ProjektMagazin
Das Fachportal für Projektmanagement

Spotlight

Wie führe ich agile Methoden ein?

Eine themenspezifische Zusammenstellung von Fachartikeln
aus dem Projekt Magazin

www.projektmagazin.de
Mehlbeerenstr. 82024 Taufkirchen
Tel: +49 89 2420798-0
Fax: +49 89 2420798-8

Wie führe ich agile Methoden ein?

Agile Software-Entwicklung zählt mittlerweile zu den etablierten Methoden. Auch Unternehmen, die bisher traditionelle Vorgehensweisen einsetzten, wollen von den Vorteilen profitieren, die agiles Projektmanagement verspricht. Allerdings ist dies nicht ohne weiteres möglich, da Agilität nur in einer dafür geeigneten Umgebung erfolgreich sein kann. In diesem Spotlight erfahren Sie, unter welchen Voraussetzungen und wie Sie agile Methoden wie Scrum oder Kanban sinnvoll einführen können, selbst in traditionellen Umgebungen. Erfahren Sie, wie Sie Tradition und Agilität verbinden können, sowohl in methodischer als auch in unternehmerischer Hinsicht.

Inhalt

Fachbeitrag

Agile Methoden im traditionellen Projektmanagement-Umfeld einsetzen

Projekte kämpfen heutzutage mit Innovations- und Marktdruck. Das Projektergebnis kann anfangs oft nur abstrakt beschrieben werden. Technische Rahmenbedingungen und Kundenwünsche verändern sich im Projektverlauf. Und die Engpass-Ressource sind gute Mitarbeiter mit hohen und seltenen Qualifikationen. Wo diese Faktoren zusammenkommen, werden seit einigen Jahren agile Vorgehensweisen angewandt, hauptsächlich Scrum und Extreme Programming (XP).

Diese Methoden kommen aus der Softwareentwicklung und werden bisher größtenteils auch genau dort eingesetzt. Jedoch ist der Trend erkennbar, dass agile Ansätze über die IT-Abteilungen auch ihren Weg zur Anwendung in anderen Unternehmensbereichen finden – und sogar in Branchen Einzug halten, die als besonders auf Sicherheit bedacht und starr bekannt sind, wie z.B. Banken, Versicherungen, Medizintechnik oder Rüstungsunternehmen. Gerade weil agile Ansätze immer weiter verbreitet werden, können sich auch hartgesottene Profis im traditionellen Projektmanagement diesen Ideen nicht mehr verschließen.

Aber was macht den agilen Ansatz aus Sicht des traditionellen Projektmanagements so reizvoll? Und wie lassen sich agile Vorgehensweisen auch in traditionellen Projektmanagement-Umgebungen einsetzen? Was bedeutet das für die Rolle des klassischen Projektmanagers? Diesen Fragen widmet sich der vorliegende Beitrag, der Projektverantwortlichen und Organisationsgestaltern Anregungen geben soll, wie sich agile Methoden auch in ein traditionelles PM-Umfeld integrieren lassen und beide Konzepte von dieser Kombination profitieren können.

Autor

Thomas Müller

Dipl.-Ing. (FH) Elektrotechnik / Nachrichtentechnik, PMP, IBM Certified Senior Projektmanager, Tätig bei IBM Rational als Spezialist und Berater

Kontakt: thomas.mueller@de.ibm.com

Mehr Informationen unter:
› projektmagazin.de/autoren

Autor

Benedict Gross

Diplom-Wirtschaftsjurist FH, PMP, zertifizierter Projektmanager GPM/IPMA, certified Project Management Consultant GPM

Kontakt: mail@b-gross.com

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in den Rubriken:

› Agiles Projektmanagement
› Projektkultur
› Unternehmenskultur

Kurze Übersicht: Was sind agile Vorgehensweisen?

Die heute als agil bezeichneten Vorgehensweisen (Scrum, Extreme Programming, die Chrystal-Familie und einige andere) haben alle eine eigene Entwicklungsgeschichte. Ihre Wurzeln liegen in den 90er Jahren. Im Februar 2001 kamen die führenden Denker dieser Methoden zusammen und verabschiedeten ein gemeinsames Credo, das sog. "Agile Manifest" (www.agilemanifesto.org). Seitdem werden diese Methoden unter dem Sammelbegriff "Agile Vorgehensweisen"

zusammengefasst und sind hauptsächlich für die Softwareentwicklung beschrieben. Dabei lässt das Agile Manifest in seiner Formulierung keinen Zweifel, dass es von Softwareentwicklern für Teams in der Softwareentwicklung verfasst wurde.

Eine Umdeutung als generellen Ansatz für das Projektmanagement erfahren die agilen Methoden erst in den letzten Jahren. Dieser schließen sich die Urheber der Vorgehensweisen sowie Berater und Buchautoren bereitwillig an. Am ergiebigsten aus Sicht des Projektmanagements ist wohl "Scrum", da es hauptsächlich einen einfachen Planungs- und Steuerungsprozess für das Management eines Entwicklungsprojekts beschreibt. Die anderen agilen Ansätze gehen hingegen sehr spezifisch auf Methoden der Softwareentwicklung ein. Aus diesem Grund scheint Scrum auch die weiteste Verbreitung zu erfahren.

Die agilen Vorgehensweisen heben sich vom klassischen Projektmanagement zunächst in der Betonung ihrer vier "agilen Werte" ab, die im Agilen Manifest als Glaubenssatz beschrieben sind:

"Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

- *Individuen und Interaktionen mehr als Prozesse und Werkzeuge*
- *Funktionierende Software mehr als umfassende Dokumentation*
- *Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlungen*
- *Reagieren auf Veränderung mehr als das Befolgen eines Plans*

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein." (Quelle: www.agilemanifesto.org)

In diesen Grundsätzen sowie den "Zwölf Prinzipien Agiler Softwareentwicklung" (siehe hierzu auch das Agile Manifest) wird den Menschen in einem Projekt eine besonders hohe Bedeutung zugemessen. Man beachte aber, dass das Agile Manifest nicht die traditionellen PM-Tugenden wie Prozesse, Werkzeuge oder Plantreue schlecht macht, wie es gelegentlich dargestellt wird. Im Gegenteil: Sie werden ausdrücklich als Werte angeführt. Gleichzeitig werden ihnen jedoch Werte gegenübergestellt, die verdeutlichen sollen, worin die traditionellen PM-Tugenden auch ihre Schwächen haben. Einen Widerspruch stellen die Begriffspaare nicht dar! Man kann funktionierende Software produzieren, die umfassend dokumentiert ist oder einen umfangreichen Vertrag verhandeln und trotzdem gut mit dem Kunden zusammenarbeiten. Aus Sicht des traditionellen Projektmanagements kann das Agile Manifest deswegen auch als Aufruf zur Ausgewogenheit verstanden werden.

Alle agilen Vorgehensweisen sind stark darauf ausgerichtet, den Mitarbeitern des Projektteams ein angenehmes produktives Arbeitsumfeld zu bereiten und die Kommunikation untereinander zu fördern. Wenn man diese Ideen vor dem Hintergrund von Softwareprojekten sieht – tatsächlich trifft das aber heute auf die meisten anderen Projektarten ebenfalls zu –, dann leisten qualifizierte Mitarbeiter den größten Beitrag zum Erfolg, sind gleichzeitig die Ressource, die am schwersten zu beschaffen ist, und das von ihnen eingesetzte Wissen ist oft zu abstrakt für eine schriftliche Dokumentation. Insofern bedeutet die Fokussierung auf Individuen und Interaktionen letztendlich

eine Form der Produktionsoptimierung, wie sie in der Industrie bezogen auf deren Produktionsmittel schon lange selbstverständlich ist.

Agil vs. Traditionell: gleicher Rahmen, verschiedene Ansätze

Ähnlich prominent wie das Agile Manifest – wenn auch nicht damit vergleichbar – ist im traditionellen Projektmanagement das "Dreieck des Projektmanagement" bestehend aus Zeit, Kosten und Leistung. Diese Restriktionen gelten natürlich auch für agil geführte Projekte, verdeutlichen aber einen grundlegenden Unterschied (Bild 1).

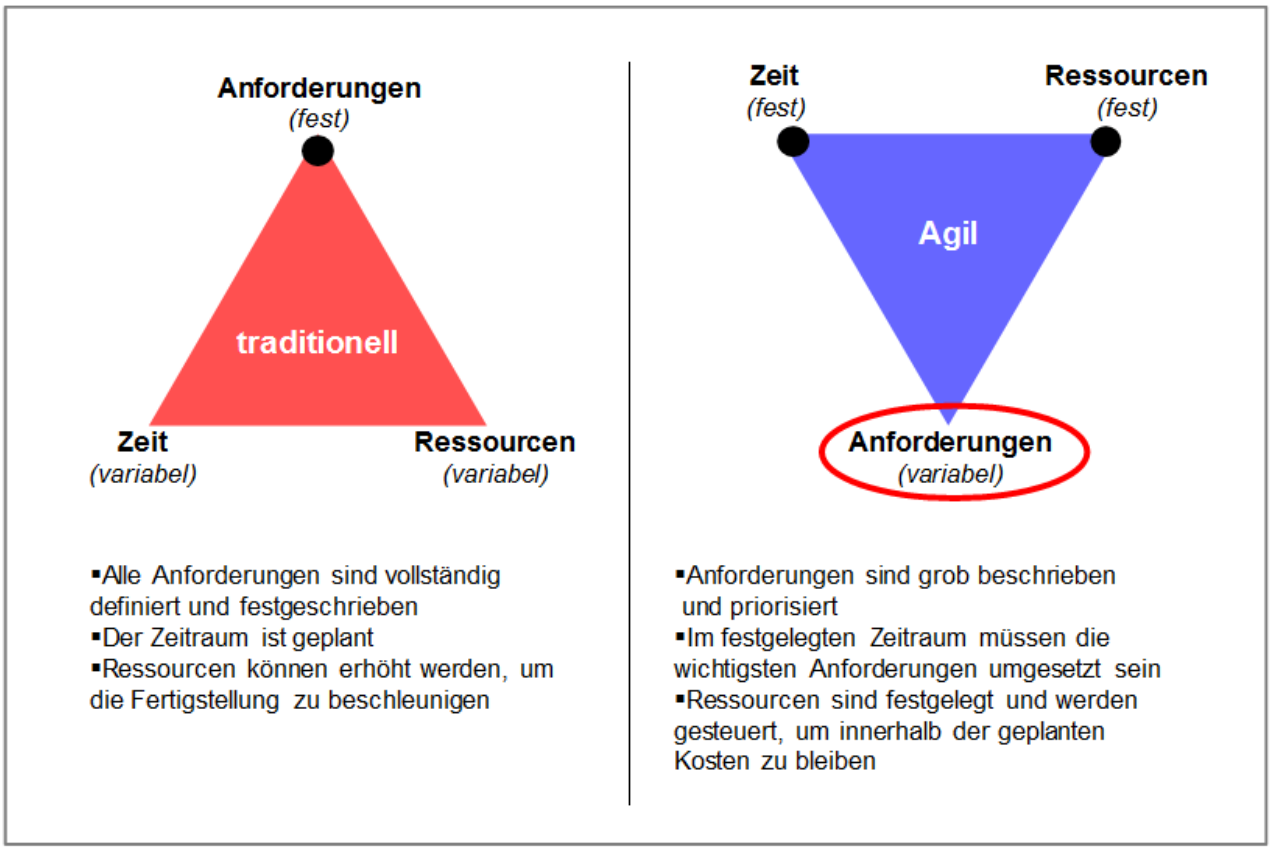

Bild 1: Das "Dreieck des Projektmanagements" für traditionelle und agile Verfahren.

Traditionelle Projekte tendieren dazu, den Leistungsgegenstand eingangs genau zu spezifizieren und dann als fix zu betrachten. Um jedoch die vereinbarten Leistungen voll zu erfüllen, ist es häufig notwendig und durchaus akzeptiert, die Zeit- und Kostenziele zu überziehen. Die agilen Vorgehensweisen basieren hingegen darauf zu akzeptieren, dass der Leistungsgegenstand anfangs unscharf ist und erst im Projektverlauf spezifiziert wird. Der Zeitrahmen und auch die verfügbaren Ressourcen werden dagegen als fix betrachtet. Agile Projekte bewegen sich also innerhalb derselben Restriktionen wie traditionell durchgeführte Projekte. Der Ansatz, mit dem den Herausforderungen eines Projekts begegnet wird, ist jedoch unterschiedlich.

So wird bei agilen Ansätzen ein "iterativ-inkrementelles Vorgehen" angewandt. Dabei findet die Produkterstellung in festen Zeitintervallen (Iterationen) von z.B. 30 Tagen statt, an deren Ende immer ein nutzbares Teilergebnis stehen muss. Es wird also nicht jahrelang auf den großen Moment der Fertigstellung hingearbeitet, sondern kontinuierlich in kleinen Schritten ein Produkt geschaffen, das früh lauffähig ist und dessen Funktionsumfang ständig erweitert wird.

Praxisbeispiel: Software-Entwicklung bei IBM

Der IT-Konzern IBM setzt seit einigen Jahren agile Vorgehensweisen in der Software-Entwicklung ein, wie z.B. Extreme Programming (XP), OpenUP, Lean Development und Scrum. Die Methoden kommen in internen Projekten zur Produktentwicklung und zur Erstellung eigener IT-Systeme genauso zum Einsatz wie in Projekten für externe Kunden. Häufig sind die Teams in diesen Projekten über Standorte in der ganzen Welt verteilt. Welche Vorgehensweise jeweils für ein Projekt gewählt wird, hängt vor allem von den bisherigen Erfahrungen der Teammitglieder mit den verschiedenen Prozessen ab – am häufigsten wird Scrum verwendet. Wichtige Grundhaltung bei der Auswahl ist es, die für den Leistungsgegenstand, Auftraggeber und das Projektteam passende Vorgehensweise zu finden und eben nicht jedemProjekt eine vorgegebene Methode überzustülpen. Insofern gilt bei der Zusammenstellung eines individuellen Managementvorgehens für ein Projekt der Leitgedanke "Pragmatismus vor methodischer Dogmatik".

Schnell zeigte sich, dass beim Einsatz agiler Methoden weniger Zeitverzüge bei den Releases auftraten und die Software bereits sehr früh eine verhältnismäßig hohe Stabilität aufwies. Allerdings ist in den ersten Releases noch nicht die gesamte geplante Funktionalität vorhanden. Dies erfordert eine geschickte Priorisierung der umzusetzenden Anforderungen, damit die Kernfunktionalität in jedem Fall verfügbar ist. So führte IBM neben den üblichen Beta-Programmen (gesamte Funktionalität vorhanden) zusätzlich "Early-Access-Programme" ein (wesentliche Funktionalität vorhanden). Dieser Softwarestand entspricht von Anfang an den grundsätzlichen Qualitätsanforderungen und ist in seinem Leistungsumfang bereits nutzbar. Die agile Vorgehensweise ermöglicht es somit, dass Interessierte auf Basis der früh funktionsfähigen Versionen weitere Anforderungen in den Entwicklungsprozess einbringen können.

Bestimmte Vorgaben gelten für alle Projekte

Neue Projekte werden im Kontext ihrer bestehenden traditionellen Umgebung und Organisation gestartet, d.h. meist nicht mit allen Freiheiten und "auf einer grünen Wiese". Es existiert also schon ein Rahmen aus Prozessen, Restriktionen und Notwendigkeiten, innerhalb dessen alle Projekte – auch die agilen – durchgeführt werden müssen. Dieser kann einerseits aus rechtlichen Rahmenbedingungen, branchenüblichen Standards oder Vorschriften bestehen, andererseits bewährte bestehende Systeme und Prozesse im Unternehmen umfassen. All dies schränkt den Spielraum ein und macht es schwer bis unmöglich, sich ad hoc von allen Standards und angestammten Vorgehensweisen einer Organisation zu trennen, was einen "revolutionären" Einschnitt bedeuten würde.

So wird der Wunsch nach einer beschleunigten Produktentwicklung es meist nicht rechtfertigen, alle Prinzipien und Arbeitsweisen auf den Kopf zu stellen. Diese Lektion haben die meisten Unternehmen schon durch Restrukturierungen und Prozessoptimierungen gelernt. Deswegen ist es bei der Einführung von agilen

Vorgehensweisen besonders wichtig, den Rahmen und die Abhängigkeiten des Unternehmens zu verstehen und mit Kenntnis der Möglichkeiten und Grenzen vorzugehen.

Ein Bereich, in dem agile Vorgehensweisen in traditionellen Unternehmen häufig besondere Schwierigkeiten haben und auf Widerstände stoßen, ist die Vertragsgestaltung. Kaum eine Rechtsabteilung wird erfreut zustimmen, Verträge über Projekte zukünftig klar terminlich und finanziell zu definieren, den genauen Leistungsgegenstand aber offen zu lassen. Es ist ein Credo des traditionellen Projektmanagements und der Unternehmensführung, dass der Liefergegenstand eines Projekts umfassend und abschließend vertraglich geregelt sein muss. Alles andere stellt ein inakzeptables Risiko dar.

Agilität wirkt sich auf die Planungssicherheit aus

Nicht nur in der Softwarebranche bestehen Abhängigkeiten zwischen den Projekten. Einfach gerechnet, steigt mit der Anzahl voneinander abhängiger Projekte eines Unternehmens auch die Anzahl der Ergebnisse, die vorausgeplant und termingerecht erstellt werden müssen. Für agil durchgeführte Projekte, in der sich die Mitarbeiter in der genauen Definition des Leistungsspektrums nur von Iteration zu Iteration festlegen wollen, schwindet in diesem Gerüst aus planerischen Fixpunkten der dringend benötigte Freiraum.

Nicht zuletzt liegt es im Interesse des Kunden, Planungssicherheit hinsichtlich seiner Investition zu erhalten. Aber was genau wird bei einer agilen Vorgehensweise die Leistung sein, die er erhält? Wie erklärt er seinem Controlling, dass er viel Geld ausgeben wird für ein neues System, das passen wird und auch funktioniert, aber nicht auf der Basis eines ausführlichen Pflichtenhefts mit eindeutig festgelegten Spezifikationen erstellt wird. Kriterien wie Kundenzufriedenheit und Passgenauigkeit einer Lösung – häufige Stärken agiler Vorgehensweisen – lassen sich nur schwer in einem Geschäftsbericht präsentieren und nur mittelbar in einer Bilanz abbilden.

Ob also Anforderungen an Projekte und deren Planungs- und Berichtswesen aus der unbeeinflussbaren rechtlichen Umwelt herrühren oder von Interessensgruppen innerhalb der Organisation – der Bedarf der Organisation an Daten und definierten Informationen aus Projekten wird nicht vor einem agilen Projekt Halt machen. Gerade die Befürworter agiler Methoden sehen es aber als Stärke an, auf Datenerfassung, die nicht der direkten Wertschöpfung am eigenen Projektgegenstand dient, zu verzichten und propagieren ein besonders schlankes Projektcontrolling möglichst mit Zettel und Stift. Ein Graus für traditionelle Projektcontroller! Und so werden in den meisten Unternehmen nicht von heute auf morgen alle Regeln außer Kraft gesetzt, um agiles Arbeiten zu ermöglichen.

Ein wesentlicher Erfolgsfaktor für den Einsatz agiler Methoden in einem Unternehmen ist jedoch die sinnvolle Integration agiler Spezifika in die vorgegebene Unternehmenswirklichkeit, in vorhandene Systeme und Prozesse. Dabei sollen die Vorteile und Stärken der Agilität genutzt werden, etablierte und sinnvolle Abläufe aber möglichst wenig gestört werden. Der bestehende und leistungsfähige Organismus eines Unternehmens muss weiterarbeiten können, während gleichzeitig Raum für Höchstleistungen in agilen Teams geschaffen wird. So schnell die Verbreitung agiler Projektmanagement-Methoden seit dem Agilen Manifest voranschreitet, so sehr ist die praktische Implementierung in großen Unternehmen noch Neuland.

Agile und traditionelle Projekte kombinieren

Bei der Integration agiler Methoden in einer traditionellen PM-Umgebung können Welten aufeinander prallen: Die Anhänger der klassischen Vorgehensweisen wollen klare Leistungsbeschreibungen und Termine, die der agilen wollen im Team von Iteration zu Iteration neu bestimmen, was wann getan wird.

Prinzipiell gibt es zwei Möglichkeiten, Agilität in einem Unternehmen zu integrieren ohne dieses in seinen Grundfesten zu erschüttern: Entweder man bildet klar abgetrennte Projektteams, die agil arbeiten; man schafft also dort Inseln, wo agile Ansätze erfolgversprechender als traditionelle sind. Oder man kannibalisiert agile Vorgehensweisen, zerlegt sie also in ihre Praktiken und Module und implementiert sie schließlich in Teilen in das traditionelle Projektmanagement.

Letztendlich stellt das Kannibalisieren eine Möglichkeit dar, die in agilen Ansätzen selbst schon enthalten ist: Extreme Programming z.B. beschreibt bewusst verschiedene Praktiken zur Softwareentwicklung, aus denen ein Anwender nach Belieben für sein Projekt wählen soll. Auch die Chrystal-Familie lädt dazu ein, selber zu definieren, für welche Projekte welche Methoden eingesetzt werden.

Integration agiler Methoden erfordert ein Umdenken

Möchte eine Organisation in ihr traditionelles Projektmanagement agile Methoden einfließen lassen, können deren Vorteile genutzt werden, ohne auf einen Schlag alle bestehenden Standards umzuwerfen. Allerdings kann man erst von agilem Projektmanagement sprechen, wenn agile Methoden auch durchgängig eingesetzt und die agilen Werte gelebt werden. Das bedeutet eine große Umstellung für das betroffene Unternehmen und hat ebenso große Auswirkungen auf die Unternehmensführung selbst. Denn die agilen Ansätze kommen mit einem Set aus Werten, die nur im Einklang mit den Werten der Geschäftsführung und der Unternehmenskultur funktionieren. Diese können wiederum Gegensätze aufwerfen: Vertrauen und freie Interaktion gegen Compliance (Regelkonformität) und Kontrolle, Selbstorganisation im Team gegen Prozessreife. So muss sich ein Unternehmen z.B. fragen, ob seine Kunden ein CMMI-Level oder agiles Vorgehen glücklicher macht. Auf einmal stehen neben der Einführung von neuen PM-Vorgehensweisen grundlegende Werte- und Kulturfragen im Raum.

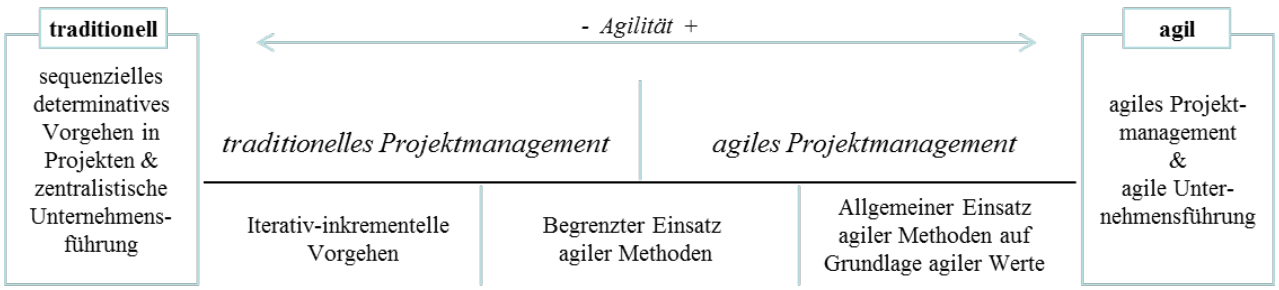

Bild 2: Auch in traditionellen Umgebungen können agile Methoden eingesetzt werden.

Je nach Sachlage bzw. Projekteigenarten, kann es sinnvoll sein, verschiedene Vorgehensweisen anzuwenden. Ob Agilität angewendet wird, hängt davon ab, ob es für das jeweilige (Teil-)Projekt Vorteile bringt und ob die

Mannschaft damit umgehen kann und will. Abhängig vom Leistungsgegenstand kann es etwa bei einem hoch innovativen Thema unumgänglich sein, mit regelmäßig iterativ neu bewerteten Zielen zu arbeiten. In einem stark reglementierten Umfeld dagegen gibt es vielleicht keine Alternative zum Wasserfall-Modell. Daher muss es die Organisation ermöglichen, dass klassisch und agil durchgeführte Projekte im selben Unternehmen und sogar im selben Gesamtprojekt neben- und miteinander koexistieren können.

Agile Projekte benötigen neue Systeme

Ein Unternehmen betreibt in der Regel ein System, in dem alle Daten konsolidiert werden, die für die Unternehmenssteuerung notwendig sind (Business-Management-System). Der Einsatz eines solchen Systems kann sogar aufgrund gesetzlicher Bestimmungen zum Konsolidieren und Berichten relevanter Geschäftsdaten verpflichtend sein. Hier werden u.a. von Projekten laufend Daten zu Aufwand (Arbeitsstunden, externe Kosten, Material) und entsprechende Prognosen erfasst. Alle Projekte müssen die dafür benötigten Informationen liefern – auch die agilen Projekte. Einer ihrer Erfolgsfaktoren ist jedoch die Einfachheit und der Verzicht auf ein aufwändiges Controlling sowie das Arbeiten mit analogen Mitteln. Anforderungsspezifikation und Fortschrittsmessung sollen auf Moderationskarten, Flip-Chart oder am White-Board geführt werden – gänzlich analog also und keinesfalls kompatibel zu digitalen Datenverarbeitungssystemen.

Zwar entwickeln mittlerweile immer mehr Hersteller Softwarelösungen für agile Teams, die Anbietervielfalt sowie die Reife der Software ist aber noch nicht so hoch wie bei Software für das traditionelle Projektmanagement. Es ist zu wünschen, dass der Einzug von PM-Software für agiles Projektmanagement auch eine Verjüngung und Verbesserung der etablierten PM-Softwaresysteme mit sich bringt, deren Staub und Muff der letzten zwanzig Jahre und unverständliche Bedienbarkeit sie zur unnützen Folter für Projektleiter mit häufig nichtssagendem Ergebnis macht.

Traditionell und agil innerhalb eines Projekts

Wie kann ein Unternehmen die Vorteile sowohl der klassischen als auch agilen Vorgehensweisen gleichzeitig nutzen? Das gelingt mit einem geschickten Kapseln agiler Projekte und klugen Schnittstellen zum traditionellen System.

Für große Vorhaben, die nicht vollständig im agilen Kontext umgesetzt werden sollen oder können, kann eine klassisch-agile Projektstruktur die Lösung sein. Die agilen Projekte sind dabei als Sub-Projekte des Gesamtprojekts oder -programms organisiert. Agil wird die eigentliche Entwicklung in den Expertenteams umgesetzt, während das klassisch geführte Hauptprojekt sämtliche Dinge wie übergeordnete Zeitplanung, Kommunikation zum Management, Ressourcenmanagement, Infrastruktur- und Lieferantenmanagement sowie alles, was mit den Finanzen zu tun hat, abwickelt (Bild 3).

Schnell wird klar, dass der Projektmanager mit seiner klassischen PMI- oder IPMA-Ausbildung keineswegs obsolet ist im Kontext agiler Projekte. Ganz im Gegenteil: Die Schnittstelle zwischen Teilprojekten und die Kommunikation zum Management sowie administrativen Einheiten eines Konzerns bildet ein klassischer Projektmanager, während der Leiter des agilen Teams, z.B. ein Scrum Master, das Entwicklerteam und die Kundenanforderungen im Fokus hat. Klassischer Projektmanager und agiler Teamleiter ergänzen sich so in ihren Aufgaben und haben keinen Widerspruch in ihrer Rolle.

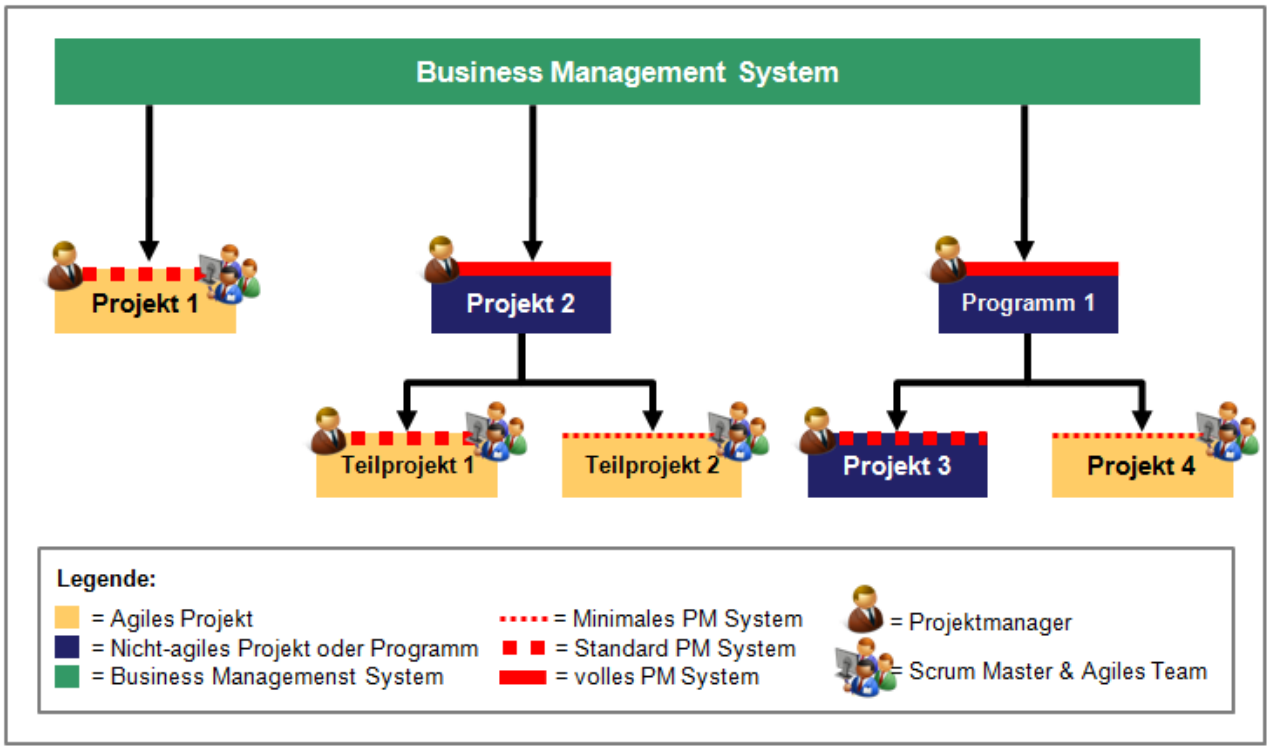

Bild 3: Koexistenz verschiedener Projektvorgehensweisen.

Ein Beispiel: Im Rahmen großer Migrationsprojekte kann und sollte weiterhin die Umstellung unzähliger Schnittstellen nacheinander mit präziser Planung der Zeitachse und abschließend definierten Fachkonzepten durchgeführt werden. Bei der parallelen Weiterentwicklung und Anpassung eines Front-Ends aber können die inhaltlichen Prioritäten, Produkt- und Bedieneigenschaften flexibel und regelmäßig nach den Wünschen des Kunden angepasst werden, während das Projekt schon läuft und das Produkt in der Entstehung ist. Der klassische Projektmanager verantwortet hier das Gesamtprojekt, der Leiter des agilen Teams ausschließlich das Teilprojekt der Front-End-Entwicklung. Andere Teilprojekte werden wiederum mit traditionellen Prozessen und Methoden durchgeführt werden (Bild 4).

Die Aufgaben des traditionellen Projektleiters unterscheiden sich abhängig von der Art des nachgeordneten Projektteams: Für traditionelle Teams ist er der erste Ansprechpartner für alle Belange des Kunden und verhandelt Anforderungen und Änderungen. Bei agilen Projektteams hingegen unterstützt ihn dabei ein in agilen Methoden ausgebildeter Teamleiter, z.B. ein Scrum Master. Die Kundenkommunikation über Anforderungen und Änderungen findet direkt zwischen Entwicklern und Kunden statt. Diese Variante fördert das eigenverantwortliche Handeln und ist sehr effizient: Die Kommunikation erfolgt auf der niedrigsten Ebene und direkt an der eigentlichen Leistungserstellung.

Der Projektmanager als "Torwächter"

Das Modell, an der Schnittstelle Projekt / Unternehmen einen klassischen Projektmanager als "Torwächter" und "Dolmetscher" einzusetzen, bietet einige Chancen, um den Aufwand für das Controlling zu reduzieren, Transparenz und Ehrlichkeit in das Berichtswesen zu bringen und das agile Projektteam vor Fremdeinflüssen zu beschützen. Ein traditioneller Projektleiter kann so den Daten- und Berichtsbedarf abwickeln, aber auch klassische Managementaufgaben, wie z.B. Finanzwesen und langfristige Projektressourcenplanung übernehmen, die nicht zum Kanon der agilen Vorgehensweisen gehören.

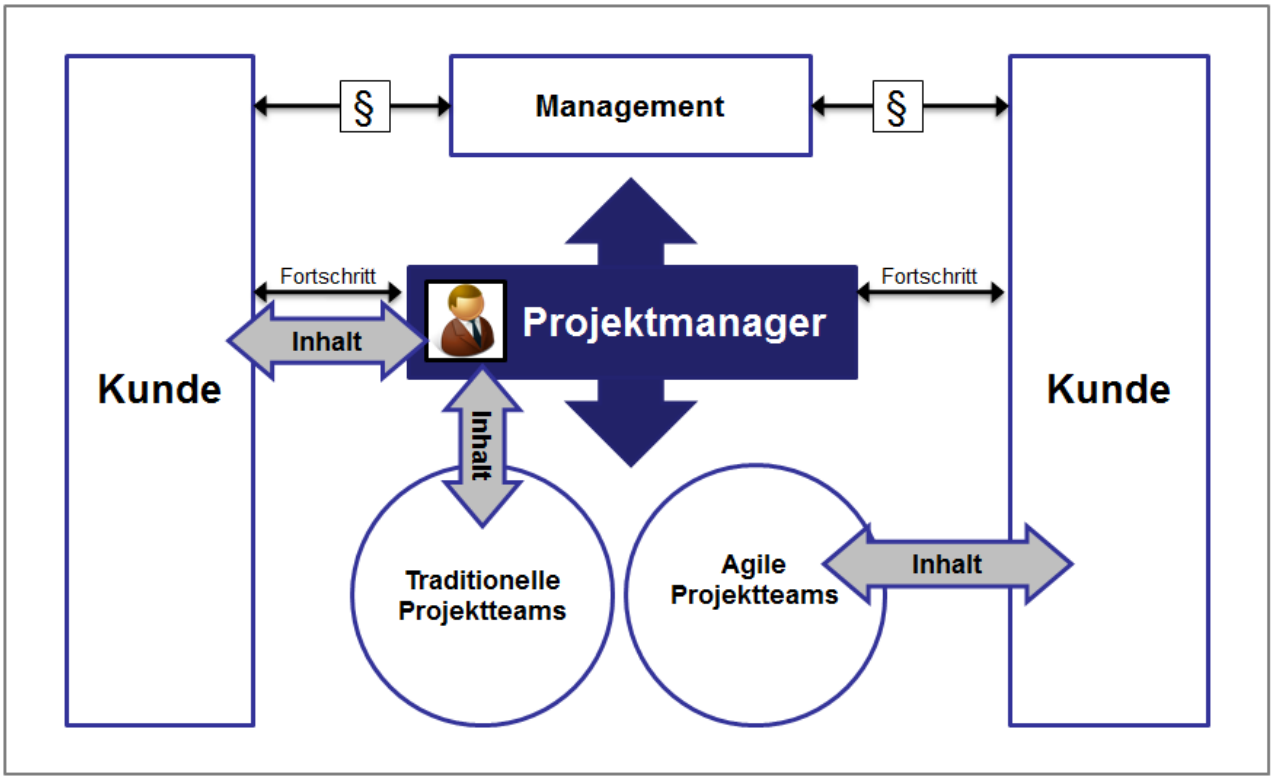

Bild 4: Der traditionelle Projektmanager ist nach wie vor die zentrale Instanz im Projektumfeld.

Besonders für das Controlling ist eine solche Schnittstelle sinnvoll. Denn häufig wird versucht, die Komplexität in Projekten durch eine immer feinere Detaillierung der Planung und Kontrolle zu beherrschen. Die Erhöhung der Detailtiefe bei der Steuerung und Überwachung komplexer Projekte führt schnell zum "Informations-Overkill", das Scheitern bzw. Auseinanderdriften von Plan, erhobenen Daten und der Projektrealität ist meistens vorprogrammiert. Paradoxerweise lässt sich oft die "Auflösungsgenauigkeit" des Controllings reduzieren und dabei trotzdem ein qualitativ besserer Überblick erlangen. Genau in dieser Hinsicht kann ein erfahrener traditioneller Projektmanager agile Teams unterstützen und solche Aufgaben von den Teams fernhalten.

Denn nur der traditionelle Projektmanager ist hierfür auch ausgebildet. Es darf nicht vergessen werden, dass agile Vorgehensweisen für das Management von (Software-)Entwicklungsteams entstanden sind. Einen umfassenden methodischen und inhaltlichen Ansatz mit Managementkompetenzen bzw. Knowledge-Areas, die über ein

einzelnes Projekt hinausgehen und auch den Kontext des Unternehmens und seines Geschäftes enthalten, bieten nur die großen und traditionellen PM-Institutionen (z.B. IPMA/GPM und PMI). Schon alleine deswegen sind die agilen Qualifikationsprogramme und Personenzertifizierungen in Inhalt und Umfang nicht vergleichbar mit dem traditionellen PMP oder den Vier-Level-Programmen der IPMA/GPM.

Es wird deswegen sicherlich spannend sein zu beobachten, wie einzelne Teile der agilen Ansätze in die nächsten Versionen der Competence-Baselines der traditionellen Institutionen aufgenommen werden und so Stück für Stück Teil der Standards im Projektmanagement werden.

Zusammenfassung

Agile und traditionelle Vorgehensweisen bilden keinen Widerspruch. Im Gegenteil, agile Ansätze können genau dort Erfolg bringen, wo die klassischen Methoden bisher schwach waren: bei der Entwicklung von Lösungen in einem dynamischen Umfeld, bei unklaren Kundenwünschen oder wenn die Projektteams aus hochqualifizierten Fachkräften bestehen. Durch ihre radikale Effizienz und Einfachheit halten sie dem traditionellen Projektmanagement hier einen Spiegel vor.

Um die Vorteile beider Welten auch in großen Unternehmen zu nutzen, kann eine agil-klassisch gemischte Projektstruktur etabliert werden, die für jedes Teilprojekt unterschiedliche Vorgehensweisen zulässt. Dafür ist es notwendig, ein sinnvolles Maß dafür zu finden, welche Daten für die Controlling-Systeme bereitzustellen sind. Durch die Schaffung "agiler Inseln" im Unternehmen lassen sich agile Vorgehensweisen leicht implementieren, ohne eine "Revolution" in der Organisation auszulösen.

Die Leiter agiler Teams konzentrieren sich auf den Entwicklungsprozess in den Teams und stellen so Qualität und Erfolg am Leistungsgegenstand der Projekte sicher. Das agile Vorgehen konzentriert sich damit bei der Wertschöpfung auf das, wofür es ursprünglich auch gedacht ist: das effiziente Führen von Entwicklerteams. Der klassische Projektmanager behält hingegen immer die Aufgabe des Gesamtverantwortlichen und der zentralen Projektführungskraft – er muss allerdings auch agile Methoden verstehen.

Fachbeitrag

Nur ein Business Case überzeugt das Topmanagement

Agile Methoden einführen – ist das rentabel?

Mit der Einführung agiler Methoden für die Software-Entwicklung versprechen sich Unternehmen oder IT-Abteilungen meist folgende Nutzeffekte: Die Fehlerzahl bei der Entwicklung soll reduziert, Entwicklungszeiten sollen verkürzt und Kundenanforderungen besser erfüllt werden. Diese Ziele sind zweifelsohne erstrebenswert und leuchten unmittelbar allen Beteiligten ein, allerdings vergessen die Verantwortlichen dabei nur allzu oft, zwei wesentliche Fragestellungen zu beantworten, bevor sie den Startschuss für die Einführung agiler Methodik geben:

- Welcher betriebswirtschaftliche Mehrwert wird tatsächlich durch die Einführung Agiler Software-Entwicklung bzw. Agilen Projektmanagements erzeugt?
- Mit welchen Kosten ist diese Umstellung insgesamt verbunden?

Oft wird übersehen, dass die Einführung agiler Vorgehensweisen erhebliche Kosten verursacht. Sollen die Software-Entwickler anstatt im Wasserfallmodell künftig agil arbeiten, so sind hierfür intensive Trainings für die einzelnen Personen, eine Neuorganisation der bisherigen internen Geschäftsprozesse, erhebliche Veränderungen der Kommunikation mit den Kunden sowie Investitionen in Software und Hardware erforderlich. Zudem enthalten die notwendigen Organisationsentwicklungsprozesse insbesondere während der Anlaufzeit ernsthafte Risikofaktoren für die Realisierung der angestrebten Nutzeffekte.

Nur wer Kosten, Nutzen und Risiken der Einführung von Agilem Projektmanagement zuvor genau ermittelt hat, kann dieses anspruchsvolle Veränderungsprojekt zielgerichtet steuern und zum Erfolg führen. Die dazu notwendigen Informationen liefert ein sogenannter Business Case (Schmidt, Projekt Magazin 4/2010).

In diesem Artikel beschreiben wir, was ein Business Case speziell für die Einführung Agiler Software-Entwicklung berücksichtigen sollte. Den Erfahrungshintergrund dafür bildet ein Praxisbeispiel, bei dem wir einen Business Case für das Project Management Office (PMO) eines Technologieunternehmens erstellten. In diesem Unter-nehmen arbeiten rund 500 Entwickler an ca. 75 aktiven Software-

Autor

Johannes Ritter

Betriebswirt, Business-Case-Experte, Partner der Unternehmensberatung Solution Matrix

Kontakt:
projektmagazin@solutionmatrix.de

Christine Marburger

Dipl.-Theologin, Senior Consultant bei Solution Matrix

Kontakt: pm@solutionmatrix.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in den Rubriken:

› Idee / Antrag / Akquisition

Service-Links

› Projektportfolio- / Programm-Management

› Projektportfoliomanagement

› Multiprojektmanagement / Projektportfolio

Entwicklungsprojekten. Die Aufgabe war, einen validen Business Case für die Einführung von Agilem Projektmanagement zu erstellen.

Agiles Projektmanagement einführen – eine echte Herausforderung

Agiles Projektmanagement in einer Organisationseinheit mit mehr als 500 betroffenen Personen einzuführen, ist ein Veränderungsprojekt, das realistisch betrachtet ungefähr zwei Jahre dauert. Oftmals haben die Initiatoren jedoch zu Beginn keine ausreichende Klarheit darüber, was mit den agilen Methoden genau verbessert bzw. welche Fehler behoben werden sollen. Dadurch entsteht bei den Beteiligten keine ausreichende Verbindlichkeit. Sie betrachten die Einführung nicht als dringend und das Projekt bleibt als halbherziger Versuch stecken. Schlimmstenfalls kehrt die Organisation wieder zum gewohnten Vorgehen zurück, da mit der unvollständig umgesetzten agilen Vorgehensweise negative Erfahrungen gesammelt werden. Um die erforderlichen Veränderungen dauerhaft umzusetzen, muss das Topmanagement die Einführung des Agilen Projektmanagements mit vollem Engagement unterstützen.

Nur ein Business Case kann das Topmanagement von den Vorteilen des Agilen Projektmanagements nachhaltig überzeugen und dadurch die notwendige Unterstützung schaffen. Das verlässliche Wissen z.B. um Kosteneinsparungen und Steigerung der Mitarbeiter- und Kundenzufriedenheit schafft die entsprechende Motivation bei den Entscheidern.

Deshalb müssen zuerst die Business-Ziele vollständig klar sein, die mit der Einführung von Agilem Projektmanagement erreicht werden sollen, bevor ein solches Projekt in Angriff genommen wird. Dies gilt nicht nur für die Organisationsveränderung, sondern insbesondere auch für die finanziellen Ziele, anhand derer sich der Erfolg des Projekts bewerten lässt. Der erste Schritt hierzu besteht darin, eine möglichst genaue Antwort zu finden auf die Frage:

Welche Ziele wollen wir mit der Einführung von Agilem Projektmanagement erreichen?

In unserem Beispielprojekt kam der erste Impuls für die Einführung agiler Methoden vom Leiter der Entwicklungsabteilung. Sein Anliegen war es, die Anzahl der Softwarefehler (Bugs) über den gesamten Entwicklungsverlauf hinweg zu minimieren, da die Freigaben neuer Versionen (Releases) immer länger dauerten. Von diesem Vorschlag konnte er auch den Leiter des PMO überzeugen.

Mit dieser Zielsetzung hatte das Unternehmen schon eine vergleichsweise konkrete Motivation. In anderen Fällen steht lediglich der allgemeine Wunsch "Wir wollen agil arbeiten!" im Mittelpunkt. Wenn der Reiz der neuen Methode die Hauptmotivation ist, dann fällt es den Beteiligten erfahrungsgemäß schwer, die angestrebte Verbesserung unternehmens- und projektbezogen genau zu spezifizieren.

Ein Business Case soll daher betriebswirtschaftliche Kennzahlen liefern, die Kosten, Nutzen und Risiken des Projekts abbilden. Dies ist auch für qualitative Zielwerte möglich. Z.B. kann die Steigerung der Mitarbeiterzufriedenheit durch die Reduktion der Mitarbeiterfluktuation quantifiziert werden.

Für einen Business Case muss die Frage daher lauten:

Wie kann ich den geschäftlichen Nutzen dieses Vorhabens konkret messen?

Die Zielformulierung: "Die Zahl der Bugs im Entwicklungsprozess soll pro Projekt um 100 reduziert werden", enthält zwar eine wichtige quantitative Aussage, ist im Sinne eines Business Cases aber immer noch zu wenig konkret. Dies ist sie erst dann, wenn sie mit einem Eurozeichen versehen werden kann, so dass sie den Entscheidern auch ohne Wissen über Agile Software-Entwicklung einleuchtet. Für einen Business Case, der diese Aufgabe erfüllt, muss man aus der Zielsetzung des Projekts quantifizierbare Kosten- und Nutzen-Vorstellungen herleiten.

In unserem Beispielprojekt entwickelte ein interdisziplinäres Projektteam die für einen Business Case tauglichen Ziele: "Produktentwicklungskosten senken" und "Umsatz durch Verkürzung der Time-to-Market steigern". Zeit- und Kostenaufwand für die Fehlerbehebung im Entwicklungsprozess beeinflussen diese beiden Ziele erheblich.

Der Business Case dient zwar der Quantifizierung des Nutzens, aber ein interdisziplinäres Team, das in diesem Fall u.a. aus Mitarbeitern der Abteilungen Entwicklung, Qualitätssicherung und Roll-Out bestand, verliert leicht aus den Augen, dass dies tatsächlich monetären Nutzen meint. Die Aussage, dass mit Agilem Projektmanagement 15 Kalendertage Produktentwicklungszeit eingespart werden können, berücksichtigt eine betriebswirtschaftliche Größe und ist deshalb als Nutzenbeschreibung bereits besser geeignet als die Reduktion der Bug-Anzahl. Für die Entscheider ist dies aber immer noch zu wenig greifbar. Stattdessen muss es einen eindeutigen Bezug zur Umsatzsteigerung geben, durch den deutlich wird, ob Kosten und Nutzen dieser Zeitersparnis einen Wechsel auf Agiles Projektmanagement rechtfertigen oder nicht.

Einen Business Case erstellen – Machbarkeitsstudie als Projekt

Für eine strategische Entscheidung mit weit reichenden Konsequenzen, wie es die Einführung Agilen Projektmanagements darstellt, ist die Erstellung eines Business Cases mit einem Vorprojekt gleichzusetzen, bei dem die wirtschaftliche Machbarkeit bzw. Rentabilität des Vorhabens überprüft wird. Für diese Machbarkeitsstudie in Projektform ist ein Lenkungsausschuss auf Entscheider-Ebene und ein Projektteam erforderlich, das den Business Case nach den Vorgaben des Lenkungsausschusses erstellt. Verfügen die Mitarbeiter über die entsprechenden Kenntnisse (vgl. Anhang) und ausreichende Erfahrung, dann können sie den Business Case auch unternehmensintern ohne externe Berater erstellen. Bei der hier vorgestellten Methode ist es zumindest bei den ersten drei Business Cases dringend anzuraten, dass externe Berater die Erstellung begleiten, die Erfahrung mit möglichen Fallstricken und das notwendige Know-how haben. Im Beispielprojekt setzte sich das Team aus Mitarbeitern der Abteilungen Entwicklung, Qualitätssicherung, Roll-Out, PMO und Finanzen/Controlling sowie zwei externen Beratern zusammen.

