird zu den guten ‘A’-Mitarbeitern gezählt, alle anderen erhalten das Attribut »Mittelmaß« oder »schlecht«. Diese Vorgehensweise hat zwei Nachteile: erstens wird die große Mehrheit der Belegschaft abgestempelt. Das ist für nichts gut und kostet nur Motivation. Zweitens ist das Schema rigide und jedes Jahr müssen die 10% schlechtesten identifiziert werden, egal ob diese wirklich schlecht sind oder nicht. Abgesehen davon, dass solche Verfahren grundsätzlich unter der Subjektivität des Managements leiden, erzeugen sie auch noch unnötige Angst und Konkurrenz unter den Mitarbeitern. Die Mitarbeiter eines Unternehmens sind seine wertvollsten und häufig auch seine teuersten Ressourcen. Warum sollten diese also ohne Grund immer wieder auf’s Neue abgestempelt und frustriert werden.

Zu dem gleichen Ergebnis kommt Reinhard Sprenger, der die Praxis der regelmäßigen Bewertung auch aus grundsätzlichen Überlegungen heraus für übergriffig hält. Er findet nicht, dass sie zu einem anständigen Unternehmen passt.

Erstaunlich also, dass sich ein solch ressourcenschädigendes Konzepte in der organisatorischen Praxis überhaupt durchsetzen konnten. Glücklicherweise nehmen immer mehr Unternehmen Abkehr von dieser organisatorischen Praxis. General Electric selbst hat die Bewertung eingestellt, aber auch Unternehmen wie Microsoft, SAP und zunehmend auch andere Unternehmen verfolgen diesen Ansatz nicht mehr.

Das ist auch gut so, denn Unternehmen sollten sich nicht auf ihre Schwächen, sondern ihre Stärken konzentrieren. Diese sicher richtige Überzeugung findet sich in vielen Managementbüchern, u.a. auch bei Malik. Anstatt also Mitarbeiter abzuwerten, sollten man ihnen eher die Möglichkeit geben, eine Reputation aufzubauen. Und genau so funktioniert *Stack Overflow*.

2.1 Was ist Stack Overflow?

Stack Overflow ist eine Frage-Antwort-Plattform für Themen der Softwareentwicklung. Der Name rührt von einem häufigen, bekannten und kritischen Fehler bei der Softwareprogrammierung her. *Stack Overflow* ist Teil der *Stack Exchange* Gruppe, die weitere Frage-Antworten-Plattformen auch für andere Spezialthemen zur Verfügung stellt. Auch *Stack Exchange* ist groß: 5 Millionen registrierte Mitglieder, knapp 4 Milliarden (!) jährliche Besuche, knapp 4 Millionen gestellte Fragen und knapp 5 Millionen gegebene Antworten.

Das Grundprinzip hinter *Stack Overflow* besteht darin, dass Menschen Fragen stellen und beantworten und sich damit eine Reputation aufbauen können. Die Reputation wird durch eine Punktzahl, einen *Score*, repräsentiert.

Das funktioniert in der Zusammenfassung folgendermaßen:

- Die Leser der Fragen und Antworten — nicht selten selbst auf der Suche nach einer Antwort — können die Frage und die Antwort jeweils um einen Zähler auf- oder abwerten.
- Jede Aufwertung der Frage addiert 5 Punkte zum Score des Fragestellers, jede Aufwertung der Antwort addiert 10 Punkte zum Score des Antwortgebers.
- Für die Akzeptanz einer Antwort durch den Fragesteller werden 15 Punkte zum Score addiert, was die Schnelligkeit einer guten Antwort belohnt.
- Beiträge und Verbesserungen zu den Antworten, sogenannte *edits*, erhöhen den Score um jeweils 2 Punkte.
- Die Abwertung einer Frage bzw. Antwort kostet den jeweiligen Besitzer 2 Punkte.
- Mit einer zunehmenden Reputation erhalten die Besitzer weitergehende Privilegien in Hinsicht auf die Frage-Antwort-Plattform und können u.a. auch stärker in die Moderation eingreifen.

2.2 Was kann man von Stack Overflow lernen?

Im Gegensatz zum Thema »Bewertung«, dass überwiegend destruktiv ist, ist das Thema »Reputation« konstruktiv. Wer Kompetenz hat und auch unter Beweis stellt, kann sich sehr schnell eine Reputation aufbauen. Einmal mehr zeigt sich, dass es andere Treiber gibt, die Menschen motivieren: anderen zu helfen, etwas zu können und das nach außen auch sichtbar unter Beweis zu stellen.

Wäre das nicht auch im Unternehmenskontext hilfreich, wenn sich die Wissensträger im Unternehmen eine sichtbare Reputation verdienen könnten und damit zu entsprechenden Themen bereichsübergreifend und firmenweit als kompetenter Ansprechpartner zur Verfügung stehen würden — über alle Grenzen hinweg und als Ansprechpartner für alle, die kompetente Informationen benötigen. Allen wäre geholfen: dem Unternehmen beim Wissensmanagement und Wissenstransfer, den Mitarbeitern, die Hilfe und Information suchen und den Mitarbeitern, die ihre Kompetenz nach außen sichtbar unter Beweis stellen und die sich durch ihre Kompetenz und ihr Engagement vielleicht auch für weitergehende Aufgaben empfehlen wollen.

