(FR)AGILES IT-MANAGEMENT

Dana Stoll
IT-Management Professional & Coach

ZUSAMMENFASSUNG

IT-Management konzentrierte sich in den letzten beiden Dekaden zunehmend auf die IT Infrastructure Library und verwandte Baukästen. Die Prozeduren von ITIL stellen *Best-Practice* zur Verfügung, die dazu geeignet ist, fragile IT-Systeme zu managen, mit einem Schwerpunkt auf Service-Qualität. Neue Produkte werden heute in einem Dialog mit dem Kunden unter Verwendung agiler Methoden erfunden. Um diese Projekte IT-technisch zu begleiten finden sich immer noch wenige Ansätze wie die Leitsätze von aliando oder DevOps. Das vorliegende Essay diskutiert die Notwendigkeit der beiden Vorgehensweisen über die von Nassim Taleb vorgeschlagene Klassifizierung in *fragile* und ~~*anti*~~*fragile* Systeme in Kombination mit dem Cynefin-System und zeigt Wege auf, wie ein integrativer, ganzheitlicher IT-Management-Ansatz aussehen kann. Inklusive typischer Irrtümer und Beispiele aus der IT-Management-Praxis.

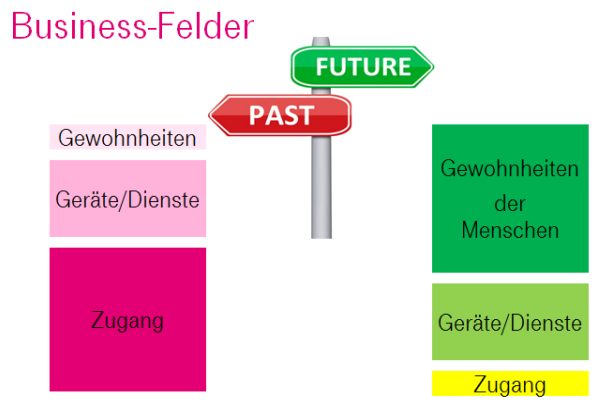
1 POST-DIGITALE INNOVATION

Zu Beginn der digitalen Revolution wurde Fortschritt maßgeblich durch die Übertragung von Produkten aus der analogen in die digitale Welt getrieben. Die Digitalisierung selbst bot durch die damit einhergehenden Vorteile ausreichend Marktpotenzial:

- Digitale Produkte sind durch den hohen Vernetzungsgrad quasi überall verfügbar.
- Die Digitalisierung kürzt den Kommunikationsprozess ab, da der zuvor für die Kommunikation nötige, analoge Transport von Medien entfällt.
- Die Digitalisierung erhöht den Grad der Automatisierung bzw. biete dafür geeignete Ansatzpunkte.

Das wesentliche, schöpferische Potenzial lag jedoch in der vorausgegangenen Erfindung der Systeme und Netzwerke innerhalb derer die digitalen Produkte erschaffen wurden. Die Übertragung der aus der analogen Welt bekannten Abläufe in definierte, digitale Kommunikationsstrukturen stellt in großen Teilen Ingenieurstätigkeit dar, auch wenn für jedes konkrete Vorhaben eine gewisse, neu hinzukommende Schaffenshöhe unterstellt werden darf.

Ebenso kann der Ausbau der Infrastruktur, z. B. die Bereitstellung von Netzzugang und dazu nötigen Komponenten (Access-Geschäft) als unternehmerische Erschließung eines bekannten Marktes mit prognostizierbarem Bedarf gesehen werden. Da die Informatik, bzw. die Digitale Industrie aus den Ingenieurswissenschaften heraus entstanden sind, liegt diese Vorgehensweise nahe.



1.1 Von Viralität nach Evolution

Diese Märkte sind jedoch weitgehend gesättigt. Neben Geräten und Diensten müssen für innovative Produkte zunehmend die Gewohnheiten der Menschen erschlossen werden. Beispiele dafür sind das Suchverhalten (Google, bing, et al.) bzw. die sozialen Netzwerke der Menschen (z. B. Facebook). Um die Gewohnheiten der Menschen in unbekannten, volatilen Märkten zu erschließen müssen innovative Produkte zusammen mit den Kunden erfunden werden. Seit der Jahrtausendwende wurde immer mehr Wert auf das „virale“ Potenzial von Neuerungen gelegt. Die ursprüngliche Vorstellung war, dass eine Erfindung solche Akzeptanz findet, dass eine kritische Masse von Benutzern erreicht wird, ab der sich ein Produkt von selbst im Schneeballsystem verbreitet.

Dabei wurde ein wesentlicher Aspekt übersehen: Virale Verbreitung funktioniert nur für zufällig überlebensfähige Kandidaten von nach dem Ingenieursprinzip gefertigten Produkten, die nach einer Phase der „Immunisierung“ bald wieder aussterben. Um echte, nachhaltige virale Verbreitung zu ermöglichen, muss man das generative Verhalten eines Virus durch das Produkt nachbilden. Das heißt: Ausgehend von einer anfänglichen, kleinen Idee wird ein Produkt im Dialog mit dem User erfunden. Um sich an den User bestmöglich anzupassen sind viele, kleine Schritte nötig, in denen man ihm funktionierendes Produkt präsentiert und seine Reaktion in die folgenden Produktentwicklungsschritte einfließen lässt.

Da sich auch das digitale Leben ständig neu erfindet, müssen sich die darin befindlichen Produkte ständig neu erfinden. Ein so entwickeltes Produkt ist nie „fertig“, sondern entwickelt sich weiter, solange sich innerhalb des von der grundsätzlichen Idee vorgegebenen Rahmens noch Innovative Kombinationen erschließen lassen. Am Anfang der Entwicklung stehen nur eine gemeinsame Idee und ein einfacher Prototyp, der schnellstmöglich mit dem Kunden zusammen „probiert“ werden muss.

War das klassische IT-Geschäft auf Quality of Service konzentriert, ist Time to Market innerhalb der Organisation dieses innovativen Dialogs ein entscheidender Faktor. Da diese evolvierende Produktentstehung sich mit klassischen Ingenieursmethoden schlecht vereinbaren lässt, erfreuen sich agile Entwicklungsmethoden heute weiter Beliebtheit. Zur agilen Umsetzung der IT-Managementaufgabe finden sich erste Ansätze wie z.B. DevOps oder die Leitlinien von aliando. Ein übergreifender, integrativer Ansatz kann folgendermaßen aussehen.

2 DER SYSTEM-LEBENSZYKLUS

Der Lebenszyklus jedes Systems (oder Produkts) von seiner Erfindung bis zu seinem Abmanagen lässt sich in vier Phasen einteilen, die auch als Quadranten des Cynefin-Modells bekannt sind:

2.1 Chaos

Am Anfang der Produktidee existiert von den Kunden- und Marktbedürfnissen nur eine schemenhafte Vorstellung. Die wesentlichen Entitäten müssen erst erschlossen und bildhaft sowie begrifflich zugänglich gemacht werden. Primäres Ziel dieser Phase ist, sich rasch in der neuen Umgebung zurechtzufinden (Novel Practice)

2.2 Komplexität

Aus der Kombination von bekannten und neu erschlossenen Produkteigenschaften entstehen neue, emergente Möglichkeiten, deren Auswirkungen und Akzeptanz nicht prognostizierbar sind. Wesentliche Strategie in dieser Phase ist, möglichst viele Kombinationen mit dem Kunden auszuprobieren und nur die weiterzuführen, die nach einer Testphase auch

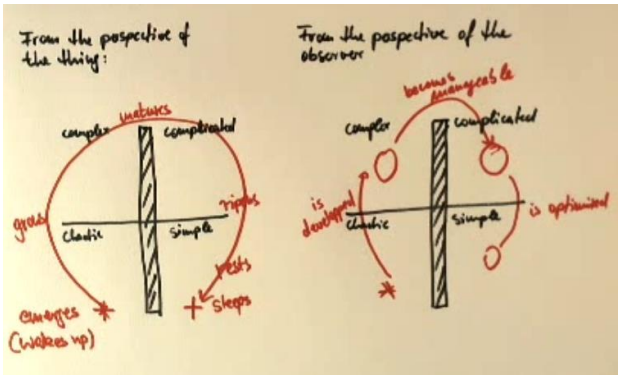
Akzeptanz bzw. Umsatzpotenzial entwickeln (Emergent Practice).