Die Begleitung durch Berater betrifft alle drei unten beschriebenen Schritte der Erstellung des Business Cases. In unserer Beratungserfahrung hat sich gezeigt, dass bereits die Projektdefinition mit Hilfe der Einfluss-Map ohne Unterstützung von außen entweder zu detailliert oder zu allgemein erfolgt. Dadurch wird entweder der Prozess zu zeitaufwendig und aufgrund der Komplexität auch zu fehleranfällig oder das Finanzmodell wird nicht aussagekräftig genug. Auch ist anfangs nicht leicht zu unterscheiden, welche Zusammenhänge ein konkretes Szenario abbilden kann und bei welchen Fragestellungen ein neues Szenario zu erstellen und zu bewerten ist. Bei der Datenerhebung ergeben sich für das Projektteam viele Fragen, wenn z.B. Experten sich nicht imstande sehen, die geforderten Schätzwerte zu liefern oder ihre Daten nicht so detailliert oder anders zusammengesetzt sind, wie es für das Finanzmodell erforderlich ist. Z.B kann es sein, dass es Experten aus dem Controlling zu ungenau erscheint,

einen durchschnittlichen Stundensatz für die Entwickler mit einer gewissen Schwankung beim minimalen und maximalen Wert zu benennen, was jedoch für den Business Case völlig ausreichend ist. Erfahrene Berater, die diesen Prozess bereits mehrfach durchlaufen haben, können hier meist einfache Möglichkeiten für die Abschätzungen aufzeigen und so die Experten und das Projektteam effizient bei der Datenerhebung unterstützen.

Die Rolle des Lenkungsausschusses

Der Lenkungsausschuss ist als fördernde Leitinstanz nötig, die einerseits für eine abteilungsübergreifende Projektdefinition sorgt und andererseits die nötige Durchsetzungsfähigkeit bei der Datenerhebung in Experteninterviews mitbringt. Je nach Anzahl der betroffenen Abteilungen kann er aus bis zu vier Entscheidern bestehen. Im Idealfall sind dies der Geschäftsführer, der Finanzvorstand und der IT-Leiter oder deren Vertreter, die ebenfalls Budgetbefugnis haben. In unserem Beispiel war der Leiter des PMO als Sponsor des Projekts ebenfalls mit im Lenkungsausschuss, denn bei der Einführung agiler Entwicklungsmethoden sollte mindestens ein Vertreter des Lenkungsausschusses ein fachliches Interesse an agilen Entwicklungsmethoden haben und gewillt sein, die erforderlichen Veränderungen zu unterstützen und durchzusetzen. Im Rahmen der Business-Case-Erstellung hat der Lenkungsausschuss folgende vier Aufgaben:

Definition von Umfang und Ziel des Projekts

Der Lenkungsausschuss bestimmt, welche Projektdefinition das Projektteam zugrunde legen und welchen Umfang der Business Case haben soll. Dabei legt er den Betrachtungszeitraum und die Zahl der Szenarien fest. Der Lenkungsausschuss gibt den Zielwert für das Projekt vor (z.B. Cashflow) und bestimmt, welche Finanzkennzahlen der Business Case liefern soll. Um die endgültige Projektdefinition mit den zu behandelnden Unsicherheiten und zu berücksichtigenden Abhängigkeiten freigeben zu können, benötigt der Lenkungsausschuss die Unterstützung durch das interdisziplinäre Projektteam, das hierzu eine "Einfluss-Map" (s.u.) anfertigt. Die Einfluss-Map, auf die sich das Team geeinigt hat, muss vom Lenkungsausschuss abgenommen sein, bevor die nächsten Arbeitsschritte erfolgen.

Qualitätskriterien für den Business Case

Die klaren Vorgaben des Lenkungsausschusses an das Projektteam gewährleisten, dass der Business Case nicht nur die formalen Anforderungen erfüllt, sondern den Entscheidern tatsächlich eine verlässliche Entscheidungsgrundlage liefert. Ein verlässlicher Business Case ist immer ergebnisoffen, er kann sowohl zum Ergebnis führen, dass sich das Projekt lohnt als auch, dass es sich nicht lohnt. Außerdem zeichnet sich ein solider Business Case durch Nachvollziehbarkeit und eine statistische Validierung der Endergebnisse aus. Es hat sich als sinnvoll erwiesen, den Lenkungsausschuss in Grundzügen mit der Business-Case-Methodik vertraut zu machen, damit dieser die Herkunft der Daten und die Berechnung der Ergebnisse nachvollziehen kann. Eine solche Transparenz schafft Vertrauen in die Ergebnisse.

Gewährleistung der Ressourcenverfügbarkeit

Um einen Business Case zu erstellen, sind Schätzwerte für bestimmte Einflussfaktoren erforderlich. Anhand der Einfluss-Map werden unternehmenseigene Experten benannt, die von Mitgliedern des Projektteams in Interviews von etwa 45 bis 60 Minuten zu diesen Schätzungen befragt werden. Mitarbeiter mit den entsprechenden Kompe-

tenzen haben diese Zeit jedoch nicht einfach in ihren Terminkalendern frei, zumal die Datenerhebung konzentriert und innerhalb eines kurzen Zeitraums erfolgen soll. Der Lenkungsausschuss unterstützt das Projektteam mit seiner Autorität dabei, die Zusagen für solche Interviews zu bekommen.

Abnahme des Business Cases und Entscheidung über weiteres Vorgehen

Zum Abschluss präsentiert das Projektteam den Business Case dem Lenkungsausschuss, so dass dieser anhand der dort gezeigten Zahlen und der darauf fußenden Handlungsempfehlung seine Entscheidung für oder gegen die Einführung agiler Software-Entwicklung fällen kann.

In drei Schritten zum Business Case

Der Weg zum Business Case lässt sich in drei Arbeitsschritte gliedern:

1. Das Projektteam definiert abteilungsübergreifend das Projekt mithilfe einer Einfluss-Map.
2. Die in der Map dargestellten Einflussfaktoren übersetzt ein Mitarbeiter des Teams in die Struktur des Finanzmodells. Anschließend führt das Projektteam Interviews mit Experten aus dem eigenen Unternehmen, um das Finanzmodell mit Daten (Intervallschätzungen) zu füllen.
3. Im letzten Schritt validiert ein Projektteammitglied die quantifizierten Endergebnisse durch eine Risiko- und Sensitivitätsanalyse.

Die Einfluss-Map

Um zu einer gemeinsamen Projektdefinition zu gelangen und diese zu visualisieren, erstellt das Team zunächst eine "Einfluss-Map", d.h. eine Art Ursache-Wirkungsdiagramm (vgl. Bild 1). Die Einfluss-Map erfasst alle wesentlichen Parameter, die den Zielwert bzw. die Zielgrößen beeinflussen. Pfeile visualisieren die Ursache-Wirkungs-Beziehungen zwischen diesen Parametern. Die Zielgrößen befinden sich ganz rechts im Diagramm, sie werden mit Sechsecken symbolisiert. Die Einflussparameter sind in Ovalen dargestellt, von ihnen führen Pfeile entweder direkt zu den Zielgrößen oder zu anderen Einflussparametern. Dabei gilt als strenge Regel, dass keine Zirkelschlüsse entstehen dürfen, da ansonsten die Erstellung des Finanzmodells nicht mehr möglich ist. Eine ausführliche Darstellung der Einfluss-Map finden Sie in Ritter und Röttgers: "Kalkulieren Sie noch oder profitieren Sie schon?" (Ritter, 2009).

Üblicherweise sammelt das Team in einem offenen Brainstorming während eines halbtägigen Workshops zunächst mögliche Einflussparameter, um einen ersten Entwurf der Einfluss-Map zu erstellen. In unserem Beispiel wurde die endgültige Einfluss-Map mit allen interdisziplinären Diskussionen iterativ über zehn Kalendertage hinweg erstellt. Bild 1 zeigt die so erhaltene Einfluss-Map in vereinfachter Form.

Eine Einfluss-Map gilt immer gleichzeitig für alle zu bewertenden Szenarien, in diesem Fall "Ist-Zustand vor der Einführung" sowie "Erwarteter Zustand nach der Einführung agiler Software-Entwicklung". Die Struktur des Finanzmodells ist für beide Szenarien identisch, was für den Vergleich eine wesentliche Voraussetzung ist. In beiden Szenarien ist z.B. "Kosten für Hardware" eine Unsicherheit, die Höhe der Kosten wird aber für jedes Szenario unterschiedlich sein. Es gibt auch Szenarien, in denen sich nicht nur die Datenwerte, sondern auch die Anzahl relevanter Unsicherheiten unter-

scheiden. In diesem Fall zeigt die Einfluss-Map das Szenario mit den meisten Parametern an. Wenn dann bei einem Szenario bestimmte Zellen des Finanzmodells nicht benötigt werden, bleiben diese ganz einfach leer.

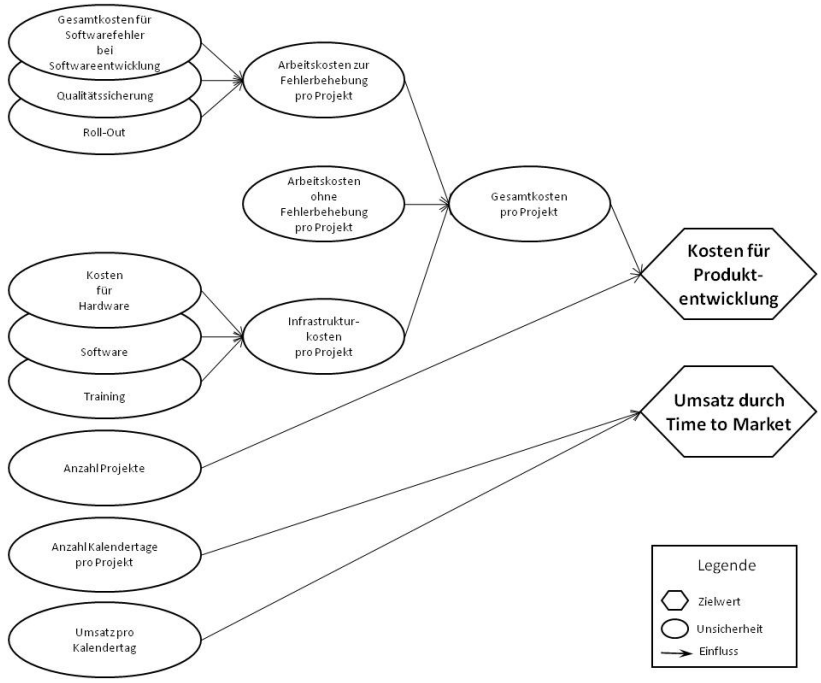

Bild 1: Einfluss-Map aus dem Beispielprojekt zur Einführung agiler Software-Entwicklung.

Die Einfluss-Map zeigt, welche Parameter für den Zielwert – in Bild 1 sind "Kosten für die Produktentwicklung" und "Umsatz durch Time to Market" die Zielwerte – verantwortlich sind. Für die ermittelten Ausgangsparameter (in Bild 1 ganz links) erhebt das Projektteam dann in Experteninterviews die erforderlichen Daten. Diese sind meist mit großer Unsicherheit belegt, was die Bewertung eines Projekts gerade so schwierig macht. Die Daten müssen deshalb in einer geeigneten Form erhoben werden (s.u.), so dass sie später statistisch validiert werden können und damit die notwendige Verlässlichkeit bieten. Die Daten auf der linken Seite der Einfluss-Map dienen als Berechnungsgrundlage für die weiteren, abhängigen Parameter, auf die jeweils mindestens ein Pfeil zeigt. Im Beispielprojekt war das ursprüngliche Ziel, Bugs durch die Einführung agiler Software-Entwicklungsmethoden zu reduzieren. Das Projektteam schlüsselte dazu die Arbeitskosten zur Fehlerbehebung entsprechend den Entwicklungsstadien "Software-Entwicklung", "Qualitätssicherung" und "Roll-Out" auf. Die getrennte Betrachtung nach den Entwicklungsstadien war notwendig, um später sinnvoll priorisieren zu können, in welchem der drei Bereiche agile Entwicklungsmethoden als erstes eingeführt werden sollten. Wie die späteren Daten zeigten, fielen die größten Kosten im Bereich “Software-Entwicklung“ an. Dort wurde später auch mit der Einführung der agilen Methodik begonnen.

Eine weitere Untergliederung z.B. in "Kosten für Fehlerlokalisierung" oder "Kosten für Fehlerbehebung", wie zunächst in der Phase des Brainstormings angedacht, hätte die spätere Datenerhebung unnötig erschwert. Das Team fasste diese Kosten deshalb im Parameter "Gesamtkosten für Softwarefehler bei Software-Entwicklung" zusammen.

Beim Erstellen der Einfluss-Map muss das Team abschätzen, welcher Detaillierungsgrad für den Business Case erwünscht ist und welcher daraus folgende Arbeitsaufwand für die Datenerhebung und Modellierung realistisch ist. Hier muss das für das Finanzmodell zuständige Projektteammitglied den jeweiligen Einfluss auf die Komplexität des Finanzmodells bewerten. Erfahrene Berater können ebenfalls meist schnell einschätzen, welcher Detaillierungsgrad sinnvoll ist.

Auch die Infrastrukturkosten beeinflussen den Zielwert "Produktentwicklungskosten", da bei der agilen Software-Entwicklung die Testläufe bereits während der Entwicklung durchgeführt werden. Die Beschaffung neuer Hard- und Software gehört deshalb mit zum Projektumfang. In unserem Beispiel ging es um die Entwicklung von Programmen für Endanwender, die auf allen aktuellen Betriebssystemen (Windows XP bis Windows 7, MAC OS X sowie Linux) und mit allen in Verwendung befindlichen Webbrowsern lauffähig sein mussten. Da zudem ungefähr 75 Projekte parallel liefen, musste eine entsprechend hohe Anzahl von Rechnern vorhanden sein, um die von der agilen Vorgehensweise geforderten Testläufe durchführen zu können. Neben der Hardware fielen auch Lizenzkosten für Software an. Diese betrafen zum einen die Ausstattung der Testcomputer pro Projekt. Zum anderen waren Software-Systeme zur Testautomatisierung und zur Versionskontrolle notwendig. Auch wenn dies ein Beispiel mit vergleichsweise hohem Testaufwand darstellt, dürfen die Software- und Hardwareinvestitionen auch in weniger testintensiven Umgebungen nicht vernachlässigt werden.

Tabelle 1 zeigt am Beispiel eines Projekts mit 13 Mitarbeitern die geschätzte Dauer und die geschätzten Kosten für Hardware, Software und Training bei agiler Software-Entwicklung.

Unsicherheit	Minimaler Wert	Wahrscheinlichster Wert	Maximaler Wert
Software	8.500 €	9.400 €	10.200 €
Hardware	2.000 €	2.400 €	2.700 €
Training und Coaching	1.200 €	1.500 €	1.900 €
Projektdauer in Kalendertagen	113	114	115

Tabelle 1: Infrastrukturkosten im Beispiel bei der Einführung agiler Software-Entwicklung.

Produkteinführungszeit reduzieren

Der zweite Zielwert, den die Einfluss-Map erfasst, ist die Umsatzsteigerung durch eine verkürzte Time to Market. Um den Nutzen agiler Methoden zu ermitteln, muss die Produkteinführungszeit in Kalendertagen pro Projekt sowohl im Ist-Zustand als auch nach der Einführung Agiler Software-Entwicklung ermittelt bzw. geschätzt werden. Für den Ist-Zustand konnte im Beispiel die Dauer eines repräsentativen Projekts aufgrund vorliegender Daten mit 129 Kalendertagen direkt ermittelt werden. Schwieriger hingegen war die Abschätzung der Zeitersparnis. Dies einzuschätzen war Aufgabe eines erfahrenen Programmierers im Unternehmen, der bereits mit Agiler Entwicklung gearbeitet hatte und daher sowohl die nötige fachliche Erfahrung als auch die Kenntnis der unternehmensspezifischen Herausforderungen hatte. Wenn dieses Know-how nicht im Unternehmen zu finden ist, muss man externe Experten für Agile Software-Entwicklung zu Rate ziehen.

Für die abschließende Quantifizierung im Rahmen eines aussagekräftigen Business Cases ist jedoch der Bezug zum Umsatz ausschlaggebend. Als Umsatz pro Produkt wurden 10.000 € am Tag ermittelt, so dass bei dem wahrscheinlichsten Wert für die Verkürzung der Time to Market von 15 Kalendertagen mit 150.000 € Umsatzsteigerung pro Projekt, bei 75 Projekten mit 11,25 Mio. € Umsatzsteigerung bezogen auf ein halbes Jahr zu rechnen ist. Bei einem Jahresumsatz des Unternehmens von 300 Mio. € bedeutet das eine Umsatzsteigerung von 7,5%.

Das Finanzmodell

Nachdem die Einfluss-Map vom Lenkungsausschuss bestätigt ist, dient sie als Vorlage für das Finanzmodell, das die Parameter und die Einflussbeziehungen von der Einfluss-Map übernimmt.

Das Finanzmodell ist mindestens ein Excel-Arbeitsblatt mit entsprechender Struktur, die den Ist-Zustand sowie den Zustand nach Einführung agiler Softwareentwicklung für einen Zeitraum von mindestens drei Jahren abbildet und damit vergleichen kann. Grob gesagt, setzt sich das Finanzmodell aus den einzelnen Parametern auf der Einfluss-Map ("y-Achse") und den Spalten zu den einzelnen Projektjahren ("x-Achse") jeweils für alle Szenarien zusammen. Letztere sind gegliedert in minimalen, wahrscheinlichsten und maximalen Wert. Die Struktur ergibt sich aus der Anordnung der Parameter.

Eine einfache Skizze zeigt diese Struktur des Finanzmodells (Bild 2). Die grün markierten Felder zeigen an, dass dieser Parameter bereits mithilfe der Simulationssoftware als Annahme definiert wurde und später in der Risiko- und Sensitivitätsanalyse das gesamte Intervall zwischen minimalem und maximalem Wert berücksichtigt wird.

A1 — Einführung von Agile Development

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Einführung von Agile Development		Ist-Zustand 2012				Ist-Zustand 2013				Agile Development 2012		
2		min. Wert	wahrschein-lichster Wert	max. Wert		min. Wert	wahrschein-lichster Wert	max. Wert		min. Wert	wahrschein-lichster Wert	max. Wert	
20	....								...				...
21	Kosten für Hardware	4.250 €	4.880 €	5.350 €		4.670 €	4.880 €	5.350 €		8.500 €	9.400 €	10.200 €	
22	Kosten für Software	1.200 €	1.400 €	1.620 €		1.200 €	1.400 €	1.620 €		2.000 €	2.400 €	2.700 €	
23	Kosten für Training	- €	- €	- €		- €	- €	- €		1.200 €	1.500 €	1.900 €	
24													
25	Infrastrukturkosten pro Projekt		6.280 €				6.280 €				13.300 €		
26													
27	...												
28													

Bild 2: Ausschnitt der Struktur des Finanzmodells in Microsoft Excel.

Die in den Befragungen ermittelten Rohdaten (s.u.) werden in der Excel-Tabelle als Ausgangsparameter eingegeben. Alle anderen Parameter werden daraus durch Formeln (s.u.) errechnet. Auf einem Tabellenblatt werden alle Berechnungen durchgeführt wie z.B. die Summe der Projektkosten pro betrachtetem Jahr. Aus diesen Ergebnissen lassen sich die gewünschten Kennzahlen wie z.B. der Kapitalwert (Net Present Value / NPV) zu zwei unterschiedlichen Zinssätzen, nämlich 15% und 30% berechnen. Auf einem weiteren Tabellenblatt wird die Differenz der Szenarien gebildet und somit die tatsächliche Einsparung durch die Einführung von Agiler Software-Entwicklung quantifiziert. Die Grafiken der Risiko- und Sensitivitätsanalyse werden in den folgenden Tabellenblättern eingebettet.

Das Finanzmodell erstellen und bearbeiten

Mitglieder des Projektteams, die ein solides Verständnis von der Systematik der Einfluss-Map besitzen, erstellen das Finanzmodell. Um dieses korrekt zu strukturieren und um die Berechnungen der Finanzkennzahlen und die Risiko- und Sensitivitätsanalyse durchführen zu können, sind eine Reihe von Qualifikationen notwendig.

Zur Erstellung des Finanzmodells sind betriebswirtschaftliche Kenntnisse in Bilanzierung und Rechnungswesen, statistische Kenntnisse sowie ein versierter Umgang mit Microsoft Excel (Specialist und Expert) und Simulationssoftware (s.u.) gefragt. Für die Berechnung der Finanzkennzahlen wie Kapitalbarwert, ROI oder Deckungsbeitrag ist Wissen in Finanzmathematik erforderlich. Die Risiko- und Sensitivitätsanalyse wird zwar von der Simulationssoftware erstellt, aber die korrekte Wahl der Verteilungsfunktion und die Interpretation der Ergebnisse setzt statistisches Grundwissen voraus.

Kenntnis der Struktur des Finanzmodells, Eingabe der Daten, Berechnung der Finanzkennzahlen und Vorbereitung des Finanzmodells für die Risiko- und Sensitivitätsanalyse sollten in der Verantwortung einer oder zweier Personen liegen, da ansonsten das Fehlerpotenzial steigt. Das Finanzmodell muss eine klare Systematik haben, die eine Person einmal vollständig durchdacht haben muss. Diese hat daher den besten Überblick und garantiert z.B. eine korrekte Erhebung der Daten, indem sie auf die genaue Form der Daten gemäß der Struktur des Finanzmodells besteht. Bei komplexen Projekten können u.U. dennoch Änderungen in Formelbezügen notwendig werden, um neue Datenformate der Struktur anzupassen.

Bereits bei der Übergabe an eine zweite Person können Anpassungsnotwendigkeiten und Eingaben übersehen werden. Mehr als zwei Personen benötigen auf jeden Fall ohne klare Absprachen. Es lässt sich festhalten: je mehr Schnittstellen es gibt, umso größer ist das Fehlerpotenzial.

Erhebung der Daten

Um zu den für den Business Case unbedingt erforderlichen Daten zu kommen, empfiehlt sich die im folgenden beschriebene Vorgehensweise.

Bei der Präsentation der Einfluss-Map vor dem Lenkungsausschuss sollten für alle Ausgangsparameter gleichzeitig diejenigen unternehmensinternen Experten namentlich benannt werden, welche die für den Business Case benötigten Daten am besten liefern können. Der Lenkungsausschuss sollte diese Auswahl explizit genehmigen und dadurch die Wichtigkeit dieser Aufgabe deutlich machen, damit die Expertenbefragungen möglichst zügig durchgeführt werden – sinnvoll ist ein Zeitraum von ca. zwei Wochen.

In unserem Beispielprojekt waren die vom Lenkungsausschuss benannten Experten ein Mitarbeiter aus dem Controlling zur Abschätzung der Infrastrukturkosten, ein Mitarbeiter der Personalabteilung, ein Projektleiter und der Leiter Qualitätssicherung zu den Arbeitskosten. Der für das Finanzmodell verantwortliche Mitarbeiter des Projektteams führte die Interviews durch, wobei ihn immer ein weiteres Teammitglied begleitete, um ihn bei den Fragestellungen zu unterstützen und mit zu protokollieren. Ein Experteninterview beansprucht etwa 30-45 Minuten.

Bei der Befragung müssen die Interviewer einerseits darauf achten, dass die Daten genau in der Form erhoben werden, die für das Finanzmodell notwendig ist und dass andererseits die Daten verlässlich sind. Fast nie liegen die Daten in der geforderten Form vor, d.h. die Experten müssen von der ihnen vertrauten Datengrundlage abstrahieren. Wenn z.B. die "Gesamtkosten für Softwarefehler bei der Qualitätssicherung" gesucht sind, müssen die Interviewer als erstes die Frage im Sinne des Business Cases präzisieren. Für dieses Beispiel würde die korrekte Frage lauten: "Wie viele Personentage braucht man bei der derzeitigen Entwicklungsweise für das Finden und Beheben der Softwarefehler im Rahmen der Qualitätssicherung und zwar jeweils in den Jahren n, n+1 und n+2?

Bitte nennen sie einen minimalen, einen wahrscheinlichsten und einen maximalen Wert, den sie mit 80%iger Sicherheit vertreten können." Genau so wird die Frage natürlich nicht gestellt, aber die Interviewer müssen den Experten deutlich machen, welche Daten sie genau von ihnen brauchen und warum sie auf diese Weise erfragt werden.

Vermutlich wird der Controller sagen, "Gesamtkosten für Softwarefehler bei der Qualitätssicherung" sei zu speziell: "Wir haben nur Daten zu den gesamten Personentagen pro Projekt." Hier ist es die Aufgabe der Interviewer, den Experten von der bisherigen eigenen Systematik in die neue fremde Systematik zu helfen. Durch geschicktes Nachfragen, durch Plausibilitätsüberlegungen oder durch die Erklärung der Anforderungen des Finanzmodells kann man die Befragten bei ihren Prognosen und Abschätzungen unterstützen. Im Beispiel kamen wir über die Frage: "Welchen Prozentsatz würden Sie der Qualitätssicherung ungefähr zurechnen?" zum Ziel. Es hilft daher auch, das Finanzmodell für alle Beteiligten vor Augen zu haben, um schnell und genau zum Ziel zu kommen.

Beim Beispiel "Gesamtkosten Softwarefehler bei der Qualitätssicherung" wird z.B. der Qualitätssicherung ein bestimmter Prozentsatz an den gesamten Personentagen pro Projekt zugeordnet. Es geht um eine neue Zusammenschau bestehender Daten und nicht darum, bisher unbekannte Daten zu erfinden. Um diese Daten auch im Nachhinein als belastbar bezeichnen zu können, empfiehlt es sich, im Finanzmodell Kommentare zum Experten, zum Vorgehen und zu möglichen Quellen zu notieren. Zugrundeliegende Annahmen wie ein Satz von 250 € pro Personentag und 15% Arbeitsaufwand für Qualitätssicherung vom gesamten Projektaufwand sollten dokumentiert sein, um eine nachvollziehbare Herleitung der eingetragenen Zahlen zu gewährleisten.

Die Unsicherheit der Daten ist eine unvermeidbare Eigenschaft eines Business Cases, der immer in die Zukunft prognostiziert. Es geht darum, diese Unsicherheit methodisch so gut wie möglich aufzufangen und nicht darum, das Ergebnis punktgenau vorauszusagen. Dieses Vorgehen erscheint gerade für diejenigen, die mit exakten Wissenschaften zu tun haben, fragwürdig und sorgt bei der Datenerhebung für Unmut. Die Befragten können die Datenerhebung für zu grob und erzwungen halten, wenn die Interviewer auf dem neuen Datenformat bestehen, das der Business Case erfordert. Meist ist es eine Frage der richtigen Übersetzung und Abstraktion, die Daten in der gewünschten Form zu erheben.

Dass die Daten gerade durch die Intervallschätzung qualitativ hochwertig sind, liegt an der Risiko- und Sensitivitätsanalyse, die den Business Case abschließt. Ohne die statistische Auswertung der Intervalle würde der Business Case tatsächlich keine verlässlichen Ergebnisse liefern. Es würde lediglich auf Grundlage des wahrscheinlichsten Wertes gerechnet, der aber als nur scheinbar exakter Wert mit Sicherheit nicht eintreffen wird. Dies entspräche der Erhebung von Punktschätzungen anstelle von Intervallschätzungen, wodurch keine ausreichende Verlässlichkeit erreicht wird, da sie das Risiko nicht ausreichend abbilden. Ein einfaches Beispiel verdeutlicht die Gefahr von Punktschätzungen: Auch wenn die Durchschnittstiefe eines Gewässers nur einen Meter beträgt, kann es dennoch Stellen geben, an denen Nichtschwimmer ertrinken können. Für eine Beurteilung möglicher Gefahren reicht eine Punktschätzung eben nicht aus.

Ein solider Business Case zeichnet sich dadurch aus, dass das Finanzmodell der festgelegten Projektdefinition streng folgt, dadurch zwei Szenarien vergleichbar macht und die Endergebnisse statistisch validiert. Diese Struktur gibt die Datenformate vor. Würde hingegen die aktuell vorliegende Form der Daten das Finanzmodell bestimmen, wäre die Vergleichbarkeit der Szenarien nicht nachvollziehbar zu gewährleisten.

Die Form der Daten spiegelt die Bedingung wider, dass Projektbewertungen immer von einem Maß an Unsicherheit gekennzeichnet sind. Deswegen werden keine Punktschätzungen erhoben, die mit Sicherheit falsch sind, sondern Intervallschätzungen, welche die Experten mit 80%iger Konfidenz vertreten können.

Auf die obige Beispielfrage könnte für das Szenario "Agile Software-Entwicklung" die Antwort z.B. lauten: "Im Jahr n+1 fallen für Fehlersuche und -behebung bei der Qualitätssicherung pro Projekt mit 80%iger Wahrscheinlichkeit minimal 0,75, am wahrscheinlichsten 1,75 und maximal 3,3 Personentage an."

Diese Aussagen zu den Kosten werden dann im jeweiligen Jahr bei minimalem, wahrscheinlichstem und maximalem Wert eingegeben. Für alle Parameter werden für jedes Jahr und jedes Szenario auf diese Weise Daten erhoben (Bild 3). Dabei können sich Vereinfachungen ergeben, wenn z.B. nicht absolute Werte über die Jahre genannt werden, sondern Relationen wie z.B. eine Jahreswachstumsrate der Projektkosten hinzugefügt wird.

	Ist-Zustand									Agile Development								
	2012			2013			2014			2012			2013			2014		
	min	w	max	min	w	max	min	w	max	min	w	max	min	w	max	min	w	max
PT für Fehlerbehebung QS	1	2	4	1,2	2,4	4,8	1,3	2,6	5,3	1,1	2,2	4,4	0,75	1,75	3,3	0,75	1,75	3,3

Bild 3 : Ausschnitt aus der Struktur des Finanzmodells, Werte für die Unsicherheit: "Personentage für Fehlersuche und –behebung in Qualitätssicherung für ein Projekt". Dabei bedeuten: min: minimaler Wert, w: wahrscheinlichster Wert, max: maximaler Wert.

Eingabe und Bearbeitung der Daten

Nachdem alle Daten in die Excel-Tabellen eingegeben worden sind, berechnet das Finanzmodell in Excel aufgrund der eingegebenen Struktur die Endergebnisse der "Produktionsentwicklungskosten" und des "Umsatz durch Time to Market". Auf Basis dieser Zahlen werden nun auch die gewünschten Finanzkennzahlen wie Return on Investment (ROI), Kapitalwert (NPV), Amortisationsdauer oder interner Zinsfuß (IRR) berechnet. Zur genaueren Berechnung des diskontierten Cashflows, ROI, NPV, internem Zinsfuß und Amortisationsdauer finden Sie weitere Informationen in: "So schreiben Sie einen Business Case, Teil 3: Betriebswirtschaftliche Auswirkungen" (Schmidt, Ritter, Projekt Magazin 6/2010).

Als wahrscheinlichste Werte für die Kosten aller Entwicklungsprojekte ergaben sich im Beispielprojekt für das Wasserfallmodell im ersten Jahr 37,28 Mio. €, beim Szenario "Agile Software-Entwicklung" 32,94 Mio. €. Die gesamte Kosteneinsparung über drei Jahre bezogen auf alle Entwicklungsprojekte hatte einen wahrscheinlichsten Wert von 13 Mio. €. Dabei ist die mögliche Umsatzsteigerung durch die frühere Produkteinführung nicht berücksichtigt. Als Grundlage für die Entscheidung konzentrierte sich der Lenkungsausschuss auf diesen Zielwert.

Risiko- und Sensitivitätsanalyse

Um eine solide Entscheidungsgrundlage zu haben, genügt es nicht, wenn das Finanzmodell lediglich die Zielwerte aus den wahrscheinlichsten Werten der Eingangsparameter ermittelt. Dies wäre nur ein grober Richtwert mit völlig unbekannter Zuverlässigkeit. Mit welcher Wahrscheinlichkeit werden die 13 Mio. € Einsparung erreicht? Die

errechneten Zielwerte benötigen eine statistische Validierung, d.h. eine Aussage über die Wahrscheinlichkeit ihres Eintretens. Diese Anforderung bestimmt bereits den Prozess der Datenerhebung: Damit man diese Aussage treffen kann, hat man alle Datensätze in Form von Intervallschätzungen erhoben, die später mithilfe einer Simulationssoftware mit Wahrscheinlichkeitsverteilungen beschrieben werden können. Weitere Angaben zur Durchführung einer Risiko- und Sensitivitätsanalyse finden Sie in: "So schreiben Sie einen Business Case, Teil 4: Sensitivität, Risiko, Empfehlungen" (Schmidt, Ritter, Projekt Magazin 7/2010)

Der Aufwand für die Durchführung der Sensitivitätsanalyse mit Hilfe einer dafür geeigneten Software ist abhängig von der Größe und Komplexität des Finanzmodells. Simulationssoftware auf dem Markt sind Add-Ons für Microsoft Excel wie Oracle Crystal Ball, Palisade @Risk und AnalyCorp Insight. Jeder für die Sensitivitätsanalyse benötigte Datensatz muss als Annahme definiert werden (Bild 4), d.h. der Anwender wählt für ihn die geeignete Wahrscheinlichkeitsverteilung und gibt ihm für die spätere Auswertung einen eindeutigen Namen. Analog definiert der Anwender die Prognosen.

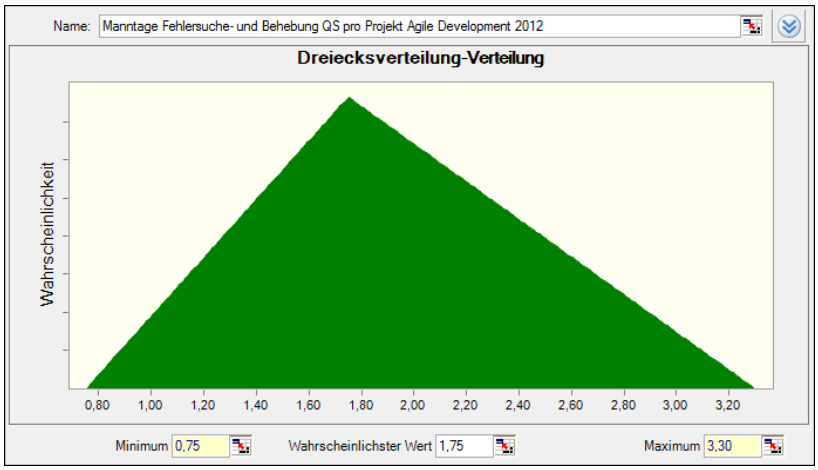

Bild 4: Die Definition einer Annahme mithilfe der Simulationssoftware.

Danach ist es nur der Klick auf den "Simulationsknopf" und man erhält eine Wahrscheinlichkeitsdichtefunktion als Ergebnis der Risikoanalyse. Bei der Simulation fließen alle Werte innerhalb des Intervalls in das Endergebnis ein, für das 100.000 mögliche Kombinationen durchgerechnet werden. Für den in Bild 3 dargestellten Parameter wählt die Monte-Carlo-Simulation in einem der 100.000 Versuche einen Wert zwischen 0,75 und 3,3 aus, der dann mit ebenfalls zufällig gewählten Werten der anderen Parameter anhand des Finanzmodells ein mögliches Ergebnis bildet. Auf diese Weise entsteht die Wahrscheinlichkeitsdichtefunktion, die im Gegensatz zu einem punktuellen Endergebnis dessen Bandbreite angibt und so eine Aussage über die Wahrscheinlichkeit des Eintreffens eines Werts erlaubt. Die in Bild 5 dargestellte Simulation des Beispielprojekts zeigt, dass das Endergebnis zwischen 12,14 und 13,86 Mio. € schwanken kann. Mit 80%iger Wahrscheinlichkeit wird jedoch ein Wert zwischen 12,61 und 13,40 Mio. € erreicht.

Mit der Risikoanalyse und einer ergänzenden Quantifizierung der wichtigsten Risikofaktoren in der Sensitivitätsanalyse, steht die Business-Case-Analyse auf so sicheren Füßen, wie es eine zukünftige Abschätzung eben sein kann. Der Lenkungsausschuss hat damit verlässliche Daten für die Abschätzung des Nutzens, dem die Einführung Agiler Software-Entwicklung im Unternehmen bringt. Auch das Projektteam hat damit verlässliche Daten für die Priorisierung der Umsetzung und ein effektives Risikomanagement. Wenn das Wissen zur Business-Case-Erstellung erfolgreich im Unternehmen verankert ist, können die Daten nach gegebener Zeit mit vergleichsweise wenig Aufwand überprüft und aktualisiert werden, um noch genauer steuern zu können.

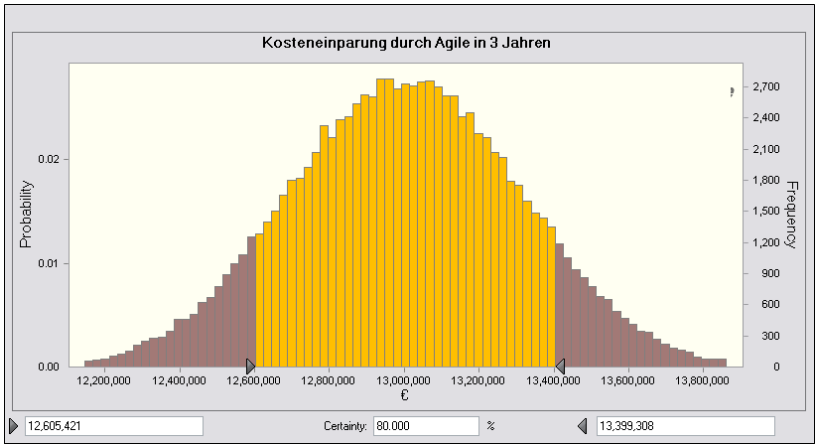

Bild 5: Simulation des Endergebnisses zu Einsparung der Kosten für Produktentwicklung durch die Agile-Einführung.

Fazit: Der Business Case holt das Topmanagement ins Boot

Die Einführung Agilen Projektmanagements für die Software-Entwicklung stellt eine grundsätzliche Umstellung dar, die einen tiefen Einschnitt in die Organisation bedeutet. Das Ausmaß an Veränderungen wird deshalb unweigerlich Ablehnung hervorrufen, die mit guten Argumenten, Werben um die Sache und Durchsetzungswillen des Topmanagements auf ein möglichst geringes Maß reduziert werden kann. Ohne das Engagement des Topmanagements wird Agiles Projektmanagement lediglich halbherzig in kleinen Pilotenprojekten betrieben und sie wird höchstens angenehme, aber im Verhältnis für den Aufwand doch zu kleine Ergebnisse bringen.

Ein Business Case in der hier vorgestellten Form erfüllt alle Voraussetzungen, um die Nutzeffekte einer Einführung Agilen Projektmanagements zu quantifizieren und damit von Anfang an effizient zu unterstützen. Da er sowohl die Befürwortung des Gesamtprojekts durch das Topmanagement und die klarere Durchsetzung der Vision von agiler Methodik im Unternehmen gewährleistet, ist dadurch zumindest eine notwendige Bedingung für ihren Erfolg erfüllt.

Literatur

- Röttgers, Frank u. Ritter, Johannes: Kalkulieren Sie noch oder profitieren Sie schon?, Frankfurt 2009
- Schmidt, Marty J. u. Ritter, Johannes: So schreiben Sie einen Business Case. Teil 1: Formalien und Einstieg, Projekt Magazin 4/2010
- Schmidt, Marty J. u. Ritter, Johannes: So schreiben Sie einen Business Case. Teil 2: Annahmen und Methoden, Projekt Magazin 5/2010

- Schmidt, Marty J. u. Ritter, Johannes: So schreiben Sie einen Business Case. Teil 3: Betriebswirtschaftliche Auswirkungen, Projekt Magazin 6/2010
- Schmidt, Marty J. u. Ritter, Johannes: So schreiben Sie einen Business Case. Teil 4: Sensitivität, Risiko, Empfehlungen, Projekt Magazin 07/2010
- Röttgers, Frank u. Ritter, Johannes: Getting Your Budget Approved, Frankfurt 2011
- Schmidt, Marty J.: Business Case Essentials. A Guide to Structure and Content, 3rd ed., Boston 2009
- Schmidt, Marty J.: The Business Case Guide, 2nd edition, Boston 2002
- Mun, Jonathan: Applied Risk Analysis. Moving Beyond Uncertainty in Business, Hoboken 2004
- Goodwin, Paul u. Wright, George: Decision Analysis for Management Judgement, 3rd ed., Chichester 2004

Anhang: Anforderungen an Projektteam und Projektleiter

Bei der Einführung von agiler Software-Entwicklung steht die Änderung des Arbeitsstils und der Organisation im Vordergrund. Deshalb müssen die Auswirkungen abteilungsübergreifender Prozessveränderungen analysiert werden. Hierfür sind entsprechend umfangreiche Kenntnisse beim Projektteam erforderlich, um eine verlässliche Abschätzung der Kosten abliefern zu können. Das Projektteam muss daher interdisziplinär zusammengesetzt sein und alle betroffenen Abteilungen mit einbeziehen.

Die Mitglieder des Projektteams sollten folgende Anforderungen erfüllen:

- Erfahrung mit interdisziplinärem Arbeiten oder zumindest die Offenheit dafür, sich auf andere Sichtweisen einzulassen.
- Fähigkeit, Verbindungen zwischen technischen und betriebswirtschaftlichen Kriterien zu sehen.
- Ausgeprägte analytische Fähigkeiten für ein gleichmäßiges Abstraktionsniveau, z.B. bei der Auswahl der Unsicherheiten und ihrer genauen Bezeichnung in der Einfluss-Map.
- Der Projektleiter muss darüber hinaus befähigt sein, abteilungsübergreifende Gruppen mit ihren heterogenen Interessen und Sichtweisen ergebnisorientiert zu moderieren. Da interdisziplinäre Teams oft eine hohe Dynamik aufweisen, muss der Projektleiter über eine Autorität verfügen, die von allen Beteiligten anerkannt wird.

Einsatz externer Berater / notwendige Fachkenntnisse für Business Case

Neben den oben angeführten, allgemeinen Anforderungen an die Mitglieder des Projektteams werden noch die in Tabelle 2 aufgeführten fachlichen Qualifikationen benötigt.

Aufgabe	Erforderliche Kompetenz mindestens eines Team-Mitglieds
Finanzmodell erstellen	Microsoft Excel-Spezialist oder Expert-Zertifizierung betriebswirtschaftliche Kenntnisse in Bilanzierung und Rechnungswesen Finanzmathematik zur Berechnung der Finanzkennzahlen (z.B. Kapitalbarwert)
Daten erheben	Gezielte Gesprächsführung. Bei Widerständen und Schwierigkeiten geeignete Lösungswege aufzeigen und zugleich auf eindeutige Ergebnisse bestehen können. Fähigkeit, ergebnisoffene, nicht beeinflussende Fragen zu formulieren.
Risiko- und Sensitivitätsanalyse durchführen	Die Risiko- und Sensitivitätsanalyse wird zwar von der Simulationssoftware erstellt, aber die korrekte Wahl der Verteilungsfunktion und die Interpretation der Ergebnisse setzt statistisches Grundwissen voraus.
Business Cases ohne Unterstützung durch Berater unternehmensintern erstellen	Erfolgreiche Durchführung von mindestens zwei sehr unterschiedlichen Projekten mit allen drei Arbeitsschritten in Begleitung durch Berater, um Detaillierungsgrad und korrekte Form der Ergebnisse einschätzen und Arbeitsaufwand für Finanzmodell und Datenerhebung abschätzen zu können.

Tabelle 2: Erforderliche Kompetenzen, um einen Business Case zu erstellen.

Fachbeitrag

Hybrides Vorgehensmodell

Agile und klassische Methoden im Projekt passend kombinieren

Agile Entwicklungs- und Projektmanagement-Methoden sind immer stärker im Kommen. Viele Unternehmen, die bereits klassische Entwicklungsmodelle wie z.B. den Rational Unified Process (RUP) oder PRINCE2 nutzen, denken mittlerweile auch über den Einsatz agiler Methoden nach. Die Gründe dafür sind vielfältig: Einerseits wünschen die Unternehmen schnellere und flexiblere Steuerungsmechanismen. Andererseits schreckt viele Projektbeteiligte der Aufwand für die zu erstellende Dokumentation ab sowie die aufwändige und kleinteilige Planung aller benötigten Aufgaben und Rollen.

Üblicherweise werden Projekte ausschließlich agil oder klassisch aufgesetzt und durchgeführt. Selbst in einer Organisation, die schon beide Philosophien unterstützt, besteht in der Praxis meist nur die Möglichkeit, das Projekt entweder nach agilen oder klassischen Prinzipien durchzuführen. Es gibt aber noch einen weiteren Ansatz, mit diesen "unterschiedlichen Welten" umzugehen: Und zwar die beiden Philosophien innerhalb eines Projekts zu kombinieren.

Wie dies in der Praxis aussehen und gelingen kann, stellen wir in diesem Beitrag vor. Zunächst beleuchten wir, für welche Anteile eines IT-Projekts sich agile Methoden besonders gut eignen und für welche Anteile eher der Einsatz traditioneller Vorgehensweisen sinnvoll ist. Danach zeigen wir anhand eines Praxisbeispiels, wie die Kombination von agilen und klassischen Methoden innerhalb eines Projekts funktionieren kann. Abschließend werden einige Erfolgsfaktoren und Lessons Learned vorgestellt.

Autoren

Dr. Michael Kirchhof

Partner und Unter–nehmensberater bei der mm1Consulting & Management PartG., Verantwortlich für den Bereich Schlankes IT-Management und Technologie- Transformation.

Kontakt:
m.kirchhof@mm1-consulting.de

Prof. Dr. Bodo Kraft

Prof. für Wirtschaftsinformatik an der FH Aachen, Schwerpunkte: Prozessmodellierung und Projektmanagement

Kontakt: kraft@fh-aachen.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Agile Methoden im traditionellen Projektmanagement-Umfeld einsetzen

› Agiles Projektmanagement – eine Einführung

› Vorgehensmodelle für Software-Projekte im Vergleich

in den Rubriken:

› Agiles Projektmanagement

› Unternehmenskultur

Vorüberlegungen

Agile Methoden eignen sich gut, um Projekte effizienter zu steuern und flexibler auf Anforderungen der Stakeholder zu reagieren. So erfordert z.B. eine Produktneuentwicklung mit vielen Unklarheiten auf der Seite des Auftraggebers geradezu ein agiles Vorgehen. Aber auch klassische Methoden haben weiterhin ihre Berechtigung und ermöglichen – ein sorgfältiges Tailoring vorausgesetzt – effiziente Projekte. So bieten sich z.B. traditionelle Steuerungsmechanis-

men für kritische Migrationsprojekte an, bei denen der Kundenauftrag in Bezug auf Termin und Umfang genau und endgültig abgegrenzt ist und zudem ein hohes Maß an formaler Dokumentation erforderlich wird.

Doch warum sollte es nur ein "Entweder/Oder" geben? Warum nicht die Stärken beider Ansätze innerhalb eines Projekts nutzen? Dies würde dazu führen, nicht mehr zu entscheiden, ob ein Projekt im Ganzen agil oder klassisch durchzuführen ist. Vielmehr liegt die Herausforderung darin, für die jeweiligen Teilprojekte den jeweils passenden Ansatz zu identifizieren. Unserer Erfahrung nach ist es insbesondere bei großen Projekten (mit mehreren Teilprojekten und über 2.000 Personentagen) vielversprechend, die Methodik innerhalb der Projektstruktur zu variieren – also einzelne Teilprojekte und Arbeitspakete unterschiedlich entsprechend ihrer Bedürfnisse zu steuern. Wir begründen das insbesondere damit, dass in diesen Projekten die jeweiligen Anteile in der Einzelbetrachtung signifikante Aufwände darstellen. Die Anwendung unpassender Vorgehensweisen führt zu Effizienzverlusten, die durch die Vorteile einer einheitlichen Gesamt-Projekt-Vorgehensweise nicht kompensiert werden. Die Investition für die Einführung des Hybriden Vorgehensmodells lohnt sich somit in jedem Fall.

Eigene Konzepte erforderlich

Entscheidet man sich dafür, die Teilprojekte mit verschiedenen Vorgehensmodellen durchzuführen, müssen diese Modelle um Konzepte erweitert werden, die es z.B. ermöglichen, die Arbeitsteilung zu strukturieren oder die Teilprojekte untereinander zu synchronisieren. Die meisten Vorgehensmodelle bieten hierzu wenig Hilfestellung, d.h. es bleibt den Unternehmen oder sogar den Projekten selbst überlassen, Wege zu finden, um diese Herausforderungen zu bewältigen. Zu betrachten ist hierbei nicht nur das Projektmanagement, sondern ebenso weitere Querschnitts-Disziplinen, wie z.B. das Anforderungsmanagement und die Qualitätssicherung.