3 React und die Bildung von Komponenten

Viele Unternehmen sind organisch oder anorganisch gewachsen. Bereiche oder Abteilungen wurden um Führungskräfte herumgeplant, manche Bereiche oder Abteilungen existieren noch, obwohl ihre historische Notwendigkeit weggefallen ist, es gab Zukäufe von Unternehmen, die nicht sauber integriert wurden usw. Als Ergebnis ist die Organisation als Ganzes nicht mehr geordnet: überlappende oder fehlende Verantwortung, ähnliche Bereich in unterschiedlichen Unternehmensteilen, unklare Zuständigkeiten, mangelnde »Ownership«, und die Vermischung von Funktionen sind nur einige Beispiele.

Unternehmen, die langfristig jedoch in der zunehmenden Komplexität ihrer Umwelt bestehen wollen, müssen sich der Herausforderung stellen und die Komplexität managen. Ohne Ordnung geht das nicht. Komplexität kann man nicht mit Chaos entgegen treten. Dafür benötigt man Organisation. Je komplexer die Unternehmensumwelt wird, desto wichtiger wird die Organisation des Unternehmens.[2]

Die ideale Organisation gibt es dabei nicht. Darin sind sich Manager und Wissenschaftler einig. Aber was ist damit gemeint? Sicher gibt es nicht die eine, allgemeingültige und immer beste Organisation für alle Zwecke. Mit Sicherheit gibt es aber eine für die Zwecke eines Unternehmens sehr gut geeignete Organisation, die mit einiger Gewissheit besser ist, als die gerade verwendete.

Würde man ein Unternehmen auf der grünen Wiese von Grund auf neu organisieren, hätte man die Aufgabe, die typischen Unternehmensbereiche anzuordnen und mit Linien zu verknüpfen. Die typischen Unternehmensbereiche sind (ohne Anspruch auf Vollzähligkeit):

Asset Management, Compliance, Controlling (für Management, Beteiligungen, Finanzzahlen, Personal und Projekte), Datenschutz, Einkauf, Erfüllung externer Anforderungen, Fakturierung, Forschung und Entwicklung, Information, Infrastruktur, Inhouse Consulting, Interne Revision, Investors Relations, IT, Kapital & Liquidität, Kommunikation, Kundenmanagement, Lagerhaltung, Lieferung & Leistung, Markenmanagement, Marketing, Materialwirtschaft, Organisation, Personal, Presse, Produktion, Produktionsmittel, Produktmanagement, Projektmanagement, Qualitätsmanagement, Rechnungswesen, Recht, Risikomanagement, Rohstoffe / Energie / Umwelt, Sicherheit, Steuerabteilung, Unternehmensentwicklung, Unternehmensleitung, Vertragsmanagement, Vertrieb, Wissensmanagement, Zulieferungen.

Die Aufgabe, die typischen Unternehmensbereiche anzuordnen, ist eine Aufgabe, der sich alle Unternehmen ab einer gewissen Größenordnung gegenübersehen. Nichtsdestotrotz scheint die gleiche Aufgabe immer und immer wieder auf's Neue gelöst zu werden. Dabei stellt sich die Frage, ob es nicht bestimmte Komponenten gibt, die die geschickte Anordnung erleichtern.

Eine ähnliche Frage muss man auch beim Design von Webapplikationen beantworten. Kann man Webapplikationen nicht mit Hilfe von Komponenten erstellen? Gibt es Komponenten, die sich dabei immer wieder verwenden lassen. Gibt es prinzipielle Unterschiede bei den Komponenten. Wie managt man die Komponenten, so dass sie in Abhängigkeit von Nutzeraktionen aktualisiert werden.

Die Antwort liefert React.

[2] Siehe auch Anders, Ulrich (2015b).

3.1 Was ist React?

React ist ein Bibliothek, die von Facebook auf GitHub öffentlich gemacht wurde und mit der Facebook, seine eigenen Webseiten kreiert. In kürzester Zeit haben knapp 1.000 Programmierer zu dieser Bibliothek beigetragen. React wird von über 4.000 Programmierern in seiner täglichen Entwicklung beobachtet und von über 60.000 Programmieren mit einem Stern als Zeichen der besonderen Anerkennung ausgezeichnet.

Dabei macht React als Bibliothek nichts anderes, als die geschickte Komponentenbildung und Komponentenwiederverwendung bei Webapplikation zu unterstützen. Zusammen mit den Bibliotheken, die einen ähnlichen Ansatz verfolgen, beeinflusst React damit signifikant die Art und Weise, wie Webapplikation jetzt und in Zukunft entwickelt werden. Offensichtlich hat React eine Antwort auf die Frage der Komplexitätsreduzierung von Webapplikation gefunden.

3.2 Was kann man von React lernen?

Komponentenbildung ist eine Antwort darauf, Struktur und Ordnung zu schaffen. Denn die Komponentenbildung zwingt die Entwickler, über eine sinnvolle Strukturierung der Komponenten nachzudenken und diese, möglichst abgeschlossen zu gestalten. Durch die Wiederverwendbarkeit von Komponenten ist React darüber hinaus eine Reaktion darauf, die gleiche Aufgabe nicht immer wieder neu lösen zu wollen, sondern auf eigene oder fremde Vorarbeiten zurückgreifen zu können.

Die Komponenten einer Webapplikation beinhalten wiederum Komponenten. Dabei kann die jeweilige Elternkomponente ihre Eigenschaft an die Kindkomponenten weitergeben. Auf der obersten Ebenen befinden sich die Hauptkomponenten. Hauptkomponenten sind als Konzept nicht neu. In der Mathematik sind Hauptkomponenten schon lange bekannt und sie werden für eine Vielzahl von Fragestellungen eingesetzt.