2.3 Kompliziertheit

In einer komplizierten Umgebung sind die wesentlichen Bestandteile erschlossen und in ihren Eigenschaften beschrieben. Kombinationen daraus lassen sich als Ingenieursaufgabe zusammenfügen. Die so erhaltenen Gebilde weisen jedoch noch einen hohen Vernetzungsgrad auf, der nur schwach gebündelt oder hierarchisiert ist. Daher sind zum Beherrschen ausreichend geschulte Experten nötig (Good Practice).

2.4 Einfachheit

Je besser das Produkt- und Marktverständnis voranschreitet, desto mehr besteht ein Produkt in seiner Umgebung aus einfachen Teilkomponenten, die weitgehend unabhängig voneinander zusammengefügt und ausgetauscht werden können. Die Management-Aufgabe ist in wesentlichen Teilen automatisiert, die fachliche Eindringtiefe für den Betrieb kann so gestaffelt werden, dass Experten nur noch in Ausnahmesituationen zu Rate gezogen werden müssen (Best Practice).



3 DIE AGILITY-FRAGILITY THRESHOLD (AF)

Der Übergang eines Systems aus einer komplexen in eine komplizierte Umgebung ist nichtlinear. Wasser, eben noch flüssig und beweglich, ändert innerhalb weniger Grad plötzlich seinen Aggregatszustand und härtet in kristallinen Strukturen aus.

Ein ähnlicher Übergang teilt die linken beiden Quadranten des Cynefin-Modells von den rechten beiden. In unterschiedliche Aggregatzustände für Services. Das heißt ein Produkt geht nicht „allmählich“ vom Zustand „komplex“ nach „kompliziert“ über. Vielmehr vollziehen diskrete Teile des Produkts diesen Schritt nach bestimmten, auslösenden Ereignissen (im Folgenden AF-Übergang genannt). Es bilden sich Eisklumpen. Für den liquiden Anteil sind agile Methoden zuständig, für den kristallinen die Methoden der Best Practice.

Aber wie findet man „flüssige“ und „feste“ IT-Systeme? Nassim Taleb liefert hier den entscheidenden Hinweis.

3.1 Fragile Systeme

Systeme in den rechten beiden Quadranten des Cynefin-Modells, also in komplizierter oder einfacher Umgebung, sind **fragil**.

Fragile Systeme kennzeichnen sich wesentlich dadurch, dass kleine, auf das System einwirkende Störungen auf das System einen verderblichen Einfluss haben und es ohne entsprechende Gegenmaßnahmen schrittweise zum Zerfall bringen. Maßnahmen zielen daher in erster Linie auf den Erhalt des Produkts, bzw. die Errichtung einer zur Gewährleistung der Service-Qualität notwendigen Schutzzone sowie Reparatur- und Wartungsmaßnahmen. Diese Maßnahmen dienen der Erhöhung der Stabilisierung.

In diesem Segment finden sich typischerweise *Cash Cows,* Massenmarkt- und Infrastrukturprodukte, die in gleicher oder ähnlicher Form von vielen Nutzern verwendet werden.

3.2 ~~(Antifr)~~agile Systeme

Systeme in den linken beiden Cynefin-Quadranten, also in chaotischer oder komplexer Umgebung, sind notwendigerweise **antifragil**.

Antifragile Systeme kennzeichnen sich dadurch, dass kleine, auf das System einwirkende Störungen sich positiv auf die Weiterentwicklung des Systems auswirken. Sie bieten den nötigen Stressor, der all diejenigen Kombinationen oder Vorgehensweisen invalidiert, die in der Umgebung nicht langfristig durchsetzungsfähig sind. Diese Maßnahmen dienen der **Adaption**.

Dabei darf Antifragilität nicht mit Robustheit verwechselt werden. Ein robustes System bleibt bei leichten Stressoren zwar unbeschadet, allerdings ergibt sich aus diesen für das System auch kein Nutzen.

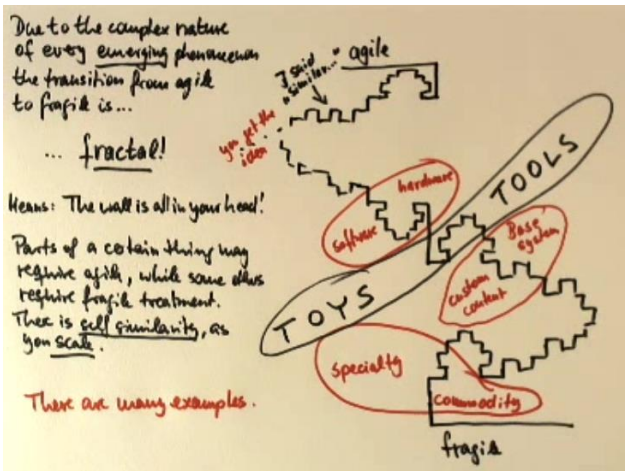
Nassim Taleb beschwerte sich darüber, dass es kein Gegenteil von „fragil“ gäbe, und erfand dafür die Antifragilität. In der IT kennen wir jedoch das Gegenteil schon seit geraumer Zeit: Wir bezeichnen diese Systeme als **agil**! In diesen beiden Quadranten finden die **agilen Methoden** Anwendung. Agil, in diesem Zusammenhang bedeutet: **schnell lernend** (und nicht etwa einfach „schnell hastend“). Um diesen Lernprozess zu treiben,

benötigt man genau diese kleinen Impulse. Das setzt jedoch voraus, dass ein System mit deren Hilfe überhaupt lernfähig ist.

In diesem Segment befinden sich typischerweise noch *Question Marks*, die je nach Potenzial in *Champs* und *Dogs* ausgelesen werden.

3.3 Diskussion

Ein System ist (als Ganzes) immer entweder fragil oder agil. Ein agiles System kann jedoch auch fragile Teilsysteme beinhalten und umgekehrt. Zentrales Element unseres (fr)agilen IT-Managements muss also der AF-Übergang für fragil gewordene Produktbestandteile sein.



Die zentrale Frage, die dabei beantwortet werden muss, ist der Schutzbedarf. Ab einem bestimmten Punkt werden Nutzer das Funktionieren eines Produktteils als so selbstverständlich betrachten, dass die Qualität dieses Teils des Service gegen die Störungen aus der Umwelt geschützt werden muss. Daher nennen wir den damit verbundenen Vorgang auch „Protection“. Ein Teilsystem oder Teilservice geht vom komplexen in einen nur komplizierten Zustand über und kann somit gemanagt werden. Würde er diesen Übergang nicht vollziehen, wäre er auf Dauer nicht beherrschbar.