So ist es bei einer Kombination agiler und klassischer Ansätze während der Projektinitialisierung wichtig, die Schnittstellen, insbesondere die Kommunikations- und Steuerungsmechanismen, festzulegen. Gerade bei großen Projekten mit mehreren Teilprojekten und großen Lenkungsausschüssen ist eine durchgängige Planung mit Fortschrittskontrolle sowie ein einheitliches Reporting und Stakeholdermanagement für den Projekterfolg elementar. Nicht zuletzt erfordern auch Compliances, z.B. CMMI, eine integrierte Sichtweise auf das Gesamtprojekt.

Ein Praxisbeispiel

Nachfolgend möchten wir den Einsatz unterschiedlicher Vorgehensmodelle innerhalb eines Projekts an einem Praxisbeispiel verdeutlichen. Das Beispiel ist an vielen Stellen vereinfacht, um auf die hier relevanten Aspekte eingehen zu können. Vor allem soll das Beispiel zeigen, wie auf Teilprojektebene die eher volatilen (geeignet für agile Methoden) und die stabilen (geeignet für traditionelle Methoden) Anforderungen identifiziert und durch entsprechende Methoden umgesetzt werden können. Darüber hinaus möchten wir beleuchten, wie die Gesamtprojektsteuerung einen konsolidierten Blick auf das Projekt erhält und wie sich einheitliche Dokumentationsanforderungen erreichen lassen.

Das Beispielprojekt mit seinen Teilprojekten wird nach dem Vorgehensmodell des Rational Unified Process (RUP) abgewickelt. Dabei ordnen wir RUP den klassischen Vorgehensweisen zu. Dies ist allgemein umstritten (s. hierzu auch Ambler, 2002 und Cohn, 2009), da durch mehrere Inkremente und die fortlaufende Beteiligung des Auftraggebers bereits agile Grundideen realisiert werden. Dennoch beschreibt der RUP ein eher komplexes und detail-

liert festgelegtes Vorgehen, das an vielen Stellen im Gegensatz zum agilen Manifest (www.agilemanifesto.org) steht. Für große Projekte ist ein sorgfältiges Tailoring der Methode, Festlegung der Verantwortlichkeiten für jede Disziplin sowie die Struktur der internen und externen Projektkommunikation elementar. Dort wo Agilität und Kundennähe erforderlich sind, schafft die Einbettung agiler Teilprojekte einen wesentlichen Mehrwert.

Worum geht es im Projekt?

Bevor wir die Kombination der Ansätze darstellen, möchten wir zunächst die wichtigsten Rahmenbedingungen im Beispielprojekt "Legacy Online" beschreiben. Ziel dieses Entwicklungsprojekts ist es, eine Legacy-Host-Anwendung, wie sie in vielen Branchen zu finden ist, über ein Web-Frontend zugänglich zu machen. Hierzu wird eine klassische Multi-Tier-Architektur implementiert, d.h. es werden gekapselte Schichten definiert. Die Schichten dienen zur Realisierung der Benutzerschnittstelle (Front-End), zur Implementierung der Sicherheitsanforderungen (Middle-Tier) und schließlich zur Anbindung an die existierende Funktionalität in der Altanwendung (Backend-Tier).

Zur Umsetzung dieses Projekts wird die in Bild 1 dargestellte Projektorganisation aufgebaut. Der Lenkungsausschuss (LA) delegiert mittels Projektauftrag die Projektverantwortung an den Gesamtprojektleiter. Dieser steuert einige Aktivitäten auf der obersten Ebene, z.B. das Anforderungs-, Stakeholder- und Risikomanagement. Andere Aktivitäten werden über weitere Projektaufträge an die Teilprojektleiter delegiert, z.B. das Konfigurationsmanagement. Insbesondere beauftragt der Projektleiter die Teilprojekte mit Entwicklungsaufgaben im Bereich "Design" und "Implementierung". Im Beispiel übernimmt je ein Teilprojekt die Realisierung einer der drei oben beschriebenen Schichten. So kümmert sich z.B. das Teilprojekt "Frontend" um die Benutzerschnittstelle mit allen Masken und deren Navigation. Ein unternehmensweit zentralisiertes Testcenter koordiniert schließlich die Testaktivitäten und stellt die Abschluss-Testate für die Launch-Freigabe aus.

Synchronisation im Projekt

Die Synchronisation zwischen Gesamtprojekt und den Teilprojekten erfolgt über Meilensteine und Phasenabschlüsse (Bild 2). Entscheidend dabei ist, dass die ausführliche Planung in den Teilprojekten erfolgt: Ausgehend von ihren Projektaufträgen planen die Teilprojektleiter ihre Arbeitspakete detailliert aus und melden die geplanten Meilensteine in aggregierter Form an den Gesamtprojektleiter zurück (bottom-up). Dieser legt dann in Zusammenarbeit mit dem Auftraggeber die zeitlichen Ziele für die jeweiligen Phasen fest.

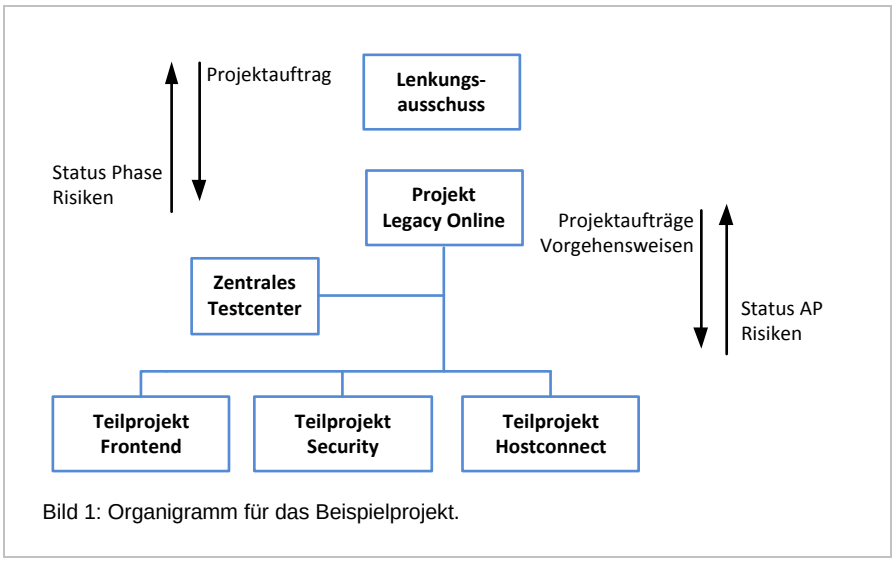

Bild 1: Organigramm für das Beispielprojekt.

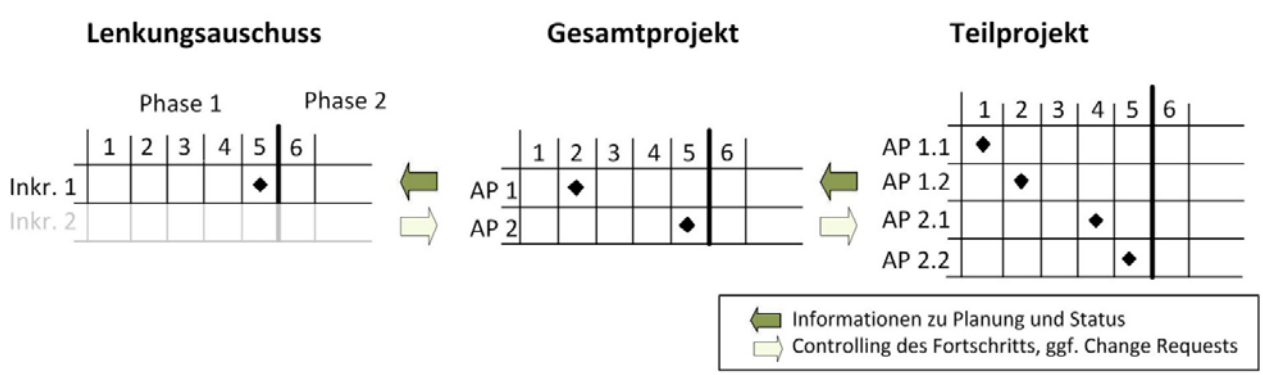

Bild 2: Verfeinerung der Aufgaben in der mehrstufigen Projekthierarchie.

Während der Projektdurchführung wird diese Planung über Statusberichte mit dem tatsächlichen Projektfortschritt abgeglichen. Das Controlling erstreckt sich dabei schrittweise von der Ebene des Lenkungsausschusses bis auf die Ebene des Teilprojekts (top-down). Veränderte Anforderungen werden – wegen des zentralisierten Anforderungsmanagements beim Gesamtprojektleiter – ebenso schrittweise an die jeweilige Ebene weiter gereicht.

Um ein solches Controlling zu ermöglichen, muss sich die Granularität entsprechend der jeweiligen Ebene ändern: Während der Lenkungsausschuss primär auf die Umsetzung der Inkremente achtet, die ausgeliefert werden sollen, verfeinert der Gesamtprojektleiter diese Inkremente zu Arbeitspaketen, die er delegieren kann. Der Teilprojektleiter verfeinert die delegierten Arbeitspakete weiter bis zu einer Granularität, die die Steuerung der Umsetzung ermöglicht. Damit ist die Skalierbarkeit bei großen Projekten erreicht.

Zuständigkeiten festlegen

Wie die Zuständigkeiten im Projekt aufgeteilt sind, zeigt Bild 3. Der Gesamtprojektleiter ("Gesamtprojekt Legacy Online") sowie die jeweiligen Teilprojektleiter sind für die Umsetzung der im Projektauftrag spezifizierten Leistungen verantwortlich. Die Gesamtverantwortung gegenüber dem Auftraggeber trägt zwar primär der Gesamtprojektleiter, dennoch sind die Teilprojektleiter durch die Delegation und Vergabe von Teilprojektaufträgen ebenso in der Verantwortung.

Die Verantwortung für das zentrale Anforderungsmanagement liegt in unserem Beispiel beim Gesamtprojektleiter. Aufgabe der Teilprojektleiter ist es, die fachlichen Anforderungen im Systemdesign und bei der Implementierung umzusetzen. Das Testcenter übernimmt die zentrale Verantwortung für das Qualitätsmanagement sowie dessen Koordination, während die Teilprojekte die Maßnahmen zur Qualitätssicherung durchführen und die Ergebnisse zurückmelden.

Dieses Beispielprojekt ließe sich in der Praxis auch ohne agile Ansätze steuern und umsetzen. Die Verantwortlichkeiten und Zuständigkeiten wären dann strikt getrennt. Vor allem für das Anforderungsmanagement bietet eine zentrale Steuerung auf Gesamtprojektebene mit dieser strikten Trennung Vorteile, da es so möglich ist, die externe Kommunikation zu bündeln und Kundenwünsche einheitlich zu beschreiben. In manchen Fällen kann es dennoch von Vorteil sein, in Teilbereichen ein flexibleres Modell anzubieten. In unserem Beispiel sind gerade für das Teilprojekt "Frontend"

	Projekt-management	Anforderungs-management	Qualitäts-management	Konfigurations-management
Gesamtprojekt "Legacy Online"	x	x	führt durch, berichtet	führt durch
Zentrales Testcenter	x	setzt um	x	führt durch
Teilprojekt "Frontend"	x	setzt um	führt durch, berichtet	x
Teilprojekt "Security"	x	setzt um	führt durch, berichtet	führt durch
Teilprojekt "Hostconnect"	x	setzt um	führt durch, berichtet	führt durch

x = verantwortlich

Bild 3: Aufteilung der Verantwortlichkeiten zwischen Gesamtprojekt und Teilprojekten.

häufige Iterationen mit dem Auftraggeber sinnvoll, um die Anforderungen zu erheben: Ausgehend von den zentralen Funktionen beschreiben Auftraggeber und Umsetzer dann zunächst gemeinsam die Anforderungen. In weiteren Durchgängen teilen sie die zunächst grob erfassten Anforderungen in sinnvolle Teilmengen auf, verfeinern diese und überarbeiten, falls erforderlich, die formalen Ergebnisse vorheriger Phasen. Die Beteiligten können bei diesem iterativen Vorgehen die Anforderungen in all ihrer Komplexität besser verstehen und erreichen ein gemeinsames Verständnis – was Voraussetzung für eine vollständige und korrekte Anforderungsdefinition ist. Um eine so intensive Interaktion mit dem Auftraggeber zu ermöglichen, bietet es sich an, das Teilprojekt "Frontend" agil zu steuern.

Das "Hybride Vorgehensmodell"

Wir erlauben in diesem Projekt explizit die Kombination verschiedener Vorgehensmodelle und bezeichnen diese Kombination als "Hybrides Vorgehensmodell". In unserem Fall bedeutet das, dass die Gesamtprojektsteuerung sowie die Teilprojekte "Security" und "Hostconnect" nach dem klassischen Vorgehensmodell RUP arbeiten, das Teilprojekt "Frontend" dagegen agil nach Scrum. Die Interaktion und die Aufgabenteilung zwischen den Teilprojekten muss neu definiert werden, um die Stärken des jeweiligen Vorgehens nutzen zu können und trotzdem eine durchgängige Projektsteuerung zu ermöglichen.

Nachfolgend gehen wir insbesondere auf die, unserer Meinung nach, erfolgskritischen Aspekte der Integration in das Gesamtprojekt ein sowie auf die Zusammenarbeit zwischen den Teilprojekten. Wir zeigen anhand der Disziplinen "Projektmanagement", "Anforderungsmanagement" und "Qualitätsmanagement", welche Auswirkungen das Hybride Vorgehensmodells hat. Dabei berücksichtigen wir sowohl die Sichtweise des Gesamtprojekts auf die Teilprojekte (top-down), als auch die erforderliche Integration der Teilprojektinformationen zum Ganzen (bottom-up).

Projektmanagement

Durch das Verwenden verschiedener Vorgehensmodelle wirkt die Projektorganisation auf den ersten Blick komplizierter, da jedes Vorgehensmodell unterschiedliche Fähigkeiten erfordert und die Art der Zusammenarbeit sich bei beiden Modellen unterscheidet. Da nur in einem Teilprojekt nach Scrum entwickelt wird, ist keine übergeord-

nete "Scrum-Koordination" erforderlich (Stichwort: "Scrum of Scrums"). Auch für die Steuerung der Teilprojekte ergeben sich aus Projektmanagementsicht keine besonderen Anforderungen: Es bleibt auf dieser Ebene bei der Delegation von Arbeitspaketen an die jeweiligen Teilprojekte. Das Vorgehensmodell spielt dabei keine Rolle, es definiert lediglich die Aufgaben der Teilprojektleitung. Da auch zur Synchronisation zwischen den einzelnen Teilprojekten weder für die Projektleitung noch für die Projektmitarbeiter eine Veränderungen des "normalen" Projektalltags erforderlich ist, entsteht insgesamt für das Projektmanagement kein zusätzlicher Aufwand.

Arbeitsauftrag und Reporting

Die Einbindung in das Gesamtprojekt erfordert allerdings eine geänderte Kommunikation bzgl. des Arbeitsauftrags und des Reportings. Hier wird das Stage-Gate-Prinzip verwendet: Betrachtet werden nur die Arbeitsergebnisse der Teilprojekte zu den vorher festgelegten Meilensteinen und Phasenabschlüssen. Ansonsten werden die Teilprojekte aus Sicht des Projektmanagements als Black Box betrachtet.

Arbeitsaufträge an das Teilprojekt "Frontend" werden in den Scrum-üblichen Ritualen erteilt, hier ist der Auftraggeber also enger eingebunden als in den anderen Teilprojekten. Die Rolle des Product Owners wird idealerweise durch den übergeordneten Projektleiter besetzt, in unserem Beispiel also durch den Gesamtprojektleiter. Kritisch für den Projekterfolg ist dabei, dass der Product Owner während eines laufenden Sprints nur Anforderungen (er)klärt und nicht verändert. Es ist Aufgabe des Product Owners, die mitunter variierenden Interessen der Stakeholder zu kapseln und zu kanalisieren – auf keinen Fall darf die Variabilität in die Entwicklung einfließen.

Das Reporting im agil durchgeführten Teilprojekt erfolgt analog zu dem Scrum-üblichen Vorgehen und basiert, vereinfacht ausgedrückt, auf "Burn-Down-Charts". Für diese Darstellung werden initial die umzusetzenden Funktionen mittels eines interaktiven Verfahrens gemäß ihrer Komplexität bewertet und als "Story-Points" aufsummiert. Die Summe der Story-Points der noch nicht umgesetzten Funktionen wird tagesaktuell erfasst, der Verlauf dieser Summe stellt den Arbeitsfortschritt (umgekehrt proportional) dar. Durch einen Vergleich dieses Werts mit der ursprünglichen Schätzung lässt sich zum Berichtszeitpunkt der Grad der Zielerreichung ermitteln. Erst der Abschluss von Arbeitspaketen wird dann wieder als Meilenstein-Erreichung berichtet. Diese Art des Reportings muss dann auf der Ebene des Gesamtprojekts berücksichtigt werden. Für den Gesamtprojektleiter unterscheidet sich die Berichtsform der agilen Teilprojekte somit nicht von der Berichtsform der klassisch geführten Teilprojekte, so dass eine einheitliche Überwachung und Steuerung möglich ist. In den aggregierten Berichten an den Projekt-Auftraggeber treten diese Unterschiede ohnehin nicht mehr in Erscheinung.

Diese Art der Fortschrittsmessung hat sogar Vorteile. Beim Messen auf Basis von Meilensteinen und Untermeilensteinen wird in aller Regel schlicht die Anzahl der erreichten Meilensteine als Maß für den Fortschritt verwendet – und damit vereinfachend angenommen, dass das Erreichen jedes Meilensteins einen gleichen Anteil an der zu leistenden Arbeit darstellt. Bei der hier beschriebenen Messung sind die Aussagen genauer, denn es fließen bereits die Expertenschätzungen des Teams (vgl. Burn-Down-Charts etc.) mit in die Fortschrittsmessung ein.

Meeting-Strukturen

Zur Zusammenarbeit mit anderen Teilprojekten hat es sich bewährt, die Meeting-Strukturen für den formalen und informellen Austausch vorher klar festzulegen. In Tabelle 1 ist ein geeigneter Fahrplan für die Zusammenarbeit

innerhalb des Projekts abgebildet: Nach dem Kick-off bearbeiten die Teilprojekte die übertragenen Arbeitsaufträge. Die Aggregation und die Abstimmung zwischen Gesamtprojekt und Teilprojekten erfolgt immer im 4-Wochen-Rhythmus im Rahmen der Lenkungsausschuss-Sitzungen. In Wochen, in denen diese Besprechungen auf Gesamtprojektebene stattfinden, müssen die angegebenen Meetings der jeweiligen Teilprojekte mindestens einen Tag zuvor sattfinden, damit für die Lenkungsausschuss-Sitzungen jeweils der aktuelle Status zur Verfügung steht. Ansprechpartner für den Gesamtprojektleiter und die direkte Koordination zwischen den Teilprojekten sind die Teilprojektleiter und im Fall des agilen Teilprojekts der Product Owner. Informationen zu den produzierten Artefakten sind zudem über das Konfigurationsmanagement für alle Projektbeteiligten transparent und abrufbar.

	PW 0	PW 1	PW 2	PW 3	PW 4	PW 5	PW 6	PW 7	PW 8	PW 9	PW 10
GP	Kick-off	JF	JF	JF	LA	JF	JF	JF	LA	JF	…
TP 1 (Scrum)		Sprint Planning	Dailies	Dailies	Sprint Review + Planning	Sprint Planning	Dailies	Dailies	Sprint Review + Planning	Dailies	…
TP 2 (RUP)		JF	JF	JF	JF	JF	JF	JF	JF	JF	…
TP 3 (RUP)			JF	JF	JF	JF	JF	JF	JF	JF	…

Tabelle 1: Koordination des Gesamtprojekts (GP) mit den Teilprojekten (TP) in den einzelnen Projektwochen (PW).

Anforderungsmanagement

Im Bereich des Anforderungsmanagements liegen die Stärken des agilen Vorgehens. Dies gilt insbesondere, wenn die Anforderungen nicht vorab vollständig erhoben werden können oder der Auftraggeber keine genaue Vorstellung von dem zu entwickelnden Produkt hat. Deshalb werden z.B. für die Entwicklung von Benutzerschnittstellen häufig agile Vorgehensweisen bevorzugt. Für die Gesamtprojektleitung und den Auftraggeber ergeben sich dadurch elementare Veränderungen.

Ein agiles Vorgehen setzt die enge Mitarbeit des Auftraggebers im Projekt voraus. Insbesondere die Anforderungsdefinitionen für einzelne Sprints werden zusammen mit dem Auftraggeber erarbeitet, aber auch die Ergebniskontrolle durch Review und Test erfordert dessen Mitwirken. Als Sprint wird hier ein Entwicklungszyklus von vier Wochen angenommen, der mit den Management-Techniken des agilen Manifests gesteuert wird.

Damit erfolgt das Anforderungsmanagement auf zwei Ebenen:

1. Die grundlegenden funktionalen und nicht-funktionalen Anforderungen für das Gesamtsystem werden im Beispielprojekt weiterhin zentral durch die Gesamtprojektleitung erhoben und verwaltet. Über spezifische Projektaufträge beauftragt diese die Teilprojekte.

2. Die Anforderungen speziell für die Benutzerschnittstelle erarbeitet das Teilprojekt "Frontend" in enger Abstimmung mit dem Auftraggeber. Diese definieren bzw. ergänzen den Projektauftrag dieses Teilprojekts.

Anforderungsmanager für das agile Teilprojekt

Für den Projekterfolg ist es elementar, dass bei der verteilten Anforderungsentwicklung ein konsistentes Gesamtbild der Anforderungen entsteht: Es dürfen zum einen keine mehrfachen oder sich überschneidenden Anforderungen spezifiziert und ggf. implementiert werden. Zum anderen dürfen aber auch keine Anforderungen vergessen werden.

Die Projektorganisation muss daher so aufgebaut sein, dass diese Konsistenz bei den Anforderungen gewährleistet werden kann. Erreichen lässt sich das, indem bei der Anforderungsdefinition ein Vertreter der Gesamtprojektsteuerung im agilen Teilprojekt als weiterer (interner) Auftraggeber mitarbeitet. Dieser Anforderungsmanager hat die anspruchsvolle Aufgabe, die inhaltliche Konsistenz der beiden Ebenen zu gewährleisten. Dabei kann man ihn unterstützen, indem die verteilt erhobenen Anforderungen in einem einheitlichen Format gespeichert und abgelegt werden.

Qualitätsmanagement

Bezüglich der Qualitätssicherung entsteht der größte Unterschied zwischen den Teilprojekten. Während in klassischen Vorgehensmodellen die Qualitätssicherung eine eher untergeordnete Rolle spielt – und dies leider viel zu häufig ein Schwachpunkt in IT-Projekten ist –, wird in agilen Verfahren (insbesondere Scrum) eine unumstößliche "Definition of Done" postuliert: Das Team darf nur dann einen Arbeitsauftrag freigeben bzw. als fertig melden, wenn eine tatsächliche Qualitätssicherung aus der Verwender-Sicht (technischer oder menschlicher Verwender) durchgeführt wurde (Beck, 2011 und Schwaber, 2004).

Ein eigener Tester im Team

In der Praxis hat es sich bewährt, in das Entwicklungsteam einen eigenen Tester zu integrieren. Die direkte räumliche Nähe ist sehr zu empfehlen – das Team erlebt Fehler beim Testen dann unmittelbarer, was in der Regel dazu führt, dass die Entwickler eine minderwertige Qualität zu vermeiden versuchen. Hinzu kommt der Effekt, dass durch das frühe Erstellen der Testfälle die späteren Abläufe im Programm bereits mental durchgespielt werden. Pro Entwicklungsteam setzen sich dadurch mindestens zwei Personen – der interne Anforderungsmanager und der Tester – sehr detailliert mit den Anforderungen auseinander. Dadurch können sie Unklarheiten und Interpretationsspielräume einfacher (und auch früher) identifizieren und eine Klärung, z.B. durch den Product Owner, einfordern.

Im Hybriden Vorgehensmodell steht es den anderen Teilprojekten frei, das Konzept des dedizierten Testers ebenfalls einzusetzen. Dieser ist als Ergänzung zum zentralen Testcenter zu sehen. Die im Teilprojekt entstehende Test-Dokumentation verbleibt im Teilprojekt und wird nur bei Bedarf veröffentlicht – muss also nicht Teil des zentralen Konfigurationsmanagements sein.

Lessons Learned und Empfehlungen

Zusätzliche Formalismen

Um die Integration des agil gesteuerten Teilprojekts in das traditionell durchgeführte (Gesamt-)Projekt sicherzustellen, empfehlen wir, in einigen Bereichen folgende zusätzliche Formalismen einzuführen. Diese ermöglichen ein einheitliches, teilprojektübergreifendes Vorgehen.

- Anforderungsmanagement: Hier können Abhängigkeiten zwischen Anforderungen berücksichtigt werden – und zwar sowohl bei deren Erhebung zu Projektbeginn als bei deren dynamischer Evaluation zur Projektlaufzeit.
- Projektmanagement: Hier kann es sinnvoll sein, eine Gesamtprojektplanung zu erstellen und das Projekt anhand dieser Planung durchzuführen und zu überwachen. Um das agile Teilprojekt in die Gesamtprojektplanung zu aggregieren, bewertet man dessen Projektfortschritt. So erhält man zudem ein aussagekräftiges Controlling des agilen Teilprojekts.
- Qualitätsmanagement: Die gesammelten spezifizierten Anforderungen lassen sich als Grundlage für die Testplanung und -ausführung verwenden. Die Anforderung, dass zu Beginn der Projektbearbeitung Anforderer und Tester zusammenarbeiten (s.o.), wirkt sich zudem sehr positiv auf die Qualität in den Teilprojekten und im Gesamtprojekt aus. Denn durch diese direkte Zusammenarbeit wird einerseits sichergestellt, dass die Tests auch tatsächlich die angeforderte Funktionalität abdecken und somit deren Umsetzung sicherstellen und andererseits, dass tatsächlich das spezifiziert wird, was gemeint und gewünscht ist.

Der zusätzliche Aufwand ist gerechtfertigt, denn die erreichten positiven Effekte kompensieren die Aufwände. Dies haben wir in Projekten festgestellt, die wir gemäß dem hier beschriebenen Hybriden Vorgehensmodell durchgeführt haben. Die Projektgröße variierte dabei, wobei sich die deutlichsten Verbesserungen bei mittleren und großen Projektvorhaben zeigten – letztlich kann man daraus schließen, dass die erreichte Verbesserung proportional zur Projektgröße ist. Ausschlaggebend dafür ist der naturgemäß steigende Projekt- und Kommunikations-Overhead bei zunehmender Projektgröße. Es geht hier quasi um einen Skalen-Effekt – je größer das Vorhaben ist, desto stärker wirken sich Fehler und Missverständnisse aus – was beim Auftreten dieser Störung in frühen Phasen des Projekts besonders dramatisch ist.

Vertrauen aufbauen

Um Vorbehalte der Projektbeteiligten bei der Einführung des Hybriden Vorgehensmodells gering zu halten, ist es wichtig, gegenseitiges Verständnis aufzubauen und keine harten Abgrenzungen zuzulassen. Denn die unterschiedlichen Wertestrukturen bei agilen und klassischen Vorgehensweisen können leicht zu Missverständnissen und Konflikten zwischen den Projektmitarbeitern führen. Während agile Vorgehensweisen eher auf Eigenverantwortung und –motivation setzen, beruhen klassische Vorgehensweisen auf dem Konzept von zentraler Planung und Kontrolle. Um das Arbeiten in einem gemeinsamen Projekt erfolgreich zu gestalten, empfehlen wir die Methodenkompetenz der Mitarbeiter aufzubauen – sei es durch gezielte Schulungsmaßnahmen oder durch temporäre Mitarbeiter in dem jeweils anders gesteuerten Teilprojekt. So wird die Basis für gegenseitiges Vertrauen und Verständnis im Projektteam aufgebaut.

Letztlich bedeutet das vorgestellte Hybride Vorgehensmodell einen Veränderungsprozess in den Unternehmen– weg von der vollständigen Spezifikation hin zum inkrementellen Arbeiten.

Informationen vernetzen

Eine Herausforderung bei der Anwendung des Hybriden Vorgehensmodells liegt zudem darin, die in Artefakten und Ergebnistypen vorliegenden Informationen zu vernetzen. Erfahrungsgemäß entwickelt sich in Organisationen und Arbeitsgruppen nach kurzer Zeit eine Best-Practice der Dokumentation und des Informationsaustauschs, die spezifisch für das Vorgehensmodell ist. Wenn nun verschiedene Vorgehensmodelle angewendet werden, so ist zumindest ein Aufklärungs- und Informationsbedarf bezüglich der jeweiligen Vorgehensweisen erforderlich. In der Praxis hat sich gezeigt, dass die größte Hürde oftmals der Sprachgebrauch ist – gleiches wird mit unterschiedlichen Namen belegt, nach kurzer "Übersetzung" lösen sich vermeintliche Konflikte und Unstimmigkeiten in Luft auf. Mit dieser Vernetzung lassen sich einerseits mehrfache Eingaben gleicher Sachverhalte vermeiden. Oftmals treten solche Mehrfacheingaben bei der Anreicherung von Anforderungen auf, z.B. durch Einzelgespräche mit den Stakeholdern oder auch bei der Dokumentation technischer Implikationen einzelner Anforderungen.

Andererseits kann man durch die Nutzung der vernetzten Informationen die Abhängigkeiten im Projekt besser erkennen. Gängige Prozess-Frameworks wie CMMI fordern eine Vernetzung (Traceability) z.B. zwischen den Anforderungen und Elementen der Disziplinen "Projektmanagement", "Test und Design" und "Implementierung". Zu solchen Elementen zählen z.B. Projektplan, Anforderungsdatenbank, Testfall-Dokumentation und Fehler-Item. Diese Vernetzung von Artefakten und Metriken ist als nächster Schritt zu untersuchen.

Basierend auf den gewonnenen Erfahrungen gehen wir davon aus, dass sich das hier beschriebene Hybride Vorgehensmodell auf andere Projektdomänen ausweiten lässt, zumindest auf Technologie-Projekte. Hier ist dann insbesondere zu untersuchen, inwieweit das Projektverständnis und die gängige Praxis sowie etwaige regulatorische und rechtliche Vorgaben sich im Hybriden Vorgehensmodell umsetzen lassen und welcher Adaptionsaufwand nötig ist.

Literatur

- Aghajani, B.; Kirchhof, M.: Agil in die Sackgasse, manage it, S. 1-6, 2010
- Ambler, S.: Agile Modeling - Effective Practices for Extreme Programming and the Unified Process, John Wiley & Sons, 2002
- Beck, K.: Manifesto for Agile Software Development, www.agilemanifesto.org (Zitat zuletzt eingesehen am 07. 12.2011)
- Cohn, M.: Succeeding with Agile: Software Development Using Scrum, Addison-Wesley Professional, 2009
- Essigkrug, A.: Agil(e) mit RUP: Gegensatz oder ideale Verbindung, Objektspektrum – Sonderbeilage Agilität, S. 8-10, 2010
- Hacker, T.; Kraft, B.; Zöll, A.: Projektzuschnitt für die inkrementelle Systementwicklung im Konzernverbund, in: Kuhrmann, M.; Linssen, O.: 18. WSGIVM: Zusammenspiel von Vorgehensmodellen und Organisationsformen, S. 1-11, 2011

- Herzog, J.; Holzmüller, G.; Schoeppe, W.: Agile Softwareentwicklung im Kontext unternehmensweiter IT-Prozesse, Objektspektrum – Sonderbeilage Agilität, S. 6-7, 2010
- Jacobson, I.; Booch, G.; Rumbaugh, J.: The unified software development process, Addison-Wesley, 1999
- Müller, T.: Welcome to Reality! Agile vs. Klassisch, Objektspektrum – Sonderbeilage Agilität, S. 16-17, 2010
- Naur, P.; Randell, B.: Software Engineering: Report of a conference sponsored by the NATO Science Committee, NATO, Scientific Affairs Division, 1968
- Reiss, M.: Hybride Vorgehensmodelle, in: O. Linssen et al.: Integration von Vorgehensmodellen und Projektmanagement, S. 1-14, 2010
- Schwaber, K.: Agile Project Management with Scrum, Microsoft Press, 2004
- Schwaber, K.; Beedle, M.: Agile Software Development with Scrum, Pearson Studium, 2008

Fachbeitrag

Scrum im Unternehmen einführen Teil 1: Einführung "von oben"

Autor

Thomas Lieder

Diplom-Informatiker, Certified Srum Master, Senior Project Manager bei der Setzwein IT-Management GmbH

Kontakt: thomas.lieder@setzwein.com

Mehr Informationen unter:
› projektmagazin.de/autoren

Autor

Katja Roth

Dipl.-Inform., PM-Fachfrau IPMA Level D, Certiefied Scrum Master, Certified Scrum Product Owner, Senior Project Manager bei der Setzwein IT-Management GmbH

Kontakt: katja.roth@gmx.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in den Rubriken:

› Agiles Projektmanagement
› Change Management
› PM im Unternehmen einführen

Der Software-Entwicklungsprozess Scrum wirkt auf den ersten Blick bestechend einfach: Es gibt nur wenige Rollen, so gut wie keine zu erstellenden Dokumente und eine überschaubare Menge an Regeln. Diese scheinbare Einfachheit verführt häufig zu der Annahme, dass auch die Einführung von Scrum in einem Unternehmen einfach durchzuführen sei. Die Praxis zeigt allerdings, dass für die erfolgreiche Umsetzung einer agilen Methode wie Scrum auch eine Reihe von Randbedingungen in der Organisation sowie der Unternehmenskultur erfüllt sein müssen.

Der Software-Entwicklungsprozess Scrum wirkt auf den ersten Blick bestechend einfach: Es gibt nur wenige Rollen, so gut wie keine zu erstellenden Dokumente und eine überschaubare Menge an Regeln. Diese scheinbare Einfachheit verführt häufig zu der Annahme, dass auch die Einführung von Scrum in einem Unternehmen einfach durchzuführen sei. Die Praxis zeigt allerdings, dass für die erfolgreiche Umsetzung einer agilen Methode wie Scrum auch eine Reihe von Randbedingungen in der Organisation sowie der Unternehmenskultur erfüllt sein müssen.

Randbedingungen für die Einführung von Scrum:

- Verfügbarer und entscheidungsbefugter Vertreter der Fachseite (Product Owner)
- Verzicht auf klassische Rolle des Projektleiters
- Tests und Deployment automatisierbar

Dieser zweiteilige Beitrag zeigt, worauf die Beteiligten bei der Einführung von Scrum in einem Unternehmen achten sollten. Anhand zweier fiktiver, aber praxisnaher Szenarien wird gezeigt, wie eine Einführung durchgeführt werden könnte. Das konkrete Vorgehen in einem Unternehmen muss dabei immer von Fall zu Fall definiert werden. Im ersten Teil erfahren Sie, wie eine Scrum-Einführung "von oben" aussieht und was es dabei zu beachten gilt. Der zweite Teil widmet sich der Einführung im Unternehmen "von unten".

Wir möchten mit diesem Beitrag allen, die an der Einführung agiler Methoden interessiert sind, einen Einblick in die prinzipielle Vorgehensweise geben. Vornehmlich richtet sich der Artikel an IT-Projektleiter und IT-Abteilungsleiter sowie an Geschäftsführer. Aber auch für Mitarbeiter der IT-Abteilungen sowie für Führungskräfte und Mitarbeiter anderer Abteilungen ist dieser Artikel interessant, da bei einer Einführung von Scrum stets das gesamte Unternehmen von der Veränderung betroffen ist.

Randbedingungen für den Einsatz von Scrum

Bevor sich die Verantwortlichen Hals über Kopf in die Einführung von Scrum stürzen, sollten sie vorher etwas Zeit investieren und sich überlegen, ob und warum Scrum ein geeigneter Prozess für die eigene Softwareentwicklung sein könnte. Denn auch wenn Scrum als Entwicklungsprozess einfach ist – die Einführung bedarf einer sorgfältigen Durchführung und lässt sich nicht "mal nebenbei" erledigen. (Zu Scrum siehe: Wirdemann, Projekt Magazin 21/2009 und Lieder/Roth, Projekt Magazin 11/2010)

So sollte den Verantwortlichen bewusst sein, dass sich die Einführung von Scrum als Softwareentwicklungsprozess nicht auf die IT-Abteilung beschränkt, sondern auf die gesamte Organisation und viele Abteilungen auswirkt. So müssen z.B. der Systembetrieb, das Marketing oder der Vertrieb auch in den Scrum-Prozess mit einbezogen werden.

Lernen als wichtiger Faktor

Um Scrum erfolgreich im Unternehmen zu etablieren, ist daher eine Unternehmenskultur wichtig, in der eine hohe Veränderungs- und Lernbereitschaft besteht. Zum einen, weil die Einführung des Prozesses selbst eine Veränderung bedeutet, zum anderen, weil sich Scrum in einem Unternehmen fortwährend weiterentwickeln soll. Das iterative Vorgehen mit konkreten Zwischenergebnissen sowie das darauf beruhende regelmäßige Feedback durch den Product Owner und die Anwender unterstützen das Lernen, sowohl bezogen auf den Scrum-Prozess als auch auf die jeweiligen Projektziele.

Lernen bedeutet in diesem Zusammenhang aber auch, Fehler machen zu dürfen – genau dies muss die Unternehmenskultur erlauben. Weiter müssen Werte wie eine gute Kommunikation, Respekt, Vertrauen und Offenheit im Unternehmen gelebt werden und dürfen nicht nur Lippenbekenntnisse sein. Nur so ist es möglich, den Prozess beständig weiterzuentwickeln und an die individuelle Unternehmensumgebung anzupassen.

Hohe Qualifikation aller Beteiligten

Für Scrum wichtige Werte der Unternehmenskultur:

- Lernbereitschaft
- Änderungsbereitschaft
- Gute Kommunikation
- Offenheit
- Respekt
- Vertrauen

Durch das iterative Vorgehen lassen sich mit Scrum auch Projekte realisieren, deren softwaretechnische Umsetzung zu Projektbeginn noch nicht klar ist. Team und Product Owner können sich dem Ziel schrittweise nähern, da Scrum nach jeder Iteration ein fertiges Produktrelease vorsieht. Dieses Vorgehen stellt hohe Ansprüche an die Qualifikation des Entwicklerteams wie auch des Scrum Masters.

Das Team sollte aus erfahrenen Entwicklern bestehen, die sich in der jeweiligen Technologie auskennen, offen für Neues sind und eine hohe Lernbereitschaft mitbringen. Für den designierten Scrum Master ist insbesondere wichtig, dass er von dem Teammitgliedern respektiert wird und über einschlägige Führungserfahrung verfügt. Im Idealfall kennen die beteiligten Personen bereits Scrum und haben dieses

Verfahren schon in anderen Unternehmen angewandt. Ansonsten ist eine Schulung bzw. ein Coaching der Beteiligten dringend zu empfehlen, damit alle den Scrum-Prozess und ihre Rollen kennen.

Weiter muss es für einen reibungslosen Projektverlauf auf Seiten des Auftraggebers (z.B. externer Kunde oder interner Fachbereich) einen kompetenten und entscheidungsbefugten Product Owner geben, der sowohl seine Rolle und die damit verbundenen Aufgaben kennt, als auch den entsprechenden Gestaltungsfreiraum hat. Das Zusammenspiel von Scrum Master, Team und Product Owner impliziert dabei auch den Verzicht auf die Rolle des klassischen Projektmanagers. Dessen Aufgaben, wie z.B. Aufwandschätzungen, Terminplanung und Reporting, werden vom Product Owner und dem Team selbst übernommen, was dementsprechend selbständig agieren können muss.

Scrum verändert "alte Muster"

Die regelmäßige und häufige Inbetriebnahme einer neuen Produktversion, die mit dem Scrum-Prozess verbunden ist, stellt darüber hinaus Anforderungen an die Entwicklungsinfrastruktur: Zum einen sollten weitgehend automatisierte Tests und ein stark automatisiertes Deployment in die Test- und Produktionsumgebungen vorhanden sein. Schließlich wird am Ende jeder Iteration ein neues Produktrelease veröffentlicht – ein hoher manueller Aufwand und lange Vorlaufzeiten für ein Deployment wären dafür hinderlich. Zum anderen sollten auch in der Softwareentwicklung selbst agile Methoden, wie z.B. Pair-Programming, Collective Code Ownership, Refactoring, Continious Integration oder testgetriebene Entwicklung eingeführt werden, um eine gute Qualität der Software trotz kurzer Entwicklungszyklen und hoher Änderungsfrequenz des entwickelten Codes sicherzustellen.

Und auch die Beziehung zu den Anwendern bleibt nicht unberührt: Am Ende jeder Iteration steht prinzipiell eine neue verbesserte Version der Software zur Verfügung, die sofort eingesetzt werden könnte. Einer sofortigen Einführung steht jedoch häufig der damit verbundene hohe Schulungsaufwand entgegen. Sofern jedoch nicht mit jedem Release alle Funktionsbereiche angepasst werden, reduziert sich auch die Zahl der Anwender, die von den Änderungen betroffen sind. Bei großen Unternehmen können dann aber immer noch Abteilungen mit zahlreichen Mitarbeitern betroffen sein, so dass dennoch ein erheblicher Schulungsaufwand anfällt. Um die notwendigen Schulungen weiter zu reduzieren, ist es möglich, mit sehr spezifischen Benutzerschnittstellen zu arbeiten: Im Idealfall sieht dann nur eine kleine Benutzergruppe die Änderungen. Die Einrichtung solcher Schnittstellen bedeutet jedoch in der Regel einen hohen Aufwand im Berechtigungsmanagement, also der Verwaltung von Benutzern und Zugriffsberechtigungen, so dass es nicht immer sinnvoll ist, eine solche Lösung umzusetzen.

Eine elegante Alternative zur Reduktion des Schulungsaufwands durch sehr spezifische Benutzerschnittstellen sind intuitiv zu bedienende Schnittstellen: Im Idealfall ist die Benutzerschnittstelle so konzipiert, dass explizite Schulungen nicht nötig sind, sondern ein Update der Onlinehilfe reicht.

Zwei Varianten bei der Einführung von Scrum

Ergeben die Vorüberlegungen, dass eine Einführung von Scrum sinnvoll und möglich ist, bleibt das konkrete Vorgehen für die Einführung festzulegen. Hierbei sind im Prinzip zwei Modelle zu unterscheiden: Die "Einführung von oben" sowie die "Einführung von unten". Die Einführung von oben wird von der Geschäftsleitung initiiert. Die Einführung von unten beginnt in der Regel auf der Team- bzw. Abteilungsebene. Die Entscheidung hierzu kann von der Geschäftsleitung

kommen, dies ist aber nicht zwingend erforderlich. Da zunächst keine unternehmensweiten Änderungen angestrebt werden, kann die Entscheidung auch auf einer unteren Ebene getroffen werden. Spätestens aber, wenn Änderungen über Abteilungsgrenzen hinweg notwendig werden, muss auch hier die Geschäftsleitung mit ins Boot geholt werden.

Die Einführung von oben geschieht mit einem offiziellen Change-Projekt sowie einem ersten unternehmensweiten Scrum-Projekt (Pilotprojekt). Wichtig ist dabei die Unterstützung durch die Geschäftsleitung. Diese Unterstützung ermöglicht es in der Regel, die bei der Einführung immer wieder auftretenden Probleme, wie z.B. geringe Akzeptanz der neuen Rollen oder Veränderungen der Unternehmensprozesse, effektiv zu lösen. Allerdings ist bei der Einführung von oben auch das gesamte Unternehmen über die Veränderung informiert. Damit ist das Pilotprojekt allen Mitarbeitern bekannt, erhält Gewicht und steht so unter einem hohen Erfolgsdruck.

Im Gegensatz dazu beschränkt sich die Einführung von unten zunächst auf ein Change- und ein Pilotprojekt innerhalb der IT-Abteilung. Dabei wird also nicht gleich das gesamte Unternehmen über beide Projekte informiert, wodurch sich auch der Erfolgsdruck auf das Projekt reduzieren lässt. Durch die Beschränkung auf nur einen Ausschnitt der Gesamtorganisation besteht aber das Risiko, dass bei einem "richtigen" Projekt in der Gesamtorganisation Probleme auftreten, die vorher nicht berücksichtigt wurden. Dadurch kann es passieren, dass zunächst eine Scheinsicherheit hinsichtlich der Durchführbarkeit von Scrum-Projekten entsteht.

Doch unabhängig davon, welchen Weg man letztendlich einschlägt: Die Einführung wird Auswirkungen auf das gesamte Unternehmen haben: "Agile tunes the engine, but the rest of the car needs an update" (Grant, 2009).

Wichtige Scrum-Begriffe

Tabelle 1 liefert eine Auflistung wichtiger Scrum-Begriffe, die im weiteren Verlauf dieses Beitrags verwendet werden. (Eine genaue Erklärung dieser und weiterer Scrum-Begriffe finden Sie in: "Agiles Projektmanagement. Scrum – eine Einführung", Projekt Magazin 21/2009).

Die Einführung von oben

Da das agile Vorgehen lange Releasezyklen durch kurze Iterationen ersetzt, die jeweils ein fertiges Produktinkrement als Ergebnis liefern, ist nicht nur die Softwareentwicklungskultur eines Unternehmens vom neuen Vorgehen betroffen. Zusätzlich werden auch zahlreiche Anforderungen an andere Unternehmensbereiche gestellt – bei einem Softwarehaus z.B. an den Vertrieb, der ein Produkt vermarkten muss, das sich in seinem Funktionsumfang ständig verändert.

Wird Scrum von oben in das Unternehmen eingeführt, lassen sich alle betroffenen Unternehmensbereiche gleichermaßen berücksichtigen und auftretende Schwierigkeiten durch die Unterstützung des Managements schnell und effektiv beseitigen.

Wichtige Scrum-Begriffe	Erklärung
Sprint / Iteration	Der Sprint bezeichnet in Scrum einen festgelegten Zeitraum (z.B. vier Wochen), in welchem das Team die Software entwickelt. Ein solcher Entwicklungszyklus besteht aus Analyse und Planung im Sprint Planning, Design, Programmierung und Test innerhalb der Umsetzung und Qualitätsüberprüfung und Prozessverbesserung in Review und Retrospektive. Am Ende des Sprints steht funktionsfähige Software zur Verfügung.
Burndown Chart	Burndown Charts zeigen im Zeitverlauf den Restaufwand eines Sprints (Sprint Burndown Chart) oder eines Releases (Release Burndown Chart) an.
Product Backlog	Priorisierte Liste, welche die Anforderungen der zu entwickelnden Software enthält.
Taskboard	Board, das sämtliche Aufgaben sowie deren Entwicklungsstand visualisiert.
Review	Im Review-Meeting präsentiert das Team die im aktuellen Sprint entwickelte Software, die vom Product Owner abgenommen wird.
Impediments	Störungen, die im Scrum-Prozess auftreten; eine Auflistung dieser Störungen erfolgt im Impediment Chart.
Velocity	Entwicklungsgeschwindigkeit des Teams innerhalb eines Sprints. Die Velocity wird anhand der entwickelten und am Ende des Sprints ausgelieferten Funktionalität gemessen.
Sprint Planning	Das Sprint Planning besteht aus zwei Meetings. Im ersten Meeting entscheidet das Team, wie viele Anforderungen aus dem Product Backlog im nächsten Sprint umgesetzt werden sollen. Im zweiten Meeting verteilt das Team eigenständig untereinander die Aufgaben des nächsten Sprints.

Tabelle 1: Wichtige Scrum-Begriffe.

Beispiel "SuperArchiv"

"SuperArchiv" ist ein mittelständisches Unternehmen mit 200 Mitarbeitern, das eine Archivierungssoftware erstellt, betreibt und kundenspezifisch anpasst. Das Produktmanagement spezifiziert jeweils im Detail die neuen Versionen der Software und schätzt den Gesamtaufwand ab, inkl. Implementierung, Anpassung der Dokumentation, Erstellung der Schulungsunterlagen, Änderung der Marketingunterlagen etc. Das Management nimmt die Spezifikation ab und priorisiert diese.

Durch den hohen Aufwand für die Erstellung und Freigabe der Spezifikation vergehen meist mehr als sechs Monate von der Entstehung einer Idee bis zum entsprechenden Produktrelease. Für die Kunden ist dies aber in der Regel zu lang. Aus diesem Grund implementiert SuperArchiv beim Kunden vor Ort spezifische Lösungen, die sich kurzfristig umsetzen lassen.

So benötigt z.B. ein Kunde eine Anbindung der Userverwaltung an den dort vorhandenen Active Directory Server. Die Integration dieses Features in das Produkt wäre enorm zeitaufwändig, da das Produktmanagement verlangt, dass dann die Benutzerverwaltung über beliebige LDAP-Server möglich sein soll. Also entwickelt SuperArchiv eine individuelle Lösung für den Kunden. Sofern dieses Feature in die folgende Produktversion integriert wird, könnten beim nächsten Releasewechsel für den Kunden wiederum zusätzliche Kosten entstehen, weil davon auszugehen ist, dass es inkompatibel mit der kundenspezifischen Implementierung ist.