In der betriebswirtschaftlichen Organisationslehre ist das Konzept der Hauptkomponenten jedoch nicht bekannt. Es wurde allerdings vom Autor dieses Artikels entwickelt und soll daher hier kurz zusammengefasst werden.[3]

3.3 Die Hauptkomponenten von Unternehmen

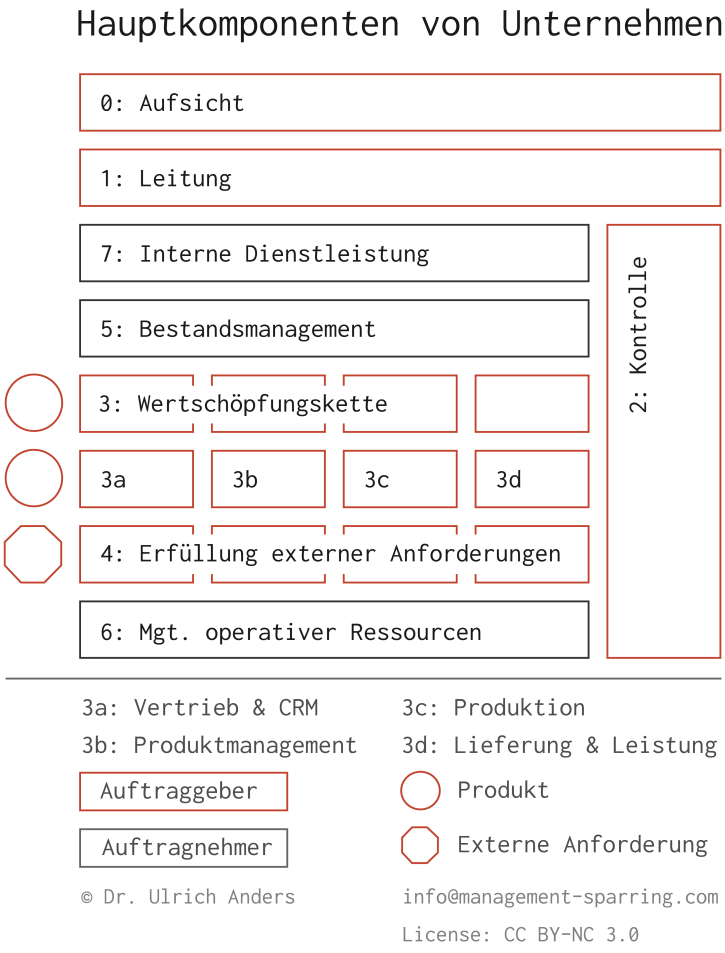
Jedes Unternehmen besteht aus vielen einzelnen Bereichen und Abteilungen, die das Ganze einer Organisation ausmachen. Diese sind in den verschiedenen Unternehmen auf die unterschiedlichste Art und Weise angeordnet, gegliedert und dimensioniert. Deshalb sind Organisationen, die man nicht kennt, auch teilweise schwer zu durchblicken. In Hinsicht auf ihre Organisation haben aber alle Unternehmen eins gemeinsam: die Hauptkomponenten — selbst wenn diese nur selten sichtbar sind. Jede Organisation lässt sich in genau acht von ihnen unterteilen. Das dem so ist, wurde an vielen Beispielen getestet und wirklich jeder bekannte Unternehmensbereich, jede Abteilung oder Funktion lässt sich mindestens einer dieser acht Hauptkomponenten zuordnen:

0. Aufsichtsfunktion
1. Leitung
2. Kontrolle
3. Produkt und Wertschöpfungskette mit

[3] Anders, Ulrich (2015a): »Die Hauptkomponenten von Unternehmen.«

- Vertrieb und *Customer Relationship Management*
- Produktmanagement inklusive Preisfindung und produktbezogenem Risikomanagement
- Produktion
- Lieferung und Leistung

4. Erfüllung gesetzlicher und regulatorischer Anforderungen
5. Bestandsmanagement (inkl. Kundenmanagement und Asset Management)
6. Management operativer Ressourcen
7. Interne Dienstleistungsfunktionen



Die den Hauptkomponenten zugeordneten Bereich und Abteilungen erben jetzt die Eigenschaften der Hauptkomponenten. Mit diesen kann auf einfache Weise Klarheit, Ordnung und Struktur geschaffen werden. Die Rollen und Verantwortungen ergeben sich nicht mehr durch individuelle Definitionen für einzelne Abteilungen, sondern sie leiten sich aus der Eigenschaft der Elternkomponente ab. Es gibt zwei prägende Eigenschaften der Hauptkomponenten:

1. **Ist Management- oder Controllingfunktionen:** Managementfunktion haben in Hinsicht auf die Entwicklung des Unternehmen eine essentielle Aufgabe: sie treffen Entscheidungen. Kontrollfunktionen hingegen müssen neutral bleiben und analysieren, messen und überwachen unabhängig die Arbeit und Entscheidungen der Managementfunktionen.
2. **Ist Auftraggeber oder Auftragnehmer:** Die Aufgabe des Auftraggebers besteht darin zu sagen, was benötigt wird, dies auch zu »bestellen« und mittels der internen Kostenrechnung auch

zu bezahlen. Die Aufgabe des Auftragnehmers besteht darin, die »Bestellung« mit geeigneter Qualität und zu möglichst günstigen Selbstkosten zu liefern.