Dabei darf das Hauptaugenmerk nicht auf dem Produkt als Ganzes liegen. Eine häufige Praxis ist die Trennung von Produktentwicklung und einer nachgelagerten Betriebsphase. Das ist zur Begleitung einer modernen Produkteinführung jedoch ungenügend. Vielmehr müssen über den Produktlebenszyklus immer wieder Produktteile ermittelt werden, die fortan wie fragile Bestandteile gemanagt werden.

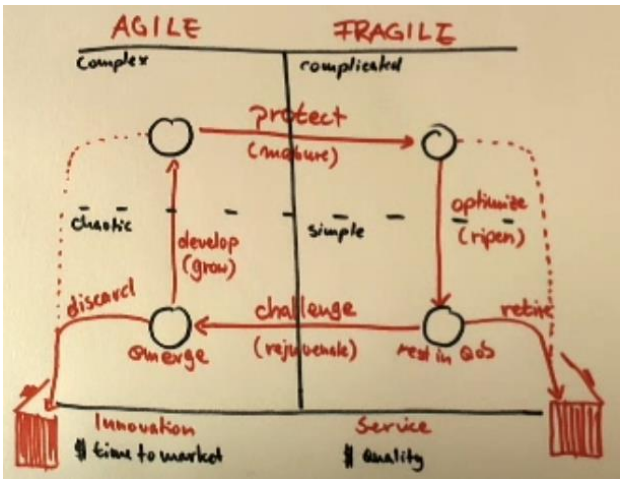
Anhaltspunkte für die Identifikation können sein:

- die Anzahl der abhängigen Teilkomponenten,
- die Anzahl der versorgten Benutzer,
- mit dem Kunden geschlossene Verträge,
- das Schadensausmaß beim Eintreten einer Störung.
- steigender Wartungsaufwand

Grundsätzlich ist eine Komponente dann als fragil anzusehen, wenn ihr Ausfall einen unmittelbaren Angriff auf den Erfolg des Gesamtprojekts darstellt, bzw. auf kleine Störungen keine sichtbare Weiterentwicklung mehr zu erkennen ist, sondern Maßnahmen zu deren Kompensation aufgesetzt werden. Die wesentlichen AF-Übergänge finden während der Weiterentwicklung von *Champs* statt.

4 (FR)AGILE IT-MANAGEMENTAUFGABE

Daraus lässt sich eine ganzheitliche, die gesamte Produktentwicklung überspannende IT-Managementaufgabe formulieren:



4.1 Emerge (Innovate)

Um Innovation zu ermöglichen, müssen innerhalb des IT-Managements speziell vorgesehene Methoden etabliert werden. Dazu gehören unter anderem:

- Die Bereitstellung von preisgünstiger Hardware und Test-Farmen
- Unkomplizierte Genehmigungsverfahren für innovative Experimente
- Mitarbeitern einen Teil ihrer Arbeitszeit für Innovation zur Verfügung stellen, projektunabhängig
- Ansprechpartner für Produkt- und Entwicklungsbereiche bereitstellen, die sich mit den innovativen Methoden und Verfahren innerhalb der gesamten IT auskennen

Hauptaugenmerk liegt darauf, möglichst wenige Widerstände zu erzeugen, so dass eine Idee kostengünstig ausprobiert werden kann, sowie möglichst viele Chancen zu schaffen, dass solche Ideen keimen können. Im Optimum hat ein Unternehmen Potenzial, allen für ein Produkt entstehenden Ideen eine Chance zu geben.

4.2 Agile Methoden

Die agilen Methoden zur Produktentwicklung legen besonderen Wert auf das Erzeugen einer raschen Abfolge von funktionierenden Prototypen, die in einem Pilot-/Alpha- oder Beta-Betrieb zusammen mit dem Kunden im Live-Betrieb weiter entwickelt werden können. Wesentliche Merkmale sind:

- Kurze Entscheidungswege
- Schnelle Entwicklungszyklen (viele Lernschritte)
- Rapid Deployment
- Steuerung über fortschreitende Priorisierung
- Abgespeckte Prozesse
- Der am besten geeignete Mitarbeiter erledigt den Job.

4.3 Best-Practice-Methoden

Die Best-Practice-Methoden stellen sicher, dass das Produkt in einer definierten Quality of Service zur Verfügung gestellt werden kann, und etablieren die dazu nötige Schutzzone. Merkmale sind:

- Fest definierte Rollen und Prozesse
- Vordefinierte Qualitätsstufen
- Hoher Standardisierungsgrad
- Sorgfältige, zeitaufwändige Planungsprozesse
- Umfangreiche Berechtigungs- und Freigabeprozesse

4.4 Retire

Das Retirement befasst sich mit dem Abmanagen von Produkten am Ende ihres Lebenszyklus, und soll hier zunächst nicht näher betrachtet werden, da es nach diesem Vorgang für den Betrieb keine Relevanz mehr besitzt. Das Retire eines Projekts kann mit klassischer Projektplanung abgewickelt werden.

4.5 Protect

Protection ist das Identifizieren fragil gewordener Anteile eines agilen Produkts, die fortan mit Best-Practice-Methoden behandelt werden müssen, da das Bereitstellen einer bestimmten Service-Qualität für das Überleben des Gesamtvorhabens essentiell geworden ist.

Solche Anteile sind typischerweise:

- Zentrale Datenbanken oder Komponenten, die für den Betrieb des kompletten Produkts unabdingbar sind,
- Plattformen und Betriebssysteme, sobald der Grad der Parallelität erhöht und ein zentrales Management nötig wird,
- Sicherheitsinfrastrukturen, sobald die Sammlung schutzbedürftiger Daten bei ihrer Kompromittierung eine signifikante Gefahr für das Projekt oder die das Projekt betreibende Organisation darstellt,
- Produktbestandteile, die für eine große Zahl von Anwendern zur „Commodity“ geworden sind, und in deren Augen nicht mehr als „Alpha“-Produkte betrachtet werden, sondern ein bestimmtes Maß an Qualität implizit vorausgesetzt wird.

4.6 Challenge

Ebenso gibt es innerhalb von fragilen Systemen immer wieder Teile, deren weiterer Verbleib in der Schutzzone in Frage zu stellen ist. Typische Anzeichen dafür sind steigende Wartungs- und Instandhaltungskosten bei dennoch starker Nachfrage. Ein Challenge beinhaltet, den Projektteil inklusive der betroffenen Projektmitarbeiter aus der eingeschwungenen Komfortzone zu befördern, um wieder Raum für echte Innovation zu ermöglichen.

Diese Aufgabe kann im Wesentlichen von erfahrenen Experten mit generalistischem Blick getragen werden. Dabei muss spürbarer Druck zur Veränderung aufgebaut werden, auch wenn sich dieser zunächst wie ein Risiko darstellt, das man aus der Sicht des Best-Practice-Betriebs eigentlich vermeiden möchte. Hierbei handelt es sich jedoch um ein notwendiges, unternehmerisch in Kauf zu nehmendes Risiko, da abzusehen ist, dass der Weiterbetrieb ohne innovativen Schub sich mittelfristig finanziell oder aus Architekturgründen nicht mehr darstellen lässt.