Der Wunsch, die Zeit von der Entstehung einer Idee bis zum fertigen Release zu verkürzen, ist für SuperArchiv einer der Gründe zu untersuchen, ob nicht das bisher stark dokumentenlastige Vorgehen durch ein agiles Vorgehen abgelöst werden kann. So fällt schließlich die Entscheidung, dass die Abteilungen "Produktentwicklung" und "Kundenprojekte", die bisher nach dem Wasserfall-Modell arbeiteten, künftig gemäß Scrum vorgehen sollen. Weil die häufigen Produkt-Releases vermarktet und in Betrieb genommen werden müssen, hat die Einführung von Scrum ebenfalls Auswirkungen auf die Bereiche Marketing und Vertrieb sowie auf den Systembetrieb.

Daher beschließt die Geschäftsführung von SuperArchiv Scrum von oben einzuführen und initiiert ein strategisches Unternehmensprojekt, dessen Ziel die schrittweise Definition eines auf die Organisation abgestimmten agilen Vorgehens einschließlich seiner Einführung ist. Das Projektteam setzt sich aus dem Geschäftsführer, dem technischen Leiter, dem Leiter der Abteilung Kundenprojekte, dem Produktmanager, dem Leiter der Vertriebsabteilung und einem externen Scrum-Coach zusammen.

Bevor es aber darum geht, die Einführung aktiv zu fördern bzw. dabei auftretende Hindernisse zu beseitigen, muss das Change-Projekt zunächst die Voraussetzungen für die Einführung und Etablierung des Vorgehensmodells schaffen.

Akzeptanz schaffen

Um eine hohe Akzeptanz für Scrum zu erreichen, muss sich die Unternehmensführung für die Einführung engagieren und den Wandel vorleben. Es bietet sich an, das Change-Projekt zur Einführung von Scrum agil durchzuführen, also ebenfalls in kurzen Iterationen. So gelingt es, kurzfristig auf Störungen zu reagieren und schon während des Veränderungsprozesses zu lernen, wie das neue Vorgehen am besten funktioniert. Vorteile lassen sich so schnell realisieren, was die Akzeptanz deutlich steigert.

SuperArchiv nutzt Scrum daher auch für das Change-Projekt und wählt eine Iterationslänge von zwei Wochen. Am Ende jeder Phase wird der Projektfortschritt geprüft und die Geschäftsleitung leitet gegebenenfalls Maßnahmen ein, um die Einführung von Scrum zu einem erfolgreichen Abschluss zu bringen. Zum Product Owner wird der CEO bestimmt, da das strategisch wichtige Projekt in diesem mittelständischen Unternehmen "Chefsache" ist. Scrum Master des Change-Projekt-Teams ist der externe Scrum-Coach. Er steht in keiner Hierarchiebeziehung zu den anderen Teammitgliedern und kann so dafür sorgen, dass das Change-Team produktiv ist.

Aufgaben des Change-Projekts

Das Change-Projekt verwaltet seine Aufgaben in einem Product Backlog. Zu den wesentlichen Aufgaben gehören:

- **Beseitigung von Störungen**
  Die wohl wichtigste Aufgabe des Change-Projekts besteht darin, das Vorgehen zu allgemeiner Akzeptanz zu führen und dabei auftretende Hindernisse zügig zu beseitigen, z.B. Ressourcenkonflikte oder Verständnisprobleme bezüglich der Scrum-Rollen.
- **Initiierung agiler Arbeitsweisen im Unternehmen**
  Dabei werden zunächst Pilotprojekte ausgewählt und das Vorgehen dort zu einer gewissen Reife geführt,

bevor es für alle Projekte ausgerollt wird. Diese Reife ist dann erreicht, wenn der Burndown in mehreren aufeinander folgenden Iterationen dem Verlauf der Ideallinie entspricht und der Product Owner im Review alle geplanten User Stories des Sprints abnimmt.

- **Ausbildungsaufgaben**
  Zukünftige Scrum Master, Product Owner und Teammitglieder müssen auf ihre neuen Rollen vorbereitet werden. Schulungen alleine reichen dafür nicht aus. Daher empfiehlt sich ein Coaching des Scrum Masters und des Product Owners während des Pilotprojekts. Diese können später selbst als interne Coaches im Unternehmen fungieren können. Häufig ist es sinnvoll, weitere direkt oder indirekt betroffene Mitarbeiter auf den Wechsel vorzubereiten. Dazu gehören einerseits die Teammitglieder, die lernen müssen, knapp dokumentierte User Stories anstelle von ausführlich spezifizierten Anforderungen in kurzen Iterationen umzusetzen. Andererseits sollten auch die Mitarbeiter aus den Abteilungen, die von der agilen Arbeitsweise nur mittelbar betroffen sind, in das neue Vorgehen eingewiesen werden, um so die Akzeptanz zu erhöhen.

- **Kommunikationsaufgaben**
  Da Veränderung oft zu Verunsicherung führt, kann die Einführung des neuen Vorgehens nur dann funktionieren, wenn alle Beteiligten vom Product Owner des Change-Projekts kontinuierlich über den Stand der Einführung auf dem Laufenden gehalten werden.

- **Anpassung des Berichtswesens**
  Scrum misst den Projektfortschritt mit einem Product Burndown Chart und kennt keine weiteren Berichte. Eine Neuausrichtung des Berichtswesens ist dementsprechend notwendig. Dabei gilt es ein einheitliches Format für Projektfortschrittsberichte im Unternehmen zu entwickeln, deren zentraler Bestandteil das Product Burndown Chart ist, das aber gleichzeitig die weiteren Anforderungen an das Reporting erfüllt. Denkbar wäre beispielsweise eine Ergänzung um den aktuellen Budgetverbrauch des Projekts.

- **Anpassung der Tool-Landschaft und Infrastruktur**
  Für eine kontinuierliche Integration sowie regelmäßige Deployments der iterativ entwickelten Software muss eine geeignete IT-Infrastruktur vorhanden sein.

Phasen bei der "Einführung von oben"

Die Einführung von Scrum im Unternehmen erfolgt in drei Phasen (Bild 1).

- **Phase 1: Analyse**
  Während der Analysephase wird der aktuelle Softwareentwicklungs-Prozess sowie die Rollenverteilung und die Schnittstellen zu den anderen Abteilungen untersucht. Daraus lassen sich Maßnahmen zur Einführung von Scrum ableiten sowie notwendige Voraussetzungen zur Einführung schaffen. Die Analysephase sollte möglichst kurz sein, da durch das iterative Vorgehen des Change-Projekts sichergestellt ist, dass Störungen frühzeitig erkannt und korrigiert werden können.

- **Phase 2: Pilotphase**
  Hier werden die in Phase 1 definierten Maßnahmen in ausgewählten Pilotprojekten umgesetzt und optimiert. Die Pilotphase ist dann beendet, wenn das Vorgehen in den Pilotprojekten etabliert ist und daraus ein unternehmensweites Vorgehen abgeleitet werden kann.

- **Phase 3: Etablierung im Unternehmen**
  Roll-out des Vorgehens im gesamten Unternehmen. Ggf. sind weitere organisatorische Veränderungen erforderlich, die in der Etablierungsphase durchgeführt werden.

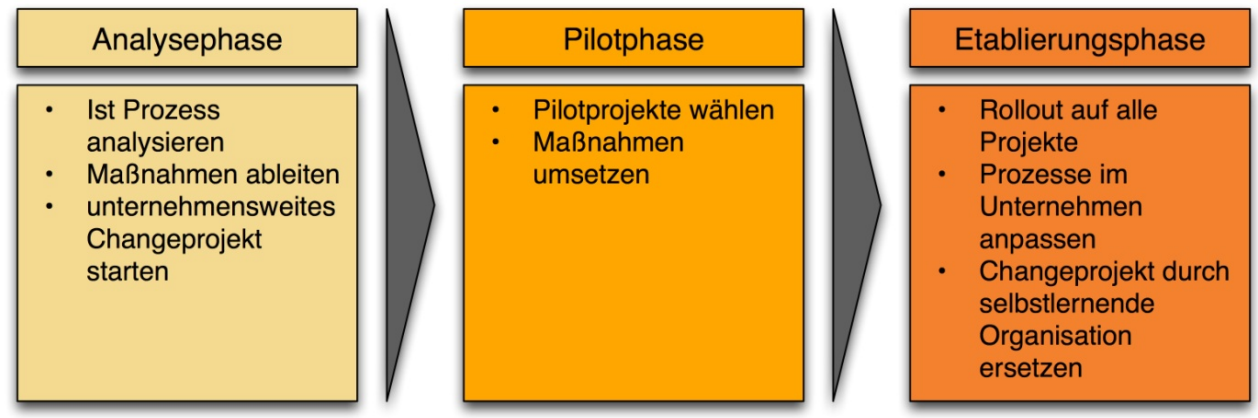

Bild 1: Phasen bei der "Einführung von oben".

Die Analysephase

Die Analysephase erlaubt es, einen grundlegenden Blick auf die Organisation zu werfen. Dies ist besonders wichtig, da ohne eine Analyse die Gefahr besteht, organisatorische Anpassungen zu unterschätzen.

Während der Analysephase geht es zunächst darum, eine Abbildung des aktuellen Rollenmodells auf die drei Scrum-Rollen Scrum Master, Product Owner und Team zu definieren. Dabei ist insbesondere zu überlegen, wie es gelingen kann, Vertreter der Fachabteilung als Product Owner in ein Projekt zu integrieren. Die einfachste Möglichkeit ist hierbei, einen kompetenten und entscheidungsbefugten Ansprechpartner des Fachbereiches direkt als Produkt Owner zu benennen. Dabei ist jedoch zu berücksichtigen, dass die Aufgabe sehr zeitintensiv sein kann – der entsprechende Mitarbeiter muss also von anderen Aufgaben freigestellt werden.

Gelingt es nicht, eine solche Person zu finden, oder kann sie nicht die notwendige Zeit aufbringen, kann ein Stellvertreter aus dem Fachbereich benannt werden. Dieser tritt gegenüber dem Team als Product Owner auf, entscheidet aber nur nach Rücksprache mit dem eigentlichen Verantwortlichen des Fachbereichs. Die dadurch entstehenden Verzögerungen werden sich aber mit hoher Wahrscheinlichkeit als Impediment in den Iterationen auswirken. Wie die Einbindung gelingen kann, muss daher immer anhand der konkreten Situation entschieden werden.

Rollen klären

Ebenso ist zu diskutieren, ob die bisherigen IT-Projektleiter in Zukunft Scrum Master sein können und sollen. In der Praxis hat es sich bewährt, die Rollen und Verantwortlichkeiten in Scrum in einer RACI-Matrix zu beschreiben. Eine solche Darstellung macht Zuständigkeiten sichtbar und offenbart eventuell vorhandene Rollenkonflikte.

Im Rahmen dieser Rollenklärung ist unbedingt zu klären, wer zukünftig die operativen Aufgaben des Projektmanagements übernimmt. Hierzu gehören z.B. das Erstellen von Reports, Berichten an das Steering-Board, Management von Lieferanten und Dienstleistern, Ressourcenmanagement und Risikomanagement. Häufig verfügt ein fachlich guter Product Owner, der Features definieren und dem Team erklären kann, nur über wenig Erfahrung mit diesen reinen Managementaufgaben. Kann der Product Owner nicht Vollzeit für das Projekt abgestellt werden, bleibt für diese Art der Aufgaben auch nur wenig Zeit. Im Zuge der Rollenklärung muss hierfür eine Lösung gefunden werden. Möglich ist es z.B., die Rolle des Product Owners auf mehrere Personen zu verteilen, die dann mit unterschiedlichen Aufgabenschwerpunkten arbeiten. Ebenso könnte ein Teammitglied bestimmt werden, das diese operativen Aufgaben übernimmt.

Im Bereich der Produktentwicklung entscheidet sich SuperArchiv dafür, den Produktmanager zum Product Owner für alle dort laufenden Projekte zu benennen, da er die Vision für das Produkt vorgibt. Vor der Einführung von Scrum wurden die drei im Produktentwicklungsbereich laufenden Projekte vom Entwicklungsleiter koordiniert. Dessen Tätigkeit bestand hauptsächlich darin, den Status der im Rahmen der Projektplanung vordefinierten Arbeitspakete zu überwachen. Die Rolle des Scrum Masters wird er künftig jedoch nicht übernehmen können, weil er aufgrund der verhältnismäßig hohen Anzahl von Scrum-Besprechungen überlastet wäre.

Stattdessen wird diese Rolle mit jeweils einem Entwickler aus dem jeweiligen Projektteam besetzt. Das Pilotprojekt wird vom Scrum-Coach unterstützt, der insbesondere in der Moderation der Besprechungen unterstützt, aber auch dabei hilft, den Arbeitsfortschritt anhand des Taskboards und des Burndown Charts zu überwachen. Im Bereich Kundenprojekte werden die Rollen auf eine andere Art besetzt: Scrum Master wird auch hier jeweils einer der Entwickler. Zum Product Owner wird der bisherige Accountmanager ernannt, weil er schon im bisherigen Vorgehen als Bindeglied zwischen Kunde und Entwicklung tätig war.

Selbstorganisation als oberstes Gebot

In Scrum geht es in jeder Iteration darum, als Team ein Ziel selbstorganisiert zu erreichen. Es zählt also weniger die Einzelleistung als die Teamleistung, gepaart mit dem Willen zum interdisziplinären Arbeiten, also der Bereitschaft auch fachfremde Aufgaben zu übernehmen.

So arbeiten in einem Projektteam der Firma SuperArchiv jeweils fünf bis sieben Entwickler, wobei jeder dieser Entwickler Experte auf mindestens einem Wissensgebiet ist. Dazu gehören Datenbankspezialisten ebenso wie Experten in der Entwicklung von Benutzeroberflächen. Enthält das Taskboard nun nur noch Datenbank-Aufgaben, helfen Spezialisten anderer Gebiete bei der Fertigstellung dieser Aufgaben aus, um das Iterationsziel zu erreichen.

Soll Scrum eingeführt werden, muss demnach festgestellt werden, wie weit die Fähigkeiten zu Selbstorganisation, eigenverantwortlichem Arbeiten und Planen bei den Projektmitarbeitern bereits ausgeprägt sind. Dies ist Aufgabe der Personalentwicklung, die mit Stärken-Schwächen-Analysen das Entwicklungspotential der Mitarbeiter identifizieren und geeignete Maßnahmen zur Ausprägung dieser Fähigkeiten ableiten muss.

Schnittstellen hohe Beachtung schenken

Scrum macht zwar Vorgaben für die projektinternen Prozesse, es gibt aber keine Antwort darauf, wie mit den Schnittstellen zu anderen Abteilungen umgegangen werden soll. Unternehmensintern sind insbesondere die Verzahnungen mit Marketing, Vertrieb und dem Systembetrieb relevant, weil diese Bereiche ihre Arbeitsweise an die kurzen Releasezyklen anpassen müssen. Besonderes Augenmerk gilt der Schnittstelle zur Qualitätssicherung, die nach jeder Iteration das entstandene Produkt-Inkrement gut getestet für den Produktivgang freigeben muss.

SuperArchiv entscheidet sich, diese Schnittstelle aufzulösen und die Tester in die Scrum-Teams zu integrieren, um so den Abstimmungsaufwand zu reduzieren. Gleichzeitig führt SuperArchiv Methoden der agilen Softwareentwicklung, wie etwa Test Driven Development und Unit Tests ein. Auf diese Weise gelingt es, trotz häufiger Code-Überarbeitung eine gleichbleibend hohe Software-Qualität sicherzustellen.

Unternehmen mit mehreren Standorten stehen vor der Frage, wie standortübergreifend durchgeführte Projekte organisiert werden sollen. Dabei ist zu klären, ob einem Projektteam nur Mitarbeiter eines Standorts zugeordnet werden sollen. Denn bei standortübergreifend arbeitenden Teams ist insbesondere die Organisation der Scrum-Besprechungen eine nicht zu unterschätzende Herausforderung.

Aber auch die Zusammenarbeit mit Kunden und Software-Lieferanten ist von der Umstellung auf Scrum betroffen. Falls das Produkt mit einem hohen Schulungsaufwand verbunden ist, wird ein Kunde nicht nach jedem Sprint ein Produkt-Update einführen wollen. Deshalb ist es in der Praxis sinnvoll, nicht jedes Produkt-Inkrement dem Kunden zu übergeben, sondern den Umfang mehrerer Iterationen zu bündeln und zu definierten Zeitpunkten auszuliefern. Ebenso wenig wird sich ein nicht agil arbeitender Lieferant auf die Lieferung von Produkt-Inkrementen einlassen, wenn es nicht seiner Arbeitsweise entspricht, regelmäßig Produkte in kurzen Zyklen zu liefern. Eine Lösung könnte auch hier die Lieferung von Teilreleases zu bestimmten Zeitpunkten sein, z.B. zu definierten Meilensteinen.

Die aus den Analyseergebnissen abgeleiteten Maßnahmen (Rollenklärung, Förderung selbstorganisierter Arbeit und Behandlung der Schnittstellen) zur Scrum-Einführung werden im Product Backlog des agilen Change-Projekts festgehalten.

Die Pilotphase

Die so identifizierten Maßnahmen werden zunächst in einigen ausgewählten Pilotprojekten umgesetzt, um das Vorgehen möglichst optimal an die Organisation anzupassen bzw. die notwendigen organisatorischen Änderungen vorzunehmen. Hier ist insbesondere die Wahl der Pilotprojekte von Bedeutung, da diese Multiplikatoren für den Erfolg des Vorgehens im gesamten Unternehmen sind. Um als Schablone für weitere Projekte dienen zu können, sollten auf der einen Seite die Pilotprojekte typisch für die Organisation sein, ohne mit einem allzu hohen Risiko behaftet zu sein. Sie sollten auf der anderen Seite dennoch eine gewisse Relevanz für das Unternehmen haben, um das Interesse am Projekterfolg sicherzustellen.

SuperArchiv bietet seit vielen Jahren das Produkt "SuperArchiv Desktop" für Windows erfolgreich am Markt an. Mit dieser Software lassen sich archivierte Dokumente anzeigen und durchsuchen, aber auch verändern. Nun

soll auch der wachsende Markt der Apple-Benutzer angesprochen werden und die Geschäftsführung möchte "SuperArchiv Desktop" ebenfalls für das Betriebssystem "MacOS X" anbieten. Auch wenn dieses Projekt sehr wichtig für die Firma SuperArchiv ist, würde eine kurze Verzögerung keinen großen Schaden bedeuten. Aus diesem Grund wird dieses Projekt als erstes Pilotprojekt für den Einsatz von Scrum bestimmt.

Weil die Einführung von Scrum einen Kulturwandel einläutet, empfiehlt es sich jedoch, behutsam vorzugehen und mit einer sehr einfachen Projekt-Konstellation zu starten, d.h. ein Projekt, das an genau einem Standort, mit einem einzigen Projektteam und möglichst wenigen Teammitgliedern durchgeführt wird. Erst wenn das Vorgehen für diese Konstellation etabliert ist, kann die Komplexität erhöht und Scrum in größeren Projekten, Projekten mit mehreren Teams oder sogar standortübergreifenden Projekten eingeführt werden.

Probleme erkennen und beheben

Das Change-Projekt hat in der Pilotphase die Aufgabe, Probleme mit dem Vorgehen so früh wie möglich zu erkennen und Maßnahmen zur Korrektur der Störung zu initiieren.

Das Projekt SuperArchiv Desktop für MacOS X etwa startet mit einer Sprintlänge von zwei Wochen. Die kurzen Releasezyklen implizieren einen hohen Integrations- und Testaufwand, so dass die Softwareentwickler kaum noch die Zeit zur Bearbeitung der Tasks auf dem Taskboard finden. Das zeigt sich deutlich an der Velocity des Teams, die stark von der geschätzten Team-Geschwindigkeit abweicht. Zur Behebung dieser Störung entscheidet man sich, die Sprintdauer auf drei Wochen zu verlängern.

In der Abteilung Kundenprojekte haben die Pilotprojekte eine deutlich höhere Komplexität, weil in den Projekten typischerweise Mitarbeiter des Kunden und der Firma SuperArchiv zusammenarbeiten. Daher führt SuperArchiv das agile Vorgehen zunächst lediglich in der Produktentwicklung ein. Als das Vorgehen in der Produktentwicklung stabil funktioniert, also die Iterationsziele regelmäßig in guter Qualität erreicht werden, soll Scrum in der Abteilung Kundenprojekte eingeführt werden.

Für ein entsprechendes Pilotprojekt kann ein langjähriger Kunde gewonnen werden, der drei seiner Mitarbeiter für das Projekt einsetzen möchte, die jedoch an ihrem jeweiligen Firmenstandort verbleiben sollen. Das Taskboard, ein Whiteboard mit Karteikarten, hängt zunächst im Büro von SuperArchiv. Ein Foto des Taskboards wird täglich vom Scrum Master an alle Teammitglieder per E-Mail verschickt. Der Verlauf des Burndown Charts ist allerdings von Anfang an weit von der Ideallinie entfernt. In der Retrospektive zeigt sich, dass die am Kundenstandort arbeitenden Mitarbeiter, die das Taskboard nicht kontinuierlich im Blick hatten, die anstehenden Aufgaben leicht aus den Augen verloren und projektfremde Tätigkeiten übernahmen. Zur Behebung dieser Störung wird kurzfristig ein digitales webbasiertes Taskboard eingeführt, auf welches das gesamte Projektteam Zugriff erhält.

Zur Problemerkennung stehen dem Veränderungsprojekt neben den Störungslisten, in denen die Störungen aufgeführt sind, auch die Burndown Charts und Risikolisten der Projekte zur Verfügung. Neben den organisatorischen Rahmenbedingungen ist, wie bereits erwähnt, in der Regel auch die Entwicklungsinfrastruktur anzupassen, so dass eine kontinuierliche Integration und häufige Deployments möglich sind.

Die Etablierungsphase

In der Etablierungsphase sind die organisatorischen Rahmenbedingungen für Scrum als unternehmensweites Vorgehen zu schaffen, so dass alle Projekte im Unternehmen gemäß dem in der Pilotphase definierten Vorgehen durchgeführt werden können. Manch ein Unternehmen wird darüber nachdenken müssen, im Rahmen der unternehmensweiten Einführung von Scrum seine Abteilungen inklusive der internen Schnittstellen neu zu organisieren.

So löst SuperArchiv den Bereich der Qualitätssicherung auf und integriert diesen in die Bereiche Produktentwicklung bzw. Kundenprojekte. Um die Bereiche Marketing und/oder Vertrieb ebenso wie den Systembetrieb in die iterative Vorgehensweise zu integrieren, stellt die Produktentwicklung nach dem Sprint Planning die Iterationsinhalte des aktuellen Sprints vor. So können diese Bereiche ihre Aktivitäten optimal auf das kommende Produktinkrement abstimmen. Möglicherweise werden sogar neue Auswahlkriterien für Zulieferer definiert. SuperArchiv nimmt sich z.B. vor, wenn es möglich ist, agil arbeitende Zulieferer auszuwählen.

Klare Trennung von Projekt- und Linientätigkeiten

In Scrum ist es üblich, dass Projektteam-Mitglieder keine weiteren Aufgaben außerhalb des Projekts übernehmen. Damit ist zwar eine recht zuverlässige Prognose der Velocity möglich, d.h. der Geschwindigkeit mit der das Projektteam seine Aufgaben bewältigt. Es hat aber auch zur Folge, dass projektfremde Tätigkeiten, etwa Aufgaben der Linienorganisation, nicht mehr nebenbei von denselben Personen bearbeitet werden können. Im Rahmen der Einführung von Scrum muss demnach ein Weg gefunden werden, der weder das Projekt noch die Aufgaben der Linienorganisation vernachlässigt.

Weil die Firma SuperArchiv die Archivierung von Dokumenten auch als Dienstleistung anbietet und eigene Archivierungsserver betreibt, werden immer wieder Entwickler benötigt, um Störungen im Betrieb zu beheben.

Zur Sicherstellung einer kontinuierlichen Entwicklungsgeschwindigkeit in den Projekten bei gleichzeitiger Gewährleistung eines funktionsfähigen Systembetriebs wählt SuperArchiv ein Rotationsprinzip: In jeder Iteration wird ein Mitarbeiter ausschließlich dafür eingeteilt, den Betrieb der Software zu gewährleisten und Störungen schnellstmöglich zu beseitigen.

Gegebenenfalls muss auch das Mitarbeiter-Bewertungssystem angepasst werden, insbesondere wenn es auf die Einzelleistung des Mitarbeiters ausgerichtet ist. Scrum honoriert den Teamerfolg, was das Bewertungssystem dann ebenfalls berücksichtigen sollte. Auch eine Ergänzung des Fortbildungsprogramms um Schulungen für Scrum Master und Product Owner ist empfehlenswert. Gleichzeitig sollte geklärt werden, ob klassische Zertifizierungsprogramme für Projektmanager, falls im Unternehmen etabliert, weiterhin zum Fortbildungsportfolio gehören sollen.

Vorteile bei der Einführung "von oben"

Ein wesentlicher Erfolgsfaktor für die Einführung von Scrum ist das Commitment der Unternehmensführung. Lebt sie das agile Vorgehen und insbesondere die entsprechenden Werte (s. oben) aktiv vor, hat dies einen positiven Einfluss auf die Akzeptanz von Scrum. Dabei kommt es auf den Mut an, Veränderungen zu initiieren, die für den

Erfolg von Scrum notwendig sind. Das kann z.B. die Einbindung einer Fachabteilung in das Projektteam in Form des Product Owners sein. Es kann aber auch bedeuten, den Umzug von Mitarbeitern in Erwägung zu ziehen, um Projektteams räumlich zusammenzuführen. Da all diese Veränderungen Unruhe in das Unternehmen bringen, steht und fällt die erfolgreiche Einführung mit einer offenen Kommunikation dessen, was gerade im Unternehmen passiert durch den Machtsponsor Unternehmensleitung.

Die Einführung von oben hat zudem den Vorteil, dass es auch die Koordination der Projekt-Schnittstellen begünstigt, da bei dieser Form der Einführung bereits das gesamte Unternehmen in den Veränderungsprozess involviert ist. So ließe sich mit den betroffenen internen Bereichen bereits zum Start des Change-Projekts ein Vorgehen zur Abstimmung innerhalb des Scrum-Prozesses festlegen. Das könnten regelmäßige Standup-Meetings sein, ebenso wie Verfahrens-Reviews nach jeder Iteration, in denen die bisherige Vorgehensweise überprüft und ggf. angepasst wird.

Unserer Erfahrung nach sollte auch die Zusammenarbeit mit Kunden und Lieferanten vorab definiert und möglicherweise sogar vertraglich festgeschrieben werden. Mit dem Kunden könnte beispielsweise vereinbart werden, dass er nach jedem Sprint am Review teilnimmt, so dass das Feedback in die Weiterentwicklung mit einfließen kann. Möglicherweise gelingt es, mit den Zulieferern Lieferungen von Teilergebnissen zu definierten Meilensteinen zu vereinbaren. Mit agil arbeitenden Zulieferern könnte man sich sogar auf iterativ-inkrementelle Lieferungen einigen. **Im Idealfall gelingt es,** Entwickler des Zulieferers soweit in das Projektteam zu integrieren, dass sie an den Daily Scrums teilnehmen.

Risiken bei der Einführung "von oben"

Wird Scrum von oben in die Organisation eingeführt, besteht die größte Gefahr darin, dass der für den Erfolg notwendige Kulturwandel mit der Einführung der neuen Prozesse und Rollen nicht Schritt halten kann. Dieses Problem kann sich dadurch äußern, dass das gewohnte Wasserfall-Denken nicht abgelegt wird. Entwickler zeigen dann möglicherweise wenig Bereitschaft, ohne eine ausführliche Konzeptphase mit der Umsetzung von Anforderungen zu beginnen, während Tester Probleme haben, ohne schriftlich fixierte Spezifikation Testfälle zu erstellen.

Diesem Problem begegnet SuperArchiv durch ein Coaching insbesondere der Product Owner. Sie lernen nicht nur die Formulierung von User Stories, sondern insbesondere, dass sie kontinuierlich mit den Softwareentwicklern und Testern kommunizieren müssen, um Anforderungen und Umsetzungswege zu klären.

Weisungsbefugnis abgeben

Weitere Risiken bei der Einführung von oben sind die Beibehaltung einer Befehlskultur, die Mitarbeiter passiv auf die Zuweisung von Aufgaben warten lässt sowie von außen erzwungene Team-Commitments, z.B. vom Product Owner. Commitments sind eigenverantwortliche Team-Entscheidungen; welcher Funktionsumfang in der nächsten Iteration erarbeitet wird, darf nicht durch Weisung von einer höheren Ebene bestimmt werden.

Die geschilderten Risiken sind darauf zurückzuführen, dass sich die Notwendigkeit zur Selbstorganisation nicht per Dekret verordnen lässt. Werden die bisherigen Projektleiter, die an die Erteilung von Befehlen gewohnt sind, zum Scrum Master ernannt, verstärken sich die Probleme üblicherweise noch. Denn erfahrungsgemäß fällt es

diesen Personen schwer, dem Team die Organisation selbst zu überlassen. SuperArchiv arbeitet deshalb mit einem Coaching-Konzept. Dabei moderiert der externe Scrum-Coach zunächst alle Scrum-Besprechungen einschließlich des Daily Standup Meetings. So lernen die neuen Scrum Master ebenso wie die Teams, wie Selbstorganisation in Projektteams funktionieren kann.

Hoher Erfolgsdruck

Auch die hohe Sichtbarkeit der Pilotprojekte im Unternehmen birgt Risiken. Sie führt unter anderem dazu, dass Probleme sofort publik werden. Hier könnte in der Organisation der Eindruck entstehen, dass das neue Vorgehen nicht funktioniert. Eine weitere Gefahr ist der Erfolgsdruck, der von Seiten der Geschäftsführung auf den Pilotprojekten lastet. Zur Demonstration eines guten Projektfortschritts könnten Product Owner nicht fertige Arbeitsergebnisse abnehmen oder Commitments ihrer Teams erzwingen, wodurch das agile Vorgehen konterkariert würde. Diese Entscheidungen (Commitments) sollen nach Scrum, wie bereits erwähnt, vom Team eigenständig und ohne Druck von außen getroffen werden.

Ausblick

Dieser erste Teil stellte die Einführung von Scrum in einem Unternehmen vor, wobei die Einführung "von oben" erfolgt. Das gesamte Unternehmen ist von der Veränderung informiert, die Geschäftsführung fungiert dabei als treibende Kraft. Die Ergebnisse des Veränderungsprojekts sowie der Pilotprojekte werden dabei offen im Unternehmen kommuniziert.

Anders verhält es sich bei der Einführung "von unten". Hier finden das Change-Projekt sowie die Pilotprojekte zunächst ausschließlich innerhalb der IT-Abteilung statt. Worauf man bei der Einführung von unten achten sollte und welche Vorteile sowie Risiken dieses Verfahren mit sich bringt, erfahren Sie im zweiten und abschließenden Teil.

Literatur

- Grant, Tom: From Agile Development To Agile Engagement, Forrester Research, 2009
- Pichler, Roman: Scrum – Agiles Projektmanagement erfolgreich einsetzen, dpunkt Verlag, 2008
- Roth, Katja; Lieder, Thomas: Agiles Projektmanagement – Häufige Stolperfallen in Scrum, Projekt Magazin 11/2010
- West, Dave; Grant, Tom: Agile Development: Mainstream Adoption Has Changed Agility, Forrester Research, 2010
- Wirdemann, Ralf: Agiles Projektmanagement. Scrum – eine Einführung, Projekt Magazin 21/2009

Fachbeitrag

Scrum im Unternehmen einführen

Teil 2: Einführung "von unten"

Autor

Thomas Lieder

Diplom-Informatiker, Certified Srum Master, Senior Project Manager bei der Setzwein IT-Management GmbH

Kontakt: thomas.lieder@setzwein.com

Mehr Informationen unter:
› projektmagazin.de/autoren

Autor

Katja Roth

Dipl.-Inform., PM-Fachfrau IPMA Level D, Certified Scrum Master, Certified Scrum Product Owner, Senior Project Manager bei der Setzwein IT-Management GmbH

Kontakt: katja.roth@gmx.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in den Rubriken:

› Agiles Projektmanagement
› Change Management
› PM im Unternehmen einführen

Bei der Einführung von Scrum in einer Organisation ist nicht nur die IT-Abteilung von der Veränderung betroffen. Auch Vertreter anderer Abteilungen müssen die Funktionsweise der Methode verstehen, damit der Software-Entwicklungsprozess produktiv eingesetzt werden kann. So ist es z.B. wichtig, dass Mitarbeiter anderer Fachbereiche ihre Rolle als Product Owner kennen. Abteilungen, wie z.B. der Systembetrieb, das Marketing oder der Vertrieb, sollten das Prinzip des iterativen Vorgehens kennen, um sich auf regelmäßig neue Produktversionen einstellen zu können.

Dieser zweiteilige Beitrag zeigt, worauf man bei der Einführung von Scrum in einer Organisation achten sollte. Der erste Teil behandelt die Einführung "von oben", bei der das gesamte Unternehmen über die Veränderung informiert ist und die Geschäftsführung dabei als treibende Kraft fungiert. Dieser zweite und abschließende Teil beschreibt, wie die Einführung "von unten" im Unternehmen realisiert werden kann.

Die Einführung "von unten"

Die Einführung "von unten" beschränkt sich ausschließlich auf die IT-Abteilung eines Unternehmens; das Change-Projekt sowie die Pilotprojekte werden nur in dieser Abteilung initiiert und durchgeführt. Die restliche Organisation wird dabei nicht über die Veränderung informiert, wodurch sich auch der Erfolgsdruck auf das Change-Projekt deutlich reduzieren lässt. Mit der Beschränkung auf nur einen Teil des Unternehmens besteht jedoch das Risiko, dass bei einem späteren "richtigen" Projekt in der Gesamtorganisation Probleme auftreten, die vorher nicht berücksichtigt wurden. Dadurch kann es passieren, dass zunächst eine Scheinsicherheit hinsichtlich der Durchführbarkeit von Scrum-Projekten entsteht.

Häufig kommt direkt aus der Entwicklungsmannschaft der Impuls, agile Methoden einzuführen. IT-Mitarbeiter lernen agile Vorgehensweisen über Bücher, Zeitschriften oder durch Berichte aus anderen Unternehmen kennen und es entsteht der Wunsch, in den eigenen Projekten ebenfalls so vorzugehen.

Kurze Analyse

Wie auch bei der Einführung "von oben" steht am Anfang eine kurze Analysephase. Dabei liegt der Fokus nicht darauf, alle denkbaren Probleme im Vorfeld zu lösen, sondern sich auf das neue Verfahren einzulassen. Da die Erfahrung zeigt, dass die wesentlichen Probleme erst bei der konkreten Durchführung auftreten, empfiehlt es sich, die Analyse kurz zu halten und möglichst schnell mit dem ersten Pilotprojekt zu starten. Die Analysephase sollte daher nicht länger als vier Wochen dauern. Im Wesentlichen werden in dieser Phase nur das erste Pilotprojekt und die dafür benötigten Rollen festgelegt. Die nachfolgende Pilotphase endet jedoch nicht zwingend mit dem Schluss des ersten Pilotprojekts, sondern kann noch weitere Pilotprojekte nach sich ziehen.

Bei der Einführung "von unten" sollte aber nicht nur das agil durchgeführte Software-Projekt im Mittelpunkt stehen; ebenfalls notwendig ist es, ein internes Change-Projekt voranzutreiben. Ziel sollte es sein, während der Projektlaufzeit möglichst viel über das agile Vorgehen im eigenen Unternehmen zu lernen und Hindernisse aus dem Weg zu räumen.

Entscheidungsbefugnis als wichtiger Faktor

Generell ist natürlich darauf zu achten, wer im Unternehmen die Entscheidung treffen darf, ein neues Vorgehensmodell einzuführen. Wenn es möglich ist, ein Veränderungsprojekt vollständig innerhalb eines kleinen Teams abzuwickeln, dann kann die Entscheidung des Teamleiters für die Einführung ausreichend sein, sofern dieser dazu berechtigt ist. Andernfalls muss eine Person diese Entscheidung treffen, die aufgrund ihrer Position in der Hierarchie dazu berechtigt ist.

Wichtig ist, dass diese Instanz gegenüber denjenigen Mitarbeitern, die an der Scrum-Einführung beteiligt sind, entscheidungsbefugt ist. Tauchen Probleme auf, welche die Einführung von Scrum stören könnten, muss diese Führungskraft ihren Mitarbeitern "den Rücken freihalten" und die notwenigen Entscheidungen treffen, damit sich die Beteiligten ganz auf die Einführung konzentrieren können.

Agile Durchführung des Change-Projekts

Wie bei den Pilot-Projekten empfiehlt es sich, auch das Change-Projekt iterativ und nach agilen Prinzipien durchzuführen. So kann das Team regelmäßig den Verlauf des Change-Projekts überprüfen, Störungen und Probleme identifizieren und sich über Verbesserungspotenziale austauschen. Das Backlog des Change-Projekts setzt sich dabei zusammen aus den notwendigen organisatorischen Änderungen, die für eine erfolgreiche Einführung wesentlich sind sowie aus den Problemen (Impediments), auf die das Change-Projekt im Laufe der Zeit stößt.

Es empfiehlt sich, für das Projekt eine Rolle ähnlich der des Scrum Masters zu etablieren. Der Inhaber dieser Rolle stellt eine direkte Verbindung zwischen den Projekten, in denen die Probleme auftreten, und dem Change-Projekt her und achtet darauf, dass die auftretenden Hindernisse priorisiert und die jeweils wichtigsten Punkte auch gelöst werden. Damit alle Beteiligten des Change-Projekts die notwendigen Änderungen mittragen, sollte das Team die Entscheidungen möglichst einvernehmlich treffen. Lässt sich kein Konsens finden, muss eine Möglichkeit bestehen, dennoch eine Entscheidung herbeizuführen, z.B. in dem die Entscheidung an die nächsthöhere Führungsebene weitergegeben wird.

Das Change-Projekt wird zunächst nur in einem kleinen Kreis in der IT-Abteilung gestartet. In der Pilotphase wird es dann nach Bedarf schrittweise ausgeweitet. Die Etablierungsphase beginnt schließlich, wenn die Einführung von Scrum nicht mehr nur auf die IT-Abteilung beschränkt ist, sondern das gesamte Unternehmen in die Einführung mit einbezogen wird. Erst wenn Scrum in allen betroffenen Abteilungen erfolgreich umgesetzt ist und verstanden wird, kann man die Einführung als abgeschlossen betrachten.

Darüber hinaus besteht nach der Einführung auch weiterhin Optimierungsbedarf. Die Verantwortlichen sollten regelmäßig überprüfen, ob das definierte Vorgehen innerhalb der Organisation noch angemessen ist. Denn gerade im Sinne des kontinuierlichen Lernens ist die Arbeit an einer agilen Vorgehensweise nie abgeschlossen.

Beispiel

Die ACME Ltd. ist ein international agierender Versandhändler, der viele Bestandteile seiner Warenwirtschaftssysteme selber entwickelt und wartet. Die Fachbereiche initiieren Projekte und beschreiben diese in detaillierten Anforderungsdokumenten. In der Vergangenheit kam es allerdings immer wieder zu Verzögerungen in den Projekten und die Entwickler setzten Funktionen nicht so um, wie sich die Fachabteilungen diese vorgestellt hatten. Folglich wurden die Spezifikationen immer detaillierter. Dokumente mit mehr als 500 Seiten waren keine Seltenheit mehr und enthielten häufig auch Datenmodelle und Pseudo-Code.

Um die Kosten der IT-Abteilung zu deckeln, ist diese seit kurzem aufgefordert, anhand der Anforderungsdefinition Festpreisschätzungen abzugeben. Die Projekte stehen darüber hinaus unter einem hohen Termindruck. Die Entwickler konzentrieren sich daher immer mehr darauf, nur genau das umzusetzen, was gefordert wurde – auch wenn sie während der Implementierung bereits merken, dass es später zu Problemen kommen wird.

Die IT-Mitarbeiter schätzen die Gefahr, selbst die Verursacher der Verzögerungen und Budgetüberschreitungen zu sein, bald als sehr hoch ein. Aus diesem Grund entsteht innerhalb der IT-Abteilung der Wunsch, zu einer neuen Aufgabenteilung zwischen den Fachbereichen und der IT zu kommen: Die Fachabteilung nennt die Anforderungen und gibt das fachliche Ziel vor, die IT definiert selbst die sinnvollste technische Lösung. Einige Entwickler äußern daher den Wunsch, agile Verfahren einzusetzen.

Phasen bei der "Einführung von unten"

Bild 1 zeigt einige Aufgaben, die in den verschiedenen Phasen bei der Einführung "von unten" durchgeführt werden müssen.

Die Analysephase

Die Analysephase dient dazu, die Rahmenbedingungen für das erste agil durchgeführte IT-Projekt festzulegen. Mindestens die folgenden Punkte sollten dabei diskutiert werden:

- **Rollendefinition**
  Scrum sieht neben dem Team nur die Rollen des Product Owners und des Scrum Masters vor. Findet das erste Pilotprojekt ausschließlich innerhalb der IT-Abteilung statt, steht ein "echter" Product Owner, d.h. ein

entscheidungsbefugter Vertreter der Fachabteilung, in der Regel nicht zur Verfügung. Es gilt also, hierfür einen Ersatz zu finden. Für das Pilot-Projekt kann ein fachlich versierter Entwickler die Rolle des Product Owners übernehmen. Das bedeutet, dass er sich nicht direkt an der Entwicklung beteiligen darf und somit auch nicht gleichzeitig Mitglied des Scrum-Teams sein kann. Ebenso ist die Rolle des Scrum Masters zu besetzen.

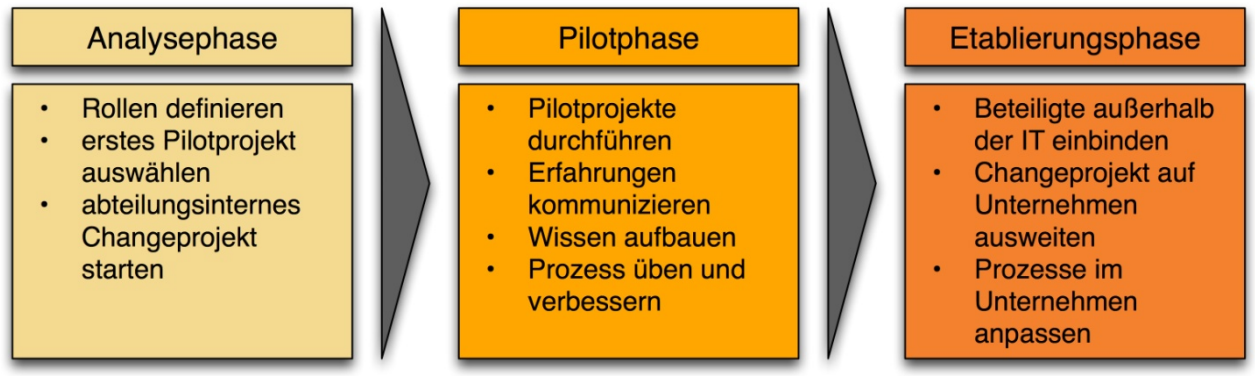

Bild 1: Phasen bei der Einführung des Scrum-Prozesses "von unten".

- **Wissensaufbau**
  Das Projektteam des Change-Projekts muss prüfen, ob die beteiligten Personen über genug Wissen zu agilen Vorgehensweisen verfügen, um das Pilotprojekt durchzuführen. Sollte kein ausreichendes Wissen vorhanden sein, kann in der Analysephase entschieden werden, Mitarbeiter zum Scrum Master oder Product Owner zu zertifizieren, um entsprechendes Wissen aufzubauen. Solche Kurse dauern in der Regel nur zwei oder drei Tage. Ist innerhalb des geplanten Projektteams nur theoretisches Wissen zu Scrum vorhanden, kann man darüber hinaus überlegen, einen Scrum-Coach hinzuzuziehen.

- **Projekt auswählen**
  Für die Pilotphase muss ein geeignetes Projekt gefunden werden. Es sollte sich hierbei um ein echtes Projekt für einen echten Kunden handeln, den ein IT-interner Product Owner vertreten kann. Das Projekt wird ausschließlich innerhalb der IT-Abteilung mit nur einem Scrum-Team durchgeführt. Möchte das Team eine rein interne Aufgabe der IT-Abteilung verwenden, besteht die Gefahr, dass das Projekt zu sehr den Charakter eines Spiels bekommt, wenn nicht der Druck eines echten Projekts und die Anforderungen eines echten Kunden vorhanden sind.

- **Change-Projekt definieren**
  Neben dem ersten Scrum-Projekt muss auch ein Change-Projekt initiiert werden, das die Änderung der Rahmenbedingungen steuert. Im Change-Projekt-Team sollten Personen vertreten sein, die Entscheidungsbefugnis haben und Konflikte lösen können. Denn in der Regel treten die ersten Probleme an der Grenze zwischen dem Pilotprojekt und seinem direkten Umfeld auf, z.B. zwischen einem Programmierer des Pilotprojekts und einem IT-Mitarbeiter, der nicht am Pilotprojekt beteiligt ist, aber Zuarbeiten seines Kollegen benötigt.

- Die Analysephase kann das Team des Change-Projekts auch dazu nutzen, ein initiales Backlog für das Change-Projekt zu erstellen. Dabei muss man jedoch berücksichtigen, dass sich in der konkreten Umsetzung

häufig andere Dinge als wichtig erweisen, als vorher vermutet wurde. Ein Backlog, das vor dem Start des Pilot-Projekts erstellt wird, ist daher Gegenstand häufiger Änderungen.

Sind die Rahmenbedingungen festgelegt, kann das Team mit dem ersten Pilotprojekt beginnen.

Die Pilotphase

Die Pilotphase startet mit dem ersten agil durchgeführten Projekt. Dieses Projekt sollte hinreichend anspruchsvoll sein, sodass es den Charakter eines "echten" Projekts hat. Es bietet sich an, ein Projekt zu wählen, dass ein Team mit fünf bis sieben Entwicklern an einem Standort durchführen kann. Die geplante Projektlaufzeit sollte dabei zwischen drei und sechs Monaten liegen. Teamgröße und Zeitdauer sollte man dabei so wählen, dass sich das Projektziel auch erreichen lässt.

Anders herum gesagt: Wird ein Pilotprojekt von vornherein unter einem zu engen Zeitrahmen gestartet, lastet man das absehbare Scheitern auch dem neuen Prozess an; die Einführung agiler Methoden wird damit deutlich erschwert. Aus diesem Grund ist es wichtig, darauf zu achten, ein Projekt auszusuchen, dessen Komplexität gering ist, damit der neue Prozess nicht unter zu vielen Störungen leidet und die Zielerreichung realistisch ist.

Erfahrene Entwickler einsetzen

Um technische Probleme zu vermeiden, sollte das Projektteam aus sehr erfahrenen Entwicklern bestehen, die gegenüber agilen Methoden offen sind. Günstig für die Teamarbeit ist es, wenn jedes Teammitglied auch Interesse an Themen außerhalb seines Fachgebiets hat. So lässt es sich vermeiden, dass Sprintziele aufgrund fehlender technischer Expertise nicht erreicht werden und dies Scrum angelastet wird.

Bei ACME Ltd. ist innerhalb der IT-Abteilung der Wunsch entstanden, ein agiles Verfahren auszuprobieren. Es wird ein Workshop mit Experten organisiert und das weitere Vorgehen definiert. Dabei wird beschlossen, ein erstes Projekt mit Scrum durchzuführen. Gegenstand des Projekts ist die Ablösung eines regelbasierten Datenimport- und exportmoduls. Dies wurde ursprünglich in Perl geschrieben und soll nun durch eine Java-Lösung ersetzt werden.

Um sich im Projekt auf das Erlernen des Prozesses konzentrieren zu können, wird das Projektteam aus technischen und fachlichen Spezialisten zusammengesetzt. Hierzu gehören z.B. die Perl-Entwickler, die seit mehreren Jahren die bestehende Schnittstelle warten und weiterentwickeln, sowie erfahrene Java-Entwickler.

Störungen erkennen und beseitigen

Parallel zu den im Unternehmen etablierten Berichtswegen sollte für das Pilotprojekt eine direkte Kommunikation zwischen dem Scrum Master und dem entsprechenden IT-Abteilungsleiter aufgebaut werden. Nur so kann der Scrum Master auftretende Probleme schnell adressieren und Ursachen beseitigen. Es ist Aufgabe des Change-Projekts, diese Änderungen voranzutreiben und festzuhalten.