Die Strukturierung einer Unternehmensorganisation in Hauptkomponenten hat viele Vorteile. Man plant zunächst die grobe Struktur. Diese ist nicht unternehmensspezifisch, sondern relativ allgemeingültig. Damit lässt sich eine Unternehmensorganisation grundsätzlich gut verstehen. Das ist wichtig für die innere Ordnung eines Unternehmens. Erst dann ordnet man die einzelnen Bereiche und Abteilungen den Hauptkomponenten zu. Daraus lassen sich dann die entsprechenden Zuständigkeiten und Rollen ableiten. Die Zuordnung einer Organisation zu Hauptkomponenten dient vielen Zwecken. Dazu gehört auch oft schon allein die Effizienzanalyse der Organisation selbst. In der Praxis offenbaren sich bereits bei der Zuordnung einer realen Organisation auf Hauptkomponenten viele ihrer Unklarheiten, die es — genau wie auch bei einer Webapplikaton — lohnt zu beseitigen.

4 RUST, »Ownership« und Verantwortung

Eine der schwierigsten Begriffe im Kontext von Unternehmen ist der Begriff der Verantwortung. Er taucht regelmäßig auf, wenn es um den Streit von Zuständigkeiten geht (»Das ist meine Verantwortung!«) oder wenn es darum geht, auf jeden Fall nicht für ein heikles Thema zur Rechenschaft gezogen werden zu wollen (»das ist die Verantwortung ... [der IT, der Personalabteilung, des Vorstands, des Risikomanagements usw.]«).

In der Tat ist es eigenartig, wenn man in den Unternehmen keine Verantwortlichen mehr für die selbst prominenten großen und kleinen Skandale der jüngeren Vergangenheit mehr findet.

Grund genug, sich das Thema der Verantwortung einmal anzusehen. Im Kontext von Verantwortung muss immer die Frage gestellt werden, Verantwortung wofür? Und hier hat in der organisatorischen Realität eine gewisse Entkopplung Einzug gehalten.

Ursprünglich erwuchs Verantwortung immer aus »Besitz«: Haus, Land, Kinder, Kraft, Begabung etc. Verantwortung bedeutete, dass man sich allumfassend um seinen »Besitz« kümmern musste. In modernen Organisationen hat man hingegen nicht mehr Verantwortung für »Besitz«, sondern nur noch für Funktionen und Aufgaben. Der Begriff der Verantwortung hat sich als von dem »Besitz« gelöst.

Das liegt daran, dass moderne Organisationen vor dem Hintergrund der Arbeitsteilung schön säuberlich in einzelne Funktionsbereiche zerlegt worden sind. Verantwortung wird heutzutage im Unternehmenskontext meist nur in Bezug auf eine Funktion verwendet. Damit hat man Verantwortung für die Aufgabe Vertrieb, Controlling, IT, Bilanzerstellung, Produktentwicklung usw. Die Verantwortung wurde also so lange zerteilt, bis sich die Gesamtverantwortung immer erst beim Vorstandsprecher oder bei dem leitenden Geschäftsführer wiederfindet. (Bei den jüngsten Verfehlungen in deutschen Konzernen musste immer erst jemand gefunden werden, der die Verantwortung übernimmt. Diejenigen zu identifizieren, die die Gesamtverantwortung hatten, schien jeweils unnöglich zu sein, da Verantwortung ja zerlegt und aufgeteilt wurde.) Ein Konzept, in dem es immer nur einen Leidtragenden gibt, nämlich den, der an der Spitze sitzt, und in dem alle anderen faktisch ein Freilos erhalten, kann langfristig nicht funktionieren.

Geht nun etwas signifikant schief, haben an dieser Sache immer viele Funktionen mitgewirkt, aber keine Funktion ist mehr allein verantwortlich. Jede Funktion hat immer nur einen kleinen Teil beigetragen.

Verursacher, die zur Rechenschaft gezogen werden könnten, gibt es damit nicht mehr. Man muss also in der Hierarchie immer weiter nach oben klettern und trifft dann letztendlich nur noch auf den Vorstandsvorsitzenden bzw. den Sprecher der Geschäftsführung, bei dem alles zusammen läuft. Dieser ist per se für alles verantwortlich und muss dann häufig genug die Verantwortung übernehmen und zurücktreten.

Ist das richtig? Eine Rücktrittsforderung an den obersten Chef, weil kein anderer gefunden, der die Verantwortung trägt, ist im Prinzip sicher falsch. Eine Rücktrittsforderung aus anderen Gründen, z.B. weil die Organisation falsch aufgesetzt wurde und damit Skandale erst möglich wurden, ist eine ganz andere Frage.

Jedenfalls widerspricht es dem Prinzip der Delegation, wenn zwar Themen und Aufgaben delegiert werden, die Verantwortung aber bei der Person an der Spitze verbleibt. Und genau das passiert bei der funktionalen Zerlegung von Verantwortung. Es ist eine bekannte und wahre Redensart, die dies trefflich zusammenfasst: »Sind zwei verantwortlich, ist am Ende keiner verantwortlich.« Ein zweites Problem tut sich bei der funktionalen Zerlegung der Verantwortung auf. Neue funktionale Aufgaben benötigen erst mal einen neuen Verantwortlichen. Solange dieser nicht bestimmt ist, fällt die Verantwortung zwischen die Stühle — und dass passiert in modernen Unternehmen häufig genug.

Mit der funktionalen Zerlegung wird eine Organisation im Extrem quasi zu einem verantwortungslosen Raum. Und genau das nimmt man bei den Skandalen wahr. Auch dass die Verantwortung für eine neue Sache keine natürliche Zuordnung hat, und damit zwischen die Stühle fallen kann, ist ein Systemfehler. So etwas dürfte eigentlich gar nicht passieren.