Die Methoden beinhalten:

- Schrittweises Eliminieren von Altkomponenten, so dass diese durch neue ersetzt werden müssen
- Festlegung von maximalen Lebensdauern für Komponenten
- Unverrückbare Deadlines für Generationssprünge
- Failover auf neue Versionen im Fehlerfall, wenn bereits Testsysteme und ausreichend Erfahrung mit der neuen Version vorliegen, so dass ein Gelingen der Migration Erfolg verspricht.
- Durchforsten der Landschaft nach Teilkomponenten, die nur noch durch „künstliche Beatmung“ am Leben erhalten werden.
- Einführen in neuer Version statt Re-Engineering, wobei nur die wirklich notwendigen Bestandteile in die neue Version überführt werden, und nebensächliche Funktionalität im weiteren Betrieb in die veränderte Umgebung neu eingepasst werden kann.
- Standardisierung von Komponenten

Zentrales Augenmerk beim Challenge ist ein Aufbrechen der Komplexität (Abhängigkeiten, Querbezüge), um so eine Neuordnung der Komponenten in einem klareren, einfacheren Modell zu ermöglichen.

Hier gilt ganz besonders der Grundsatz: „Wenn du es nicht ersetzen kannst, mach’s nicht kaputt!“

4.7 Coupling

Diese Vorgehensweise bedingt, agile Systeme mit fragilen Systemen interagieren. In den wenigsten Unternehmen kann man agile Systeme völlig losgelöst von bestehender (fragiler) Infrastruktur entwickeln. In der Praxis führt diese Zusammenarbeit häufig zum Streit über an die Komponenten zu stellende Anforderungen oder anzuwendende Methoden. Besonderen Wert bei der Kopplung agiler und fragiler Methoden muss darauf gelegt werden, dass:

- die antifragilen Teile so **lose** angekoppelt sind, dass die Quality of Service der fragilen Teile im Schadensfall nicht oder nur in geringem Maße beschädigt wird,
- der Schaden entweder durch die Robustheit des antifragilen Systems abgefedert wird oder zumindest das Risiko als akzeptabel klassifiziert wurde,
- falls nötig Reparaturmethoden bekannt und der Reparaturaufwand überschaubar und akzeptabel ist,
- die innovativen Kandidaten durch zu festes Coupling an fragile Anteile nicht von den starren Prozeduren der Best Practice in ihrer Entwicklung gebremst werden,
- durch die Innovation keine Sonderlocken geschaffen werden, deren Auflösung hohe Folgeaufwendungen nach sich zieht

4.8 Allgemeines

Grundsätzlich findet ein innovatives Produkt in einem Unternehmen ähnliche Bedingungen vor, wie innovative Organisationen insgesamt in ihren Märkten:

- Die Investition erfolgt schrittweise in überschaubarem Rahmen, damit trotz fehlender Business-Prognosen Budget-Obergrenzen definiert und das unternehmerische Risiko begrenzt werden können.
- Am Ende jeder Phase erfolgt ein Review, das die gesamte Unternehmung komplett in Frage stellt und auf dessen Basis über die Weiterführung des Vorhabens neu entschieden wird. Der erstellte Prototyp wird gegen die entstandenen Kosten und die bisherige Kundenresonanz sowie erschlossenen Marktpotenziale geprüft.
- Typischerweise muss die Finanzierung für die Anschlussphasen innerhalb des Unternehmens für jede Phase neu erkämpft werden.
- Die Skalierung des Systems erfolgt in Schritten. Entwickelt sich der Kandidat zum Champion, stellt sich nicht die Frage wie am besten, sondern viel häufiger wie innerhalb einer gewissen Zeit überhaupt finanzierbar skaliert werden kann um die rasch steigende Nachfrage zu decken.

5 DAS GROSSE MISSVERSTÄNDNIS

In vielen Unternehmen befindet sich die IT-Organisation gegenüber den Produktbereichen heute in der Defensive. Sätze wie „Das geht mit der IT nicht!“ sind seit der

Jahrtausendwende an der Tagesordnung. Dabei geben sich sowohl IT-Organisation als auch Produktbereich größte Mühe, bestmöglich mit dem Gegenüber zusammenzuarbeiten. Es muss also ein systematisches Missverständnis zugrunde liegen, das diese Grabenkämpfe befördert.

Durch die Fokussierung auf die Quality of Service, im Wesentlichen getragen von der Einführung von Best-Practice-Frameworks wie ITIL, aber auch durch externe Forderungen wie das KontraG, SOX, ISO9000/20000, IT-Governance und andere Rahmenbedinungen, wurden die IT-Organisationen systematisch dazu gedrängt, ihre Systeme wesentlich wie fragile Systeme zu behandeln. Dazu trägt auch bei, dass bei unklarer Anforderungssituation – wie im innovativen Fall – die Fachseite aus eigener Ambitioniertheit tendiert eher zu hohe anstatt zu niedrige Anforderungen zu stellen. Die IT-Organisationen sind aus ganzheitlichem Blickpunkt daher zu sehr auf fragile Systeme fokussiert, also quasi **überfragilisiert**.

Die Produkt- und Entwicklungsabteilungen hingegen sind von einem hohen Innovationsdruck getrieben. Daher haben sich dort seit Mitte der 80er Jahre bereits agile Methoden etabliert. Viele Ansätze, die „aus dem Bauch“ heraus praktiziert wurden, würde man heute dem zuschreiben, was unter den agilen Methoden zusammengefasst wird, seit ein dafür geeignetes Vokabular erschlossen wurde. Produkt- und Entwicklungsbereiche sind aus ganzheitlicher Sicht daher auf antifragile Produkte fokussiert, also eher **überagil**.

Misstrauen zwischen den Bereichen entsteht besonders dann, wenn:

- agile Bedürfnisse mit fragilen Forderungen beantwortet werden und umgekehrt, wenn
- auf fragile Bedürfnisse agile Antworten erfolgen.

Konkret: Wenn zur Innovation eines Produkts agile Prozesse benötigt werden, das Vorhaben aber nur dann umgesetzt werden soll, wenn es innerhalb eines Best-Practice-Rahmenprozesses betrieben wird, dann entsteht ein notwendiger Konflikt, denn der Rahmenprozess bietet für die iterative Entwicklung nicht die nötige Flexibilität.

Wenn an einem Qualitätsprodukt, das sich ggf. im festen Verbund mit anderen fragilen Produkten befindet, Änderungen nach agilen Methoden erfolgen sollen, ohne dass für diesen bestimmten Teil bewusst ein o.g. Challenge initiiert wurde, entsteht ein notwendiger Konflikt, denn die agilen Methoden können die geforderte Quality of Service des fragilen Produkts nicht sicherstellen und sind aus Produktsicht somit lebensgefährdend.

Die meisten mir bekannten Vorwürfe zwischen Produkt-, Entwicklungs- und IT-Organisationen haben ihre Ursache in Misstrauen, das auf diesem Missverständnis basiert.

5.1 Überfragilität mobilisieren

Um Überfragilität zu mobilisieren hilft es, sich folgende Umstände ins Gedächtnis zu rufen:

- Ein innovatives Produkt entsteht aus einem Dialog mit dem Kunden. Der genaue Umfang kann im Vorfeld nicht definiert werden, weder funktional, noch finanziell. Das zu fordern trägt zum Abbruch des Vorhabens, aber nicht zu dessen Lösung bei.
- Die Vorgehensweise muss unterwegs erfunden werden. Das birgt die Chance, während der weiteren Entwicklung Expertise mit in das Produkt einfließen zu lassen und einen Dialog zu etablieren, der Vertrauen schafft.
- Methoden wie „Scrum“ organisieren im Wesentlichen diesen Dialog.
- Die Etablierung dieses Vertrauens benötigt Zeit.
- Methoden, die für Touring-Maschinen geschaffen wurden, funktionieren im Allgemeinen nicht für komplexe Akteure oder Produkte, da Touring-Maschinen fragile Systeme sind. Das gilt auch für die meisten Prozessdiagramme, die auf Menschen angewendet werden sollen.
- Innovation benötigt über Algorithmen hinausgehende, intelligente Beiträge von Mitarbeitern. Innovation ist weder standardisierbar, noch automatisierbar.