Ein typisches Beispiel ist die Störung des Scrum-Teams durch Supportanfragen, z.B. aus zuvor entwickelten Systemen. So verständlich diese Anfragen sind, sie stellen dennoch eine Unterbrechung der Teamarbeit dar und führen bei häufigem Auftreten dazu, dass Sprintziele nicht erreicht werden können. Je nach konkreter Situation kann das Team des Change-Projekts verschiedene Lösungswege erarbeiten. Die Szenarien reichen von einem Untersagen der Anfragen, über das Einrichten spezifischer "Sprechzeiten" bis zum Aufbau dedizierter Supportteams. Welche Lösung die passendste ist, muss immer im Kontext des laufenden Pilotprojekts durch das entsprechende Change-Projekt entschieden werden.

In der ACME Ltd. arbeitet das Team bereits an der fünften Iteration. Doch die anfängliche Euphorie ist inzwischen verflogen, da in den letzten beiden Iterationen das jeweilige Iterationsziel nicht erreicht werden konnte. In der Retrospektive stellt sich heraus, dass die Perl-Entwickler neben der Arbeit im Projekt immer wieder auch an Problemen und Erweiterungen der alten Schnittstelle arbeiteten. Die Arbeit außerhalb des Projekts hat in den letzten Wochen stark zugenommen, so dass nicht alle User Stories umgesetzt werden konnten.

Bei einer genaueren Analyse fällt allerdings auf, dass nur wenige der gemeldeten Probleme wirklich kritisch waren – häufig handelte es sich um kleine Erweiterungen. Um das Problem zu lösen, wird beschlossen, die eingehenden Anfragen zunächst zu kategorisieren und nur die kritischen Probleme sofort zu beheben. Für diese werden dann entsprechende Taskkarten angelegt, um nach dem Sprint ermitteln zu können, wie viel Mehraufwand hierdurch verursacht wurde. Dies kann das Team dann in den Planungen für die nächsten Iterationen berücksichtigen. Alle anderen Aufgaben werden in das Backlog des agilen Projekts aufgenommen und priorisiert. Mit diesem Vorgehen lassen sich die Verzögerungen im agilen Projekt durch Erweiterungen an der alten Schnittstelle sichtbar machen und es wird sichergestellt, dass solche funktionalen Erweiterungen gleich in die neue und nicht die alte Schnittstelle integriert werden.

Mehrere Projekte agil durchführen

Wurde das erste Projekt erfolgreich abgeschlossen, gilt es, die gewonnene Erfahrung zu verteilen und zu vertiefen. Dies gelingt am besten, wenn die Mitglieder des Pilotteams jeweils den Kern der neuen Teams bilden. Denn im nächsten Schritt können anstatt eines Projekts bereits zwei oder drei Projekte gleichzeitig agil arbeiten. Dafür kann das alte Team in zwei oder drei etwa gleich große "Subteams" aufgeteilt und mit neuen Teammitgliedern versehen werden.

Da nun mehrere Teams parallel nach Scrum arbeiten, ist zu erwarten, dass bisher verdeckte Ressourcenkonflikte zu Tage treten. Auch die Entwicklungsinfrastruktur, d.h. Entwicklungssysteme, Testrechner, Sourcecode-Verwaltung usw., wird durch die kontinuierliche Integration und häufige Tests wesentlich stärker belastet. Das begleitende Change-Projekt muss dafür sorgen, diese Probleme zu lösen. In der Praxis hat es sich häufig als nützlich erwiesen, die Verfahren weiter zu automatisieren und Hardware-Ressourcen zu erweitern.

Erste Erfolge kommunizieren

Parallel zur Projektarbeit gilt es auch, erste Erfolge zu kommunizieren. Mitglieder des Pilotteams können z.B. ihr Projekt in abteilungsinternen Meetings vorstellen und dabei Erfolge, aber auch – hoffentlich gelöste – Probleme

thematisieren. Durch solche Treffen sowie durch die informelle Kommunikation zwischen den Kollegen verbreitet sich das Wissen schließlich über die ersten agilen Projekte aus der IT-Abteilung hinaus in andere Bereiche des Unternehmens und kann dort ein Interesse an agilen Methoden wecken. Im Idealfall löst dies in anderen Abteilungen den Wunsch aus, in zukünftigen Projekten ebenfalls agile Methoden einzusetzen.

Die Etablierungsphase

Die Pilotphase endet im Prinzip, sobald sich Scrum in der IT-Abteilung etabliert und als praktikabel erwiesen hat und auch in anderen Bereichen des Unternehmens ein ernsthaftes Interesse an agilen Methoden besteht.

Die Einführung verlässt damit ihr "Versteck" in der IT-Abteilung und tritt hinaus in das "Rampenlicht" des Unternehmens. Einerseits ermöglicht dies, jetzt Fachleute aus den anderen Abteilungen als Product Owner zu gewinnen, da sich durch die offizielle Einbeziehung des gesamten Unternehmens auch Änderungen außerhalb der IT-Abteilung vornehmen lassen. Andererseits ist nun auch mit dem Änderungswiderstand einer großen Organisation zu rechnen: Nicht jeder Beteiligte wird agilen Methoden offen gegenüber stehen. Einzelne mögen sogar ein Scheitern der neuen Methode bevorzugen. Dem Risiko, dass Scrum am Widerstand des gesamten Unternehmens scheitert, kann man am besten begegnen, wenn die Einführung ohne übertriebene Erwartungen als Chance kommuniziert wird, die gegenwärtigen Prozesse zu verbessern.

Das Change-Projekt ausweiten

Um Scrum unternehmensweit zu etablieren, muss das Change-Projekt auf die gesamte Organisation ausgedehnt werden. Dafür ist die Unterstützung durch das obere Management notwendig. Dieses erweiterte Change-Projekt hat die Aufgabe, die organisatorischen Probleme zu lösen, die bei der Durchführung agiler Projekte entstehen können. Es entspricht damit dem Change-Projekt, das bei der Einführung "von oben" von Beginn an vorhanden ist. Um den Umgang mit Scrum im gesamten Unternehmen zu üben, empfiehlt es sich, mit einem Softwareprojekt zu starten, das ein einziges Scrum-Team – höchstens jedoch zwei Scrum-Teams – umsetzen kann (zur Skalierung von Scrum, siehe z.B. Pichler, 2008).

Die Scrum Master und Scrum-Teams sollten dabei bereits über Erfahrung mit dem Scrum-Prozess verfügen und bereits mehrere agile Projekte erfolgreich durchgeführt haben. Die weitere Verbreitung von Scrum in das Unternehmen findet dann durch die Product Owner der jeweiligen Fachbereiche statt. Dabei ist darauf zu achten, dass der Product Owner gegenüber dem Scrum-Team mit nur einer Stimme sprechen darf. Sind also mehrere Vertreter unterschiedlicher Fachbereiche involviert, die ggf. noch verschiedene Meinungen vertreten, so müssen diese geklärt werden, bevor mit dem Team kommuniziert wird.

Bei der ACME Ltd. sind mittlerweile mehrere Projekte erfolgreich mit Scrum durchgeführt worden. Auch außerhalb der IT-Abteilung entsteht vermehrt ein Interesse an der Methode. Nach einigen Workshops beschließt die Geschäftsführung, ein agiles IT-Projekt mit direkter Beteiligung der Fachbereiche durchzuführen. Es soll sich dabei um ein neues System für den Online-Shop handeln. Schnell wird klar, dass Anforderungen aus mehreren Fachbereichen berücksichtigt werden müssen. So haben der Einkauf, das Marketing und auch die Kundenbetreuung viele und zum Teil widersprüchliche Anforderungen.

Ein einzelner Product Owner kann daher nicht gefunden werden. ACME bildet aus diesem Grund ein Product-Owner-Team aus Stellvertretern der einzelnen Fachbereiche. Dieses Team sichtet und bewertet alle Anforderungen und priorisiert diese im Backlog. Für jede Iteration wird im Product-Owner-Team ein Ansprechpartner für das Scrum-Team bestimmt. Dem Scrum-Team bleibt damit der komplexe Prozess der Anforderungspriorisierung verborgen und es arbeitet wie üblich mit einem Product Owner.

Der Product Owner und weitere Stakeholder benötigen nicht zwingend eine hohe Erfahrung mit Scrum. Ggf. können Coaches sie in einem Scrum-Projekt unterstützen. Weitere Schnittstellen des Projekts, z.B. zu externen Zulieferern sollten minimiert werden, um die Gesamtkomplexität des Projekts überschaubar zu halten. Das erste "große" Projekt muss nicht so einfach sein, wie beim Vorgehen "von oben" vorgeschlagen, die Herausforderungen sollten aber dennoch überschaubar sein.

Auf gute Kommunikation achten

Zudem ist es bei der unternehmensweiten Einführung wichtig, der Kommunikation viel Aufmerksamkeit zu schenken. Der Projektstatus der Software-Projekte und des Change-Projekts, das Wissen über agile Methoden sowie die gewonnen Erfahrungen sollten regelmäßig im Unternehmen verbreitet werden.

Um den agilen Werten "Offenheit" und "Vertrauen" gerecht zu werden, können die Projektbeteiligten in der Mitarbeiterzeitschrift regelmäßig über die Fortschritte der agilen Projekte informieren. Dort können sie auch über die bereits durchgeführten organisatorischen Änderungen berichten, wie z.B. über die direkte Zuordnung der QA-Mitarbeiter zu den Entwicklungsteams oder die enge Zusammenarbeit zwischen den Key-Account-Managern und den Product Ownern. Das Backlog und der Burndown-Chart der Projekte sind darüber hinaus jederzeit im Intranet für alle einsehbar.

Erst wenn die Organisation agile Methoden nicht mehr als Ausnahme ansieht und die Unternehmenswerte agile Vorgehensweisen unterstützen (s. Teil 1), kann die Einführung als erfolgreich abgeschlossen betrachtet werden.

Vorteile und Risiken bei der Einführung "von unten"

Für eine Einführung "von unten" spricht, dass nach einer kurzen Analysephase schnell mit der Umsetzung begonnen werden kann. Da nicht sofort das gesamte Unternehmen an das neue Vorgehen angepasst werden soll, sind die Auswirkungen der Einführung zunächst überschaubar. In der Regel lässt sich ein einzelnes Projekt innerhalb der IT-Abteilung recht gut von störenden Einflüssen abschotten, sodass ein Projekterfolg einfach erzielbar ist.

In dieser "Isolierung" steckt aber auch eine Herausforderung: Es gilt, ein Pilotprojekt zu finden, das auch abgeschottet durchführbar ist. Für organisatorische Schnittstellen, z.B. zu Vertrieb, Marketing, Qualitätssicherung oder Systembetrieb, müssen pragmatische Lösungen gefunden werden. Diese können zu Lasten der reinen Scrum-Lehre gehen, z.B. wenn zunächst keine Ansprechpartner aus den Fachabteilungen zur Verfügung stehen. Hat man aber eine entsprechende Kapselung gefunden, ermöglicht die Einführung "von unten" ein Lernen und Ausprobieren des neuen Prozesses nahezu ohne öffentlichen (Erfolgs-)Druck.

Da bei ACME ein Großteil des Geschäfts über den Internet-Shop abgewickelt wird, führt ein Ausfall der Systeme sofort zu einem Einnahmeverlust. Der ACME Systembetrieb, d.h. die Abteilung, die für den Betrieb der Serversysteme zuständig ist, stellt daher hohe Anforderungen an ein Release und lässt Änderungen an produktiven Systemen nur einmal je Quartal zu. Die jeweiligen Iterationsergebnisse der agilen Projekte werden daher am Ende der Iteration zunächst nicht sofort in die Produktivumgebung eingespielt, sondern in einem separaten System gesammelt und kumuliert einmal im Quartal produktiv gesetzt.

Risikofaktor "Product Owner"

Hat sich Scrum innerhalb der IT-Abteilung bewährt, folgt im nächsten Schritt der Sprung in die "Unternehmensöffentlichkeit". Dieser Schritt birgt das größte Risiko bei der Einführung "von unten". Denn erst jetzt können die organisatorischen Probleme in der Gesamtorganisation zu Tage treten. Es besteht die Möglichkeit, dass diese Hindernisse unterschätzt wurden.

So ist es zwar möglich, innerhalb der IT-Abteilung mit dem Ersatz eines "echten" Product Owners erfolgreich die ersten Projekte durchzuführen. Soll aber z.B. für eine bestimmte Fachabteilung eine Softwarelösung entwickelt werden, verfügt dieser Ersatz in der Regel nicht über spezifische Kenntnisse der notwendigen Anforderungen. Er kann deswegen auch keine für die Praxis wichtigen Entscheidungen hinsichtlich der Umsetzung der Funktionen treffen. Zwar kann er Details zu den Funktionen klären und auch die Reihenfolge der Umsetzung definieren, er ist in der Regel aber nicht entscheidungsbefugt, um Funktionen zu streichen oder hinzuzufügen.

Erst wenn Projekte unternehmensweit durchgeführt werden, kann sich auch ein "richtiger" Product Owner an den Projekten beteiligen. Bei Projekten, die mehrere Fachabteilungen betreffen, kann es aber eine Herausforderung sein, eine Person zu finden, die den Gesamtumfang des Projekts fachlich durchdringt und außerdem für das Gesamtprojekt entscheidungsbefugt ist. In einem solchen Fall ist das bei der ACME Ltd. verwendete Product-Owner-Team eine mögliche Lösung.

Erfolgsfaktoren und Fallstricke

Egal ob die Einführung "von oben" oder "von unten" erfolgt – es gibt für beide Varianten gemeinsame Erfolgsfaktoren und Fallstricke. Diese sind besonders auf der Ebene der Unternehmenskultur zu finden: Eine offene Unternehmenskultur in der Vertrauen und Respekt vorherrscht und die von einer offenen Feedback-Kultur geprägt ist, sind die wichtigsten Erfolgsfaktoren für eine Einführung agiler Methoden. Am einfachsten lässt sich die Einführung umsetzen, wenn die Mitarbeiter von sich aus diese Veränderung wünschen. Im Idealfall lässt sich das durch eine offene Kommunikation erreichen, die Scrum als interessant und vorteilhaft darstellt.

Doch auch nachdem die Einführung zunächst erfolgreich abgeschlossen wurde, lauern in der täglichen Arbeit mit Scrum eine Reihe von Stolperfallen. Diese gilt es zu erkennen und die Ursachen zu beseitigen (s. Lieder, Roth, Projekt Magazin 11/2010), um den Prozess am Leben zu erhalten. Für eine erfolgreiche Einführung muss z.B. berücksichtigt werden, dass es sich nicht nur um die Änderung eines Prozesses handelt, sondern häufig auch organisatorische Änderungen notwendig sind, wie ein neues Rollenverständnis der Verantwortlichen in den Fachabteilungen sowie bei Lieferanten oder Kunden. Der erfolgreiche Einsatz agiler Methoden setzt eine offene Unternehmenskultur voraus. Dazu

gehört z.B. ein ehrlicher Austausch untereinander, um den Prozess laufend zu verbessern. Die beteiligten Personen auf dem Weg dorthin mitzunehmen, ist häufig die größte Herausforderung bei der Einführung von Scrum.

Ideen und Werte berücksichtigen

Alle Beteiligten müssen die Ideen und Werte kennen und berücksichtigen, die hinter den in Scrum verwendeten Methoden stehen. Ansonsten lassen sich die Vorteile von Scrum in der Regel nicht realisieren. So ist es z.B. nicht ausreichend, vom "Daily Scrum" nur die Idee eines täglichen kurzen Informationsaustausches zu übernehmen. Es muss verstanden und gelebt werden, dass das Team im Daily Scrum Tagesziele definiert und deren Erreichung reflektiert. Wird "nur" berichtet, womit man sich beschäftigt hat und nicht analysiert, warum die gesteckten Ziele nicht erreicht wurden, ist es erfahrungsgemäß kaum möglich, vorhandene Hindernisse aufzudecken.

Die Veränderung benötigt Zeit

Die in der Regel notwendige Änderung der Organisation und Firmenkultur lässt sich nicht kurzfristig erreichen. Für eine erfolgreiche Einführung agiler Methoden sollte daher mindestens ein Zeitraum von ein bis zwei Jahren eingeplant werden. Das ist einerseits darauf zurückzuführen, dass es einige Zeit dauert, den agilen Prozess an die Organisation anzupassen, die Mitarbeiter zu schulen und das Verfahren in Pilotprojekten zu optimieren. Andererseits brauchen die Mitarbeiter Zeit, um sich an die Veränderung im Unternehmen zu gewöhnen und sich mit dem neuen Vorgehen zurechtzufinden. Schließlich verlangt die Einführung von Scrum in vielen Fällen von den Mitarbeitern, die über viele Jahre gewachsene Orientierung aufzugeben und durch neue Werte und Prozesse zu ersetzen. In erster Linie hängt die notwendige Zeit für die Einführung von der vorhandenen Unternehmenskultur ab und erst in zweiter Linie von der Größe der betroffenen Organisation.

In einer offenen Feedbackkultur gelingt es recht schnell ein funktionierendes agiles Verfahren zu etablieren. Probleme werden dabei im Sinne eines kontinuierlichen Verbesserungsprozesses in "Daily Standup Meetings" oder Retrospektiven offen angesprochen und zeitnah beseitigt; die Organisation lernt. In einer großen Organisation, die Scrum "von oben" einführt, werden die Erkenntnisse aus den Softwareprojekten im Change-Projekt zusammengetragen, um diese Erfahrungen zu multiplizieren. Dies kann z.B. die erfolgreiche Einführung von "Test Driven Development" an einem Standort des Unternehmens sein, das fortan auch in anderen Projekten eingeführt wird.

In einer Unternehmenskultur, die auf Hierarchie und "Command-and-Control" basiert, fehlt den Mitarbeitern hingegen in der Regel das Vertrauen, um auf Schwachstellen aufmerksam zu machen. Das Change-Projekt würde sozusagen "verhungern" und Lerneffekte bleiben aus. Eine Einführung von Scrum wäre somit unabhängig von der Unternehmensgröße problematisch.

Seine Rolle kennen

Organisatorische Änderungen kann in der Regel nur die Geschäftsleitung initiieren. Um entsprechende Anpassungen im Rahmen der Scrum-Einführung vornehmen zu können, muss die Geschäftsleitung hinter der Umstellung stehen und diese vorantreiben.

Nicht vernachlässigt werden darf auch die Aus- und Weiterbildung der Beteiligten. Insbesondere Product Owner und Scrum Master müssen mit den Aufgaben der jeweiligen Rolle vertraut sein. Eine Zertifizierung ist dabei sicherlich hilfreich, in der Regel aber alleine nicht ausreichend.

Scrum verändert alte Strukturen

Agile Methoden wie Scrum definieren lediglich eine "Architektur" für ein Vorgehen – sie geben jedoch nur wenige Anhaltspunkte für eine konkrete Implementierung im Unternehmen. Daher ist die erfolgreiche Umsetzung von der Theorie in die Praxis schwierig. Viele Stolperfallen bei der Einführung und in der Umsetzung agiler Verfahren lassen sich vermeiden, indem man ausgewiesene Experten agiler Methoden hinzuzieht.

Nicht zuletzt sollte bedacht werden, dass durch die Einführung agiler Methoden die vorhandenen Rollen und damit entsprechende Karrierepfade verändert werden. Gibt es in einem Unternehmen z.B. einen Karriereweg vom Softwareentwickler über den Softwarearchitekten zum Projektleiter, so lässt sich dieser Karrierepfad nach der Einführung von Scrum nicht mehr aufrecht erhalten: Die Rollen sind schlicht nicht mehr vorhanden. Den Mitarbeitern müssen daher entsprechende Alternativen angeboten werden. Nicht vergessen werden sollte, dass bei einer Änderung der Arbeitsbedingungen auch der Betriebsrat zu berücksichtigen ist. Dieser muss beispielsweise davon überzeugt werden, dass es beim Daily Scrum Meeting lediglich um das Verfolgen des Fortschritts im Sprint und nicht um eine personenbezogene Performancemessung geht.

Die Einführung "von unten" ist vorzugsweise für kleine und mittelständische Unternehmen mit einer einzigen IT-Abteilung geeignet. Das beschriebene Vorgehen basiert darauf, dass die Mitarbeiter des ersten Pilotprojekts den Kern weiterer agiler Projekte bilden. Auf diese Weise gelingt es, Schritt für Schritt alle Projekte der IT-Abteilung agil durchzuführen. Ergänzend dazu kann das abteilungsinterne Change-Projekt das Vorgehen an den Schnittstellen der Pilot-Projekte klären, z.B. die Zusammenarbeit mit den Fachbereichen, die die Anforderungen liefern und das System abnehmen, dem Systembetrieb, der das System betreiben muss, oder Dienstleistern, die Komponenten für das System bereitstellen.

Diese Art der Einführung ist für große und verteilt arbeitende Unternehmen nicht möglich. Einerseits würde die Einführung viel zu lange dauern, andererseits würde einem abteilungsinternen Change-Projekt die Macht fehlen, gewisse Veränderungen, wie etwa die Vereinbarung kurzer Releasezyklen mit dem Systembetrieb zu initiieren.

Fazit

Agile Methoden bieten durch das iterative Vorgehen die Möglichkeit, sogar Projektziele strukturiert zu erreichen, für die das konkrete Vorgehen zu Projektbeginn noch nicht ganz klar ist. Um die Vorteile dieser Methoden realisieren zu können, müssen bei der Einführung allerdings eine Reihe von Randbedingungen beachtet werden:

- Scrum ist vor allem bei Softwareentwicklungsprojekten mit einer klaren Vision sinnvoll.
- Die Einführung bedeutet eine Prozessänderung und bringt die Einführung neuer und den Verzicht auf einige etablierte Rollen mit sich.

- Die Änderung des Vorgehens hat häufig auch organisatorische Auswirkungen und macht Änderungen der Unternehmenskultur notwendig.
- Probleme und Widerstände bei der Einführung sind normal und müssen gelöst werden.
- Die Einführung von Scrum kann nicht nebenbei geschehen, sondern muss geplant und bewusst angegangen werden.

Die Einführung agiler Methoden ist kein Selbstzweck, sondern dient dazu, Geschäftsziele zu erreichen. In diesem Sinne ist es nur natürlich, dass einerseits agile Methoden das Unternehmen verändern und andererseits, dass sich agile Methoden bei der Anwendung im Unternehmen ändern (s. West, Grant, 2010). In diesem Sinne ist die Arbeit an den eigenen agilen Prozessen nie abgeschlossen.

Literatur

- Grant, Tom: From Agile Development To Agile Engagement, Forrester Research, 2009
- Pichler, Roman: Scrum – Agiles Projektmanagement erfolgreich einsetzen, dpunkt Verlag, 2008
- Roth, Katja; Lieder, Thomas: Agiles Projektmanagement. Häufige Stolperfallen in Scrum, Projekt Magazin 11/2010
- West, Dave; Grant, Tom: Agile Development: Mainstream Adoption Has Changed Agility, Forrester Research, 2010
- Wirdemann, Ralf: Agiles Projektmanagement. Scrum – eine Einführung, Projekt Magazin 21/2009

Fachbeitrag

Erfolgschancen erhöhen

Scrum mit Scrum im Unternehmen einführen

Agile Methoden werden immer populärer. Heute nutzen mehr als 84% aller Unternehmen zumindest in Teilen ihrer Softwareentwicklungsprojekte agile Praktiken (VersionOne, 2013). Natürlich wird nicht jedes Projekt agil durchgeführt – die Anzahl dieser Projekte steigt jedoch erheblich. Während 2002 nur ca. 2% aller Projekte und weniger als 5% der Applikations-Neuentwicklungen mit agilen Methoden durchgeführt wurden, sind heute fast 9% aller Projekte und 29% der Applikations-Neuentwicklungen agil unterwegs (Standish, 2011). Der prominenteste Vertreter der agilen Familie ist dabei Scrum. In einer Forrester-Umfrage (2012) gaben 81,5% der Befragten (Projektmanager, die bereits agile Methoden einsetzten) an, Scrum zu nutzen.

Scrum-Einführungen scheitern häufig

Der Einsatz agiler Methoden geschieht nicht zufällig, sondern wegen der erzielten Erfolge. Dies legt die Frage nahe, warum "nur" 29% der Neuentwicklungen mit Scrum erfolgen. Ein Grund liegt darin, dass es sehr schwierig ist, agil zu werden (Cohn, 2009). Forrester bringt es auf den Punkt: Die Zukunft der Agilität leuchtet hell – aber nur wenn man sich um das Change Management kümmert, das für die Einführung im Unternehmen notwendig ist (Forrester, 2012). Dies geschieht leider nicht immer. Entsprechend scheitert auch die Einführung agiler Methoden häufig, insbesondere wenn dies in einem skalierten Umfeld, also bei vielen voneinander abhängigen Teams, geschehen soll. Hauptgründe sind dabei die mangelnde Erfahrung mit agilen Methoden, mangelhaftes Verständnis für die Auswirkungen auf die Organisation sowie Unternehmenskulturen, die im Widerspruch zur Agilität stehen (VersionOne, 2013).

Vor diesem Hintergrund beschreibt der vorliegende Artikel, wie mithilfe eines Scrum-Projekts die Scrum-Einführung (oder die Einführung anderer agiler Methoden) im Unternehmen gelingen kann (Maximini, 2013). Dabei steht vor allem im Fokus, wie innerhalb des Change-Projekts das Führungsteam aussehen kann und welcher Aufgaben sich diese sogenannte "Führungskoalition" annehmen muss, um der Einführung zum Erfolg zu verhelfen.

Autor

Dominik Maximini

Dipl.-Wirtschaftsinf. (FH), Professional Scrum Trainer, Managing Consultant bei der NovaTec Consulting GmbH

Kontakt:
dominik.maximini@novatec-gmbh.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

Scrum im Unternehmen einführen

Wie Sie Akzeptanz für Veränderungen erreichen

sowie in den Rubriken:

› Agiles Projektmanagement

› PM im Unternehmen einführen

› Change Management

Service-Links

Dienstleister
› Projektmanagementeinführung und –optimierung

Termine
› Agiles Projektmanagement

Bücher
› Agiles Projektmanagement
› Change Management

Software
› Agiles Projektmanagement

Die Führungskoalition

Empirische iterative Vorgehensweisen wie Scrum verkörpern ein anderes Wertesystem, als dies gerade in vielen großen Unternehmen (insbesondere wenn diese in gesättigten Märkten agieren) üblich ist. Während bei Scrum Fokus, Offenheit, Transparenz, kontinuierliche Verbesserung, Respekt und Mut gelebt werden, legen manche Unternehmen einen stärkeren Fokus auf Prozesstreue, langfristige Vorausplanung, Hierarchien und Stabilität. Diese Sichten schließen sich zwar nicht zwangsläufig gegenseitig aus, können aber durchaus zu Konflikten führen.

Mit der einfachen Deklaration der Scrum-Einführung ist es daher in aller Regel nicht getan. Wenn Sie die agilen Werte so im Unternehmen verankern wollen, dass diese gelebt werden und dadurch auch die Arbeit zwischen verschiedenen Abteilungen und Bereichen harmoniert, müssen Sie außerdem die reine Projektebene verlassen und auch in andere Unternehmensbereiche hineinwirken. Diese Wirkung können Sie aber nur dann erzielen, wenn die jeweilige Führungsebene die Scrum-Einführung aktiv unterstützt.

Die Führungskoalition richtig besetzen

Das Team der Führungsebene, das den Organisationswandel begleitet, wird von John Kotter (2011) als "Führungskoalition" bezeichnet. Es muss aus Personen bestehen, die den Wandel wollen, verstehen, umsetzen können und die Macht haben, ihn durchzusetzen. Je nach Kontext führt das zu unterschiedlichen Besetzungen der Führungskoalition (siehe Tabelle 1).

Selbstverständlich müssen die Mitglieder der Führungskoalition ihre zeitliche Verfügbarkeit entsprechend sicherstellen. In aller Regel ist die Scrum-Einführung so wichtig, dass sie sogar andere Themen zurückstellen müssen – immerhin werden neue Paradigmen eingeführt, die das Potential haben, die bestehende Unternehmenskultur zu verändern. Ist die Verfügbarkeit der Führungskoalition nicht gegeben, sinkt die Erfolgswahrscheinlichkeit drastisch, während negative Folgen, wie z.B. Mitarbeiterfluktuation, inkonsistente Prozesse oder sinkende Produktivität in ihrer Häufigkeit und hinsichtlich der Schwere ihrer Auswirkungen erheblich zunehmen.

Ebene, auf der Scrum eingeführt werden soll	Höchste benötigte Führungskraft	Hinweise
Projekt	Projektleiter	Konflikte des Einführungsprojekts mit dem Rest der Organisation sind vorprogrammiert. Insbesondere als Pilotprojekt zu empfehlen.
Abteilung	Abteilungsleiter	Innerhalb der Abteilung kann das sehr gut funktionieren. Probleme treten meist dann auf, wenn die Unterstützung der Bereichsleitung fehlt, weil dann abteilungsübergreifende Veränderungen nur sehr schwer möglich sind.
Bereich	Bereichsleiter	Zum Beispiel im Entwicklungsbereich können hier schon exzellente Ergebnisse erzielt werden. Schwierig wird es bei der Zusammenarbeit mit anderen Bereichen wie Produktmanagement, Vertrieb oder Einkauf. Es kann auch zu Konflikten mit Controlling und Human Resources kommen.
Unternehmen	Geschäftsführer / CEO	Der schwierigste Weg betrifft die Veränderung der Unternehmenskultur. Dafür sind – bei Erfolg – die Potentiale für das Unternehmen am größten. Bei Erfolg treten hier keine der oben genannten Probleme mehr auf. Es kann allerdings beim Austausch mit anderen Marktteilnehmern in Einzelfällen zu Konflikten kommen, wie z.B. Kunden oder Auftragnehmern.

Tabelle 1: Leitung der Führungskoalition im Kontext.

Beispiel

Nehmen wir zum Beispiel Herrn Müller. Er ist Gesamtprojektleiter eines 16 Mio. Euro teuren IT-Projekts in einem großen Unternehmen der Automobilindustrie. Er hat bereits Erfahrungen mit agilen Methoden gesammelt. In seinem aktuellen Projekt hat er etwa zwei Drittel des Projektbudgets für Softwarethemen eingeplant; der Rest wird für die Hardwareentwicklung benötigt. Er sieht für sein Projekt signifikante Vorteile mit agilen Vorgehensweisen und möchte daher Scrum einführen.

Herr Müller muss sich jetzt entscheiden: Will er Scrum in einem Softwareteilprojekt, im Gesamtprojekt oder sogar darüber hinaus einführen? Nach einigen Diskussionen mit seinen Kollegen und Vorgesetzten entscheidet er sich dafür, agile Methoden in den Software-Teilprojekten einzuführen. Er ist sich im Klaren darüber, dass es zwischen seinen Hard- und Software-Teilprojekten zu Konflikten kommen wird, da die Hardwareentwicklung mit einem V-Modell-Ansatz weiterarbeitet. Dies nimmt er in Kauf. Da es sich um die Veränderung einzelner Teilprojekte handelt, kann Herr Müller als Gesamtprojektleiter der Führungskoalition vorstehen – er verfügt über genug Motivation, die Kenntnisse und die Macht für den Wandel.

Agile Transitionen beginnen oft mit der Einführung auf Teilprojekt-Ebene. Bitte beachten Sie, dass ich im Folgenden ganz bewusst keine Lösung dafür skizziere, wie Hard- und Software-Entwicklungsprojekte agil harmonieren können. Auch das Thema der Skalierung, also der Einführung vieler voneinander abhängiger Teams, wird nicht behandelt. Der Fokus dieses Beitrags liegt ausschließlich auf dem Thema "Scrum mit Scrum einführen".

Die Scrum-Einführung mit Scrum

Es hat sich bewährt, Scrum mit Scrum einzuführen. Das bedeutet, dass die Werte, Rollen, Ereignisse und Artefakte von Scrum auch von der Führungskoalition genutzt werden, um das Einführungsprojekt zu steuern. Ziel ist dabei, auch als Führungskraft (parallel zu den betroffenen Projektteams) Erfahrungen mit agilen Methoden zu sammeln, ständige Transparenz über den Fortschritt zu erlangen sowie die betroffenen Mitarbeiter durch das eigene Vorbild zu inspirieren. Agile Methoden mit einem vorhersagbaren Ansatz (z.B. Wasserfall oder V-Modell) einzuführen, ist ein Widerspruch in sich und wird dazu führen, dass Ihre Mitarbeiter (sobald sie Agilität verstanden haben) Sie im Stich lassen.

Beachten Sie, dass Ihr Unterfangen scheitern wird, wenn die Werte nicht vorgelebt werden. Nehmen Sie z.B. eine Führungskoalition, die sich misstraut – in einem solchen Klima kann keine Transparenz entstehen und wird sich folglich auch in Entwicklungsprojekten nicht implementieren lassen.

Die Führungskoalition mit Scrum-Rollen besetzen

Im Hinblick auf die Führungskoalition bedeutet die Scrum-Einführung mit Scrum, dass auch hier die Scrum-Rollen "Entwicklungsteam", "Scrum Master" sowie "Product Owner" vertreten sein müssen. Der Product Owner ist dabei verantwortlich für das Ergebnis – also in diesem Fall den Wandel des Unternehmens und die Einführung von Scrum. Er muss über die nötige Autorität verfügen und sein Team motivieren können. Er ist zwingend die "höchste benötigte Führungskraft", wie in Tabelle 1 dargestellt.

Als Product Owner ist er selbstverständlich auch für das Stakeholdermanagement zuständig. "Stakeholder" sind dabei die internen Kunden, die Führungskoalition, aber auch Führungskräfte und Meinungsmacher aus anderen Bereichen der Organisation. Seine "Kunden" sind alle, die von der Veränderung betroffen sind – potentiell also das Team, die Abteilung, der Bereich oder sogar das gesamte Unternehmen.

Der Scrum Master als Prozessexperte

Der Scrum Master hat in diesem speziellen Kontext die Aufgabe, seine Kollegen in der Führungskoalition als Prozessexperte sowie Moderator zu unterstützen und erhöht dadurch die Effizienz der Führungskoalition. Er hilft allen Beteiligten, Scrum zu verstehen und sich in die jeweils eigene Rolle einzufinden. Manchmal erinnert er auch als "schlechtes Gewissen" an noch offene Aufgaben.

Er sollte der beste verfügbare Scrum-Experte im Unternehmen sein. Gibt es einen solchen noch nicht, empfehle ich dringend, für die Übergangszeit auf einen externen Berater zurückzugreifen. Eine Doppelrolle Scrum Master / Product Owner ist strikt verboten, da dies zu Interessenskonflikten und potentiell schlechten Ergebnissen führt.

Fach- und Führungskräfte für das Entwicklungsteam

Das Entwicklungsteam entwickelt Lösungen und trägt den Wandel ins Unternehmen hinein. In der Regel besteht es aus Führungskräften und vereinzelten Fachexperten, wie z.B. ein Software-Architekt, Domänenspezialist oder eine wichtige informelle Führungskraft (z.B. ein lange im Unternehmen tätiger Entwickler, dessen Einschätzungen und Bewertungen als Meinungsführer ein hohes Gewicht haben). Hier werden Probleme analysiert, Strategien erstellt, operationalisiert und konkrete Lösungen vor Ort implementiert. Sprich: Hier wird gearbeitet! Wenn Ihr Unternehmen groß ist und die Scrum-Einführung mehr als eine Hand voll Führungskräfte betrifft, kann es eventuell sinnvoll sein, ein zusätzliches "Umsetzungsteam" ins Leben zu rufen. Ein solches Umsetzungsteam besteht in aller Regel aus den Führungskräften der nächst niedrigeren Ebene und hat die Aufgabe, die in der Führungskoalition erarbeiteten Strategien und Aufgaben umzusetzen. Dieser Fall wird im vorliegenden Artikel allerdings ausgeklammert.

Das Entwicklungsteam sollte im Regelfall aus drei bis neun Personen bestehen. Scrum Master und Product Owner werden hier nicht dazugerechnet. Alle drei Rollen zusammen bilden die Führungskoalition.

Beispiel

Herr Müller überprüft, wer die höchste benötigte Führungskraft für seine Scrum-Einführung ist. Er stellt fest, dass er dies selbst ist, da die Einführung nur auf Projektebene stattfindet. Folglich muss er die Rolle des Product Owners übernehmen. Bei der Suche nach einem geeigneten Scrum Master stößt Herr Müller auf Frau Schnell. Sie hat bereits fünf Jahre Erfahrung mit Scrum in zwei Unternehmen gesammelt und macht einen engagierten sowie kompetenten Eindruck. Beide einigen sich und Frau Schnell wechselt als Scrum Master zu Herrn Müller ins Projekt. Gemeinsam wählen Sie die Mitglieder für das Entwicklungsteam aus. Die vier Teilprojektleiter der Softwareprojekte sind eine offensichtliche Wahl. Hinzu kommen zwei erfahrene Entwickler sowie ein Qualitätsmanagement-Experte, der dem gesamten IT-Projekt zugeordnet ist. Zusammen sind es also sieben "Entwickler", die gemeinsam mit Product Owner und Scrum Master als Führungskoalition das Ziel der Scrum-Einführung vorantreiben.

Das Product Backlog

Sind die Rollen verteilt, müssen als nächstes die Anforderungen für die Scrum-Einführung in einem Product Backlog festgehalten werden. Dieses sollte auf der Vision des Wandels aufbauen. Auch muss das Product Backlog nicht von Anfang an vollständig sein – wichtig ist, dass es vollständig genug ist, um anzufangen. Es ist normal, dass sich während des Einführungsprojekts Anforderungen ändern, neue hinzukommen oder alte obsolet werden.

Es ist zu empfehlen, die gesamte Führungskoalition in die Erstellung des Backlogs einzubeziehen. Die Granularität der Elemente sollte so gewählt sein, dass mindestens jede Woche eine Aufgabe erledigt werden kann. So lässt sich sicherstellen, dass der Fortschritt deutlich sichtbar ist, was die Motivation steigert und die Möglichkeit bietet, bei Fehlentwicklungen schnell korrigieren zu können.

Beispiel

Herr Müller setzt sich mit seinem Team zusammen und erstellt mit Hilfe der gemeinsamen Kreativität in einem Brainstorming die erste Version des Product Backlogs. Dabei identifiziert er die aus seiner Sicht wichtigsten Elemente:

1. Die Arbeitsweise der Führungskoalition klären.
2. Die Vision des Wandels an alle Mitglieder der Teilprojekte kommunizieren.
3. Die Projektmitarbeiter in Scrum ausbilden.
4. Die Entwicklungsziele (zu entwickelnde Features) der einzelnen Teilprojektteams eindeutig sortieren und in Form eines Product Backlogs aufbereiten. Die Formulierung kann dabei z.B. in Form von User Storys erfolgen.
5. Die technische Infrastruktur für die kontinuierliche Integration (Continuous Integration) bereitstellen lassen.
6. An andere Projekte sowie die Linienorganisation kommunizieren, was geplant ist und wie in Zukunft darüber informiert wird.
7. Metriken implementieren, welche die Leistungen der Teams messen.
8. Mit dem Project Management Office besprechen, wie deren Anforderungen auch mit agilen Prozessen erfüllt werden können.
9. Das Qualitätsmanagement direkt in die Teilprojektteams integrieren.

Zwar werden noch weitere Aufgaben erkannt, diese werden jedoch ungeordnet mit niedriger Wichtigkeit im Backlog notiert.

Die Führungskoalition ist sich darüber im Klaren, dass diese Liste nicht vollständig ist. Auch können in einem anderen Kontext andere Punkte wichtiger sein. Für diesen konkreten Fall erscheint das Backlog aber vollständig genug, um mit der Veränderung beginnen zu können.

Arbeitsweise der Führungskoalition

Die Führungskoalition muss selbstverständlich nicht nur die Scrum-Rollen übernehmen, sondern auch ihre Arbeitsweise entsprechend anpassen. Das Herz von Scrum ist dabei der Sprint, also die Iteration. Je kürzer die Iteration, desto schneller kann auf neue Entwicklungen reagiert werden. Die maximale Länge einer Iteration bei der Scrum-Einführung mit Scrum beträgt auch hier vier Wochen, die kürzeste sinnvolle Dauer eine Woche. Wird eine längere Dauer gewählt, sind Ergebnisse auch erst nach dieser Zeitspanne transparent und für Außenstehende ist kein Fortschritt erkennbar.

Am Ende jeder Iteration muss ein fertiges Produktinkrement geliefert werden. Im Falle einer Scrum-Einführung ist das natürlich keine Software, sondern Prozesse, Dokumente, Entscheidungen, Genehmigungen, Tools usw. Präsentieren Sie diese Ergebnisse am Ende jeder Iteration allen Betroffenen und Interessierten. Ziel ist dabei einerseits, die Fortschritte sichtbar zu machen und sie zu kommunizieren. Andererseits soll Feedback von den Stakeholdern eingeholt werden, um die Scrum-Einführung an den Bedürfnissen der Betroffenen auszurichten und das zu tun, was diesen einen schnellen Nutzen bringt. Sofern Sie etwas (im wahrsten Sinne des Wortes) Greifbares vorzuweisen haben, lassen Sie die Stakeholder dieses auch anfassen (z.B. Dokumente) oder ausprobieren (z.B. Tools). So erreichen Sie eine hohe Aufmerksamkeit und echtes Feedback zum Ergebnis.

Mit Retrospektiven die Produktivität erhöhen

Neben diesem Sprint Review gehört auch die Durchführung einer Retrospektive zu den Aufgaben der Führungskoalition. Hier wird erarbeitet, was bei der Zusammenarbeit der Führungskoalition gut und weniger gut geklappt hat. Außerdem werden konkrete Maßnahmen beschlossen, mit denen sich die Zusammenarbeit verbessern lässt. Wird dieses Instrument zur kontinuierlichen Verbesserung regelmäßig und ernsthaft genutzt, können sich erhebliche Produktivitätssteigerungen ergeben. Außerdem werden Probleme (wie z.B. die Unzufriedenheit eines Mitarbeiters, fehlende Prozesse und Tools oder Konflikte mit anderen Bereichen) frühzeitig erkannt und können behoben werden, bevor sie ernsthaften Schaden anrichten.

Transparent planen im Daily Scrum

Um die Einführung agiler Methoden mit Scrum schnell und nachhaltig zu gestalten, ist auch die Einführung eines Daily Scrums notwendig. Dabei trifft sich die Führungskoalition täglich zur selben Zeit am selben Ort, um sich zu synchronisieren, neue Probleme zu identifizieren und die Projektplanung zu aktualisieren. Dieser tägliche Planungsworkshop sollte nicht länger als 15 Minuten dauern – sonst wird er ineffizient und wird von den Teammitgliedern abgelehnt.

Es gibt auch Teams, die sich nicht täglich treffen möchten. Dies ist ein deutlicher Hinweis darauf, dass die Einführung von Scrum nicht die höchste Priorität für sie hat. Das ist nicht unbedingt schlimm, kann aber bedeuten, dass die Tragweite einer Scrum-Einführung unterschätzt wird. Ich empfehle daher, jeden Sprint neu zu überdenken, ob die Häufigkeit der Daily Scrums ausreicht.

Beispiel

Herr Müller und sein Team einigen sich schnell darauf, alle zwei Wochen ihre Ergebnisse den Stakeholdern vorzuführen, Feedback einzusammeln und das weitere Vorgehen zu planen. Außerdem einigen sie sich darauf, je-

den Morgen um 8 Uhr in der Kaffeeküche die neuesten Informationen auszutauschen und ggf. die Sprintplanung zu überarbeiten. Frau Schnell trägt dabei wesentlich zum Erfolg bei, indem sie abschweifende Diskussionen unterbricht und die Führungskoalition ständig auf die aktuellen Ziele fokussiert.

Damit die Sprintplanung transparent wird, hängt das Team ein Sprint Backlog in Form eines Taskboards in die Kaffeeküche, auf dem die aktuellen Aufgaben aus dem Product Backlog zu sehen sind. So ist jeden Tag ersichtlich, wer was bis zum nächsten Daily Scrum erledigen möchte und was insgesamt noch nötig ist, um das Sprintziel zu erreichen.

Das Taskboard besteht in diesem Fall aus den im Scrum-Umfeld häufig anzutreffenden Spalten "Product Backlog Item", "Offen", "In Bearbeitung" und "Fertig". Im Sprint Planning werden zu den Product Backlog Items kleinschrittige Aufgaben geplant und auf das Taskboard gehängt. Zu Beginn des Sprints hängen somit alle Aufgaben in "Offen". Beginnt ein Mitglied der Führungskoalition die Arbeit an einer Aufgabe, bewegt er diese in die Spalte "In Bearbeitung". Sobald sie abgeschlossen ist, wandert sie zu "Fertig".

Die ersten Erfolge werden sichtbar

Die Planung funktioniert von Anfang an für das Team und das Review erzeugt sehr gutes Feedback. Als Ergebnis des ersten Sprints zeigt das Team um Herrn Müller den Stakeholdern auf, wie diese sich in den Wandel einbringen können. Außerdem wurde der Start des ersten agilen Projektteams vorbereitet. Diese konkreten Fortschritte und der ständige Fokus auf Transparenz begeistern die anwesenden Kollegen. Es kommt auch Feedback: Welche Kennzahlen hat die Führungskoalition vorgesehen, um den eigenen Erfolg sowie den des ersten agilen Projektteams zu messen? Dieser Punkt ist zwar im Backlog der Führungskoalition bereits enthalten, Herr Müller nimmt die Rückfrage allerdings zum Anlass, ihn schon für den nächsten Sprint einzuplanen.

Als etwas schwieriger erweist sich die Retrospektive: Der größte monierte Punkt an Herrn Müller und einigen anderen Teammitgliedern ist ihre mangelhafte Verfügbarkeit. Dadurch konnten einige Ziele (namentlich die Schulung der Projektbeteiligten und die Erstellung eines teilprojektübergreifenden Product Backlogs) nicht erreicht werden. Nach langen – teils auch hitzigen – Diskussionen einigen sich die Mitglieder der Führungskoalition darauf, einige Termine so umzulegen, dass auch die zeitlich stärker belasteten Kollegen teilnehmen können. Im Gegenzug sorgen diese Kollegen dafür, dass ihnen mehr Zeit für die Arbeit an der Scrum-Einführung zur Verfügung steht. Dabei sorgt ein Satz von Frau Schnell für den Durchbruch: "Entscheiden heißt verzichten!"

Auf diese Weise kommt die Führungskoalition um Herrn Müller gut voran. Im nächsten Sprint widmet sich das Team den Metriken, die zur Erfolgsmessung des Einführungsprojekts sowie des ersten agilen IT-(Teil-)Projekts herangezogen werden sollen. Man einigt sich auf "Zufriedenheit der Mitarbeiter", "durchschnittlich geschaffener Produktwert pro Entwickler" und "Anzahl der pro Sprint neu hinzugekommene Bugs".

Für die Einführung der Mitarbeiter in die Scrum-Methodik wird je Team ein ganztätiger Workshop initiiert, dessen Vorbereitung und Leitung Frau Schnell übernimmt. Die Workshops werden dabei einem Review durch die Teilnehmer unterzogen, um daraus kontinuierlich Verbesserungen abzuleiten. Schließlich könnten diese Workshops unternehmensweit genutzt werden, falls Scrum unternehmensweit ausgerollt werden sollte.

Auch die Beteiligten in den anderen Projekten sowie in der Linienorganisation werden über den Fortschritt der agilen Einführung auf dem Laufenden gehalten. Dies geschieht nicht nur in den Sprint Reviews, sondern Herr Müller informiert alle Interessierten aktiv in monatlichen Informationsveranstaltungen. So stellt er sicher, dass die ersten Erfahrungen auch in andere Projekte und Organisationseinheiten transportiert werden. Von den Erfahrungen profitieren insbesondere die Kollegen, die ebenfalls über die Einführung agiler Methoden nachdenken.

Wie ging es weiter?

Sechs Monate später: Herr Müller hat in seinen vier Software-Teilprojekten Scrum eingeführt. Nach einigen anfänglichen Schwierigkeiten läuft es gut. Dazu hat wesentlich beigetragen, dass die Mitglieder der Führungskoalition die Sorgen und Nöte der Projektteams am eigenen Leib erfahren konnten und dies für alle Beteiligten deutlich sichtbar wurde. Die Leistung der Teams ist jetzt besser als zuvor und die Zufriedenheit der Mitarbeiter ist enorm gestiegen. Die hohe Transparenz vermeidet böse Überraschungen und sorgt bei allen Beteiligten für Klarheit. Besonders freut Herrn Müller, dass er Änderungen am Projektumfang ohne große Probleme zu jedem Sprintwechsel bei seinen Teams platzieren kann.