Für Führungskräfte und Manager ist dies eine an und für sich komfortable Position. Sie werden für vermeintlich große Verantwortung bezahlt, können auf dieser Grundlange aber nie (oder nur selten) wirklich zur Rechenschaft gezogen werden, da die verantwortete Funktion in einer arbeitsteiligen Welt immer nur einen Teil beiträgt.[4] Die höhere Bezahlung von Managern sollte sich also statt durch die Übernahme von Verantwortung mit anderen Kriterien rechtfertigen lassen: durch Leistung, Kompetenz, bestimmten Fähigkeiten, Wissen, Beanspruchung oder durch eine größere Jobunsicherheit.

Was ist also schief gelaufen mit dem Thema »Verantwortung«. Ist »Verantwortung« als Konzept gescheitert? Nein, ohne Verantwortung geht es nicht. Die Übernahme von Verantwortung ist unbedingt notwendig, um ein Unternehmen voran zu bringen. Jedoch muss Verantwortung immer mit dem Konzept »Ownership« einhergehen, sonst ist Verantwortung reine Augenwischerei.

Ein Beispiel soll das verdeutlichen. Einem Gärtner wurde die Verantwortung für die Pflege des Rasens in einem Garten übertragen. Er mäht den Rasen, fegt das Laub und düngt ihn entsprechend. Der Rasen im Garten wächst und gedeiht. Der Gärtner wurde der Verantwortung für seine Aufgabe gerecht. Ein Rosenstock im Garten fühlt sich hingegen nicht wohl. Er hat zu wenig Sonne und zu wenig Wasser. Der Gärtner kümmert sich nicht, denn seine Verantwortung besteht ja darin, sich um den Rasen zu sorgen — die Rose verkümmert. Anders wäre es hingegen, hätte der Gärtner die »Ownership« für den Garten übertragen bekommen. In diesem Fall hätte er sich auch um dem Rosenstock gekümmert — vielleicht sogar zuungunsten des Rasens, denn er hat ja beschränkte Ressourcen. In der Summe würde es aber dem Garten als Ganzes besser gehen.

Diese Situation findet man auch regelmäßig in Unternehmen vor. Wer das nicht glaubt, der braucht ja nur den »Owner« eines Produktes oder eines IT-Systems zu suchen: er wird ihn in der Regel nicht

[4] Das gilt natürlich nicht für Aktivitäten, die einen Straftatbestand darstellen.

finden.

Dass mangelnde »Ownership« ein signifikantes Problem darstellt und hohe Fehleranfälligkeit mit sich bringt, haben auch die Entwickler der Programmiersprache RUST erkannt.

4.1 Was ist RUST?

Was ist RUST? RUST ist eine neue Programmiersprache, die gleichzeitige und parallele Prozesse erlaubt, und derzeit ziemlich viel Aufmerksamkeit erfährt. RUST hat bei verschiedenen vorher existierenden Programmiersprachen Anleihen genommen, führt aber auch, um Sicherheit zu gewährleisten, neue Konzepte ein. Eines dieser Konzepte ist »Ownership«.

Die Entwickler von RUST haben nämlich erkannt, dass mangelnde »Ownership« (in diesem Fall von Variablen) Ursache für viele Fehler im Programmablauf sind. Das Prinzip »Ownership« stellt deshalb sicher, dass es immer nur genau einen »Owner« geben kann, der eine Sache verändern kann und sich damit um die Sache kümmern muss. Die »Ownership«–Regeln umfassen:

1. »Ownership« kann weitergeben werden, d.h. der Owner kann seinen Besitz an einen anderen Owner weitergeben.
2. Ein Owner kann seinen Besitz auch »verpachten«. Dabei kann er bestimmen, ob die Sache von dem »Pächter« verändert werden darf oder nicht.
3. Der Owner kann seinen Besitz auch an mehrere Nutzer zu Benutzung geben. Eine Sache, die von mehr als einem Nutzer benutzt wird, kann niemals von einem Nutzer verändert werden.

4.2 Was kann man von RUST lernen?

Das Konzept »Ownership« würde auch in der organisatorischen Realität sicher eine Reihe von Problemen lösen. »Ownership« ist prinzipiell nicht in Hinsicht auf eine Aufgabe definiert, sondern in Hinsicht auf eine faktisches oder virtuelles (in der Regel anfassbares) Ding. »Ownership« wird verliehen, das heißt, die Dinge gehen natürlich nicht faktisch in den Besitz eines »Owners« über.

Die Sachen gliedern sich in vier Bereiche: Produkte, externe Anforderung, Bestände und operative Ressourcen. All diese Elemente finden sich im Übrigen auch in den Hauptkomponenten von Unternehmen wieder.

Zu den Produkten eines Unternehmens gehören die externen und internen Produkte oder Dienstleistungen sowie deren Komponenten oder Module und natürlich auch die Ergebnisse von Projekten. Zu den externen Anforderungen die Dinge, die zu deren Erfüllung notwendig sind, als beispielsweise die Bilanz und G&V, bestimmte Berichte, Mitteilungen oder Beantragungen.

Zu den Beständen gehören beispielsweise Kunden, Verträge, Finanzmittel, Wissen, physische Bestände (wie zum Beispiel Gebäude) und Beteiligungen.

Zu den operativen Ressourcen gehören Human Resources, IT-Systeme, Informationen, Zulieferer usw.

Um gleich von vorneherein etwaigen Missverständnissen vorzubeugen. An manchen Stellen ist der Begriff des »Owners« problematisch, zum Beispiel im Kontext von Human Resources oder bei Zulieferern. Menschen haben natürlich keine Besitzer und wer dies so liest, möchte das Konzept der »Ownership« explizit falsch verstehen. Im Zusammenhang mit Human Resources kann man »Owner« vielleicht etwas ungelenk, aber am besten mit »hauptverantwortlicher Kümmerer« übersetzen.