Um Schaden von beteiligten, fragilen Systemen abzuwenden, lassen sich folgende Forderungen aufstellen:

- Agile und fragile Systeme müssen lose gekoppelt werden. Durch die Art und Weise der Kopplung kann sich das fragile System wirksam schützen.
- Die Ausarbeitung dieser Kopplung ist ein notwendiger Konflikt.
- Teile, die fragile Systeme direkt betreffen, müssen identifiziert und innerhalb der Best-Practice-Methoden abgewickelt werden.
- Während der Projektlaufzeit kann zu jedem Projekt-Review auch ein Review der Komponenten erfolgen, die möglicherweise fragil geworden sind und die *protectet* werden müssen.

5.2 Überagilität kanalisieren

Um den Tatendrang von Überagilität zu kanalisieren, können die folgenden Vorstellungen helfen:

- Wesentliche Aufgabe einer IT-Organisation ist die Sicherstellung der Quality of Service für Cash-Cows und Massenmarktprodukte. Diese Produkte sind Umsatzträger. Die daraus erwirtschafteten Gewinne sichern das aktuelle Geschäft, finanzieren also auch die Innovation.
- Aus dieser Warte heraus sind die Einkünfte aus möglichen, zukünftigen Produkten bestenfalls dubios.
- Im operativen Geschäft, das den Hauptanteil des IT-Betriebs ausmacht, schlägt die unmittelbare Gefährdung bestehender Umsatzträger im Allgemeinen die Anforderungen aus mittelfristigen Potenzialen. Diesem Phänomen, das auch aus dem Gesundheits- und Finanzwesen hinreichend bekannt ist, muss aktiv entgegengesteuert werden.
- Die Produktbereiche können durch Vermeidung überambitionierter Forderungen, wo tatsächliche Ziele noch nicht genannt werden, dazu beitragen, diesen Konflikt zu entschärfen, damit ein beidseitiger, realistischer Blick auf das Mögliche entstehen kann. Wesentlich überzogene Forderungen führen wahrscheinlich zu Schutzreaktionen zu Gunsten der fragilen Systeme und letztlich zum Abbruch des innovativen Vorhabens.

Dieser Prozess muss vor allem in frühen Produktphasen aktiv von Experten moderiert werden, die diese Konflikte kennen und über geeignete Methoden zu deren Lösung verfügen (das „A-Team“).

6 TYPISCHE, VERMEIDBARE FEHLER

6.1 Innovate-Fehler

- Die Methoden zur Zusammenarbeit agiler Produktentwicklung mit der IT-Organisation sind unbekannt oder ungenügend (ein Verweis auf ggf. „abgekürzte“ Verfahren oder einzelne, ausgelassene Phasen genügt nicht!)

  Es müssen nicht nur Phasen abgekürzt, sondern der komplette Rahmenprozess in viel kürzeren Zyklen mehrfach durchlaufen werden. Ebenfalls sind für die Verfahren wesentliche Rahmenparameter für die Innovation häufig noch unbekannt.

- Projekte werden von der IT-Organisation nur unterstützt oder begleitet, wenn ein Projekt nach dem Best-Service-Prozess dafür etabliert wurde.

  Die Voraussetzungen zur Projektpriorisierung können von der Innovation im Vergleich zu Projekten an fragilen Systemen selten erfüllt werden, da künftiges Potenzial gegen konkrete Umsätze selten argumentiert werden können. Durch die fehlende Prognostizierbarkeit antifragiler Systeme sind die Rahmendaten zu weich, um mit den harten Fakten der QoS-Welt zu konkurrieren.

- Es existieren keine Test-Farmen oder andere Möglichkeiten, neue Produkte in einem ersten Schritt auszuprobieren.

6.2 Agile Prozessfehler

- Teilweise verfügen Firmen zwar über Testfarmen und Entwicklungslabore, in denen neue Software getestet werden kann. In diesen lässt sich jedoch eine agile Produktentwicklung mit Live-Kunden, also ein Wachstum vom Prototypen über Pilot-, Alpha- und Beta-Betrieb, oft nicht abbilden. Ein Alpha-Betrieb übersteigt ggf. die Zielsetzung der Testfarm. Allerdings existiert in den Unternehmen dann keine kostengünstige Lücke, um das System bis zur Produktreife unterbrechungsfrei weiter zu betreiben. Der Betrieb in für fragile Systeme vorgesehenen Umgebungen ist für diese Phase häufig zu teuer und führt zum Einstellen des Projekts.

- In einigen typischen Bereichen wie Sicherheit werden während der komplexen Entwicklungsphase eines Produkts frühzeitig überzogene Forderungen gestellt. Sicherheitsforderungen sind dafür anfällig, da „ein bisschen“ Sicherheit genauso wenig hergestellt werden kann wie „100%ige“ Sicherheit. Daher tendieren vor allem größere Organisationen dazu, die tatsächliche Gefährdung eher zu überschätzen, und haben zusätzlich gleichzeitig einen umfangreichen Katalog an Requirements, der sich aus der Schublade ziehen lässt, „just to be on the safe side…“

Dahingegen ist in kleineren Organisationen zu beobachten, dass die Bedrohungen teilweise unterschätzt werden, insbesondere dann, wenn sich die Erarbeitung der zu treffenden Maßnahmen als zusätzliche Herausforderung darstellt.

Die Lösung des Problems besteht in der Verteilung der zu erfüllenden Security Requirements in die verschiedenen Betriebsphasen nach zu erwartendem Benutzer-Wachstum und dadurch entstehender, schutzbedürftiger Datensammlung. Die Verteilung muss von einem Experten für das konkrete Vorhaben vorgenommen werden. Dabei darf das Projekt durch zu frühe, übertriebene Forderungen nicht in seiner Entwicklung gefährdet werden, aber auch keine erhebliche Schutzlücke entstehen, die das Vorhaben oder das beherbergende Unternehmen nachhaltig schädigen können.

- Die IT-Organisation sieht sich nicht zuständig für agil zu entwickelnde Projekte.

  Dadurch schiebt die IT-Organisation wesentliche Schritte auf, die später kaum bewältigbaren Aufgaben in den „Protect“-Phasen führen und trägt aktiv zu den Grabenkämpfen und Misstrauensbekundungen bei. Besser ist es, den agilen Entwicklungsprozess von Anfang an zu begleiten und dabei besonderes Augenmerk auf die Protection fragil gewordener Projektanteile sowie das sich entwickelnde Coupling zur bestehenden Systemlandschaft zu legen. Dazu müssen innerhalb der IT-Organisation organisatorische Möglichkeiten, und vor allem Management-Unterstützung vorgesehen werden.

- Engineering-Methoden werden irrtümlicherweise auf komplexe Komponenten angewendet, wo anstatt Evolution stattfinden müsste.