Die Führungskoalition arbeitet mittlerweile sehr effizient und alle schätzen die Zusammenarbeit. In letzter Zeit häufen sich jedoch trotz der nachweisbaren Erfolge die Probleme mit anderen Abteilungen und Bereichen. Nicht alle Kollegen stehen – teilweise aus gutem Grund – Scrum wohlwollend gegenüber. Auch rumpelt es immer wieder bei den Hardwareteilprojekten, da diese mit der Geschwindigkeit der Software-Entwickler nicht mithalten können. Herrn Müller dämmert es, dass es an der Zeit ist, auch andere Bereiche der Organisation von Scrum zu überzeugen, wenn er die aktuell aufkeimenden Probleme nachhaltig lösen möchte. Aber ist dies der richtige Ansatz für die anderen Bereiche? Herr Müller beschließt, das herauszufinden…

Fazit

Wer Scrum im Unternehmen einführen möchte, muss dafür Scrum einsetzen. Nur so kann sichergestellt werden, dass die Mitglieder der Führungskoalition Scrum nicht nur im Detail verstehen, sondern auch, dass die betroffenen Mitarbeiter sehen, dass nicht "Wasser gepredigt und Wein getrunken" wird. Die Regeln von Scrum sind kurz und einfach. Sie aber in der Praxis umzusetzen, ist angesichts der alltäglichen Realität sehr schwer. Nur wenn diese Schwierigkeiten überwunden werden, kann Scrum so exzellent implementiert werden, dass Zufriedenheit sowie Produktivität der Mitarbeiter schnell und nachhaltig steigen. Die Schwierigkeiten beginnen dabei schon am allerersten Tag der Einführung: Wann wird das Daily Scrum durchgeführt? An welchem Ort? Wie schaffe ich es, täglich dabei sein zu können – und dann auch noch jeden Tag einen Fortschritt vorzuweisen?

Wichtig ist auch, dass die Führungskoalition die für Scrum notwendigen Werte und Prinzipien aktiv vorlebt und einfordert. Diese sind vor allem: Fokus, Offenheit, Transparenz, Inspektion, Adaption, Respekt, Mut und Engagement. Es ist nicht möglich, Transparenz und Offenheit in seinen Projekten zu erhalten, ohne diese nicht auch umgekehrt zu gewähren. Es ist klar, dass dies nur in einem Klima des Vertrauens erreicht werden kann.

Dieser Artikel hat nur einen Aspekt beschrieben, der für die Einführung von Scrum wichtig ist. Die Einhaltung dieser Vorgehensweise sorgt allerdings dafür, dass alle Aspekte der Scrum-Einführung effektiver, effizienter, trans-

parenter und mit einer höheren Erfolgswahrscheinlichkeit vonstattengehen, als dies bei einem ungeplanten oder vollständig vorausgeplanten Ansatz der Fall wäre.

Machen Sie uns transparent, was Ihnen an diesem Artikel gut und was Ihnen weniger gut gefallen hat. Teilen Sie uns in Form eines Kommentars oder per Mail an redaktion@projektmagazin.de mit, welche Themen Sie im Kontext dieses Artikels am brennendsten interessieren. Wir werden dies gerne in die weitere Themenplanung einfließen lassen.

Literatur

- Cohn, M.: Succeeding with Agile. Software Development Using Scrum, Amsterdam, Addison-Wesley Longman, 2009
- Forrester: November 2011 Global Agile Software Application Development Online Survey, April 2012, Forrester Research, Inc.,
- Kotter, J.: Leading Change: Wie Sie Ihr Unternehmen in acht Schritten erfolgreich verändern, München, Vahlen, 2011
- Maximini, D.: Scrum – Einführung in der Unternehmenspraxis. Von starren Strukturen zu agilen Kulturen, Berlin Heidelberg, Springer Gabler, 2013
- Standish: Chaos Manifest. The Laws of CHAOS and the CHAOS 100 Best PM Practices, The Standish Group International, Inc., 2011
- VersionOne: State of Agile Survey: Seventh annual survey. Version one, 2013: http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf, zuletzt eingesehen am 05.07.2013

Fachbeitrag

So gelingt die Kanban-Einführung

In der IT kommt die Veränderungsmethode Kanban immer häufiger zum Einsatz. Dies ist auch nicht weiter verwunderlich: Denn mit einfachen Mitteln führt Kanban schnell zu nachhaltigen Verbesserungen. Es verspricht klare Arbeitsabläufe, eine gleichmäßigere Auslastung sowie kürzere Durchlaufzeiten und lässt sich ohne weiteres auf jeden bestehenden IT-Prozess sofort anwenden. Dabei lässt sich Kanban nicht nur in der IT, sondern auch in anderen Bereichen einsetzen, die Aufgaben nach Prozessen bearbeiten, wie z.B. im Kundensupport, im Marketing, im Portfolio-Management oder in Medienagenturen.

In der Praxis wird jedoch häufig übersehen, dass es bei Kanban nicht bloß um die Anwendung einer neuen Methode geht. Um Kanban erfolgreich einzuführen, braucht es vielmehr ein umsichtiges Veränderungsmanagement, dessen Durchführung durchaus anspruchsvoll ist. Der folgende Beitrag beschreibt, wie Sie dies in der Praxis bewerkstelligen können. An einem Beispiel wird gezeigt, wie sich die Kanban-Einführung im Bereich der Softwareentwicklung über mehrere Abteilungen hinweg und unter Berücksichtigung aller Betroffenen erfolgreich gestalten lässt.

Ein typisches Fallbeispiel

Manfred Bauer ist Leiter der Softwareentwicklung eines Infrastruktur-Unternehmens mit knapp 800 Beschäftigten. In den letzten Jahren sind immer mehr Aufgaben und Kompetenzbereiche an das Team von Herrn Bauer herangetragen worden. Dies erhöhte nicht nur den Leistungsdruck, sondern sorgte auch für einige Unklarheiten:

- Einzelne Mitarbeiter wissen nicht, von wem sie wann welche Arbeitsaufträge erhalten. Vieles geschieht auf Basis persönlicher Zurufe ganz nach dem Motto: "Wer lauter schreit, kommt früher dran."
- Das Team hat keine Übersicht, welche Entwicklungsaufgaben aktuell hinzugekommen, in Arbeit oder erledigt sind.
- Es gibt weder eine Koordination der einzelnen Auftraggeber noch eine klare interne Priorisierung der Aufgaben.

Von einem befreundeten Kollegen hörte Herr Bauer in dieser Zeit erstmals von Kanban – und ist sofort "Feuer und Flamme". Die Praktiken erscheinen bestechend einfach und versprechen eine kompakte Lösung der bestehenden Probleme:

Autor

Dr. Siegfried Kaltenecker

Managementtrainer, Supervisor & Coach, Certified ScrumMaster, Geschäftsführer der Loop Organisationsberatung GmbH

Kontakt:
siegfried.kaltenecker@loop-beratung.at

Dr. Klaus Leopold

Dipl.-Informatiker, akkreditierter Kanban-Trainer und -Coach, Inhaber der LEANability e.U.

Kontakt:
klaus.leopold@klausleopold.com

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Software-Kanban – eine Einführung

› Scrum und Kanban sinnvoll kombinieren

in den Rubriken:

› Projektabwicklung

› Prozessmanagement

› Change Management

Service-Links

› Kanban in der IT

- Durch die Visualisierung des vorhandenen Softwareentwicklungs-Prozesses werden die Abläufe ebenso transparent wie die einzelnen Aufgaben (Tasks).
- Die Definitionen für spezifische Arbeitstypen (z.B. Anforderungen, Wartungsaufgaben oder Bugfixes), WIP-Limits und Serviceklassen kanalisieren den Arbeitsfluss und stellen das unkoordinierte Push- auf ein geordnetes Pull-System um (Bild 1). Mit Serviceklassen ist hier eine Klassifizierung der einzelnen Aufgaben gemeint, mit der sich diese nach Auswirkungen und Risiken unterscheiden lassen. So können diese Tasks z.B. in die Klassen "Standard", "Beschleunigte Bearbeitung" und "Fester Liefertermin" aufgeteilt werden. (Zu WIP-Limits und anderen Grundbegriffen von Kanban siehe auch "Software-Kanban – eine Einführung", Projekt Magazin 04/2011)
- Zukünftig werden alle relevanten Stakeholder regelmäßig in ein sog. "Queue Replenishment Meeting" eingeladen, in welchem gemeinsam die anstehenden Arbeiten besprochen und an die Mitarbeiter verteilt werden.

Sogleich macht sich Herr Bauer daran, die internen Arbeitsprozesse seines Teams zu Papier zu bringen. Er definiert WIP-Limits, spezielle Serviceklassen und Metriken, mit denen fortlaufend die Durchlaufzeit und der Durchsatz gemessen werden können. Am Ende kann es Herr Bauer kaum erwarten, die neue Methode seinem Team vorzustellen und dann voll in Richtung Verbesserung durchzustarten…

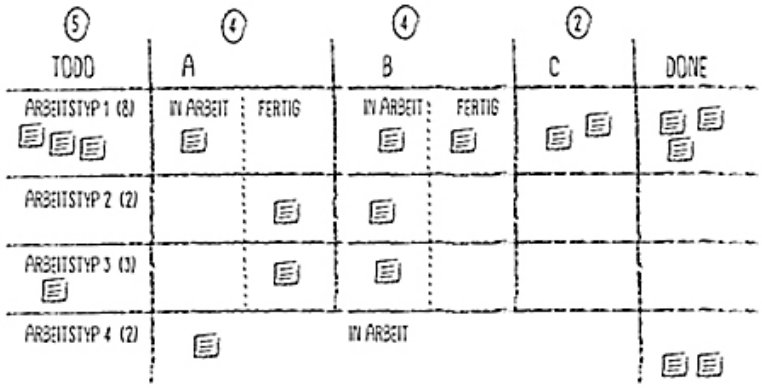

Bild 1: Kanban-Board mit WIP-Limits und Arbeitstypen.

Die große Ernüchterung

Zwei Monate später ist Herr Bauer mehr als ernüchtert. Von seinem ursprünglichen Feuer ist kaum etwas übrig geblieben. Weder konnten die Durchlaufzeiten reduziert noch die tatsächlichen Engpässe geklärt werden. Stattdessen ist die Skepsis einiger Teammitglieder ebenso angewachsen wie das Unverständnis von Seiten der Kunden. Wieso sollten sie auf einmal Rücksicht auf WIP-Limits nehmen? Warum sollten sie nicht mehr direkt zu den Mitarbeitern gehen können, um ihnen direkt den Arbeitsauftrag für eine Sonderlösung zu erteilen? Und die Mitarbeiter selbst sagen, dass sie nicht wegen jeder Kleinigkeit zum Board gehen und Tickets umhängen wollen. Schließlich ist auch Herr Bauers Vorgesetzter, der IT-Hauptabteilungsleiter, nicht sonderlich begeistert vom Verlauf des angekündigten "Innovations"-Projekts. Doch was war passiert?

Kanban-Mechaniken sind nicht genug

Warum haben sich die kontinuierlichen Verbesserungen nicht eingestellt? Was hat Herr Bauer falsch gemacht? Eine erste Antwort auf diese Fragen könnte lauten, dass die Verwendung der Kanban-Mechaniken alleine eben nicht genug sind. Aus unserer Sicht hat Herr Bauer vor allem drei wesentliche Dinge versäumt:

- Das Softwareentwicklungsteam wurde weder in die genaue Problemdefinition noch in die Lösungsfindung einbezogen. Stattdessen hat Herr Bauer alles alleine gemacht. Damit hat er einerseits gemeinsames Lernen und ein besseres Verständnis für die laufenden Prozesse verhindert. Und andererseits hat er die veränderungstypischen Zweifel und Befürchtungen weiter genährt, dass Veränderungen einmal mehr von oben verordnet werden und nur noch mehr Arbeit mit sich bringen.
- Die Stakeholder, in diesem Beispiel vor allem die Kundenvertreter, das Projektmanagement und das Linienmanagement, wurden nicht davon überzeugt, dass eine bessere Leistung in der Softwareentwicklung nur über eine bessere Abstimmung zu realisieren ist – und nicht über höheren Druck. Darüber hinaus entkräftete Herr Bauer die Vorbehalte der Kunden sowie des Top-Managements nicht ("Bringt mir das überhaupt was?" etc.).
- Die Kommunikation schrumpfte weitestgehend auf einseitige Informationen zusammen. Statt die wichtigsten Stakeholder an der Erstellung des Kanban-Boards und der zukünftigen Arbeitsabläufe zu beteiligen, wurde dies von Herrn Bauer alleine ausgearbeitet und gewissermaßen als "Fertiggericht" serviert. Dadurch ließ sich der Mehrwert des Queue Replenishment Meetings ebenso wenig realisieren wie der Nutzen von Daily Stand-ups oder regelmäßiger Retrospektiven.

Kanban braucht ein professionelles Change Management

Tatsächlich zeigt unsere Erfahrung als Kanban-Trainer und Organisationsberater, dass in der Begeisterung über die bestechend einfachen Regeln und Tools immer wieder das professionelle Management der kontinuierlichen, in kleinen Schritten erfolgenden Veränderung (evolutionäres Change Management) vergessen wird. Vor allem in größeren Unternehmen ist dieses Veränderungsmanagement jedoch alles andere als trivial. Es umfasst im Wesentlichen folgende vier Phasen (vgl. Leopold/Kaltenecker, 2012):

- die **Klärungsphase**, in der die persönlichen Verbesserungsideen in eine unternehmerische "Kanban-Initiative" übersetzt, Erwartungen und Ressourcen definiert sowie die eigene Rolle reflektiert werden
- die **Diagnosephase**, in der alle für die Veränderungsinitiative rele-vanten Stakeholder ihre Sichtweisen und Interessen abgeben und diese zu einem gemeinsamen Katalog von Wünschen und Anliegen verdichtet werden

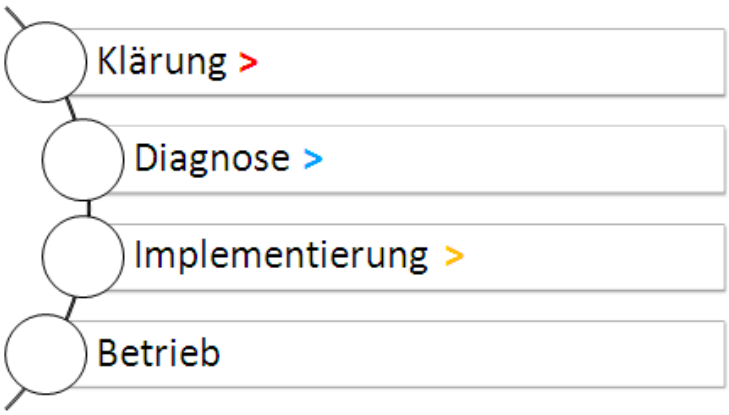

Bild 2: Die vier Phasen der Kanban-Einführung.

- die **Implementierungsphase**, in der dieser Katalog in ein maßgeschneidertes "Systemdesign" transferiert und mit den Stakeholdern ein spezifisches Prozedere vereinbart wird, um Kanban schließlich in den Betrieb zu nehmen

- die **Phase des Betriebs**, in der es um aktive Führung im Sinne einer kontinuierlichen Verbesserungsarbeit geht

Im Folgenden sehen wir uns an einem weiteren Beispiel diese einzelnen Schritte genauer an und zeigen, warum sie für den Erfolg jeder Kanban-Initiative zentral sind.

Die Klärungsphase

Heidrun Schreiber ist seit vier Monaten COO eines Schweizer Softwareunternehmens, das sich auf die maßgeschneiderte Aufarbeitung von Finanzinformationen für Banken, Versicherungen oder Agenturen spezialisiert hat. Von Anfang an ist Frau Schreiber bestrebt, sich einen möglichst guten Überblick über ihren Verantwortungsbereich zu verschaffen. Dabei stellt sie als "Prozess-Ownerin" vor allem einen hohen Verbesserungsbedarf im Bereich "Request for Change" fest, in dem die bestehende Software an die speziellen Kundenwünsche angepasst wird (z.B. für neue Produkte oder bei auftauchenden Business Opportunities):

- Es fehlt offensichtlich an Prozessqualität, da die Durchlaufzeiten sehr hoch sind, viele Aufgaben lange liegen bleiben und auch die gelieferte Software fehleranfällig ist.
- Obwohl eine intensive Zusammenarbeit zwischen den Abteilungen Business, Informatik, Qualitätssicherung und Operations notwendig ist, um die Kunden wunschgemäß zu betreuen, arbeiten alle Abteilungen vorwiegend "siloorientiert", d.h. Arbeiten werden weitgehend isoliert vom Gesamtzusammenhang abgearbeitet und dann "über den Zaun" zur nächsten Abteilung geworfen. Das verstärkt die klassische Schnittstellenproblematik und fördert diverse Grabenkämpfe.
- Die Kommunikation zwischen den Abteilungen läuft sehr formal ab. Viele Informationen werden per E-Mail verteilt, ein gemeinsames Monitoring gibt es ebenso wenig wie Retrospektiven. Dies verhindert ein gemeinsames Lernen und vorhandene Probleme werden nicht gelöst.

Aufgrund ihrer bisherigen Managementerfahrung weiß Frau Schreiber um die Auswirkungen derartiger Probleme. Ein Request-for-Change-Prozess, der nicht so funktioniert, wie er eigentlich sollte, ist eine echte Belastung des gesamten operativen Geschäfts und verhindert eine zielorientierte Wertschöpfung. Glücklicherweise hat Frau Schreiber in ihrem letzten Unternehmen bereits gute Erfahrungen gemacht, die genannten Probleme mit Hilfe von Kanban zu lösen. Dementsprechend ist sie auch entschlossen, in ihrer neuen Funktion darauf zu setzen.

Bereits im nächsten Bereichsmeeting stellt sie ihren Entschluss den Abteilungsleitern vor und holt erste Rückmeldungen zu ihrem Vorschlag ein. Zudem kontaktiert sie noch vor dem nächsten Jour fixe des Top-Managements den für die IT verantwortlichen CIO, um ihre Sichtweisen mit ihm abzugleichen. Erfreulicherweise gibt es sowohl unter den Abteilungsleitern als auch beim CIO ein ähnliches Problembewusstsein. Zudem reagieren alle Angesprochenen positiv auf die Idee eines evolutionären Veränderungsprozesses. Im Jour fixe werden gemeinsam mit dem CEO noch einmal die wichtigsten Fragen durchgegangen, damit aus der Kanban-Idee eine unternehmerische Initiative wird:

- Warum Veränderung? Wie sieht der "Case for action" aus? Was droht, wenn nichts unternommen wird? Was sind daher die wichtigsten Ziele der Veränderungsinitiative?
- Warum Kanban? Wieso ist der evolutionäre Ansatz gut geeignet für eine Verbesserung des Request-for-Change-Prozesses? Welche Probleme werden damit in Angriff genommen? Welchen Nutzen darf sich der

CEO, der Bereich Request for Change, das gesamte Unternehmen davon versprechen? Und wie trägt Kanban zu einer höheren Wertschöpfung bei?

- Was braucht die Kanban-Initiative, um möglichst effektiv für Veränderung zu sorgen? Wie sieht die laufende Kommunikation aus? Welche Zeit- und Personalressourcen werden benötigt? Wie stehen die Kosten dem Nutzen gegenüber? Und was braucht Frau Schreiber, um ihrer Rolle als "Kanban Change Leader" gerecht zu werden?

An dieser Stelle ist Überzeugungsarbeit angesagt. Um das Top-Management für die Kanban-Initiative zu gewinnen, sollten Sie also gute Argumente parat haben. Nachfolgend ein paar Beispiele für mögliche Antworten auf die wichtigsten Fragen:

Warum Kanban? Warum ist der evolutionäre Veränderungsansatz gut geeignet?

- Da das Unternehmen wirtschaftlich erfolgreich und in seinen internen Prozessen ohne größere Reibereien funktioniert, gibt es keinen Grund für große Veränderungsmaßnahmen. Lokale Verbesserungen sind jedoch sehr wohl sinnvoll und notwendig. Kanban hilft dabei, indem es einerseits die bestehenden Strukturen sowie Rollen respektiert und damit Umstellungskosten minimal hält und andererseits kontinuierlich für kleine Verbesserungsschritte sorgt.

Wie trägt Kanban zu einer höheren Wertschöpfung bei?

- Durch das Sichtbarmachen des Arbeitsflusses werden rasch Verbesserungspotenziale, insbesondere Engpässe und Blockaden deutlich. Die Visualisierung hilft dabei, diese konsequent zu bearbeiten. Weiter sorgt die Limitierung der laufenden Arbeit dafür, dass sich der Durchsatz erhöht und mehr Wert für den Kunden generiert wird. Schließlich ermöglichen Serviceklassen, das wirtschaftliche Risiko hinter jedem Arbeitstyp verständlich zu machen.

Was braucht die Kanban-Initiative, um möglichst effektiv für Veränderung zu sorgen?

- Wie die Erfahrung vieler Initiativen zeigt, lebt Kanban von der Beteiligung aller wichtigen Stakeholder. Die für den Arbeitsprozess relevanten Kräfte müssen identifiziert, Schlüsselfiguren möglichst früh eingebunden und die für die gesamte Wertschöpfung wichtigsten Verbesserungswünsche erhoben werden.

Was braucht Frau Schreiber, um ihrer Rolle als "Kanban Change Leader" gerecht zu werden?

- Frau Schreiber braucht natürlich Mut und einen langen Atem. Sie braucht aber vor allem auch Unterstützung von ihren Führungskräften, um möglichst alle relevanten Entscheider aus dem Projektmanagement, dem Sales-Bereich, der Softwareentwicklung und dem Betrieb für Kanban zu gewinnen. Zudem bereitet sich Frau Schreiber mit einem Kanban-Training umfassend auf ihre Rolle vor und zieht punktuell externe Experten zur Unterstützung heran.

Die Diagnosephase

Nachdem Frau Schreiber den Segen des Top-Managements erhalten hat, erfolgt in mehreren Teilschritten eine umfassende Diagnose des Status quo.

Schritt 1: Persönliche Retrospektive

Der erste Schritt sieht eine persönliche Retrospektive vor: Obwohl Frau Schreiber bereits vieles durchdacht hat, nimmt sie sich noch einmal eine halbe Stunde Zeit, um ihre bisherigen Beobachtungen festzuhalten. Was läuft im Request-for-Change-Prozess gut? Auf welche Stärken und Erfolge lässt sich bauen? Welche Verbesserungen sind bereits gelungen? Und was läuft nicht gut? Welche Misserfolge gab es? Wo ist definitiv Verbesserungsbedarf gegeben? Im Sinne des Brainwritings hält Frau Schreiber jede einzelne Antwort auf einem eigenen Post-it fest.

Schließlich sammelt sie diese Post-its auf einem dreispaltigen A3-Blatt und clustert diese. In der ersten Spalte stehen alle positiven Aspekte, in der zweiten Spalte alle negativen. In der dritten Spalte hält sie ihre Schlussfolgerungen fest: Was fällt mir auf? Welche zentralen Herausforderungen sehe ich? Und wie passt das zur skizzierten Verbesserungsstrategie? Diesbezüglich fällt Frau Schreiber vor allem auf, dass wichtige Arbeiten nicht termingerecht fertig werden und es keine gezielte Steuerung gibt, sodass vieles im berühmt-berüchtigten "Feuerlösch-Modus" erfolgt.

Mit dieser strukturierten Visualisierung führt sich Frau Schreiber noch einmal ihre aktuellen Überlegungen vor Augen und kann ihre Startposition kritisch prüfen. Insbesondere der Wert der gemeinsamen Visualisierung und die kontinuierliche Einbindung aller am "Request for Change" beteiligten Arbeitsbereiche bestätigen sich für Frau Schreiber.

Schritt 2: Retrospektive im Führungsteam

Der zweite Schritt erfolgt in ihrem Abteilungsleiter-Team. Gemeinsam mit den Abteilungsleitern führt sie eine zweite Retrospektive mit denselben Fragestellungen durch. Das gibt Frau Schreiber die Gelegenheit, ihre eigenen Einschätzungen durch wichtige Aspekte zu ergänzen und unterschiedliche Perspektiven zu diskutieren. Zudem wird das weitere Vorgehen erörtert und die gemeinsame Verantwortung für den Erfolg der Kanban-Initiative bekräftigt.

Die Führungsrunde bestätigt unisono, dass die Priorisierung der Arbeiten oft nicht klar ist. Wenn eine Priorisierung geschieht, dann erfolgt diese zumeist durch Kundenbeschwerden oder durch Druck von Seiten des Top-Managements (offenbar vor allem von Frau Schreibers Vorgänger). Folgerichtig wird am Ende der Retrospektive vor allem das Thema "Wirtschaftliches Risiko" sowie eine entsprechend klare Klassifizierung bestimmter Arbeiten diskutiert.

Schritt 3: Großgruppenveranstaltung

Im dritten Schritt werden die Sichtweisen aller 41 Mitarbeiter eingeholt, die in den unterschiedlichen Abteilungen und Unternehmensbereichen am Request-for-Change-Prozess mitwirken. Deren Nahsicht auf vorhandene Probleme und Herausforderungen stellt für Frau Schreiber eine unverzichtbare Quelle für die Verbesserungsarbeit dar. Dazu wird eine halbtägige Großgruppenveranstaltung für eine erweiterte Retrospektive geplant, die Frau Schreiber gemeinsam mit dem CIO sowie allen Abteilungsleitern aus dem Business, der Softwareentwicklung, der Qualitätssicherung und dem Betrieb vorbereitet.

Die Veranstaltung soll sowohl eine vertiefte Diagnose ermöglichen als auch die Mitarbeiter auf breiter Front für die Initiative gewinnen. Aus diesem Grund entschließt sich das "Kanban Change Team", die neu betitelte Führungsrunde, bei der Großgruppenveranstaltung die verschiedenen Diskussionsgruppen mit einer fach- und bereichs-

übergreifenden Mischung von Mitarbeitern zu besetzen. Entsprechende Vorabinformationen über die Großgruppenveranstaltung verbreiten die Führungskräfte in ihren jeweiligen Abteilungsmeetings.

Nach der Durchführung wertet das "Kanban Change Team" die Veranstaltung aus. Als bestätigt sieht die Runde die Unklarheit des konkreten Arbeitsprozesses, der Reihenfolge und der Bedeutung bestimmter Arbeiten an. Als überraschend empfand man vor allem die vielstimmige Kritik an der mangelnden Abstimmung sowie die Kritik an den fehlenden Feedbackschleifen von anderen Teams, von Kundenseite, aber auch durch das Management selbst: Passt die Qualität dessen, was wir unseren Kollegen zur Weiterbearbeitung liefern? Passt die Geschwindigkeit? Wird effizient weitergearbeitet? Ist das Ganze auch wirklich im Sinne des Kunden? Und last but not least: Haben wir jetzt gute Arbeit geleistet – oder nicht?

Auf der Tagesordnung des "Kanban-Change-Team"-Meetings steht allerdings nicht nur die Innensicht, sondern auch der kritische Außenblick. Mithilfe einer einfachen Moderationstechnik wird gemeinsam eine Landkarte der Stakeholder entworfen, die Einfluss, Kontaktintensität und Beziehungsqualität sichtbar macht (s. hierzu "Stakeholder Management" auf der Platform for Agile Management http://p-a-m.org/2012/07/stakeholder-management). Auf diesem gemeinsamen Bild aufbauend, bestimmt das Change Team nun strategische Folgeschritte für die Kanban-Einführung, in deren Zentrum der direkte Kontakt mit den wichtigsten Stakeholdern steht.

Schritt 4: Stakeholder befragen

Die als besonders relevant identifizierten Stakeholder, also in unserem Beispiel das Projektmanagement, die Qualitätssicherung, der Sales-Bereich als Kundenvertreter sowie der Bereich Operations werden anschließend kontaktiert, über das Verbesserungsvorhaben informiert und um ein kurzes Gespräch gebeten. In diesen Gesprächen, die in Form eines Interviews durchgeführt werden, erhebt Frau Schreiber nun auch die Außensicht. Dem Leitfaden für die internen Retrospektiven folgend, werden weitere Produktmanager, Vertreterinnen aus dem Marketing und ausgewählte Sales Representatives vor allem zu zwei Punkten befragt:

1. Was passt bisher in der Zusammenarbeit und worauf können wir folglich auch in Zukunft bauen?
2. Womit sind die jeweiligen Gesprächspartner unzufrieden und was davon sollte unbedingt verbessert werden?

Am Ende des Gesprächs, das bewusst aufs Zuhören und Verstehen und nicht auf Diskussion angelegt ist (deswegen das Format des Interviews), werden die Stakeholder noch genauer über die Kanban-Initiative informiert und zu einem Basistraining eingeladen (siehe "Implementierungsphase").

Die Ergebnisse der Interviews werden im "Kanban Change Team" reflektiert und mit der ermittelten Innensicht abgeglichen. Dabei bestätigt sich, dass eine termingerechte Lieferung und eine klare Ordnung der Arbeiten für alle Stakeholder zentrale Kriterien sind. Darüber hinaus wird der Wert der Visualisierung unterstrichen, da die meisten Befragten gerne wissen wollen, in welchem Stadium sich die von ihnen beauftragten Arbeiten gerade befinden. Transparenz und Vorhersehbarkeit stellen also zentrale Anforderungskriterien für die Verbesserungsarbeit dar.

Auf der Basis der geclusterten Interviewergebnisse wird eine erste Zuordnung vorgenommen, wie die wichtigsten Verbesserungsthemen mit Kanban-Elementen adressiert werden können (siehe Tabelle 1). Darüber hinaus wird der letzte Schritt der Diagnosephase vorbereitet.

Wichtigste Verbesserungsthemen "Das sollten wir dringend angehen"	Die nächsten Schritte "So hilft uns Kanban"
"Ich will wissen, wann meine Arbeit fertig wird." "Die Termintreue sollte sich erhöhen." "Ich möchte im Sinne der Kunden dringende Arbeiten vorziehen können."	Diese Anliegen können durch die Einführung von Serviceklassen, WIP-Limits und dem Queue Replenishment Meeting adressiert werden.
"Mir ist unklar, woran das Team gerade arbeitet." "Es fehlt an Übersicht, wo gerade welche Arbeiten durchgeführt werden." "Wir wissen nicht, warum bestimmte Tasks liegen bleiben."	Diese Informationen können durch Visualisierung und die Arbeit mit einem gemeinsamen Kanban-Board bereitgestellt werden.
"Das Verständnis für die einzelnen Prozessschritte ist nicht bei allen vorhanden." "Wir lernen nicht aus einmal gemachten Fehlern, sondern wiederholen sie." "Ich möchte einen Request-for-Change-Prozess und nicht drei oder vier verschiedene."	Diese Probleme werden durch ein gemeinsames Prozessverständnis und regelmäßige Meetings bearbeitet. Sowohl das Daily Stand-Up vor dem Board als auch abteilungsübergreifende Retrospektiven aller am Request-for-Change-Prozess Beteiligten sind für ein entsprechendes Teambuilding hilfreich.

Tabelle 1: Verbesserungsthemen und Umsetzungsmöglichkeiten.

Schritt 5: Verbesserungsideen abstimmen

Dieser letzte Schritt konzentriert sich darauf, die wichtigsten Rückschlüsse aus dem nunmehr verdichteten Gesamtbild zu ziehen: Wo wird insgesamt der größte Verbesserungsbedarf gesehen? Und wie hilft Kanban, uns auch tatsächlich zu verbessern (siehe hierzu ebenfalls Tabelle 1)? In kurzen, persönlichen Meetings lädt Frau Schreiber die wichtigsten Stakeholder nochmals ein, um die konsolidierten Verbesserungsvorhaben vorzustellen und gemeinsam mit ihnen kritisch zu prüfen.

Am Ende ihrer diagnoseorientierten Schrittfolge hat Frau Schreiber bereits eine Reihe essentieller Ergebnisse zusammengetragen:

- Sie hat einen gemeinsam mit allen relevanten Stakeholdern konsolidierten Katalog an Anforderungen und Wünschen in Händen, der in der Implementierungsphase punktgenau bearbeitet und mit hohem Commitment umgesetzt werden kann. Schließlich wurden die notwendigen Verbesserungsschritte nicht von Frau Schreiber oder einem "Prozess-Spezialisten", sondern von allen Betroffenen gemeinsam definiert.
- Durch die umfassende Diagnose hat Frau Schreiber allerdings weit mehr erreicht als eine rein inhaltliche Konsolidierung. Sie hat sowohl die internen als auch die externen Stakeholder des Bereichs frühzeitig in ihre Initiative einbezogen und bewusst Räume für die Mitgestaltung geschaffen. Das stärkt das Verständnis für den Prozess und fördert die effektive Zusammenarbeit.
- Dadurch konnten negativen Emotionen, die für viele Veränderungsvorhaben typisch sind, z.B. Skepsis gegenüber Kanban oder Befürchtungen, dass dies primär Mehrarbeit mit sich bringt, frühzeitig abgefangen werden. Change wird viel weniger als aufgezwungen erlebt und entsprechend erlitten, sondern sinnhaft selbst gestaltet.

- Allein durch die intensive Kommunikation auf allen Ebenen hat Frau Schreiber schon in der Diagnosephase für eine Menge positiver Entwicklungsimpulse im Sinne von Kaizen gesorgt.

Die Implementierungsphase

Die Implementierungsphase gliedert sich wiederum in drei Abschnitte.

Schritt 1: Basistraining

Unter dem Titel "Kanban für Entscheider" hält Frau Schreiber ein halbtägiges Basistraining ab, zu dem nicht nur alle Mitarbeiter, sondern auch alle Stakeholder eingeladen sind. In diesem Basistraining wird Kanban kurz und bündig vorgestellt. An praktischen Beispielen wird das Wie, aber auch das Warum der Verbesserungsarbeit erörtert. Durch den gezielten Einsatz der Diagnoseergebnisse (vgl. Tabelle 1) wird auch auf breiter Ebene für ein fundiertes Verständnis gesorgt.

Schritt 2: Systemdesign-Workshop

Am Ende des Basistrainings entscheidet jede Abteilung für sich, welche zwei bis drei Delegierte mit der Ausarbeitung des Systemdesigns betraut werden sollen. Dadurch lassen sich die spezifischen Perspektiven und Interessen aller am Prozess beteiligten Fachabteilungen berücksichtigen und die Gefahr von anschließenden Abwehrreaktionen ("Das ist nicht unser Ding!") werden minimiert. In dem zweitägigen Systemdesign-Workshop werden insbesondere

- die Grenzen des Kanban-Systems festgelegt.
- Arbeitstypen identifiziert: Welche Stakeholder erteilen welche Arbeitsaufträge und an wen werden diese weitergegeben?
- Arbeitsschritte gefunden: In welchen Stufen verlaufen die Tätigkeiten für die einzelnen Arbeitstypen?
- Kapazitäten und WIP-Limits definiert: Wie viel Arbeit kann das System maximal aufnehmen, damit ein kontinuierlicher Arbeitsfluss gewährleistet bleibt?
- Serviceklassen bestimmt: Wie unterscheiden sich Aufgaben in puncto Auswirkungen und Risiken? Welche Leistungen können in welchen Zeiträumen geliefert werden?
- die tatsächlichen Verbesserungen kontinuierlich gemessen: Welche Dimensionen sollen untersucht werden, um Fortschritte nachzuweisen und weitere Verbesserungspotenziale herauszufiltern?
- Abstimmungsformate und -intervalle für den Betrieb vereinbart: Welche Meetings braucht es? In welchen Abständen sind Meetings sinnvoll?

Zum Ende des Systemdesign-Workshops sind viele Optionen besprochen, Arbeitsabläufe simuliert und wegweisende Entscheidungen getroffen worden. Freilich gäbe es noch Stoff für viele weitere Diskussionen. Ziel des Workshops ist jedoch nicht die Erstellung eines perfekten Systemdesigns. Vielmehr wird eine pragmatische Lösung gesucht, mit der sich Kanban in Betrieb nehmen lässt und die eine kontinuierliche Verbesserungsarbeit im Alltag vorantreibt.

Schritt 3: Meeting zur Inbetriebnahme

Schließlich fehlte nur noch die finale Zustimmung der Stakeholder, um mit Kanban in den Arbeitsalltag zu starten. Im entsprechend genannten Meeting "Kanban-Inbetriebnahme" präsentiert Frau Schreiber das entworfene Board und erläutert den zukünftigen Betrieb anhand konkreter Arbeitsbeispiele (Bild 3). Zudem vereinbaren alle Beteiligten konkrete WIP-Limits, Serviceklassen, Meetings und diejenigen Metriken, anhand derer Mitarbeiter wie Kunden die tatsächliche Verbesserung messen können. Auch dieses Meeting hat nicht zum Ziel, bis ins kleinste Detail jeden denkbaren Fall durchzuspielen. Vielmehr soll, wie Frau Schreiber immer wieder hinweist, ein pragmatischer Startpunkt definiert werden, um die kontinuierliche Verbesserungsarbeit in den Arbeitsalltag zu übertragen.

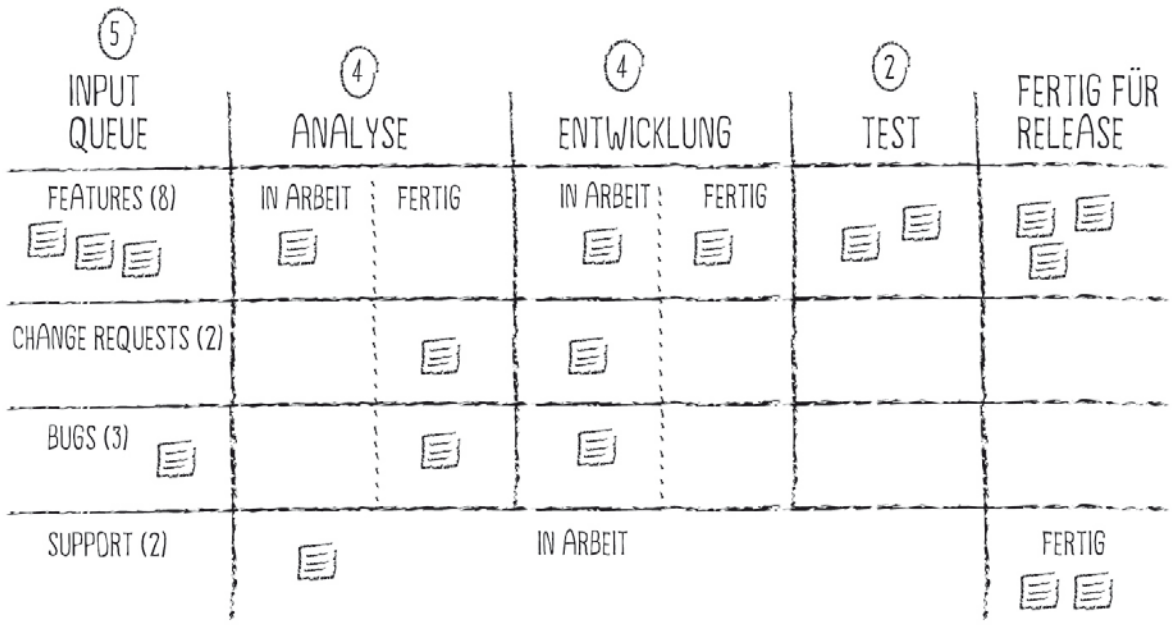

Bild 3: Das fertige Kanban-Board aufgeteilt nach Arbeitstypen und mit WIP-Limits.

Die Betriebsphase

Nach nur zwei Wochen Vorbereitungszeit hat Frau Schreiber den "kanbanisierten" Request-for-Change-Prozess in den Realbetrieb übernommen. Nach einer kurzen Eingewöhnungsphase zeigten sich die ersten Verbesserungen:

- Das Kanban-Board wird konsequent bespielt. Die vielen Bewegungen der Tickets sind eine starke Botschaft und die Mitarbeiter haben Spaß, damit zu arbeiten.
- Es wird bereichsübergreifend zusammengearbeitet. Durch die zunehmende Vertrauensbildung und den sichtbaren Fortschritt, aber auch durch den neu erwachten Sportsgeist bei auftauchenden Problemen entsteht ein eigener, bisher völlig unbekannter Spirit.

- Delegierte aller wichtigen Stakeholder (u.a. aus Projektmanagement, Sales oder der Qualitatssicherung) koordinieren sich in den zweiwöchigen Queue-Replenishment-Meetings und legen dort die weitere Bearbeitung der Aufgaben fest.
- Daily Stand-ups werden ebenso regelmäßig durchgeführt wie die wöchentlichen Retrospektiven, zu denen nach vier Wochen das erste Mal auch Vertreter des Sales-Bereichs zur kritischen Prüfung eingeladen sind.
- Bereits die ersten Messungen zeigen Fortschritte, nach acht Wochen hat sich die durchschnittliche Durchlaufzeit um 70% verbessert.

Dennoch zeigt sich in diesen ersten Wochen auch sehr deutlich, dass vieles noch suboptimal läuft und weitere Verbesserungsschritte anstehen. Allein der Umstand, dass das große Kanban-Board für den gesamten Prozess in einem eigenen Raum hängt und weder für alle ständig sichtbar noch leicht zugänglich ist, kann nur als Übergangslösung gesehen werden. Die angezeigten Strukturänderungen, vor allem die Umsiedelung einiger Teams für eine größere räumliche Nähe zueinander wie zum Board, ist jedoch ein großer Veränderungsschritt, der ebenfalls gut vorbereitet und kommuniziert werden muss.

Kaizen ohne Unterlass

Im Betrieb stößt das nach wie vor sehr aktive "Kanban Change Team" um Frau Schreiber rasch auf weiteren Verbesserungsbedarf. Einerseits geht es dabei um technische Verbesserungen wie die Korrektur von zu hohen WIP-Limits, die Einführung einer neuen Serviceklasse oder eine bessere Bearbeitung der Metriken. Andererseits geht es auch um die weitere Optimierung der Zusammenarbeit, die durch besser vorbereitete Retrospektiven, durch rotierende Moderationsverantwortung oder das Einfordern von mehr Kommunikationsdisziplin in Angriff genommen wird. Kaizen bleibt in diesem Sinne nicht nur ein schönes Wort, sondern wird auch aktiv gelebt.

Auch die "Kanban-Mission" von Frau Schreiber ist mit der Inbetriebnahme keinesfalls zu Ende. Weder findet die kontinuierliche Verbesserungsarbeit von selbst statt noch ändert sich automatisch die Unternehmenskultur. Vielmehr braucht es im Betrieb von Kanban ebenso professionelle Führungsarbeit wie in der Einführungsphase. Insbesondere die Arbeit an der Fehlerkultur hat für Frau Schreiber eine besonders hohe Priorität. Schließlich merkt sie an vielen kleinen Auseinandersetzungen, dass sich die frühere Praxis der wechselseitigen Schuldzuweisung noch nicht in Luft aufgelöst hat. Dieses und weitere Probleme in kleinen Etappen zu lösen, ist für Frau Schreiber unerlässlich, um im Sinne des evolutionären Change Managements kontinuierlich den Kanban-Prozess zu verbessern…

Fazit

Um Kanban bestmöglich einzusetzen, braucht es nicht bloß ein fundiertes Wissen um spezielle Kanban-Praktiken. Vielmehr ist ein profundes Verständnis von Veränderungsgesetzen und professioneller Führung vonnöten. Schließlich handelt es sich bei Kanban um eine Methode des Change Managements.

Evolutionäre Veränderung beginnt mit einem umsichtigen Einführungsprozess. Die Art, wie Probleme identifiziert, systemische Lösungen vorbereitet und alle relevanten Stakeholder zu einer gemeinsamen Verbesserungsarbeit eingeladen werden, stellt schon früh die Weichen für den Erfolg einer Kanban-Initiative. Wird auf diese Weichenstel-

lung verzichtet und professionelle Vorbereitung vernachlässigt, droht diese Initiative leicht in der Sackgasse zu landen. Die "Just-do-it"-Philosophie reicht definitiv nicht aus, um nachhaltige Verbesserungen auf den Weg zu bringen.

Das von uns vorgestellte Phasenmodell kann diesbezüglich als "Navigationshilfe" dienen. Schritt für Schritt werden die notwendigen Klärungen vorgenommen und Wissen aufgebaut. Professionell geführt, bringt dieser Prozess ein hohes Maß an Aufmerksamkeit und Kommunikation mit sich. Und verwirklicht, ganz nebenbei, selbst ein lebendiges Stück Kaizen.

Literatur

- Leopold, Klaus; Kaltenecker, Siegfried: Kanban in der IT, Hanser Verlag, 2012
- Kanban in der IT, Die Webseite zum Buch: www.kanbaninit.com, zuletzt eingesehen am 29.6.2012
- Roock, Arne: Software-Kanban – eine Einführung, Projekt Magazin 04/2011
- Roock, Arne: Scrum und Kanban sinnvoll kombinieren, Projekt Magazin 14/2011
- Platform for Agile Management: www.p-a-m.org, zuletzt eingesehen am 2.7.2012
- Andrezak, Markus: Enterprise Kanban at mobile.de, www.slideshare.net/mandrezak, zuletzt eingesehen am 2.7.2012, sowie www.portagile.com, zuletzt eingesehen am 2.7.2012

Fachartikel

Höhere Produktivität durch geringere Auslastung

Den Output des Teams mit Kanban optimieren

Herr Paulsen schwitzte wie verrückt. Nach dem letzten großen Projekt, das er als Projektleiter (mehr oder weniger) erfolgreich abschließen konnte, hatte er sich seinen Urlaub mehr als verdient. Also nahm er sich zwei Wochen frei, packte seine Koffer und setzte sich in sein Auto Richtung Süden. Aber schon nach ein paar Stunden auf der A7 passierte es: Die Autos wurden immer langsamer, der Verkehr immer zähflüssiger, und schließlich gab es einen Stau. Nichts ging mehr!

Kurz darauf begann das obligatorische Hupkonzert. Und zu allem Überfluss fiel auch noch seine Klimaanlage aus! In dieser Stress-Situation gingen seine Gedanken auf merkwürdige Reisen: "Aus Projektleiter-Sicht ist ein Stau eigentlich super, denn die Autobahn ist zu 100% ausgelastet. Effizienter geht es nicht! Ich wäre froh, wenn ich das einmal in einem meiner Projekte hinbekommen würde." Herr Paulsen schüttelte den Kopf, um diesen Gedanken zu verscheuchen, denn er war absurd. Staus waren eine Katastrophe, sie nützten niemandem. Punkt. Aber die Idee ließ ihn nicht mehr los. "Offensichtlich gibt es einen Unterschied zwischen einer Autobahn und einem Projekt. Auf einer Autobahn stellt eine hohe Auslastung ein großes Problem dar, in einem Projekt ist sie erstrebenswert", dachte er. Das war merkwürdig, denn er kam einfach nicht drauf, warum eine hohe Auslastung auf einer Autobahn negativ sein sollte, in einem Projekt hingegen positiv.

Dann wurde er von einem lauten Geheul aus seinen Gedanken gerissen: Ein Polizeiwagen mit Blaulicht fuhr rechts auf dem Standstreifen an ihm vorbei. "Ein Glück, dass wenigstens der Standstreifen nicht voll ausgelastet ist", dachte er, "denn sonst würde noch nicht einmal die Polizei oder ein Krankenwagen bei einem Notfall durchkommen." Dann war er sofort wieder bei seinen Projekten. Hier gab es auch immer wieder Notfälle, um die sich sofort jemand kümmern musste. "Wäre es nicht vielleicht auch eine gute Idee, hierfür eine Art Notfall-Team zur Verfügung zu haben, das nicht voll ausgelastet ist, um schnellstmöglich reagieren zu können?" Plötzlich kam wieder Bewegung in die Autoschlange, und der Stau löste sich auf. Ein Glück! Diese Gedanken waren einfach zu verrückt.

Autor

Dr. Arne Roock

studierte Germanistik und Politikwissenschaft, Prokurist der it-agile GmbH, Certified ScrumMaster (CSM), zertifizierter Trainer für Zeitmanagement

Kontakt: arne.roock@it-agile.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› So gelingt die Kanban-Einführung

› Software-Kanban – eine Einführung

in den Rubriken:

› Prozessmanagement

› Ressourcen

Service-Links

Dienstleister
› Prozessoptimierung

Termin
› Prozessmanagement

Buch
› Prozessmanagement

Auslastung als problematische Metrik

Die Auslastung der Mitarbeiter ist in vielen Organisationen nach wie vor die Haupt-Metrik, um Projekte zu steuern. "Je höher die Auslastung, umso erfolgreicher das Projektmanagement", lautet eine unausgesprochene Gewissheit aus dem Projektmanagement, die so fundamental ist, dass es schwer fällt, in anderen Kategorien zu denken. Dabei gibt es durchaus schwerwiegende Probleme, die mit der Optimierung der Auslastung verbunden sind. Zwei davon hat Herr Paulsen auf der Autobahn bereits bemerkt.