Jedes der vorgenannten Dinge benötigt also einen eindeutigen »Owner«. Letztendlich entscheidet nur der »Owner«, was mit dem, was er besitzt, passieren darf.

Dabei kann er sein Besitztum gemäß dem RUST-Konzept weiterreichen, »verpachten«, »verleihen«, oder zur Benutzung durch mehrere Benutzer freigeben. Dabei muss er gleichzeitig entscheiden, ob die Benutzer Änderungen durchführen dürfen (also beispielsweise an einem IT-System). Die Eindeutigkeit des Besitzverhältnis wird dabei aber niemals aufgegeben.

Ebenfalls entscheidet allein der Besitzer, welche Modifikationen an seinem Besitztum herbeigeführt werden. Nicht also derjenige, der eine Modifikation durchführt ist verantwortlich, sondern der Besitzer, der die Modifikation akzeptiert hat.

Damit ist gewährleistet, dass die Verantwortungen im Unternehmen klar und eindeutig verteilt sind. Sie sind nämlich an die »Ownership« einer Sache gekoppelt.

Am Beispiel des Gärtners wird diese Situation noch einmal klarer. Dem Gärtner ist nichts vorzuwerfen, da er sich lediglich — so wie es seine Aufgabe verlangte — um den Rasen gekümmert. Leider verkümmerten dabei die Rosen. Was fehlte war die »Ownership« für den Garten. Hätte diese existiert, wäre der Gärtner mit Sicherheit auch beauftragt worden, sich mit um die Rosen zu kümmern. Wir erweitern jetzt das Beispiel: Seit Neuestem gibt es in dem Garten einen kleinen Haustierbestand. Solange es noch keine Funktion gibt, deren Aufgabe darin besteht, sich um Tiere zu kümmern, und die der »Owner« beauftragen kann, muss er sich eben selbst kümmern. Verantwortungslücken können so erst gar nicht entstehen.

Eine Organisation, in der die »Ownership« geklärt ist, — und das zeigt RUST ganz deutlich — ist weniger anfällig für Fehler und Misswirtschaft. Die Verantwortungen sind eindeutig, Delegation ist möglich, denn die Unternehmensspitze delegiert nicht mehr nur funktionale Verantwortungen, sondern teilt seine initiale, allumfassende »Ownership« auf und reicht diese mit der zugehörigen Verantwortung weiter.

5 REDUX und die Reduzierung von Komplexität

Komplexität war ja bereits ein Thema. Wir haben festgestellt, dass das Umfeld von Unternehmen immer komplexer wird und dass Unternehmen innere Ordnung benötigen, um dieser Komplexität zu begegnen. Denn mit Chaos kann man Komplexität auf keinen Fall beherrschen.

Was aber ist Komplexität? Komplexität bedeutet, dass sich ein System oder Prozess in seinem Verhalten nicht eindeutig vorhersagen lässt, dadurch dass

- in ihm mehrere Elemente vorhanden sind,
- zwischen den Elementen gegenseitige Beziehungen existieren und
- diese Beziehungen nichtlinear sind.

Eine Situation ist also umso komplexer, je mehr Elemente involviert sind, je stärker diese Elemente sich gegenseitig beeinflussen, je weniger gradlinig diese Beeinflussung stattfindet und je weniger gut dadurch eine Veränderung vorhersagbar ist.

Es gibt einige Grundsätze, die es ermöglichen, Komplexität zu reduzieren. Die Prinzipien der Komplexitätsreduzierung sind 1. Modularisieren, 2. Hierarchisieren, 3. Modellieren, 4. Visualisieren sowie 5. die Trennung von Zustand und Aktion.[5]

Ein komplexes System ist unter anderem auch deshalb komplex, weil in ihm so viele Aktionen passieren, dass der resultierende Zustand des Systems nicht mehr nachvollziehbar oder vorhersagbar ist.

Das haben auch die Entwickler von Redux erkannt und dafür eine Lösung gefunden.

5.1 Was ist Redux?

Redux ist wiederum eine Software-Bibliothek, die es ermöglicht die Komplexität in Webapplikationen zu reduzieren. Dies geschieht dadurch, dass die jeweiligen Zustände der Applikation streng von den Aktionen in der Applikation getrennt werden. Die Zustände werden also in einem separaten Modul verwaltet. Aus dem jeweils aktuellen Zustand und einer Aktion resultiert immer ein neuer Zustand. Der jeweils letzte Zustand ist also die Folge einer großen Anzahl von vorhergehenden Zuständen, die durch eine einzelnen Aktionen erzeugt wurden. Das nennt man auch das *State-Action-Model*. Damit lassen sich in einer Applikation selbst die komplexesten Wege und unerwartetsten Nutzerverhaltensweisen auf relativ einfache Art nachverfolgen und eventuelle Fehler, die auf diesem Weg aufgetaucht sind, korrigieren.

5.2 Was können wir von Redux lernen?

Unternehmen sind einer Reihe von internen und externen Einflüssen unterworfen, die auch noch stark voneinander abhängen. Oft gelingt es Entscheidern, Controllern oder Arbeitsgruppen nicht mehr, diese vielen Faktoren zu ordnen und hier einen Handlungsstrang zu entwickeln. Das Denken in Zustanden kann hier helfen, die Komplexität signifikant zu reduzieren. Nicht mehr die möglicherweise nachteiligen und unvorhersehbaren Konsequenzen einer Handlung stehen im Vordergrund, sondern die Entwicklung eines neuen Zustandes.