  Die zum Anwenden von Engineering-Methoden nötigen Vorarbeiten sind aufwändig. Das Produkt muss im Wesentlichen in seinen Teilen umfänglich beschrieben sein. Diese Konzeption erstreckt sich oft über viele Monate, die für die Produktevolution verloren gehen. Am Ende dieser Konzeption steht oft ein geistiges, hypothetisches Produkt, in das zwar viele Expertenmeinungen eingeflossen sind, dessen Grundannahmen jedoch bereits in den ersten tatsächlichen Produktiterationen verworfen werden müssen.

6.3 Protect-Fehler

- Viele Unternehmen verfügen über getrennte Entwicklungsabteilungen, die auch über eigene

Infrastrukturen verfügen, um den Alpha- und Beta-Betrieb abzudecken. Diese sind oft aus der Notwendigkeit heraus entstanden, dass die von der Best-Practice-Infrastruktur angebotenen Möglichkeiten in frühen Phasen zu kostspielig sind. Daher laufen die innovativen Projekte über einen langen Zeitraum parallel, bis über ihre definitive Marktreife befunden wird.

Zu diesem Zeitpunkt versucht man dann, die Projekte als Ganzes in den Best-Service-Betrieb zu überführen, was im laufenden Betrieb eine gewaltige Aufgabe darstellt. Das führt häufig dazu, dass diese Produkte über einen langen Zeitraum hinweg im Parallelbetrieb weiter betrieben werden. Dabei können sie nicht von der gemeinsamen Infrastruktur der Best-Practice-Landschaft profitieren (Availability-, Capacity-, Change-, Release-Management-Prozesse, usw.), und es werden kostspielige Parallel-Lösungen geschaffen. Dieser **Big-Bang-Protect** funktioniert genauso schlecht wie Big-Bang-Produkteinführungen am Markt.

Besser ist es, rechtzeitig Teilbereiche des Produktes zu identifizieren, die in den Best-Practice-Betrieb überführt werden müssen und diese Aufgabe in Stufen, getrieben von der jeweiligen, tatsächlichen Produktsituation wahrzunehmen.

- Eine IT-Organisation depriorisiert Maßnahmen, die als „Protect“ gedacht sind, die in die Best-Practice-Prozesse eingebracht werden, da sie sie fälschlicherweise für innovative Anteile hält.

  Schon im innovativen Betrieb kann es von Vorteil sein, gewisse Funktionalitäten oder lose Kopplung mit Best-Practice-gemanagten Systemen sicherzustellen, z.B. um zu gewährleisten, dass im Support die Standard-Methoden zur Identifikation von Kunden benutzt werden, bevor ein innovatives System zum Einsatz kommt, oder der Aufruf dieses Systems für die Agenten nur aus der Standard-Arbeitsumgebung heraus möglich ist, die wesentliche Rahmenbedingungen sicherstellt, die für das innovative Projekt selbst nicht möglich sind, wie z.B. das Vorhandensein eines offenen Tickets. Diese Maßnahmen können, wenn sie als Standalone-Maßnahmen eingebracht werden, selten innerhalb der üblichen Priorisierungsverfahren wirtschaftlich abgebildet werden.

  Besser ist es, diese Maßnahmen eigens zu kennzeichnen und darüber separat zu befinden.

- Die Kostenrechnungsmodelle sehen innovative Finanzierungsmodelle nicht vor. Protect-Maßnahmen werden ebenfalls häufig depriorisiert, wenn die IT-Organisationen pauschal als Cost-Center betrieben werden und eine Finanzierung von Aufwendungen für Protect-Maßnahmen direkt über die Projektbudgets nicht möglich ist. Innovation wird dann ggf. durch Kostensparmaßnahmen verhindert, was verheerende Folgen für den mittelfristigen finanziellen Erfolg oder die infrastrukturelle Partitionierung des Unternehmens haben kann.

  Besser ist es, die Kostenrechnungsmodelle speziell für innovative Projekte flexibel zu gestalten und innovative Anteile von IT-Kosten nicht über Gemeinkosten abzubilden, sondern den Projekten direkt zuzuordnen.

- Die Protection wird für fragil gewordene Komponenten zu spät durchgeführt. Die daraus resultierende mangelnde Quality of Service hat verheerende Auswirkungen auf die weitere Akzeptanz durch die Nutzer.

- Die Protection wird für Komponenten zu früh durchgeführt, so dass das Produkt in seiner weiteren Entwicklung gebremst wird. Da die Idee nun bereits am Markt platziert ist besteht die Chance, dass man die bereits erworbenen User an Konkurrenzangebote verliert.

6.4 Challenge-Fehler

- Komponenten, die dringend erneuert werden müssen, werden nicht identifiziert. Dadurch erhöht sich schrittweise die Komplexität der Gesamtlandschaft, ohne dass die notwendige Neustrukturierung ihrer Teilkomponenten vorgenommen wird. Somit sind nicht nur die einzelnen Teile, sondern die gesamte Best-Service-Landschaft fragil geworden. Der Ausfall einzelner Komponenten kann große Teile in Mitleidenschaft ziehen, da die Zusammenhänge im Gesamtsystem immer undurchschaubarer werden. In besonders unübersichtlichen Situationen kann es sogar vorkommen, dass ein Wiederanfahren eines Systems nicht möglich ist, da Komponenten abhängig voneinander entstanden sind und jede der Beiden zum Start den Betrieb der anderen Komponente bereits voraussetzt.

  Um solche Abhängigkeiten zu reduzieren, müssen regelmäßig und systematisch Komponenten identifiziert werden, für die ein Challenge durchgeführt werden muss.

- Challenge wirft nicht aus der Komfort-Zone. Challenge bedeutet im eigentlichen Sinne, dass der Betrieb einer Komponente aufgegeben wird, damit schnellstmöglich die Notwendigkeit geschaffen wird, sie durch eine neue Komponente zu ersetzen.

- Der Challenge migriert zu viele Altlasten. Ziel des Challenge ist es, ein möglichst minimales Neusystem zu etablieren, das neben einer besseren strukturellen Integration auch bezüglich seiner inneren Komplexität minimiert ist. Dazu gehört es, nicht mehr benötigte Daten bei der Erneuerung über Bord zu werfen und nur das

wirklich benötigte Mitzunehmen. Dazu muss oft ein spezieller Druck aufgebaut werden. Das kann auch für Funktionalitäten sinnvoll sein, die bei einer Neuentwicklung mit großer Wahrscheinlichkeit anders erstellt werden, als sie in der gegenwärtigen Landschaft implementiert sind.

- Es werden Komponenten gechallenget, die kurzfristig nicht ersetzt werden können. Aus diesem Fehler erwuchs einst die Direktive „Never touch a running system“. Besser wäre es, zu formulieren: „Never break a system you cannot replace!“

6.5 Best-Practice-Fehler

- Agile Methoden werden auf fragile Systeme angewendet. Schwieriger zu erkennen ist, wenn die agilen Methoden auf Systeme angewendet werden, die mit fragilen Systemen zu starr gekoppelt sind, und dadurch in Mitleidenschaft gezogen werden können.
- Die Best-Practice-Anteile werden beim Aufsetzen der agilen Projekte nicht sauber herausgearbeitet. Auch agil betriebene Projekte benötigen in späteren Projektphasen Schnittstellen zu bestehenden Systemen. Teilweise haben diese Systeme jedoch deutlich längere Vorlaufzeiten und die Kopplung muss in einem bestehenden Best-Practice-Prozess miterheblichen Fristen vorgenommen werden. Daher müssen solche Kandidaten bereits in einer frühen Phase identifiziert und deren vorsorglich beauftragt werden, ggf. mit einem „Opt-out“, sollte sich die Kopplung zu einem späteren Zeitpunkt als überflüssig erweisen.
- Die Mobilisierung eines Systems wird nicht ausreichend abgesichert. Um iterative Vorgehensweise zu ermöglichen ist es nötig, dass die Fachseiten konfigurativ per Trial and Error neue Prozesse erschließen. An solche Umgebungen sind besondere Anforderungen zu stellen (vgl. „Sandbox“).
- Die Methoden der Best Practice werden nicht in einem kontinuierlichen Verbesserungsprozess vereinfacht.
- Die Qualitätssicherungsmaßnahmen innerhalb der Best Practice sind nicht ausreichend.