1. Zum einen führt eine zu hohe Auslastung dazu, dass die Geschwindigkeit heruntergeht. Und dies nicht linear, sondern ab einem gewissen Grad exponentiell! Durch die Warteschlangentheorie wissen wir schon seit geraumer Zeit, dass es grob fahrlässig ist, Netzwerke dauerhaft zu mehr als 80% auszulasten, weil sie ab diesem Punkt schlagartig langsamer werden. Erstaunlich, dass gerade wir in der IT diese Erkenntnisse komplett ignorieren, wenn es um Projektmanagement geht! Natürlich sind IT-Projekte keine Computer-Netzwerke, unsere Mitarbeiter sind ja auch keine Computer. Aber die Effekte, die sich durch zu hohe Auslastung ergeben, sind trotzdem sehr ähnlich.

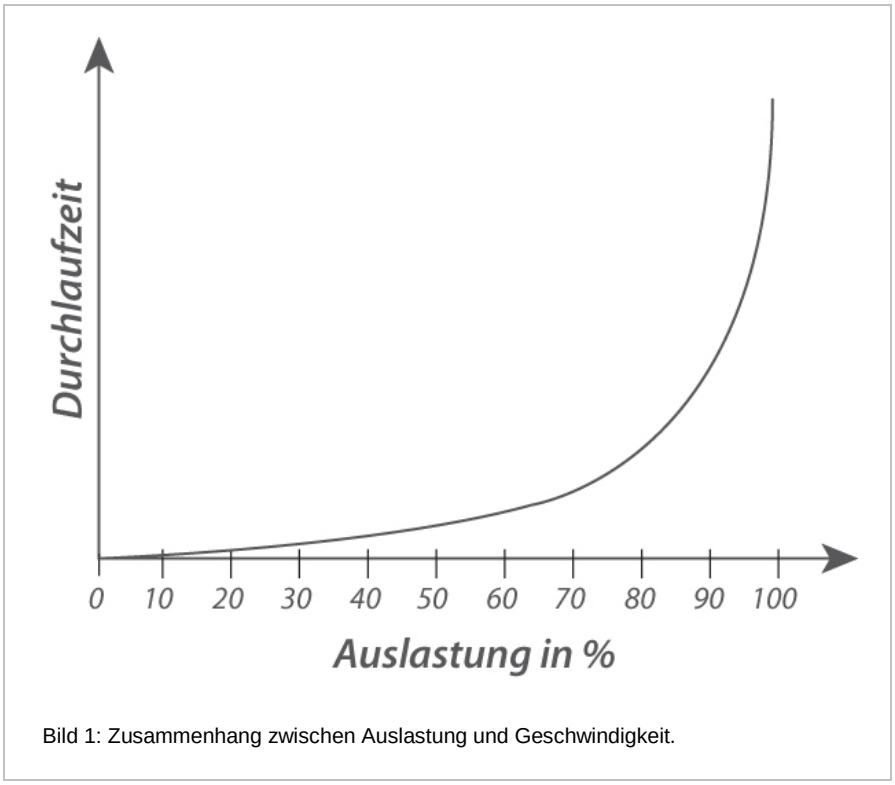

Bild 1: Zusammenhang zwischen Auslastung und Geschwindigkeit.

2. Eine hohe Auslastung führt zu geringerer Flexibilität. Wenn alle Mitarbeiter zu 100% ausgelastet sind und sich dann die Priorität der Aufgaben ändert (z.B. weil unser Kunde eine tolle neue Idee hat, die er sofort ausprobieren möchte), so müssen wir entweder warten, bis wieder Mitarbeiter verfügbar sind, oder es müssen begonnene Aufgaben unterbrochen werden (was Kosten bei der erneuten Einarbeitung verursacht). Das andere Extrem würde bedeuten, dass unsere Mitarbeiter zu 0% ausgelastet sind und sich in ständiger Bereitschaft befinden, um dann sofort auf solche Ereignisse reagieren zu können; dies wäre das Pendant zur Berufsfeuerwehr. 0% Auslastung ist sicherlich wirtschaftlicher Irrsinn, auch wenn es maximale Flexibilität bedeuten würde. Auf der anderen Seite scheint aber auch 100% Auslastung nicht immer optimal zu sein. Die Wahrheit liegt also – wie so oft – zwischen den beiden Extremen. Wo genau, hängt vom jeweiligen Kontext ab: Wie viel Flexibilität benötigen wir? Was ist sie uns wert? Wie groß sind die Kosten, die entstehen, wenn wir nicht flexibel genug sind?

Die beiden genannten Punkte dürften für sich schon ein wichtiges Argument gegen zu hohe Auslastung darstellen, aber es kommt noch schlimmer! Sehen wir uns Nummer 3 und 4 an.

3. Hinter der Optimierung der Auslastung steckt die unausgesprochene Vermutung: "Wenn alle Mitarbeiter voll ausgelastet sind, dann verbringen sie den größtmöglichen Anteil ihrer Arbeitszeit mit wertschöpfenden Tätigkeiten, was dann in der Summe maximale Wertschöpfung bedeutet." Diese Gleichung stimmt so allerdings nicht! Denn eine hohe Auslastung wird häufig durch viele parallele Aufgaben und somit viele Kontextwechsel erkauft ("In diesem Projekt gibt es für Herrn Meier gerade nichts zu tun, also stecken wir ihn vorübergehend in ein anderes Projekt"). Ein erheblicher Anteil der Arbeitszeit wird dann für Einarbeitung und unnötige Kommunikation bzw. Dokumentation verschwendet. Aber selbst, wenn wir diese Kontextwechsel einmal außer Acht lassen, gibt es ein Problem. Denn voll ausgelastete Mitarbeiter mögen effizient sein (Effizienz = "Die Dinge richtig tun"); das bedeutet jedoch noch lange nicht, dass sie auch effektiv sind (Effektivität = "Die richtigen Dinge tun"). Und genau dieser Fokus, sich auf die wirklich wichtigen Dinge zu konzentrieren, geht schnell verloren, sobald wir zu sehr auf die Auslastung achten. Denn das einfachste Mittel, um die Auslastung zu erhöhen, besteht darin, sich die nächstbeste bzw. einfachste Aufgabe zu nehmen und an ihr zu arbeiten. Dies wird aber höchstwahrscheinlich nicht die Aufgabe mit dem höchsten Business Value sein. Im schlimmsten Falle führt eine hohe Auslastung also nur dazu, dass wir die falschen Dinge schneller erledigen!

4. Eine hohe Auslastung macht Systemverbesserungen so gut wie unmöglich. Wenn wir eine Kultur der kontinuierlichen Verbesserung (Kaizen) etablieren möchten, dann müssen wir dafür sorgen, dass die Mitarbeiter "Slack-Zeiten" haben; also Zeit, in der sie nicht an ihren regulären Aufgaben arbeiten. Diesen Slack können sie dann verwenden, um sich selbst weiterzubilden, sich mit Kollegen auszutauschen, Wissensinseln abzubauen und auf andere Weise das System, in dem sie arbeiten, zu verbessern. All dies ist extrem wichtig, und wir alle wissen das, aber es wird ständig vom Tages- und Projektgeschäft verdrängt. Und selbst wenn wir uns diese Dinge in unseren To-Do-Listen notieren – solange wir keinen Slack haben, werden wir sie niemals angehen! Dies ist fatal, denn in der heutigen Zeit kann es sich eigentlich kaum noch ein Unternehmen erlauben, sich nicht ständig zu verbessern. Oder um es etwas zynisch mit Edwards Deming zu sagen: "It"s not necessary to change. Survival is not mandatory." ("Sich zu verändern, ist nicht notwendig. Man muss ja nicht überleben.")

Warum ist uns die Auslastung überhaupt so wichtig?

Eine hohe Auslastung verlangsamt also unser System und ist keinesfalls erstrebenswert, wie Herr Paulsen eingangs noch angenommen hatte. Sie kann zu geringerer Flexibilität führen, birgt die Gefahr, sich auf die falschen Dinge zu konzentrieren und verhindert Systembesserungen! Wenn auch nur die Hälfte dieser Punkte stimmt, stellt sich sofort die Frage, warum wir denn überhaupt so ein großes Gewicht auf die Auslastung legen?

Die Hauptaufgabe eines Projektmanagers besteht darin, dafür zu sorgen, dass Projekte erfolgreich durchgeführt werden. Metriken sollen der Projektleitung zeigen, dass das Projekt möglichst von Anfang an richtig läuft bzw. deutlich machen, wann gegengesteuert werden muss. Die Auslastung spielt in diesem Zusammenhang deshalb eine große Rolle, weil sie so einfach zu messen ist. Dabei ist die Auslastung der Mitarbeiter eigentlich gar nicht so interessant für den Erfolg des Projekts. Wichtiger sind vielmehr die Leistungsfähigkeit und der Fortschritt des Teams, der Abteilung oder der gesamten Organisation. Aber die Leistungsfähigkeit ist schwierig zu messen, also nehmen wir uns eine Proxy-Metrik (also eine Metrik, die etwas Anderes misst, als das, was wir eigentlich wissen möchten): die Auslastung.

Dies erinnert ein wenig an den alten Witz: Ein Betrunkener kriecht nachts auf allen vieren unter einer Straßenlaterne herum. Als ein Polizist ihn fragt, was er dort tut, antwortet er: "Ich suche meinen Wohnungsschlüssel." Der

Polizist forscht weiter: "Haben Sie ihn denn hier verloren?", worauf die Antwort lautet: "Nein, irgendwo dort hinten. Aber da ist es zu dunkel zum Suchen!"

Ein alternativer Ansatz: Flow

In den letzten Jahren hat Kanban als Methode für evolutionäres Change Management rapide an Bedeutung gewonnen und wird von immer mehr Unternehmen eingesetzt (s. auch "Software-Kanban – eine Einführung", Projekt Magazin 04/2011). Eine Wurzel von Kanban ist Lean, also das Toyota Production System. Aus dem Lean-Gedankengut wurde die Idee des "Flow" übernommen und auf die Wissensarbeit übertragen. Flow bedeutet zunächst einmal, dass unsere Arbeitspakete gleichmäßig durch unser System fließen (z.B. von der Analyse über Entwicklung und Test bis zu Integration und Deployment). Unser Ziel besteht also darin, kontinuierlich werthaltige Aufgaben an unsere Kunden (die ja auch interne Kunden wie z.B. andere Teams sein können) auszuliefern. Die Vorteile eines solchen Flows sind bestechend:

1. Unsere Kunden bekommen nicht am Ende eines Projekts einen großen Stapel an fertigen Aufgaben, sondern immer wieder kleinere Zwischenergebnisse. Im besten Fall können Sie diese bereits produktiv einsetzen und evtl. sogar schon Geld verdienen. Selbst wenn dies nicht der Fall ist, können sie häufig bereits im Produktivsystem getestet werden. Daraus ergibt sich nützliches Feedback für die Entwicklung aller kommenden Arbeitspakete.
2. Auch auf einer anderen fachlichen Ebene erhalten wir in diesem Flow-Modell wertvolles Feedback: Entwickeln wir überhaupt das System, das sich unser Kunde wünscht? Gab es Missverständnisse bei der Anforderungsdefinition? Können wir aus den ersten erledigten Aufgaben etwas für die kommenden Arbeitspakete lernen?
3. Kontinuierlich Dinge fertig zu stellen führt zu einer besseren Vorhersagbarkeit und Planbarkeit. Wir vermeiden hierdurch das bekannte Phänomen "Alle Aufgaben sind zu 80% fertig", was in der Regel zu bösen Überraschungen am Projektende führt.

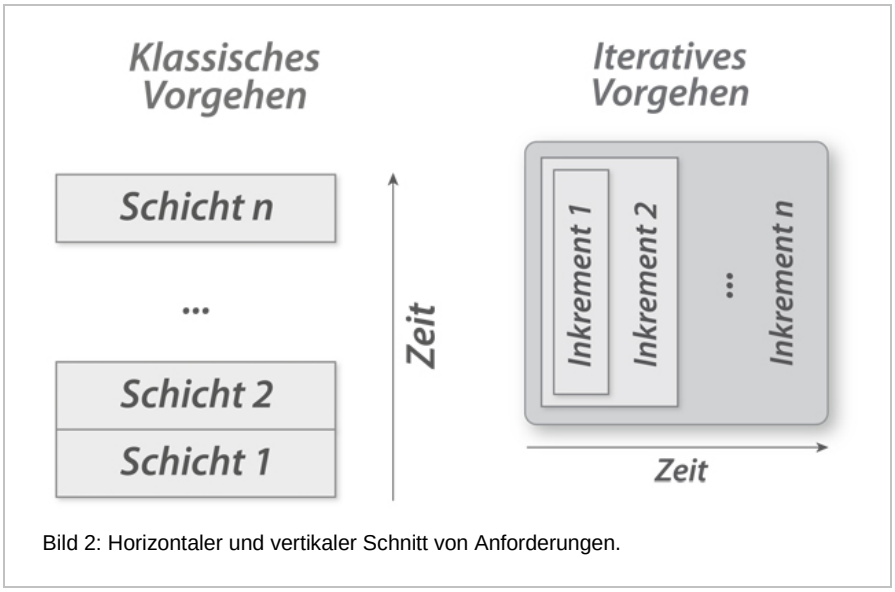

Bild 2: Horizontaler und vertikaler Schnitt von Anforderungen.

Dieses Flow-Modell ist in der Theorie einfach, in der Praxis jedoch schwierig umzusetzen und erfordert in der Regel eine längere Zeit. Die größte Herausforderung besteht im Zuschnitt der Anforderungen: Damit wir wirklich nützliches Feedback bekommen können und eine bessere Planbarkeit erhalten, ist es nämlich empfehlenswert, dass die Aufgaben vertikal, also fachlich geschnitten sind. Das bedeutet, dass wir unser System nicht, wie dies klassisch in der Regel geschieht, Schicht für Schicht entwickeln – also z.B. zuerst die Datenbank, dann die Fachlogik und am Schluss die Oberfläche. Stattdessen wird das System iterativ entwickelt; d.h. idealer-

weise sollte jede Anforderung die Form eines Inkrements haben, also die nötigen Teile *aller* Schichten enthalten (vgl. Bild 2). Ein Arbeitspaket, das in einer Änderung der Datenbank besteht, ohne dass diese Änderung auch an der Oberfläche sichtbar wird, erfüllt dieses Kriterium nicht oder nur sehr ungenügend. Dieses Arbeitspaket lässt sich schwerer testen, und seine Fertigstellung gibt kaum Aufschluss darüber, wie weit wir mit unserem Projekt bereits sind. Darüber hinaus – und dies ist vielleicht das größte Problem – kann der Endkunde uns zu so einem Arbeitspaket kein Feedback geben, weil ihm das technische Verständnis fehlt.

Verbesserung des Flows

... in der Theorie

Der Flow-Ansatz ist nichts, was man einmal "installiert" und dann nach einer gewissen Zeit erreicht hat. Vielmehr handelt es sich um ein Vorgehen zur kontinuierlichen Verbesserung: Egal wie gut oder schlecht unser Flow im Moment ist, wir können ihn stets ein Stück weit verbessern, um unsere Kunden früher und häufiger zufriedenzustellen (oder noch besser: zu begeistern), schneller Feedback zu erhalten und häufiger zu lernen.

Der Mechanismus hierfür ist verhältnismäßig einfach: Wir visualisieren den momentanen Flow an einem Kanban-Board und nehmen Messungen vor, um Möglichkeiten zur Verbesserung aufzudecken. Dann führen wir Maßnahmen durch, von denen wir uns eine Verbesserung des Flows versprechen. Spannend wird es nun, sobald die Maßnahmen Wirkung zeigen und die Aufgaben besser fließen. Dann sollten wir nämlich die "Latte ein Stück höher legen". Um dies zu tun besteht ein einfaches Mittel darin, den Work in Progress (WIP) zu reduzieren, also die Menge an angefangenen Aufgaben in unserem System. Nun wird der Flow vermutlich sofort ins Stocken geraten, wir erkennen neue Möglichkeiten zur Verbesserung, überlegen uns neue Maßnahmen usw.

... und in der Praxis

Schauen wir uns das einmal anhand eines Beispiels an. Als Herr Paulsen aus seinem Urlaub zurückkehrt, lässt ihn die Idee nicht mehr los, dass Flow eine entscheidende Optimierungsgröße ist. Also entscheidet er sich gemeinsam mit dem Team dafür, Kanban einzusetzen, um den Flow zu verbessern und somit häufiger und vorhersagbarer auszuliefern und die Kundenzufriedenheit zu erhöhen. Als erster Schritt werden der bestehende Workflow sowie alle aktuellen Aufgaben an einem Kanban-Board visualisiert (Bild 3).

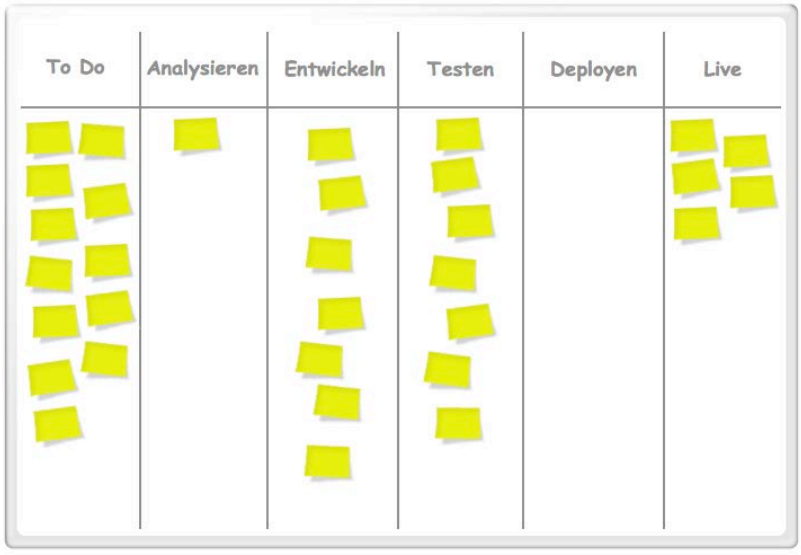

Bild 3: Am Kanban-Board zeigt sich, wo der Flow stockt.

An dem Board wird sofort deutlich, dass die Aufgaben extrem ungleich verteilt sind: In der Entwicklung staut sich Arbeit, ebenso im Test, während sich in der Analyse nur wenige Aufgaben befinden und im Deployment momentan gar nichts zu tun ist. Insgesamt befinden sich bei Analyse, Entwickeln, Testen und Deployen aktuell 15 Aufgaben in Bearbeitung; der Work in Progress (WIP) beträgt somit 15. Das Team einigt sich darauf, dass es nicht noch mehr Aufgaben gleichzeitig bearbeiten möchte und legt folglich einen maximalen WIP von 15 für das System fest. D.h. bevor die Teammitglieder eine weitere Aufgabe beginnen, muss zuerst eine andere Aufgabe abgeschlossen sein.

Feinere Visualisierung

Nun kann es verschiedene Gründe für den Stau beim Entwickeln und Testen geben, der auf dem Board sichtbar wird: Vielleicht ist es auch nur ein kurzfristiges Phänomen, das sich nach einiger Zeit von selbst wieder erledigt. Um ein besseres Bild zu bekommen, entscheidet sich das Team, die Visualisierung weiter zu verfeinern. Zum einen werden nun die Aktivitätsspalten in "Doing" und "Done" unterteilt, um zu sehen, an welchen Aufgaben gerade aktiv gearbeitet wird und welche nur auf den nächsten Prozessschritt warten (Bild 4). Dabei zeigt sich, dass die meisten Aufgaben im Test bereits fertig getestet sind und nur darauf warten, deployed zu werden. Dies scheint Herrn Paulsen ein Indiz dafür zu sein, dass das Deployment zu selten stattfindet. Nun wäre es allerdings voreilig, einfach zu entscheiden, häufiger zu deployen. Weil der Deployment-Prozess kaum automatisiert ist, wäre es betriebswirtschaftlicher Irrsinn, ab sofort wöchentlich oder sogar täglich auszuliefern. Die spannende Frage lautet also: "Was können wir tun, um die Transaktionskosten für ein einzelnes Deployment zu senken?" Darüber möchte Herr Paulsen sich einmal mit einem Admin unterhalten.

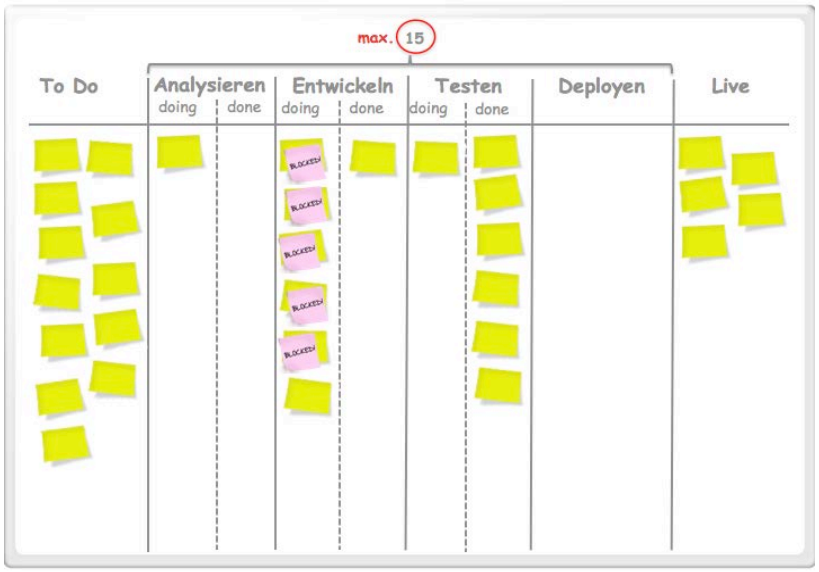

Bild 4: Die roten Blockade-Sticker und die volle Wartespalte nach dem Test weisen auf Verbesserungspotenzial hin.

Als weitere Maßnahme werden Blockaden durch rote Haftnotizen sichtbar gemacht. Dadurch wird deutlich, warum sich so viele Aufgaben in der Entwicklung befinden: fünf von sieben Tickets sind blockiert, d.h. die Entwickler können gerade nicht aktiv an diesen Aufgaben weiterarbeiten. Interessanterweise ist auf vier dieser fünf Tickets zu lesen: "Warten auf Zuarbeit von Team Green".

Probleme erkennen und lösen

Als Reaktion darauf, dass die Entwickler immer wieder auf diese Zuarbeit warten mussten, hatten sie stets neue Aufgaben begonnen, bis auch diese blockiert waren. Herrn Paulsen war zwar bewusst gewesen, dass es Abhängigkeiten zwischen seinem Team und Team Green gab, aber dass die Auswirkungen so groß sind, verschlägt ihm schon die Sprache! Als Projektleiter sieht er es als eine seiner Hauptaufgaben an, dem Team dabei zu helfen, Probleme zu lösen.

Er entwickelt verschiedene Ideen, wie man das Problem dauerhaft beheben könnte, z.B. indem man regelmäßige Abstimmungsmeetings zwischen beiden Teams etabliert; oder indem ein Teammitglied aus Team Green im Team von Herrn Paulsen mitarbeitet; oder indem sein Team sich selbst die Skills aneignet, über die im Moment anscheinend nur Team Green verfügt. Folglich setzt er einen kurzen Workshop an, um diese Ideen mit seinem Team zu diskutieren und verabredet sich mit dem Projektleiter von Team Green zum Essen, weil sich aus dem Workshop wahrscheinlich Änderungen in der Zusammenarbeit beider Teams ergeben werden.

Drei Monate später hat sich einiges getan: Eine Taskforce aus Entwicklern und Admins hat sich gebildet und mit der Automatisierung des Deployment-Prozesses begonnen. Das Team kann nun regelmäßig alle zwei Wochen ausliefern – demnächst vielleicht sogar wöchentlich. Außerdem gibt es eine engere Zusammenarbeit mit dem Team Green: Jeweils ein "Abgesandter" aus jedem Team besucht die Standup-Meetings des anderen Teams. Außerdem werden gemeinsame Planungsmeetings durchgeführt. Dadurch haben sich die Wartezeiten durch Team-Abhängigkeiten bereits deutlich verkürzt. Der Flow ist also spürbar besser geworden.

Den WIP reduzieren

Aber das Team möchte sich immer weiter verbessern. Also beschließt es gemeinsam mit Herrn Paulsen, das WIP-Limit von 15 auf 11 zu reduzieren. Sofort gerät der Flow wieder ins Stocken, denn die Tickets stauen sich in der Spalte "Testen". Der Grund dafür liegt darin, dass ein wichtiger Experte für Integrationstests dem Team nur einen Tag pro Woche zur Verfügung steht. Also werden Ideen gesammelt, wie sich dieses Problem beheben lässt, um so den Flow wieder ein Stück zu verbessern. Das Team entscheidet sich dafür, ein kleines Experiment durchzuführen: Würde es helfen, wenn der Experte nicht einen ganzen Tag in der Woche für das Team arbeiten würde, sondern zwei halbe Tage? Herr Paulsen bespricht diesen Vorschlag in der Projektleiter-Runde und bekommt grünes Licht. Nach vier Wochen nimmt das Team eine Auswertung des Experiments vor. Und siehe da! Der Flow hat sich durch diese einfache Maßnahme tatsächlich deutlich verbessert.

Den Flow verbessern, egal was die Auslastung sagt

In diesem Beispiel ist es dem Team von Herrn Paulsen gelungen, die Leistungsfähigkeit ihres Systems deutlich zu verbessern. Sie können nun häufiger ausliefern, planbarer arbeiten, Risiken durch kürzere Feedbackschleifen reduzieren und (wahrscheinlich) die Kundenzufriedenheit erhöhen. Die Haupt-Antriebsfeder für diese Verbesserungen ist Flow; die Auslastung der Mitarbeiter spielt hingegen keine Rolle. Und höchstwahrscheinlich wird die Auslastung sogar sinken: Indem das Team seinen WIP reduziert, wird es immer wieder vorkommen, dass einzelne Mitarbeiter nicht an ihren regulären Aufgaben arbeiten können. Es entsteht also Slack.

Diese Slack-Zeiten können dann dazu verwendet werden, um beispielsweise das System zu verbessern, Kollegen zu helfen oder an Automatisierungen zu arbeiten. Slack stellt also kein Problem dar, sondern ist eine notwendige Voraussetzung für die kontinuierliche Verbesserung!

Staffellauf

Vielleicht ist es aber auch gar nicht nötig, sich von der Auslastung als Metrik zu verabschieden, sondern wir müssen nur den Gegenstandsbereich ändern. Wie wir gesehen haben, bringt eine hohe Auslastung von Menschen große Probleme mit sich. Wenn wir aber statt der Menschen die Aufgaben in den Fokus nehmen, dann ergibt sich daraus eine nützliche Perspektive. So lange ständig an den Aufgaben gearbeitet wird (und zwar an den Aufgaben mit der höchsten Priorität), ist es erst mal nebensächlich, ob die Menschen ausgelastet sind: Besser Menschen warten auf Arbeit als andersherum! Dieser Punkt wird sehr schön deutlich, wenn man sich einen Staffellauf vorstellt.

Angenommen, jeder einzelne Läufer gibt sein Bestes und läuft Rekordzeit. Haben wir dann automatisch ein erfolgreiches Team? Nein, denn die Einzel-Zeiten sind völlig irrelevant. Was einzig zählt, ist die Gesamt-Zeit, die das Staffelholz unterwegs ist. Wenn nämlich der erste Läufer am Ende seines Abschnitts das Staffelholz einfach auf den Boden wirft und der zweite Läufer es erst suchen und aufsammeln muss, ist das schlecht für unsere Zeit. Noch schlechter ist es, wenn zwischen den Läufern ein großer Stapel mit Staffelhölzern liegt. Wenn nun ein keuchender Läufer ankommt und sein Holz übergeben möchte kann er dies nicht, denn der nächste Läufer sagt zu ihm: "Tut mir leid, aber ich muss erst einmal diese 84 anderen Staffelhölzer wegbringen, dann kommt dein Holz an die Reihe!"

Das hört sich absurd an? Beim Staffellauf sicherlich. Aber in Projekten arbeiten wir genau so! Wir achten auf eine hohe Auslastung der Mitarbeiter und effizientes Arbeiten, aber wir verschwenden wahnsinnig viel Zeit, weil Aufgaben regelmäßig lange Zeit herumliegen, ohne dass an ihnen gearbeitet wird. Und es kommt noch schlimmer! Denn besonders viel Zeit wird in der Regel bei Übergaben verschwendet. Typisch ist dieses Szenario: Die Entwickler geben ihr Bestes und arbeiten sehr effizient ihre Aufgaben ab. Danach werden sie an die Tester übergeben. Diese sind jedoch noch mit anderen Aufgaben beschäftigt, also warten die neuen Tickets erst einmal in einer Warte-Spalte. Jetzt nehmen die Tester die nötigen Tests vor (wiederum sehr effizient) und schieben die Tickets danach in eine Pufferspalte mit einem Namen wie "Bereit für Deployment". Weil aber nur alle 4 Wochen deployed wird, liegen die Tickets hier manchmal 20 Tage oder länger herum, ohne dass an ihnen gearbeitet wird! In so einer Situation ist es nur ein Tropfen auf dem heißen Stein, an der Effizienz der Entwickler und der Tester zu arbeiten. Stattdessen sollte man sich ernsthaft Gedanken darüber machen, was man tun müsste, um häufiger zu deployen.

Hinzu kommt, dass Aufgaben häufig blockiert sind, weil wir auf die Zuarbeit oder Informationen von anderen Teams oder Stakeholdern warten. Und schließlich verschwenden wir Zeit mit Nacharbeiten. Zeitdruck, unklare Regeln und schlechte Kommunikation führen dazu, dass Aufgaben hin- und herwandern und immer wieder angefasst werden müssen. Hierdurch entstehen unnötig viele Übergaben und Einarbeitungsaufwände. All diese Phänomene bekommen wir in den Blick, wenn wir uns auf die Auslastung der Aufgaben konzentrieren und von der Auslastung von Personen abrücken. In diesem Sinne ist der Slogan aus dem Lean-Thinking zu verstehen: "Watch the baton, not the runner!" ("Das Staffelholz beobachten, und nicht den Läufer!")

Zusammenfassung

Im Projektmanagement ist es immer noch üblich, die Auslastung der Mitarbeiter als wichtige Metrik zu verbessern, um den Zustand und den Fortschritt des Projekts zu überwachen. Die Auslastung ist zwar einfach zu messen, aber sie zeigt nicht das, was wir eigentlich sehen möchten. Denn wir können mühelos eine hohe Auslastung erreichen und trotzdem sehr ineffektiv arbeiten.

Interessanter als die Auslastung ist die Leistungsfähigkeit unseres Teams und unserer Organisation. Diese zeigt sich am Flow, also daran, wie schnell und wie regelmäßig wir werthaltige Aufgaben für unsere Kunden ausliefern. Durch einfache Visualisierungstechniken (z.B. wie sie im Rahmen von Kanban angewendet werden) können wir Flow-Probleme schnell sichtbar machen. An diesen Stellen verbirgt sich Verbesserungspotenzial.

Wie sich zeigt, ist eine hohe Auslastung schädlich für guten Flow. Denn zum einen verlangsamen wir durch Überlastung unser System und machen uns unflexibel. Zum anderen benötigen wir Slack (also freie Zeiten), um den Flow nachhaltig zu verbessern.

Fachbeitrag

Prozesse verschlanken, Leistung steigern

Scrum und Kanban sinnvoll kombinieren

Das agile Framework Scrum führt schon länger kein Nischen-Dasein mehr, sondern ist inzwischen in vielen Bereichen der Software-Branche akzeptiert und wird immer mehr zum Mainstream. Aber dem aufmerksamen Beobachter bleibt kaum eine Verschnaufpause – denn kaum hat er sich mit den (ziemlich revolutionären) Scrum-Ideen angefreundet, hört er immer öfter von Kanban. Besonders seit dem Erscheinen des Buches "Kanban" von David Anderson im Jahr 2010 (deutsche Übersetzung 2011) hat Kanban einen festen Platz in der Familie der Agilen Methoden eingenommen.

Gehört Scrum damit der Vergangenheit an? Müssen sich Teams und Unternehmen zwischen Scrum und Kanban entscheiden? Oder lassen sich Scrum und Kanban miteinander kombinieren? Dieser Artikel beschreibt Gemeinsamkeiten und Unterschiede von Scrum und Kanban und zeigt verschiedene Möglichkeiten auf, wie sich beide Ansätze sinnvoll kombinieren lassen.

Projektleiter und Manager, die ihre Teams und IT-Abteilungen agiler machen möchten, erhalten so eine Orientierung, welcher Weg für sie geeignet sein könnte. Und Teams, die bereits Scrum einsetzen, erhalten Anregungen, wie sie sich durch Kanban weiter verbessern können.

Autor

Dr. Arne Roock

studierte Germanistik und Politikwissenschaft, Prokurist der it-agile GmbH, Certified ScrumMaster (CSM), zertifizierter Trainer für Zeitmanagement

Kontakt: arne.roock@it-agile.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in der Rubrik:

› Agiles Projektmanagement
› PM im Unternehmen einführen
› IT-Projekte

Was ist Scrum?

Bei Scrum geht es darum, iterativ und inkrementell Software zu entwickeln (obwohl man Scrum im Prinzip auch für andere Zwecke verwenden kann, z.B. für die Planung einer Hochzeit). Entscheidend sind dabei die kurzen Feedback-Zyklen: Es werden neue, kleine funktionsfähige Häppchen (Inkremente) der Software erstellt und begutachtet (im besten Fall sogar gleich produktiv gestellt). Das Feedback, das man zu den letzten Inkrementen bekommt, geht dann in die weitere Entwicklung mit ein. Der Workflow von Scrum ist dabei so einfach gestaltet, dass er sogar auf einen Bierdeckel passt (Bild 1).

Der Gedanke dahinter ist, dass wir ein neues Software-Produkt niemals auf der "grünen Wiese" komplett spezifizieren können. Vielmehr starten wir mit den Dingen, die wir schon sicher wissen, sammeln unterwegs Feedback und spezifizieren dann nach und nach jeweils gerade so viel, wie wir für die nächsten paar Wochen benötigen.

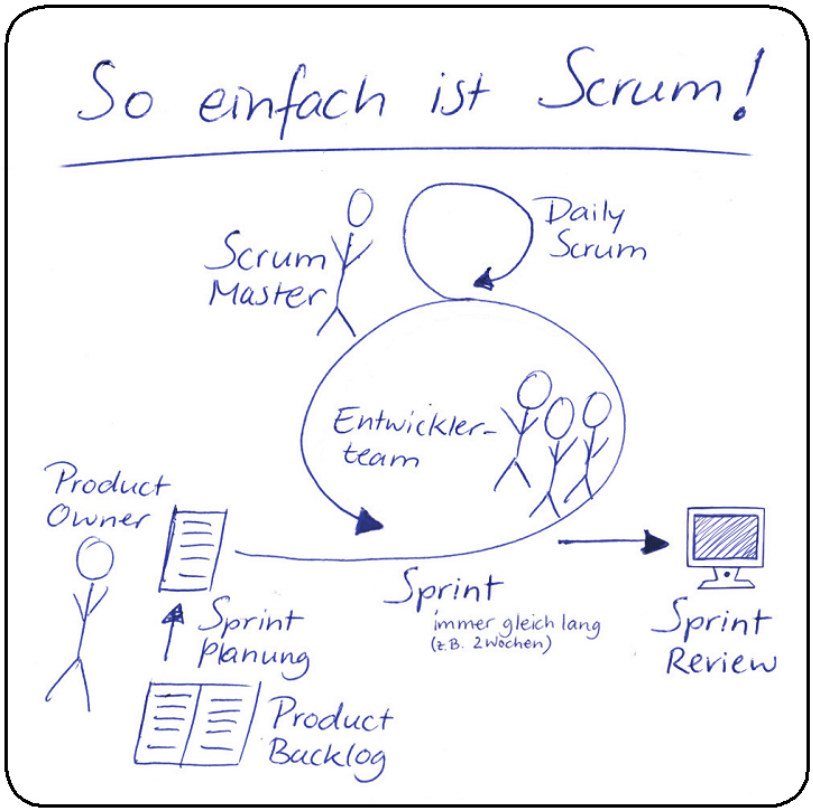

Bild 1: Die Grundlagen von Scrum sind so einfach, dass sie auf einen Bierdeckel passen.

Wie etliche Erfahrungsberichte zeigen, führen erfolgreiche Scrum-Implementierungen zu mehr Effektivität, höherer Qualität und deutlich fokussierterer Arbeit – und somit letztlich zu besseren Produkten und höherer Kundenzufriedenheit. Allerdings ist Scrum kein Selbstgänger, und in etlichen Fällen scheitern Scrum-Einführungen oder bleiben auf halbem Wege stehen (oft "ScrumBut" genannt). Der Grund dafür ist, dass es zwar sehr einfach ist, die sog. "Low hanging Fruits" von Scrum zu ernten (z.B. erhöhte Transparenz und bessere Kommunikation), aber es nicht gelingt, wirkliche organisatorische Änderungen durchzusetzen – vor allem das Zulassen und Fördern der Selbstorganisation und damit die Abkehr vom Command-and-Control-Management.

Ausführliche Informationen zu Scrum finden Sie in dem Artikel "Agiles Projektmanagement. Scrum – eine Einführung" von Ralf Wirdemann (Projekt Magazin 21/2009).

Was ist Kanban?

Ursprünglich war Kanban eine Methode zur Produktionssteuerung, die bei Toyota entwickelt wurde und in der Automobilherstellung längst zum Standard gehört. Wenn heute allerdings von Kanban im Zusammenhang mit der IT die Rede ist (oft auch "Software-Kanban" genannt), dann ist damit eine Methode des "Evolutionären Change Managements" gemeint, die David Anderson bei den Unternehmen Microsoft und Corbis entwickelt hat und die sich seit 2009 rapide um die ganze Welt verbreitet.

Nach dem Motto "Nur was wir sehen können, können wir auch verändern" beruht Kanban auf der Visualisierung der Wertschöpfungskette und der umzusetzenden Aufgaben. Dafür wird ein "Kanban-Board" erstellt, auf dem die verschiedenen Prozessschritte, die eine Aufgabe bis zur Fertigstellung durchläuft, als Spalten abgebildet sind. Als Board kann ein Whiteboard, eine Pinnwand oder auch ein elektronisches Tool dienen. Die Aufgaben werden als sogenannte "Tickets" dargestellt. Dies können z.B. Haftnotizen oder Karteikarten sein, auf denen die wichtigsten Informationen festgehalten sind.

Ein wesentlicher Bestandteil von Kanban ist "Kaizen": Es soll eine Kultur der kontinuierlichen Verbesserung etabliert werden, in der das Team permanent daran arbeitet, den Prozess zu verbessern. Dafür muss das Team Maßnahmen beschließen und umsetzen, die dazu führen, dass immer schneller ein immer größerer Geschäftswert für den Kunden ausgeliefert wird (wobei auch interne Kunden wie Fachabteilungen, andere Teams usw. gemeint sein können).

Ausführliche Informationen zu Kanban finden Sie in dem Artikel "Software-Kanban – eine Einführung" von Dr. Arne Roock (Projekt Magazin 4/2011).

Scrum und Kanban kombinieren

Bei der Frage nach dem Zusammenhang zwischen Scrum und Kanban kursieren allerlei Halbwahrheiten und arg vereinfachte Darstellungen, die häufig auf eine Gegenüberstellung der Unterschiede verkürzt werden und so ein "Entweder-Oder" nahelegen. Tatsächlich haben Teams und Organisationen auf der ganzen Welt inzwischen gezeigt, dass sich Scrum und Kanban auf vielfältige Weise sinnvoll miteinander kombinieren lassen.

Gemeinsamkeiten von Scrum und Kanban

Der Grund dafür liegt darin, dass Scrum und Kanban durch eine gemeinsame Denkweise geprägt sind, die viele elementare Prinzipien aus dem Lean Thinking und der agilen Softwareentwicklung enthält. So ist für beide Methoden grundlegend, dass die Teams fokussiert daran arbeiten, schnell und häufig Produkte mit einem hohen Geschäftswert für den Kunden auszuliefern. Um dies zu erreichen, ist es notwendig, "mit leichtem Gepäck" zu reisen, also den organisatorischen Aufwand auf das Nötigste zu reduzieren.

Deshalb versuchen sowohl Scrum als auch Kanban, mit einem Minimum an "Upfront-Planung" auszukommen – also ohne eine umfangreiche und langfristige Planung vor der eigentlichen Entwicklung. Stattdessen sollen Anforderungen "Just in Time" spezifiziert und nach einer strikten Priorisierung umgesetzt werden. Weiter ist es äußerst wichtig, schnelles Feedback zu den ersten Arbeitspaketen einzuholen und diese Lernerfahrungen immer wieder in die weitere Entwicklung zu integrieren.

Schließlich spielt das Prinzip der Einfachheit in beiden Methoden eine entscheidende Rolle: Prozesse, Tools und Arbeitspakete sollen so einfach wie möglich gehalten werden ("Do the simpliest thing that could possibly work"). Damit ist jedoch nicht gemeint, dass es einfach wäre, Scrum oder Kanban langfristig erfolgreich einzuführen. Denn dafür sind nach und nach verschiedene organisatorische Hürden zu nehmen, wie z.B. die Abkehr vom "Mikromanagement", also der ständigen Kontrolle von Einzelheiten, oder das Fördern von selbstorganisierten Teams.

Unterschiede von Scrum und Kanban

Betrachtet man die Unterschiede zwischen Scrum und Kanban, so fällt als erstes auf, dass Scrum eindeutig zur Familie der agilen Entwicklungsmethoden gehört, während Kanban terminologisch und konzeptionell eine deutliche Nähe zu Lean aufweist. Das wird z.B. daran deutlich, dass "Flow" in Kanban eine entscheidende Rolle

spielt und hier mehr Gewicht auf quantitatives Management gelegt wird. Damit ist gemeint, dass gezielt einfache Messungen durchgeführt werden (z.B. Durchlaufzeiten der Tickets), um anhand dieser Messungen Verbesserungspotenziale zu erkennen. Das Ziel der Flow-Verbesserung in Kanban erklärt auch, warum beispielsweise die Engpasstheorie hier eine größere Rolle spielt als in Scrum.

Was konkrete Rollen, Meetings und Artefakte angeht, macht Scrum deutlich mehr Vorgaben und gibt praktischere Anleitungen für die initiale Einführung. Insgesamt lässt sich der Hauptunterschied so zusammenfassen: Scrum ist ein Management-Framework, das bei konsequentem Einsatz zu tief greifenden Änderungen führen wird, während Kanban in erster Linie eine Change-Management-Methode ist. Führt man sich diesen Unterschied vor Augen, ist es nur allzu logisch, dass sich beide sinnvoll miteinander kombinieren lassen.

Lohnenswerte Kombination

Warum lohnt es sich, darüber nachzudenken, Scrum und Kanban miteinander zu kombinieren? Ein Hauptgrund besteht darin, dass Scrum sich in erster Linie auf die reine Entwicklung konzentriert, während Kanban darauf ausgelegt ist, weitere Teile der Wertschöpfungskette abzubilden und zu verbessern – also z.B. auch Prozesse, die vor der Entwicklung liegen (Upstream-Prozesse) wie Sales und Produktmanagement, und der Entwicklung nachgelagerte Prozesse wie Wartung und Systemadministration (Downstream-Prozesse).

Aus diesem Grund ist es oft auch einfach, eine Zustimmung des mittleren und oberen Managements für Kanban zu bekommen. Dies wird durch die Lean-Nähe von Kanban erleichert, da viele Konzepte des Lean Thinking schon seit Längerem im Top-Management bekannt sind. Neben der unternehmensweiten Prozessverbesserung ist Kanban zudem gut geeignet, um eine bestehende Scrum-Implementierung kontinuierlich zu verbessern und so nach und nach an den jeweiligen Kontext der Organisation anzupassen.

Räumliche Nähe der Teams ist von Vorteil

Sowohl Scrum als auch Kanban funktionieren am Besten bei Teams, die räumlich nicht getrennt arbeiten. Dies ist kein Wunder, denn *alle* Prozesse und Methoden, die wir kennen, funktionieren am Besten, wenn das Team sich an einem Ort befindet. Wenn eine räumliche Trennung unumgänglich ist, sollte darauf geachtet werden, dass die Kommunikation so direkt wie möglich ist, um auch tatsächlich Verbesserungen zu erreichen: Face-to-Face-Kommunikation ist immer besser als eine Videokonferenz. Diese ist immer noch besser als eine Telefonkonferenz. Und Telefonkonferenzen sind der E-Mail- oder Chat-Kommunikation vorzuziehen bzw. sollten durch diese ergänzt werden.

Häufig besteht die Tendenz, die Kommunikation im Wesentlichen auf elektronische Tools (z.B. Issue-Tracker) zu verlagern, z.B. indem Kommentare direkt in die Tickets geschrieben werden. Diesem Trend sollte man von Anfang an entgegenwirken, denn es lässt sich beobachten, dass hierbei Informationen verloren gehen können und es häufig zu Missverständnissen kommt. Und die Erfahrung zeigt, dass die Software-Entwicklung von verteilt arbeitenden Teams umso besser funktioniert, je öfter die Teams sich gegenseitig besuchen und ein paar Tage zusammenarbeiten – auch wenn die Kosten hierfür unverhältnismäßig hoch erscheinen.

Voraussetzungen für eine erfolgreiche Einführung

Um Scrum oder Kanban einzuführen, ist zu Beginn verhältnismäßig wenig Aufwand nötig: zwei bis drei Tage Schulung bzw. Workshop reichen aus, um die Mechanismen zu verstehen und erste eigene Schritte zu gehen. Allerdings zeigt sich immer wieder, dass es typische Fallstricke bei der Einführung gibt, die leicht übersehen werden: Sich keine Zeit für eine regelmäßige Reflexion und für Prozessverbesserungen zu nehmen, stellt wohl den häufigsten Fehler dar.

Darüber hinaus wird Scrum häufig vorschnell "auf die eigenen Bedürfnisse angepasst". Unangenehme Aspekte, wie z.B. strukturelle Veränderungen, werden einfach weggelassen. Was Kanban angeht, so lässt sich häufig ein "Over-Engineering" feststellen, d.h. in das initiale Board werden bereits alle Eventualitäten hineinmodelliert, anstatt mit dem bestehenden Prozess zu beginnen und diesen schrittweise zu verbessern.

! Um solche Fehler zu vermeiden, ist es empfehlenswert, sich regelmäßig Feedback von außen einzuholen. Dies muss nicht unbedingt ein externer Coach sein; häufig finden sich auch in anderen Abteilungen oder Geschäftseinheiten kompetente Kollegen, die die Funktion des außenstehenden Beobachters und Ratgebers übernehmen können.

Für eine Kombination von Scrum und Kanban in der Praxis werden nachfolgend vier erprobte Kombinationsmöglichkeiten näher beleuchtet.

Die ScrumBan-Story

Wie oben bereits dargestellt wurde, handelt es sich bei Scrum um ein Management-Framework, während Kanban eine Methode ist, um in kleinen Schritten systematische Veränderungen herbeizuführen. Um Kanban überhaupt einsetzen zu können, braucht man also einen Prozess, den man visualisieren und verbessern kann. Wenn man keinen definierten Prozess hat, wird dies durch Kanban sofort deutlich. Denn wie sollen jetzt die Spalten auf dem Board benannt werden? Und wie weiß ich, welche Tickets in welche Spalte gehören?

Kanban allein kann also nicht zur Entwicklung von Software verwendet werden, sondern setzt stets auf einen existierenden Prozess auf, um diesen zu verbessern. Allerdings ist es möglich, sich bei der Kanban-Einführung ad hoc auf einen sehr einfachen Prozess zu einigen und diesen dann allmählich zu verfeinern. Der einfachste Prozess besteht aus den drei Schritten "To Do" / "Doing" / "Done" (Bild 2).

Sprints verbessern

So lässt sich Kanban dafür verwenden, die Sprints (Umsetzung einer Iteration) in einer bestehenden Scrum-Implementierung zu verbessern. Diese Kombination wird als "ScrumBan" bezeichnet (Ladas, 2008). Dabei vereinbart das Scrum-Team explizite WIP-Limits für User Storys, an denen während des Sprints parallel gearbeitet werden darf. Dieses Vorgehen hat den Vorteil, dass am Sprint-Ende die wichtigsten Storys (Anforderungen) fertig gestellt sind, selbst wenn nicht alle geplanten Storys des Sprints umgesetzt werden konnten.

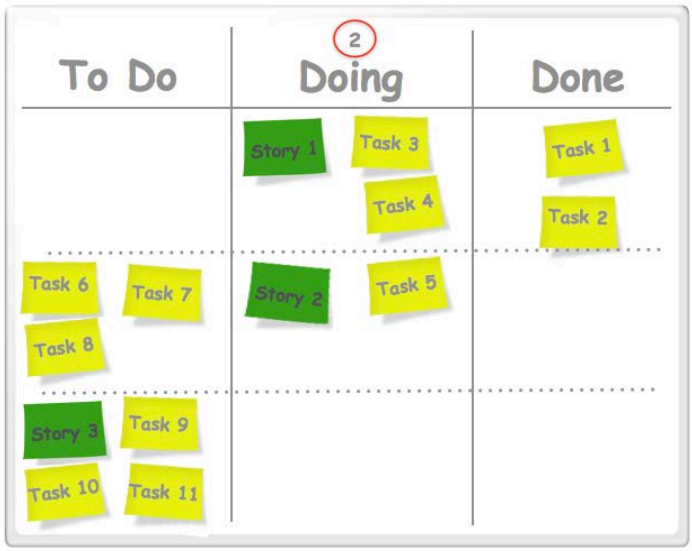

Bild 2:Scrum-Taskboard, bei dem ein Limit für User Stories eingeführt wurde, an denen gleichzeitig gearbeitet werden darf.