Die Trennung von Zustand und Aktion klingt als Konzept relativ trivial. Sie ist aber ein relativ weitreichendes Gedankenkonzept. In der Regel "denken" Unternehmen in Aktionen und Maßnahmen. Viel sinnvoller ist es hingegen in Zuständen zu denken und die notwendigen Aktionen aus den gewünschten Zuständen abzuleiten. Wie soll eine IT-Landschaft in 1, 2 oder 3 Jahren aussehen. Auf welchen internationalen Märkten will das Unternehmen in 1, 2 oder 3 Jahren präsent sein und seine Produkte verkaufen? Wie soll die Produktpalette in 1, 2, oder 3 Jahren prinzipiell aussehen. All diese Zustände kann man visualisieren und vorstellen. Die Strategie ist dann die Summe der Aktionen, die es ermöglichen sollen, diesen relativ gut spezifizierten zukünftigen Zustand auch zu erreichen.

In Zuständen zu denken, macht die Zukunft konkret. Zustände bieten ein nachvollziehbares Ziel und sie helfen dem Management, die Kraft des Unternehmens zu bündeln und zu konzentrieren und sich nicht von der Komplexität der Umgebung beherrschen zu lassen.

6 SCRUM, Agilität und die Iteration

Eine der größten Herausforderungen, denen sich Unternehmen heutzutage gegenüber sehen, ist, sich ständig anpassen zu müssen. Vorbei sind die Zeiten, in denen man von langer Hand planen, dann testen und dann Dinge einführen konnte. Zu schnelllebig sind die Zeiten geworden. Das Feedback der

[5] Siehe Anders, Ulrich (2105b).

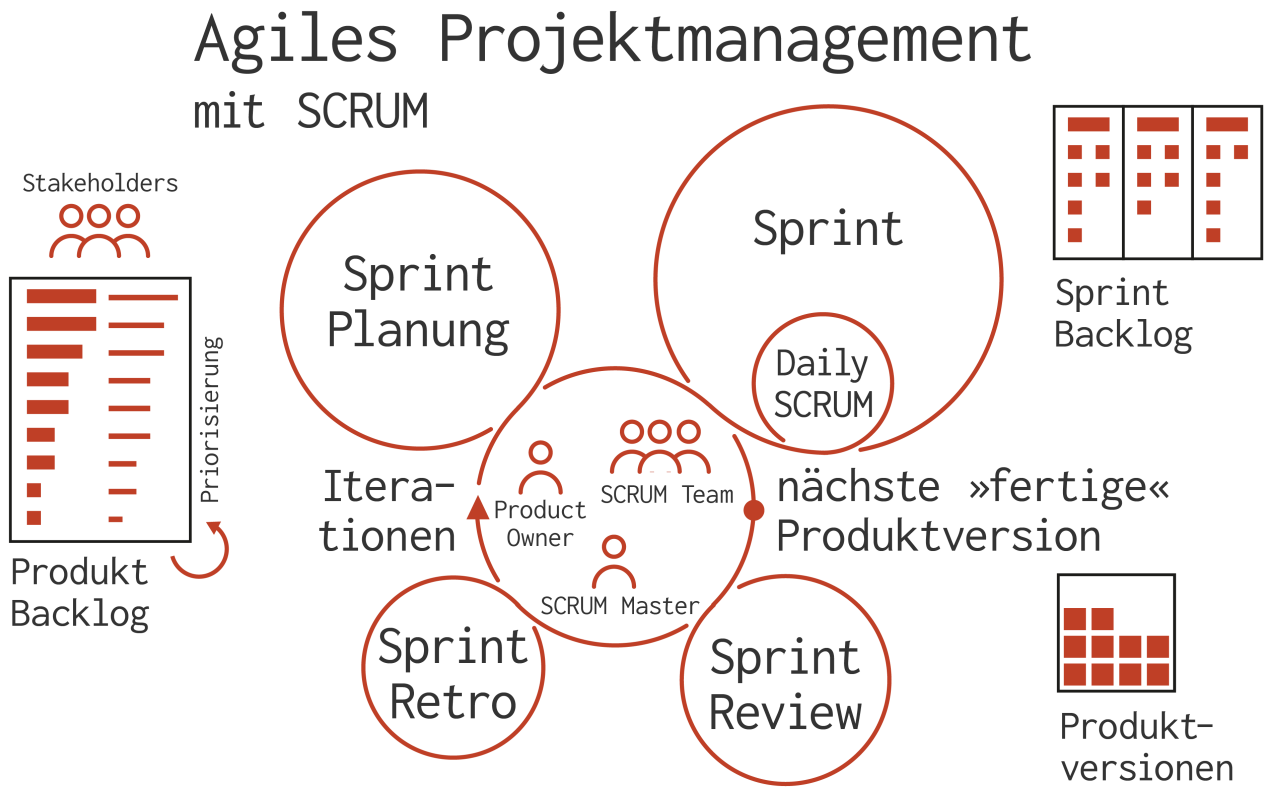
Kunden kommt innerhalb von wenigen Tagen und wird über die sozialen Medien auch noch um ein Vielfaches verstärkt.

Unternehmen, die es nicht gelernt haben, sich kurzfristig an den Bedarf ihrer Kunden anzupassen, haben mittelfristig wenig Chancen zu überleben. Die Zeitungen sind voll mit solchen Beispielen. Agilität ist gefragt. Aber wie erreicht man das in Unternehmen? Zu viele sind nach wie vor geprägt durch klassische hierarchische Strukturen, durch ein mittleres Management mit einer viel zu hohen Verweildauer an gleicher Stelle, durch eine relative Undurchlässigkeit für neue Ideen, wenn sie von unten aus der Organisation kommen, durch einer strikten Trennung zwischen IT- und Non-IT-Abteilungen und durch bürokratischen Prozesse.