6.6 Retire-Fehler

- Das Retire von Altsystemen wird zu lange hinausgezögert, so dass Kapazitäten unnötig lange gebunden werden, die für andere Arbeiten nicht zur Verfügung stehen.
- Abhängige Komponenten erhalten durch verzögertes oder übervorsichtiges Retire eines Produktes kein Challenge. Dadurch erhöht sich die Komplexität, und damit die Fragilität der Gesamtlandschaft mit teilweise katastrophalen Folgen bei unvorhergesehenen Ereignissen.

7 STRATEGIEN

7.1 Fragile Strategien

Als Idealvorstellung für ein fragiles System kann ein vollständig robustes System dienen. Robustheit bedeutet, dass kleine Störungen über einen längeren Zeitraum nicht zum allmählichen Ableben des Systems führen. Allerdings sind 100% robuste Systeme in der Praxis nicht zu erreichen, und der Aufwand, sie jenseits der 99% herzustellen, steigt mit jeder Nachkommastelle. Daher kann fehlende Robustheit von der Service-Organisation, in die die Systeme eingebettet sind, aufgefangen werden. Im Optimalfall sind nicht nur die Systeme robust, sondern bis zu einem bestimmten Grad auch die Organisation. Da die Organisation, neben definierten Prozessen noch komplexe Akteure umfasst, ist im Idealfall die Organisation der Service-Organisation selbst sogar antifragil und besitzt über ein hohes Maß an Kontingenz.

7.1.1 Die Fragilen Prinzipien

- Fragiles lernen funktioniert langsam. Eine Methode wird erst dann eingeführt, wenn sie sich tausendfach in der Praxis bewährt hat.
- Keine Experimente.
- If you cannot fix it, don't break it.
- Besser regelmäßig die Backups testen.
- Lieber zweimal nachsehen.
- Besser noch einmal nachfragen, ob es auch wirklich erledigt ist.
- Erst der Plan, dann die Ausführung.
- Vereinbarungen sind verbindlich, und Terminpläne unbedingt einzuhalten.
- In der Best Practice steht, wie man es am besten macht, und es gibt wirklich keinen Grund, warum jetzt jeder eine eigene Version davon einführen muss.
- Qualität hat ihren Preis.
- Wenn etwas auseinanderzubrechen droht, muss es rechtzeitig ins Trockendock.
- Wartung wird nicht aufgeschoben.
- Finger weg.
- Ich bleibe nebendran stehen und schau dir über die Schulter, bis es fertig ist.

7.1.2 Präventive Maßnahmen an den Systemen

Präventive Maßnahmen an den Systemen erhöhen im Wesentlichen deren Robustheit. Dazu zählen:

- Die Verwendung von Komponenten mit höherer Qualität, die sich in längerer Lebensdauer oder besserer „Belastbarkeit", d.h. Benutzbarkeit auch bei sich ändernden Umgebungsparametern niederschlägt.
- Die Bildung von Redundanzen. Redundanzen in fragilen Systemen müssen möglichst „gleichartig" gestrickt sein, so dass ein Knoten im Fehlerfall die Aufgabe eines anderen Knotens mit übernehmen kann.
- Zeitversetzte Redundanzen, z. B. Backups.
- Standardisierung dient im Wesentlichen dazu, die Komplexität der Redundanzenbildung zu verringern, so dass weniger unterschiedliche Komponente eingesetzt, verknüpft, im Lager vorgehalten, usw. werden müssen.
- Präventive Wartungsarbeiten.

7.1.3 Präventive Maßnahmen an der Organisation

Präventive Maßnahmen an der Organisation umfassen:

- Die Etablierung von Monitoring- und Überwachungssystemen, die einen Ausfall zwar nicht vorhersagen, aber feststellen können.
- Die Etablierung von Notfall-Prozeduren, um möglichst schnell den Normalbetrieb wieder aufnehmen zu können.
- Das Training der Notfallprozeduren in der Organisation, damit diese im Ernstfall auch tatsächlich angewendet werden können. Fehlt dieser Schritt, sind die Prozeduren zwar dokumentiert, aber in der Praxis nicht existent.
- Erhöhte Aufmerksamkeit und Handlungsbereitschaft in kritischen Phasen

Im operativen Betrieb muss besonders darauf geachtet werden, das über die Köpfe verteilte Sachwissen ausreichend und aktuell zu halten, z.B. durch regelmäßiges Auffrischen von Trainingsmaßnahmen.
In der Praxis zeigt sich die mangelnde aktive Präsenz der nötigen Informationen in den Köpfen der Mitarbeiter meist durch eine häufig erwähnte Forderung nach besserer Dokumentation. Diese Dokumentation ist jedoch in viele Fällen im Ernstfall gar nicht zu sichten, da das den Zeitrahmen sprengen würde und im Ernstfall die kognitiven Fähigkeiten, insbesondere die Informationsaufnahme und deren Verarbeitung durch die Ausschüttung von Stresshormonen stark eingeschränkt sind.

7.2 Agile Strategien

Die Strategien der linken beiden Cynefin-Quadranten haben zur Aufgabe, die Antifragilität der dort betriebenen Vorhaben sicherzustellen. Dazu gehören:

- Das Herstellen eines gewissen Maßes an Handlungsfähigkeit (Kontingenz), damit neue Möglichkeiten überhaupt erschlossen werden können.

- Das etablieren einer bestimmten Form von Stress, damit die Vorhaben gezwungen sind, durch Überkompensation die notwendigen Fortschritte zu durchlaufen. Das geschieht teilweise automatisch durch natürliches Interesse und Lernvorgänge, kann jedoch aus „interner Konkurrenz“ resultieren, solange diese Konkurrenz grundsätzlich angstfrei bleibt.

7.2.1 Die agilen Prinzipien

- Wachstum tut weh.
- Du bist nie zu alt, auf die Nase zu fallen.
- Es ist nie zu spät, wieder aufzustehen.
- Stress tötet Produktivität.
- Ergebnisse werden dadurch erzielt, nicht erreichbar zu sein.
- Die faulste und einfachste Art und Weise, etwas zu tun, ist die intelligenteste.
- Wenn etwas kaputt geht, wirf’s weg und mach’s neu.
- Wenn der Motor ausgeht, wirf ihn wieder an. Er war nur aus, es ist nichts falsch.
- Persönliches Chaos ist produktiver als zentrale Organisation.
- Zwei Leute denken besser als nur einer.
- Pläne von heute sind morgen veraltet. Dokumentation auch.
- Richtung statt Ziel.
- Ran an die Buletten.
- Ungefähre Vorstellungen sind völlig ausreichend.
- Führen von vorne. Am besten im Zweifelsfall selbst Hand anlegen können. Zumindest aber kompetenter Sparringspartner sein.
- Formulare? Was sind Formulare? Listen? Ugh!