Eine Antwort gibt SCRUM.

6.1 Was ist SCRUM?

SCRUM ist eine Projektmanagementmethode. Seinen Namen hat SCRUM von dem gleichnamigen Gedränge beim American Football. Anders als bei der klassischen Projektherangehensweise, ein Projekt von vorne bis hinten durchzuplanen, geht SCRUM in Iterationen vor und erweitert das SCRUM-Produkt sozusagen in Schritten.



Es gibt viele Vorteile dieses Ansatzes, insbesondere denjenigen, dass die Nutzer nach jeder Iteration die nächste Version des SCRUM–Produktes präsentiert bekommen und anschließend ihre Prioritäten und Nutzeranforderungen neu spezifizieren können. Damit werden SCRUM–Produkte immer ganz nah an den Bedürfnissen der Nutzer entwickelt. Die Methode empfiehlt sich vor allem auch dann, wenn die Nutzer die Anforderungen an das Endprodukt ex ante auch noch nicht final kennen, weil

sich viele Anforderungen erst ergeben, wenn man erste Versionen des SCRUM-Produktes betrachten kann.

In SCRUM–Projekten gibt es drei Rollen. Die SCRUM–Teammitglieder, den SCRUM–Master und den »Product-Owner«, der die zentrale Rolle spielt. Er repräsentiert die Nutzer oder Kunden des Endprodukts, stellt damit die Anforderungen an das Projekt und bekommt am Ende das Ergebnis des Projekts übergeben. Das SCRUM–Team entwickelt die jeweils nächste Version des SCRUM–Produkts autonom und wird dabei vom SCRUM–Master unterstützt.

6.2 Was können wir von SCRUM lernen?

SCRUM zeigt zunächst auf, wie wichtig es ist, nah am Kunden — sei er intern oder extern — zu agieren und wie das geht. Statt mit final ausgearbeiteten und dann unveränderbaren starren Anforderungen arbeitet SCRUM in Iteration, die konstantes Nutzerfeedback ermöglichen. Damit kann man viel schneller und flexibler reagieren. Vielleicht brauchen Organisationen in Zukunft viel mehr solcher flexiblen Strukturen, die flexibel, zeitlich begrenzt und agil arbeiten statt unbeweglich, langfristig und fest in die Organisationshierarchie eingebunden zu sein. Mit solchen Strukturen kann ein Unternehmen jedenfalls viel schneller auf neue Anforderungen des Marktes oder seiner Kunden reagieren.

Bei SCRUM treffen wir auch einen alten Bekannten wieder: das State-Action-Model. Am Ende einer Iteration hat ein SCRUM-Produkt einen neuen Zustand erreicht. Dazwischen finden mit den SCRUM-Sprints die Aktionen statt.

Zu Beginn jeder Iteration werden die Anforderung der Nutzer werden durch den »Product-Owner« spezifiziert. Und schon wieder begegnen wir einem Bekannten. Die Notwendigkeit der »Ownership«, die natürlich auch einen »Product-Owner« fordert, wurde bereits im Kapital über RUST dargelegt.

Die Tatsache, dass verschiedene Ansätze für unterschiedliche Herausforderungen gleiche Lösungen bieten, beweist einmal mehr die Sinnhaftigkeit der Lösung.

7 Zusammenfassung

Das Softwareumfeld hat eine Reihe von Lösungen für seine Herausforderungen gefunden: Zusammenarbeit in Projekten, Motivation zur Mitarbeit, Aufbau von Reputation, Konzentration auf Stärken, Komponentenbildung zur Lösung immer wiederkehrender Probleme, »Ownership« zur Fehlerreduzierung, Komplexitätsreduzierung durch Trennung von Zustand und Aktion, agiles und iteratives Verhalten zur Erhöhung von Kundennähe, Reagibilität und Geschwindigkeit.

Damit stellt das Softwareumfeld eine Reihe von Ansätzen bereit, die helfen können, die absolut vergleichbaren Probleme und Herausforderungen in Unternehmen zu bewältigen. Die Unternehmen, die Managementliteratur oder die BWL müssen sich nur bedienen und die Anwendung übertragen. Dabei ist es von großem Vorteil, dass die Ansätze im Softwareumfeld nicht nur bloße Konzepte sind, sondern schon tausendfach erprobt und angewendet werden. Das Wissen ist da und frei verfügbar. Man muss sich nur bedienen.

8 Referenzen

- Anders, Ulrich (2015a): »Die Hauptkomponenten von Unternehmen«.

  http://management-sparring.com/pdf/Die-Hauptkomponenten-von-Unternehmen.pdf
- Anders, Ulrich (2015b): »Komplexität im Unternehmensumfeld«.

  http://management-sparring.com/pdf/Komplexitaet-im-Unternehmensumfeld.pdf
- Gallup Studie (2014): »Engagement Index«.
- GitHub: https://github.com/open-source
- Malik, Fredmund (2014): »Führen – Leisten – Leben«.
- Pink, Daniel (2010): »Drive: The Surprising Truth About What Motivates Us«
- React: https://facebook.github.io/react/
- Redux: http://redux.js.org/
- Rubin, Kenneth S. (2012): »Essential Scrum: A Practical Guide to the Most Popular Agile Process«
- RUST: https://www.rust-lang.org/en-US/
- Sprenger, Reinhard K. (2015): »Das anständige Unternehmen: Was richtige Führung ausmacht – und was sie weglässt«.
- Stack Overflow: http://stackoverflow.com/
- The state of the octoverse 2016, https://octoverse.github.com/.
- Welch, Jack (2003): »Jack«