7.2.2 Präventive Maßnahmen an den Systemen

- Preiswerte Komponenten und Plattformen
- Möglichst einfache Bausteine, die Komplexität ergibt sich aus den Kombinationen
- Prototypische Produktentwicklung

7.2.3 Präventive Maßnahmen an der Organisation

- direkte Kommunikation im gesamten Team mit hoher Abstimmungsrate,
- Praxis des „Un-Management“
- flache (am besten gar keine) Hierarchien
- gestaffelte Projekt- und Budgetfreigaben, die sich als eigenständige Generationen innerhalb der Entwicklung sehen anstatt von mehreren Phasen innerhalb eines Gesamtprojekts, das erst am Ende ein Produkt liefert.

8 FINANZPROBLEME …

8.1 Wofür Geld ausgeben?

Die Priorisierung innerhalb der Best-Practice-Prozesse läuft üblicherweise so:

> Ein Prophet ruft vom Berg: „Wenn du mir 5 Millionen Euro gibst, bau ich dir diese Maßnahme, und dann wirst du 8 Millionen Euro ernten!“

Woher weiß der Prophet das, wenn sich Innovation gar nicht prognostizieren lässt?

Wenn mir früher jemand begegnet ist, der mir gesagt hat: „Gib mir 5000 Euro, dann wird jemand kommen, der gibt dir dafür 8000!“, dann war ich immer äußerst skeptisch. Ich verstehe nicht, wie sich das als allgemeine Praxis in der Geschäftswelt etablieren konnte. An diesem grundlegenden Sachverhalt ändert sich nichts, egal wie plausibel und detailliert die Business-Pläne auch aussehen mögen. Ob eine Maßnahme greift, lässt sich erst feststellen, wenn man sie ausprobiert.

Wenn dir jemand sagt, er wüsste im Vorhinein, wie viel Profit eine Maßnahme abwerfen wird, dann schick ihn weg.

Es ist besser, zunächst viele Ansätze auszuprobieren und dann nach anfänglichen Phasen nur diejenigen weiterzuentwickeln, die sich auch als tatsächlich plausibel erweisen. Man könnte jetzt geneigt sein, anzunehmen, ohne konkretes Ziel vor Augen sei das ein Fass ohne Boden. Tatsächlich ist es umgekehrt: Durch die prototypische Vorgehensweise wird gerade vermieden, dass für Vorhaben, die sich nach ersten Tests mit den Kunden nicht als vielversprechend erweisen, weiter Geld investiert wird. Die Vorgehensweise ist empirisch.

Das ist besser, als das ganze Geld auf eine Karte zu setzen. Sind die Millionen erst in die Entwicklung einer Maßnahme geflossen, die sich dann doch nicht als Champion erweist, ist die Versuchung groß, dort noch weitere Mittel nachzuschießen, um die Maßnahme „doch noch irgendwie zum Fliegen zu bringen“. Doch das ist Doktor spielen an einem ohnehin todgeweihten Patienten.

8.2 Kostensenkung bei fragilen Systemen

In schwierigen Zeiten hört man vor allem bei großen Organisationen immer wieder, dass Kosten gespart werden müssen. Das ist richtig und falsch.

Grundannahme meiner Argumentation ist die Folgende: **Die Umsätze, die ein fragiles System in der Lage ist, zu erzeugen, nehmen über die Zeit beständig ab.** Das heißt: Kostensenkende Maßnahmen können im Sinne der Gesamtwirtschaftlichkeit nur kurzfristige Rentabilität erzielen.

Kostensenkung ist genau dann richtig und notwendig, wenn die vorhandene Kostenstruktur (z.B. durch Verpflichtungen, Schulden) sich auf das Gesamtunternehmen fragilisierend auswirkt. Dann hat

das Unternehmen die Aufgabe, diese Teile einem Challenge zu unterwerfen, um sich zu mobilisieren.

Neue Umsätze entstehen mittelfristig aber immer aus den innovativen Maßnahmen der agilen Welt, die in der Lage sind, sich zu Champions zu entwickeln. Kosten senken darf nicht heißen, keine Mittel mehr aufzuwenden, diese innovativen Maßnahmen durchzuführen.

Kosten senken darf auch nicht bedeuten, Maßnahmen nicht durchzuführen, wenn sich dadurch die Fragilität der Systeme der Service-Organisation zu erhöhen (wie z. B. auf nötige Wartungs- oder Instandsetzungsarbeiten zu verzichten). Wer diese Strategie anwendet, der schädigt bewusst mittelfristig das Unternehmen, um kurzfristige Effekte darstellen zu können.

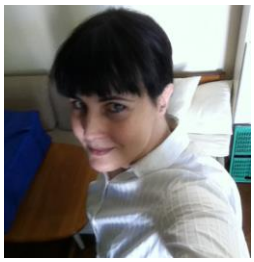
Wer Kosten senken möchte sollte sich vornehmlich um Maßnahmen kümmern, die:

- die Komplexität des Gesamtsystems reduzieren,
- Verbindlichkeiten abbauen, die die Kontingenz der Organisation einschränken,
- bei fragilen Systemen Redundanzen abbauen, die im Regelbetrieb nicht benötigt werden,
- Altsysteme abzumanagen, die immer noch Ressourcen binden, aber nicht mehr zwingend benötigt werden,
- Projekte einstellten die zwar Innovation versprechen, aber nicht mit geeigneten Methoden entwickelt werden,
- Vorhaben beenden, die trotz sich nicht einstellender Erfolgsaussichten trotzdem weiter finanziert werden,
- Systemteile identifizieren, die auf Infrastruktur mit zu hoher Quality of Service betrieben werden, weil sie üblicherweise teurer ist als das Projekt erfordert,
- die gelieferte Quality of Service auf ein für das Produkt dienliches Maß beschränkt,
- dort zusätzlich Redundanzen schaffen, wo Kompensationsmaßnahmen (Reparaturen, Instandsetzung) hohe, wiederkehrende Kosten verursachen,
- dort Standardisierung vorantreibt, wo durch gemeinsame Verwendung von Ressourcen die Systemlandschaft vereinfacht und dadurch Kosten gespart werden können,
- ungenutzte oder unausgelastete Ressourcen identifizieren.

8.3 Kostensenkung bei agilen Systemen

Kostensenkungsmaßnahmen für agile Systeme sind:

- Kürzere Entwicklungszyklen innerhalb der prototypischen Entwicklung,
- Konsequentes Verwerfen von Ansätzen, die nach erster Produkterfahrung keine Aussicht auf Generierung von Umsätzen versprechen,
- Eliminieren langwieriger Planungsszenarien und Diskussionen, bevor eine Idee konkret ausprobiert wird (das sind die eigentlichen Kostenfresser)
- Schaffung von Infrastrukturen zur Abwicklung solcher Projekte (Innovation Farm), damit Ressourcen für nachfolgende Versuche wiederverwendet werden können.
- Vermeiden, Menschen zu demotivieren, die mit Ansätzen befasst waren, die nicht überlebt haben. Wenn man deren Motivation erhält, und sie ihren Drive in neue Projekte stecken können, profitiert das Unternehmen wahrscheinlich mehr als es durch Entlassung von Mitarbeitern auf der fragilen Seite jemals einsparen könnte.



Dana Stoll ist IT-Management Professional und realisiert seit 20 Jahren Projekte im und um das Internet. Dabei bilden betriebliche Aspekte, insbesondere in Kombination mit agiler Entwicklung einen wesentlichen Schwerpunkt ihrer Arbeit.