Wenn aber während des Sprints kontinuierlich Storys fertig gestellt werden, dann ist es oft eine gute Idee, dass der Product Owner sie auch einzeln während des Sprints abnimmt (und nicht gesammelt am Sprint-Ende). Diese Arbeitsweise ist vollständig kompatibel zu Scrum und wird durch eine Kombination mit Kanban befördert.

Mit der regelmäßigen Abnahme der Storys kann die Software-Entwicklung den Charakter eines kontinuierlichen Flows annehmen und das Team zu interessanten Diskussionen führen: Wenn wir regelmäßig einzelne Storys fertig stellen, ist es dann nicht auch sinnvoll, sie einzeln (oder in kleinen Bündeln) zu releasen? Was müssten wir dafür tun? Ist es vielleicht sinnvoll, einen Admin mit im Team zu haben, damit fertige Storys jeweils sofort ausgeliefert werden können? Möchten wir den Sprint-Rhythmus verkürzen? Oder Sprints ganz abschaffen? Müssen wir wirklich für alle Storys Aufwands-schätzungen abgeben? Oder reicht es vielleicht aus, die durchschnittliche Durchlaufzeit einer Story zu kennen?

Vielleicht kommt dabei nach einer Weile etwas heraus, das gar nicht mehr nach Scrum aussieht. Vielleicht wird aber auch deutlich, dass die bestehende Scrum-Implementierung Defizite aufweist und es sinnvoll ist, mehr nach "Scrum-by-the-Book" vorzugehen – also sich (vorerst) sehr strikt an die Scrum-Standards zu halten. In jedem Falle sollten durch die Kombination aus Scrum und Kanban schrittweise Verbesserungen sichtbar werden.

Die Wartungs-Story

Die Scrum-Einführung bei XING ab dem Jahr 2008 kann in gewisser Hinsicht als prototypisch für viele andere Organisationen angesehen werden (OBJEKTspektrum, 2010): Zuerst wurde Scrum bei einzelnen Entwicklungsteams eingeführt und nach ersten Erfolgen auf alle Entwicklungsteams ausgerollt. Nach einer Weile stellte sich dann fast naturgemäß die Frage, ob nicht auch das Wartungsteam nach Scrum vorgehen sollte.

An dieser Stelle zeigten sich dann aber einige Schwierigkeiten: Weil Wartungsteams mit einem großen Anteil sehr dringender, nicht-planbarer Aufgaben zu tun haben, sind zweiwöchige und selbst einwöchige Sprint-Rhythmen zu lang. Um flexibel genug auf Notfälle o.ä. reagieren zu können, bräuchte man eintägige Sprints oder sogar noch kürzere Zyklen. Dadurch würde aber der organisatorische Aufwand für die verschiedenen Meetings unverhältnismäßig hoch, denn es müssten täglich ein Standup Meeting, ein Sprint Planning, ein Sprint Review und eine Sprint-Retrospektive durchgeführt werden.

Hinzu kommt, dass Schätzungen bei den meisten Wartungsaufgaben überflüssig sind, denn die Aufgaben müssen in jedem Fall in einer bestimmten Reihenfolge erledigt werden. Schließlich arbeiten Wartungsteams nicht kontinuierlich an einem Produkt, sondern bekommen Aufgaben aus verschiedenen anderen Teams bzw. Abteilungen. Dadurch wird es sehr schwierig, sinnvolle Sprintziele zu definieren, hinter deren Umsetzung auch das Wartungsteam steht. Ohne solche Sprintziele verliert Scrum jedoch einen großen Teil seiner Stärke.

Die Lösung bei XING (und in anderen Organisationen wie z.B. mobile.international) bestand darin, für das Wartungsteam einen einfachen Ad-Hoc-Prozess zu definieren und diesen mit Kanban schrittweise zu verbessern. Dieser Prozess bestand aus den Schritten "Not started" / "in Progress" / "QA" (Qualitätssicherung) / "Done".

Serviceklassen abbilden

Für Wartungsteams (wie für alle Arten von Service-Teams) passt Kanban deshalb besonders gut, weil durch das Konzept der "Serviceklassen" elegante Möglichkeiten gegeben sind, um mit Notfällen und fixen Terminen umzugehen und trotzdem zu gewährleisten, dass kontinuierlich an wichtigen Aufgaben gearbeitet wird. Beispielsweise kann sich ein Team darauf einigen, dass Standard-Aufgaben durch gelbe Haftnotizen repräsentiert werden, Notfälle durch rote Haftnotizen und langfristige Projekte durch blaue Haftnotizen.

Nun definiert das Team Priorisierungsregeln für die verschiedenen Serviceklassen. Die gelben Tickets werden strikt von oben nach unten abgearbeitet, die roten Tickets haben Vorfahrt und werden sofort bearbeitet, sobald ein Teammitglied freie Kapazitäten hat. Die blauen Tickets schließlich sind wichtig, aber nicht dringend. Sie können bei Bedarf beiseite geschoben werden, um Platz für Notfälle zu machen. Um allerdings zu gewährleisten, dass dies nicht permanent geschieht, wird eine "feste Bandbreite" definiert, z.B. dass 20% der Gesamt-Kapazität des Teams darauf verwendet wird, an den blauen Tickets zu arbeiten.

Die Portfolio-Story

Kanban gibt zwar vor, dass es Tickets geben muss, die sich über das Board bewegen. Es macht jedoch keinerlei Aussagen darüber, welche Art von Aufgaben durch diese Tickets repräsentiert werden. In den ersten Kanban-Implementierungen hatten die Tickets in etwa die Granularität von kleineren User Storys oder Bugs. Was würde aber passieren, wenn wir in einer anderen Dimension denken und jedes Ticket für ein komplettes Projekt stünde? Ließe sich damit nicht mehr Transparenz für das Portfolio-Management herstellen und so auch ein besserer Flow unserer Projekte?

Diese Frage stellte sich im Jahr 2009 das Unternehmen mobile.international, Europas größte Online-Plattform für Gebrauchtwagen, als deren Entwicklungsteams bereits erfolgreich mit Scrum arbeiteten und dadurch ein scheinbar kaum zu verwirklichendes Migrationsprojekt (1:1-Vollmigration von Perl nach Java) gestemmt hatten.

Kanban in einer höheren Ebene

Nachdem die Entwicklungsteams Scrum erfolgreich einsetzten, wollte man auch das Portfolio-Management transparenter und flexibler gestalten. Also wurde ein Backlog von allen Projekten aufgebaut, die umgesetzt

werden sollten. Im Pull-Verfahren setzten die Teams dann ein Projekt nach dem anderen um – strikt nach den vorher festgelegten Prioritäten. Es handelt sich hier also um eine Kombination aus Scrum auf der Entwicklungs-Ebene und Kanban auf der Portfolio-Ebene.

Die Projekte stellen Tickets auf dem Kanban-Board dar und durchlaufen so von der Idee bis zur Livestellung die gesamte Wertschöpfungskette. Sobald ein Ticket in die Spalte "Entwicklung" gelangt, ist das jeweilige Entwicklungsteam, das sich das Ticket gezogen hat, für die Entwicklung zuständig. Das Projekt wird dann mit Scrum umgesetzt und in einzelne User Stories und Tasks "herunter gebrochen".

Das Portfolio-Board bei mobile.international sieht natürlich vollkommen anders aus als typische Kanban-Boards bei Entwicklungs- oder Wartungsteams: Als Spalten auf dem Board sind die Prozessschritte "Idea" / "Vision" / "Envisioning" / "Development" / "Live To Site" abgebildet. Eine sehr informelle Projektidee (Idea) wird in ein konkretes Ziel mit einer einfachen Produktvision (Vision) konkretisiert, bevor das Product Backlog mit einzelnen User Storys erstellt wird (Envisioning). Danach folgt die eigentliche Entwicklung (Development) und schließlich die Live-Stellung (Live To Site).

Portfolio-Kanban hat bei mobile.international nicht nur für mehr Transparenz und Flexibilität geführt, sondern auch zu einer stärkeren Fokussierung auf die jeweils wichtigsten Projekte, die dann mit Scrum jeweils schnell und agil umgesetzt werden konnten (zu Portfolio-Kanban s. auch "Feeding the Kanban" von Markus Andrezak in der deutschen Übersetzung von Anderson 2011).

Die Games-Story

Der Autor Clinton Keith beschreibt eine interessante Kombination aus Scrum und Kanban in der Spieleproduktion (Keith, 2010). Um z.B. ein neues Action-Spiel im großen Stil herzustellen, gibt es zwei unterschiedliche Phasen: Die "Pre-Production" und die "Production". Die Pre-Production ist durch ein hohes Maß an Kreativität und Ungewissheit gekennzeichnet: Bringt es mehr Spaß, wenn der Held in einem Auto oder auf einem Motorrad durch Paris rast? Soll er mit einer Eisenstange oder mit einem Ninja-Schwert angegriffen werden? Und wie detailgetreu muss der Eiffelturm aussehen, wenn der Held nur einmal kurz an ihm vorbeifährt? Um all diese Dinge herauszufinden, ist der iterative Ansatz von Scrum hervorragend geeignet: Die strikten Timeboxes zwingen zur Fokussierung und verhindern, dass sich die Teammitglieder in Details verlieren, von denen noch nicht einmal klar ist, ob sie jemals benötigt werden.

In der Production hingegen ist weniger Kreativität gefragt, sondern hier geht es in erster Linie darum, Dinge konkret umzusetzen und hochzuskalieren. Wenn z.B. klar ist, wie detailliert der Eiffelturm benötigt wird, dann gilt Ähnliches auch für das Louvre. Und wenn wir wissen, wie ein Porsche animiert wird, dann ist der Schritt zu einem Ferrari nicht mehr weit. Weil die Variabilität in der Production viel geringer ist als in der Pre-Production, sind hier Iterationen nicht nötig, und die Arbeit lässt sich nach einem Fluss-Prinzip organisieren, in dem die einzelnen Arbeitspakete schrittweise übergeben werden (z.B. vom Graphik- an den Sound-Designer). Die Production wird mit Kanban gesteuert und kontinuierlich verbessert, um sicherzustellen, dass hier kein Wasserfall mit all den bekannten Problemen entsteht, wie z.B. langsames Reagieren auf die Marktsituation oder ein sehr spätes Nutzer-

Feedback. Das Entwicklerteam stellt kontinuierlich einzelne Arbeitspakete fertig und erhält so stetig ein direktes Feedback.

Der Kontext ist entscheidend

Wie die verschiedenen Beispiele gezeigt haben, sind Scrum und Kanban keine exklusiven Alternativen, sondern beide Ansätze lassen sich auf verschiedene Weise sinnvoll miteinander kombinieren. Welche dieser Kombinationen im Einzelnen sinnvoll ist, hängt natürlich vom jeweiligen Kontext ab:

- Wenn Sie für die Entwicklung bereits Scrum einsetzen, aber Ihre Scrum-Teams Schwierigkeiten haben, am Ende jedes Sprints fertige User Storys abzuliefern, dann sollten Sie vielleicht über einen ScrumBan-Ansatz nachdenken, um ein fokussierteres Arbeiten innerhalb der Sprints zu erreichen.
- Für die Skalierung von Scrum von den reinen Entwicklungsteams auf Wartungsteams ist es oft ratsam, über einen Kanban-Ansatz für die Wartungsteams nachzudenken. Hierfür lässt sich häufig ein einfacher Ad-Hoc-Prozess definieren (wenn es nicht schon einen gibt), um diesen mit Kanban zu steuern und zu verbessern. So lässt sich besser mit Notfällen und unplanbaren Aufgaben umgehen als mit Scrum.
- Falls Sie mit der Arbeit Ihrer Scrum-Teams zufrieden sind, aber mehr Transparenz, Flexibilität und Fokussierung auf einer Projekt- oder Programm-Ebene wünschen, dann ist es eine gute Idee, sich näher mit Portfolio-Kanban zu beschäftigen. So können Sie die gesamte Wertschöpfungskette visualisieren und neben der eigentlichen Entwicklung auch das Produktmanagement und das Deployment mit in den Blick bekommen. Portfolio-Kanban ist darüber hinaus auf einer Abstraktionsebene angesiedelt, die auch für das obere Management interessant ist.
- Haben Sie innerhalb Ihrer Entwicklung unterschiedliche Arten der Umsetzung, wie Pre-Production und Production, dann sollten Sie sich näher mit der Scrum-Kanban-Kombination beschäftigen, wie sie Clinton Keith beschreibt.

Allerdings sollten Sie sich davor hüten, einfach einen Ansatz 1:1 zu kopieren, der bei einer anderen Organisation gut funktioniert! Jede Organisation agiert in ihrem eigenen Umfeld und hat mit unterschiedlichen Herausforderungen zu kämpfen. Arbeiten Sie also immer an Ihrer eigenen Lösung!

Womit anfangen?

Auch wenn Scrum und Kanban sich keineswegs ausschließen, so kann es eine große Herausforderung darstellen, beides gleichzeitig einzuführen. Wenn Sie vor der Wahl stehen, welchen Schritt Sie zuerst gehen möchten, dann können die folgenden Fragestellungen interessant sein:

- Müssen Sie Ihre Entwicklung sehr schnell Verändern, und bietet sich jetzt die Chance – oder der Zwang dazu, z.B. weil die Konkurrenz Sie abzuhängen droht? Dann denken Sie im ersten Schritt über eine konsequente Scrum-Einführung nach und lassen Kanban in einem späteren Schritt folgen.
- Möchten Sie mehr Teile Ihrer Wertschöpfungskette verändern und nicht nur die reine Software-Entwicklung? Oder sind die letzten Change-Initiativen gescheitert, und Sie möchten jetzt nachhaltige Veränderungen in

kleinen Schritten ohne größere Widerstände einführen? Dann ist es wahrscheinlich die bessere Wahl, mit Kanban zu beginnen.

Typische Fallstricke

Die Prinzipien von Scrum und Kanban sind recht simpel – die erfolgreiche Einführung ist es nicht! Anforderungen jetzt User Storys zu nennen, macht noch lange kein Scrum. Und Haftnotizen an ein Whiteboard zu hängen, macht nur einen sehr kleiner Teil von Kanban aus! Beide Ansätze werden – wenn auch auf unterschiedlichen Wegen – zu Veränderungen in Ihren Teams und Ihrer Organisation führen. Wenn Sie diese Veränderungen nicht ernsthaft wollen, dann ist weder Scrum noch Kanban das Richtige.

! Hilfreich ist es z.B. mit Kunden, Partnern oder Beratern zu sprechen, die bereits Erfahrung mit der Einführung von Scrum und Kanban haben, um ein besseres Bild zu bekommen, welche Konsequenzen sich durch die Einführung ergeben können.

Kaizen nicht vergessen

Was die (vermeintliche) Kombination von Scrum und Kanban angeht, so lässt sich in letzter Zeit häufig ein schlechter Lösungsansatz (sog. "Anti-Pattern") beobachten: Bei einer Scrum-Einführung zeigen sich schnell erste Erfolge. Aber dann kommt ein Punkt, an dem der weitere Fortschritt mühsam wird und unangenehme Veränderungen verlangt. Viele Teams und Organisationen scheuen vor diesen Veränderungen zurück, begeben sich zurück in ihre Komfortzone und nennen diese halbherzige Scrum-Implementierung dann Kanban.

Natürlich ist es vollkommen legitim, nur die "Quick Wins" von Scrum mitzunehmen, wenn man dadurch die gewünschten Ergebnisse erzielt. Wenn danach jedoch noch nicht einmal kleine Verbesserungen angestrebt werden, dann sollte man diesen Zustand nicht Kanban nennen, denn dies widerspricht dem Kaizen-Gedanken, der untrennbar mit Kanban verbunden ist. Wenn der Kaizen-Ansatz nicht ernst genommen wird, gerät der Prozess nach einiger Zeit nicht nur ins Stocken, sondern er wird vielleicht sogar immer weiter degenerieren, so dass die gewonnenen Vorteile nur kurzfristige Erfolge sind.

Fazit

Scrum und Kanban sind keine strengen Alternativen in dem Sinne, dass man sich für eines entscheiden müsste. Wie viele Beispiele aus der Praxis zeigen, sind eine Reihe sinnvoller Kombinationen möglich. Die Einführungsstrategien unterscheiden sich jedoch in der Regel deutlich voneinander: Während Scrum große Veränderungen zu Beginn nötig macht, bevorzugt Kanban eine evolutionäre Strategie, in der Änderungen in kleinen Schritten durchgeführt werden. Je nach Kontext sollte man deshalb unterscheiden, in welcher Weise Scrum und Kanban kombiniert werden sollen und welches Vorgehen zuerst eingeführt wird.

Allerdings gilt sowohl für Scrum als auch bei Kanban: Über die Zeit wird es zu größeren Veränderungen kommen. Darüber sollten Sie sich bewusst sein!

Literatur

- Anderson, David (2011): Kanban. Evolutionäres Change Management für IT-Organisationen, dpunkt Verlag, 2011
- Keith, Clinton: Agile Game Development with Scrum, Addison-Wesley Longman, 2010
- Kniberg, Henrik; Skarin, Mattias: Kanban and Scrum. Making the most of both, 2010
- Ladas, Corey: Scrumban. Essays on Kanban Systems for Lean Software Development, 2008
- OBJEKTspektrum: Kanban bei XING: Wer am Ziel ist, irrt sich, Ausgabe 2/2010
- Dr. Roock, Arne: "Software-Kanban - eine Einführung", Projekt Magazin 4/2011
- Wirdemann, Ralf: "Agiles Projektmanagement. Scrum – eine Einführung", Projekt Magazin 21/2009

Fachbeitrag

Lesson learned

Zehn Jahre agil – das wurde teuer!

Vor rund zehn Jahren führten wir mit großer Begeisterung bei der microTOOL GmbH, einem Berliner Softwarehersteller, agile Software-Entwicklung ein. Wir entwickelten die neue Version eines wichtigen Produkts nicht mehr nach traditionellen Vorgehensweisen wie Wasserfall oder "Command & Control", sondern mit eXtreme Programming (XP). Später führten wir Scrum ein und die Erfolge bestätigten uns in dieser Entscheidung. Noch vor drei Jahren hätte unser Erfahrungsbericht über agile Software-Entwicklung deshalb den Titel "Sieben Jahre agil – das hat sich gerechnet" getragen.

In den letzten drei Jahren wandelte sich jedoch unsere ursprüngliche Begeisterung in eine sehr differenzierte und kritische Sichtweise. Wir mussten lernen, dass eine rein agile Vorgehensweise langfristig betrachtet erhebliche Gefahren für die Qualität des Produkts birgt. Die Konsequenz hieß für uns allerdings nicht, zu den alten Methoden zurückzukehren, sondern Scrum so zu modifizieren, dass einheitliche Qualitätsstandards gewährleistet sind. Im Folgenden schildern wir unsere Erfahrungen mit agiler Software-Entwicklung und beschreiben unsere Lösung für das Dilemma zwischen Agilität und Produktqualität.

Ausgangspunkt: die Werte unserer Unternehmenskultur

Das mittelständische Unternehmen microTOOL ist auf die Entwicklung von Software-Werkzeugen für Requirements Engineering sowie integriertes Projekt- und Anforderungsmanagement spezialisiert. Vier grundlegende Werte prägen unsere Unternehmenskultur und damit auch unsere Vorgehensweisen bei Produktentwicklung und Projektmanagement:

Vertrauen der Kunden und Anwender

Wie bei allen Softwareproduzenten ist auch für uns das Vertrauen, das Kunden und Anwender uns entgegenbringen, ein zentraler Wert. Dieses Vertrauen zahlt sich letztlich in Lizenz-, Wartungs- und Dienstleistungsumsatz aus.

Autor

Ursula Meseberg

Dipl.-Math, Mitbegründerin, bis 2013 Geschäftsführerin der microTOOL GmbH

Kontakt:
Ursula.Meseberg@microtool.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Agiles Projektmanagement. Scrum – Eine Einführung

sowie in den Rubriken:

› Agiles Projektmanagement

› IT-Projekte

› Informations- und Kommunikationstechnologie

Service-Links

Dienstleister
› PM-Einführung und –optimierung

Software
› Agiles Projektmanagement

Termin
› Agiles Projektmanagement

Buch
› Agiles Projektmanagement

Qualität von Architektur und Code

Um Vertrauen zu gewinnen und dauerhaft zu erhalten, ist eine hohe Qualität der Software-Produkte unverzichtbar. Für uns spielt bei der Qualität von Architektur und Code die Erweiterbarkeit eine besondere Rolle. Durch sie haben wir den entscheidenden Wettbewerbsvorteil, kundenspezifische Anpassungen und Erweiterungen unserer Produkte anzubieten zu können.

Wissen und Erfahrung

Um Software mit hoher Qualität zu entwickeln, braucht man das Wissen und die Erfahrung von Mitarbeitern. Da Mitarbeiter wechseln, ist es notwendig, das Wissen im Unternehmen auch personenunabhängig zu halten. Dafür verwenden wir unter anderem ein Firmenwiki und Entwickler-Blogs. Aber dokumentiertes Wissen droht immer zu veralten. In den Algorithmen zur automatisierten Softwareproduktion steckt das Knowhow über die technische Architektur unserer Produkte, dies ist das wertvollste Wissen von microTOOL.

Kultur des Miteinanders und Soft Skills

Und schließlich ist für jedes Unternehmen die Kultur des Miteinanders wichtig: Wie wird kommuniziert? Wie werden Lösungen entwickelt? Wie löst man Konflikte? Wie geht man mit Krisen um? Brainstorming, moderierte Meetings zur Problemlösung, Software-Entwurf-Sessions im Team mit Metaplantechnik, Mediation bei Konflikten – das alles ist in der Unternehmenskultur von microTOOL heute verankert.

Diese vier Werte sind eng miteinander verknüpft. Wenn nur einer ins Wanken gerät, so sind unweigerlich auch die anderen gefährdet. Diese Werte haben unseren Weg stark beeinflusst, den wir mit der agilen Software-Entwicklung gegangen sind.

Erste Begegnung mit agilen Konzepten: XP vs. V-Modell

Im Jahr 2002 starteten wir ein Projekt mit dem Ziel, eine unterstützende Software für den Einsatz des V-Modells 97 zu entwickeln. Das V-Modell ist der deutsche Entwicklungsstandard für IT-Vorhaben der öffentlichen Hand – in der Version V-Modell 97 folgt es dem traditionellen Wasserfallmodell. Das derzeit aktuelle V-Modell XT erschien 2005 und enthält auch agile Vorgehensweisen.

Für die Entwicklungsteams war der Ansatz des V-Modells plausibel, so dass eine einheitliche und standardisierte Vorgehensweise bei der Software-Entwicklung die Qualität der Entwicklungsergebnisse verbessert. Aber alle waren sich darin einig, dass für Teamgrößen von drei bis zehn Entwicklern das V-Modell 97 ungeeignet ist, da es für große Projekte mit Auftraggeber und Auftragnehmer gedacht ist.

Da stellte ein Mitarbeiter einigen Kollegen das Buch "eXtreme Programming eXplained – Embraced Change" von Kent Beck (Beck, 1999) vor. Der Kontrast zum V-Modell hätte nicht größer sein können: Kommunikation, Einfachheit und schnelle Rückkopplung im Gegensatz zu umfangreichen Prozessbeschreibungen. Extreme Programming (XP) schien genau auf unsere Situation zu passen, da es die Eigenschaften hatte, die Produktentwicklung in ei-

nem sich rasch wandelnden Markt braucht. Wir entschieden uns spontan, XP einzuführen und begannen sofort damit, technische Praktiken aus XP umzusetzen. Dazu zählten vorrangig:

- kontinuierliche Integration mit einem automatisierten Build-Prozess (d.h. möglichst tägliches Erstellen einer Version mit allen neu programmierten Funktionen)
- gemeinsame Verantwortung aller Entwickler für den Code auf Basis einer zentralen Versionsverwaltung
- automatische Tests beim Build-Prozess
- Refactoring (d.h. automatisierte Optimierung des Codes nach einheitlichen Qualitätskriterien, wie z.B. systematische Benennung von Variablen und Funktionen)
- Pair Programming (d.h. zwei Programmierer arbeiten zusammen und wechseln sich als Ersteller und Beobachter ab) in begrenztem Umfang

Planungsprozess mit Iterationen

Wir verankerten die Iterationen im Planungsprozess. Es gab zwar bereits vorher Iterationen, allerdings ohne planerische Grundlage. Im Rahmen der Release-Planung planten wir nun neben Zielen, Produktfeatures und Release-Terminen auch die Iterationen mit den zu realisierenden Anforderungen, Entwickleraktivitäten und Terminen. Basis waren Anforderungen des Produktmanagements, Kundenanforderungen, Kundenwünsche, Supporttickets und Ideen, die nach einem einheitlichen Schema erfasst und in unserem System für Customer-Relationship-Mangement verwaltet wurden. Produktmanager und Projektleiter übernahmen die Planung. Das Projektteam am Planungsprozess zu beteiligen, erschien uns zu diesem Zeitpunkt noch als ein zu hoher Grad an Innovation.

Anforderungsmanagement

Zusätzlich etablierten wir ein intensives Anforderungscontrolling. Der Produktmanager priorisierte dabei jede neue Anforderung. Der Projektleiter nahm hoch priorisierte, neue Anforderungen nach Möglichkeit zusätzlich in die Planung oder sogar in die schon laufende Entwicklung eines Releases auf. Die sich daraus ergebenden Terminverschiebungen nahmen wir in Kauf. Zu diesem Zeitpunkt verfügten wir noch über keine Metriken, um den Projektfortschritt zu überwachen. Vorhanden war lediglich eine Statusübersicht aller Anforderungen, die wir mithilfe einer einfachen Ampelnotation stets aktuell hielten.

Agile Methodik steigert Kundenzufriedenheit

Das Fazit für diese Zeit fällt mit Blick auf das Wertesystem durchweg positiv aus. Das iterative Vorgehen und das kontinuierliche Anforderungscontrolling ermöglichten uns, besser auf Kunden und Markt zu reagieren. Die Kundenzufriedenheit stieg. Die Qualität von Architektur und Code stand als Wert bei den Teammitgliedern hoch im Kurs und verbesserte sich zusehends. Dazu trug neben den genannten technischen Praktiken aus XP vor allem der inkrementelle Software-Entwurf bei. Jede Komponente, die in einer Iteration entwickelt werden sollte, musste in UML (Unified Modeling Language) entworfen werden. Anschließend führten der Projektleiter und architekturerfahrene Entwickler, die nicht am Entwurf einer Komponente beteiligt waren, ein Entwurfsreview durch. Grundlage für Entwurf und Review bildeten die von Robert C. Martin (Martin, 2003) formulierten, objektorientierten Entwurfsprinzipien, wie

z.B. Dependency Inversion, Liskov Substitution und Open-Close-Prinzip. Ziel dieser Prinzipien ist es, eine Komponentenarchitektur aufzubauen, die sich leicht ändern und erweitern lässt. Ein online verfügbares Prozesshandbuch dokumentierte das Wissen über die anzuwendenden Entwurfsprinzipien (s. Bild 1). Diese Prinzipien und die Vorgehensweisen waren bei allen Entwicklern akzeptiert und sie arbeiteten intensiv mit dem Handbuch. Es begleitete den gesamten Prozess und wurde, solange diese Vorgehensweise zum Einsatz kam, regelmäßig aktualisiert.

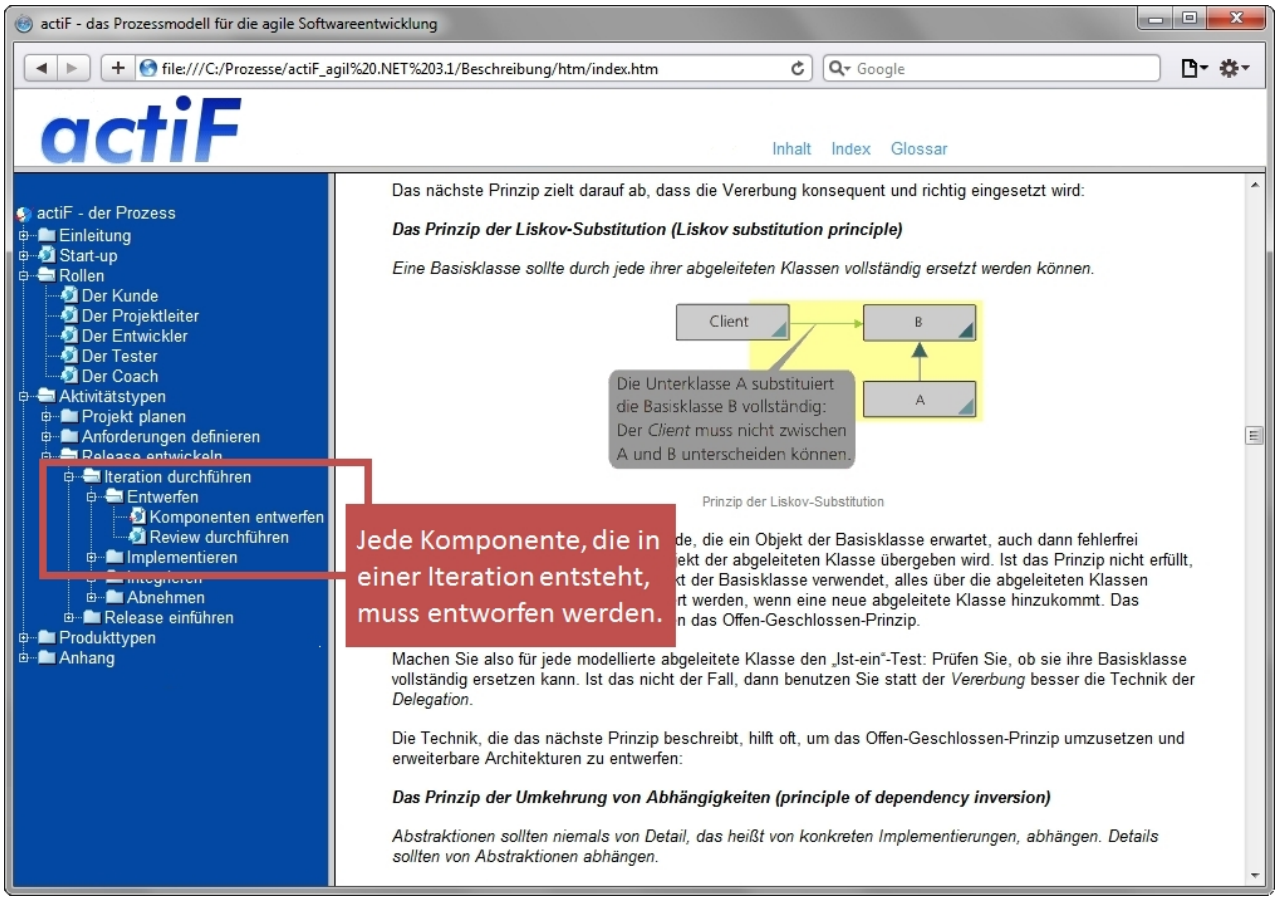

Bild 1: Das Prozesshandbuch – Regeln für das gemeinsame Vorgehen.

Scrum: Schluss mit den fließenden Terminen!

2007 entdeckten wir Scrum (vgl. Gloger, 2013) für uns. Mit seinen einfachen Regeln und ritualisierten Abläufen, der festen Iterationslänge, den selbstorganisierenden Teams und der Beteiligung der Entwickler an der Iterationsplanung erschien uns Scrum als vielversprechende Lösung für das Problem der sich ständig verschiebenden Iterations- und Release-Termine. Sowohl Entwickler als auch Management unterstützten die Einführung von Scrum, Widerstände gab es keine. Wir orientierten uns bei der Scrum-Einführung an den Prinzipien des Agilen Manifests (Kent, 2001), in denen es sinngemäß heißt: Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie brauchen – dann werden sie es schon richten.

Genau so haben wir es gemacht:

- Wir wählten ein Projekt innerhalb der Produktentwicklung aus, um mit Scrum zu starten. Gegenstand des Projekts war die Weiterentwicklung einer PM-Software, unser damals umsatzstärkstes Produkt.
- Die Infrastruktur für das Team wurde geschaffen: Alle Teammitglieder fanden in einem gemeinsamen Teamraum Platz, der mit Pinnwänden und Interaktionsmaterial ausgestattet wurde.
- Die Teammitglieder wurden hausintern in Scrum geschult. In externen Schulungen erwarben drei Mitarbeiter Zertifizierungen als Scrum Master, einer als Product Owner.
- Nach einem initialen Sprint begann das Team, den erlernten Scrum-Ablauf praktisch umzusetzen.
- Der Product Owner bezog das Team in die Sprintplanung ein. Gemeinsam sammelten wir Erfahrungen mit agilen Schätztechniken wie Planning Poker.
- Wir richteten ein Product Backlog ein, das wir mit unserer eigenen PM-Software verwalteten. Wir erweiterten unsere PM-Software zusätzlich um Funktionen, mit denen unter anderem Burndown Charts und Velocity Charts tagesaktuell erzeugt werden konnten.

Das Scrum-Projekt löste sich vollständig von der "alten", im Prozesshandbuch dokumentierten Vorgehensweise. Die technischen Praktiken von XP fügten sich in das Vorgehen nach Scrum reibungslos ein. Das Prozesshandbuch mit den darin verankerten Entwurfsprinzipien wurde im Scrum-Projekt ad acta gelegt. Bis 2009 entwickelte es sich zu einem Scrum-Vorzeigeprojekt: Mit hoher Konzentration arbeiteten die Teammitglieder am funktionalen Zuwachs eines unserer wichtigsten Produkte. "Liefern, liefern, liefern! " wurde zum Mantra des Teams. Das drückte sich nicht zuletzt in der Begeisterung aus, mit der das Burndown Chart allabendlich – trotz vorhandener Toolunterstützung – mit der Hand fortgeschrieben wurde und für alle sichtbar an der Wand hing (s. Bild 2).

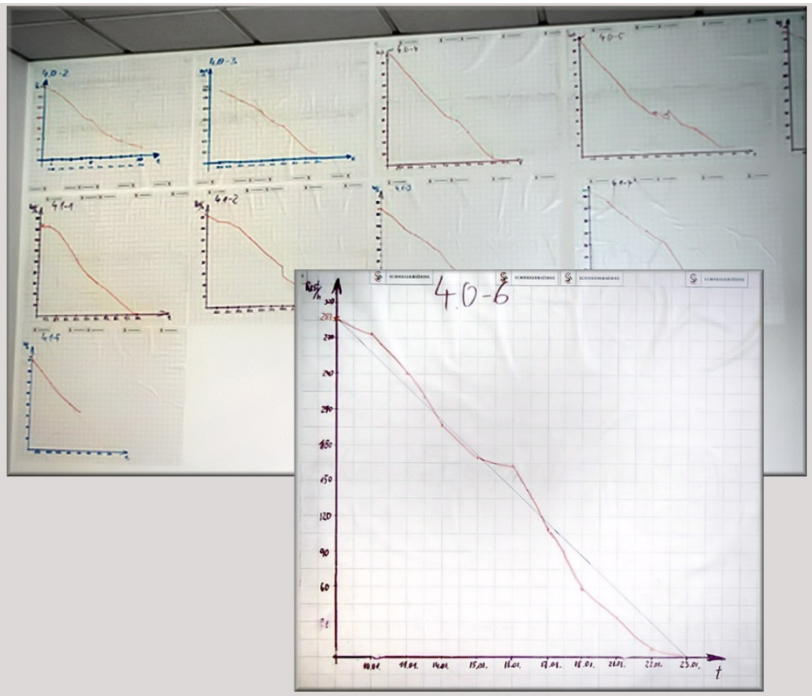

Bild 2: Endlich Metriken! Burndown Charts aus dem Jahr 2009.

Positiv zu vermerken ist: Ab diesem Zeitpunkt hatten wir Metriken, die den Projektfortschritt messbar und für uns sichtbar machten. Einen Wermutstropfen gab es für das Team: Der Product Owner war selten vor Ort. Er entwi-

ckelte, wie es typisch für Softwareproduzenten unserer Größenordnung ist, gemeinsam mit Schlüsselkunden Anforderungen zur Weiterentwicklung unserer Software. Diese Anforderungen flossen dann, von ihm priorisiert, in das Product Backlog ein. In der Kommunikation mit ihm war das Team weitgehend auf E-Mail und Telefon angewiesen.

Die Abnahme von Sprint-Ergebnissen und Releases konzentrierte sich auf die für die Anwender sichtbare Funktionalität. Verantwortlich für die Abnahme war der Product Owner. Unterstützung erhielt er von unseren Beratern, die für die Kundenbetreuung zuständigen waren. Soweit es möglich war, nahmen sie als Vertreter der Anwender an der Abnahme teil.

Nachdem sich eine gewisse Routine im Scrum-Ablauf eingestellt hatte, führte das Team Retrospektiven ein. Sie fanden halbjährlich jeweils nach Abnahme eines Releases statt. Ziel war es, das Vorgehen und die Zusammenarbeit kontinuierlich zu verbessern. Moderiert wurden die Retrospektiven durch in Scrum ausgebildete Kollegen aus dem Beratungsteam. Schwierigkeiten bei der Umsetzung von Scrum wurden thematisiert und Lösungswege beschlossen. Die Techniken, die bei den Retrospektiven eingesetzt wurden, gehen auf Esther Derby und Diana Larsen (Derby, Larsen, 2006) zurück. Bild 3 zeigt Material aus dieser Zeit.

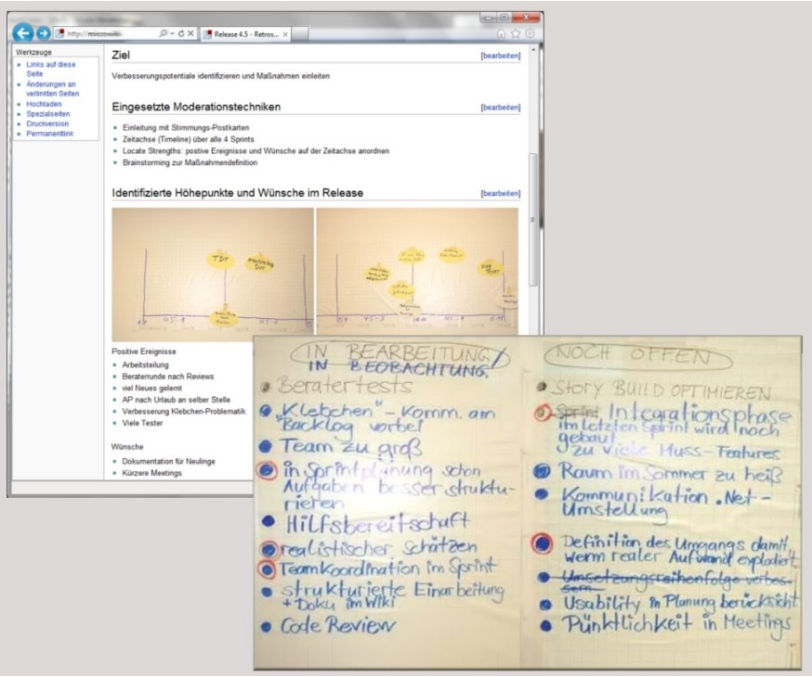

Bild 3: Ergebnisse aus Release-Retrospektiven – dokumentiert im Firmen-Wiki.

Das Fazit der ersten Scrum-Jahre: Das Team entwickelte in kurzer Zeit eine reichhaltige und kluge Produktfunktionalität. Es konnte regelmäßig neue Releases veröffentlichen. Die Kunden waren zufrieden. Mögliches Konfliktpotenzial wurde in den Retrospektiven aktiv bearbeitet. Die Kultur des Miteinanders verbessert sich. Motivation und Selbstbewusstsein des Teams stiegen.

Die schöne, heile, agile Welt gerät ins Wanken

Im Jahr 2010 mehrten sich kritische Rückmeldungen, vor allem von Unternehmen mit einer hohen Anzahl an Installationen des mit Scrum erstellten Produkts: Diese Anwender meldeten immer häufiger Performance-Probleme. Aber nicht nur die Software wurde beständig langsamer, die Kunden mussten auch immer länger auf Toolerweiterungen warten, die sie beauftragt hatten. Die Kundenzufriedenheit geriet ebenso ins Wanken wie der vermeintliche Wettbewerbsvorteil der einfachen Erweiterbarkeit des Produkts.

Wir gründeten umgehend eine Task-Force. Sie sollte den Problemen auf den Grund gehen, die Performance messen, den Code analysieren und die Probleme beseitigen. Das war leichter gesagt als getan, denn die Task-Force musste feststellen, dass es zwar viel Wissen in wenigen Köpfen, aber wenig Software-Dokumentation gab. "Funktionierende Software über umfassende Dokumentation" – diesen Punkt aus dem agilen Manifest hatte das Team offenbar zu eng interpretiert und damit einen der Grundwerte des Unternehmens außer Kraft gesetzt.

Es kamen noch weitere unerfreuliche Dinge zutage: Bei dem Produkt, das im Vorzeige-Scrum-Projekt weiterentwickelt wurde, handelte es sich um eine datenbankbasierte Client-/Server-Lösung mit einer Mehrschichtenarchitektur. Dies bedeutet, dass sich zwischen Benutzeroberfläche und der eigentlichen Datenbank mehrere logische Software-Ebenen ("Schichten") befinden. Ziel einer Mehrschichtenarchitektur ist es, die eigentliche Datenbank, sowie die Logik zur Speicherung, Beschaffung, Verarbeitung und Darstellung der Daten zu trennen. Der Vorteil: Änderungen, z.B. der Darstellung der Daten auf dem Bildschirm, betreffen nur eine Schicht und nicht die gesamte Software.

Im Falle unserer Software verfügten die Entwickler über Software-Werkzeuge, mit denen sie die Datenbankzugriffsschicht vollständig generieren konnten. Das heißt, sie waren auf sehr elegante und bequeme Art in der Lage, Datenbankzugriffe zu realisieren und machten davon intensiven Gebrauch. Leider hat eine Schichtenarchitektur auch einen Nachteil: Das "Durchreichen" der Daten von der Datenbank über die Datenzugriffsschicht bis in die Schicht der Verarbeitungsfunktionen kostet bei großen Datenmengen Zeit. Die Antwortzeiten steigen aus Sicht des Benutzers an. In diesem Fall empfiehlt es sich, unter Umgehung der Schichtenarchitektur direkt auf die Datenbank zuzugreifen.

Die Analyse der Task-Force brachte zutage, dass mal der eine, mal der andere Zugriffsweg und manchmal sogar beide in Kombination benutzt wurden, um Datenbankauswertungen auf den Bildschirm zu bringen. Es gab kein homogenes Zugriffskonzept. Vielmehr hatte jeder Entwickler seinen eigenen Stil verwirklicht, was letztlich zu den Performanceproblemen führte.

Leider war das nicht der einzige Architekturdefekt, der zum Vorschein kam: So musste die Task-Force z.B. feststellen, dass Verarbeitungsfunktionen, die eigentlich in eine obere Schicht (Service-Schicht) gehörten, in der darunter liegenden Schicht des Datenmodells (Entity-Schicht) angesiedelt waren. Service-Operationen mit Geschäftslogik befanden sich dadurch an einer Stelle, wo es eigentlich nur Validierungsmethoden hätte geben dürfen. Der eigentliche Nutzen der Mehrschichtenarchitektur wurde dadurch torpediert. Im Ergebnis war eine große monolithische Komponente entstanden. Kein Wunder, dass das Erweitern des Produkts immer aufwendiger wurde.

So teuer kam uns die Agile Methodik zu stehen

Die Kosten zur Behebung des Schadens waren erheblich, die wesentlichen Positionen waren:

- Drei Personenjahre Entwicklungsaufwand waren notwendig, um die Qualität von Architektur und Code wiederherzustellen.
- Wir mussten die Weiterentwicklung des Produkts über einen Zeitraum von sechs Monaten anhalten, um die Personalkapazität dafür zu schaffen.
- In Folge büßten wir Umsatz im Neukundengeschäft ein, der sich allerdings nur schwer beziffern lässt.

Glücklicherweise konnten wir durch offene Kommunikation wenigstens einen Imageschaden vermeiden. Inzwischen sind ein korrigiertes Release und ein weiteres Folgerelease auf dem Markt und erfolgreich bei den Kunden im Einsatz.

Wie konnte es dazu kommen?

In den Prinzipien zum Agilen Manifest heißt es eigentlich: "Die besten Architekturen, Anforderungen und Entwürfe entstehen in selbstorganisierenden Teams" Warum schien das aber nicht in unserem zu gelten? Wie konnte die Architektur solchen Schaden nehmen? Wir identifizierten mehrere Ursachen, die vier wichtigsten waren:

- Der dominierende Wert im Team war der Funktionszuwachs, genauer gesagt: der Zuwachs an von außen sichtbarer Funktionalität.
- Dadurch war die Architektur, d.h. die innere Sicht des Produkts, im Team kein zentrales Thema mehr.
- Refactoring auf Entwurfsniveau fand kaum noch statt. Das Team machte sich die Haltung zu eigen: Der Entwurf entsteht inkrementell beim Bauen.
- Das Team hatte zwar eine gemeinsame Vision von einem funktional vielfältigen Softwareprodukt. Aber es hatte das "Big Picture", den Bauplan von dem zu erstellenden Gebäude verloren.

Wo waren Product Owner und Scrum Master in dieser Situation? Der Product Owner, der als interner Auftraggeber eines Softwareprodukts nicht nur an der Funktionalität, sondern mit Blick auf den Markt auch an der Qualität Interesse hätte haben müssen, war nur begrenzt verfügbar und selten vor Ort. Heute wissen wir: Es reicht einfach nicht aus, wenn der Product Owner nur zur Sprintplanung und Abnahme im Team erscheint. Der Scrum Master trat in seiner Rolle nicht in Erscheinung. Die Rolle war, auch das ist aus heutiger Sicht klar, zu schwach besetzt. Der Mitarbeiter war nach seiner fachlichen Qualifikation ausgewählt worden, verfügte aber nicht über das nötige Durchsetzungsvermögen. Mit wachsendem Erfolg bei der Selbstorganisation vermisste das Team einen seine Rolle ausfüllenden Scrum Master nicht.

Rückeroberung der Werte

Um die eingangs beschriebenen Unternehmenswerte wieder zur Geltung zu bringen, zogen wir weit reichende Konsequenzen. Wir setzten den Entwicklungsprozess aus technischer Sicht grundlegend neu auf und veränderten unser Projektmanagement in wesentlichen Punkten.

Entwicklungsprozess: Modellgetrieben

Parallel zur laufenden Produktentwicklung investierten wir, gefördert vom Bundesministerium für Bildung und Forschung, von 2008 bis 2011 in eine neue Entwicklungstechnik, welche die Arbeit der Entwickler grundsätzlich veränderte. Die modellgetriebene Entwicklungstechnik stützt sich auf eine Kombination aus Entwicklungstool und Komponentendatenbank (s. Bild 4).

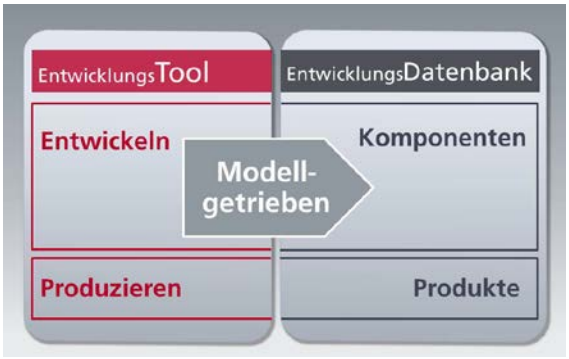

Bild 4: Die modellgetriebene Entwicklung basiert auf Entwicklungstool und Komponentendatenbank.

In der Datenbank sind Komponenten gespeichert, die zu Produkten für den Markt oder zu individuellen Kundenlösungen kombiniert werden können. Einige der Komponenten müssen in jedem Produkt vorkommen, andere sind optional. Von einigen Komponenten gibt es Varianten, aus denen der Kunde eine Auswahl für seine spezielle Lösung treffen kann. Die so entstehenden Produkte werden ebenfalls wieder in der Datenbank verwaltet. Neue Komponenten entwickeln wir konsequent modellgetrieben, das heißt auf eine hoch automatisierte Weise mit dem Entwicklungstool.

Der Einsatz modellgetriebener Entwicklung veränderte die Arbeitsweise des Teams grundsätzlich: Für jede neue Komponente erstellen die Entwickler nach festen Regeln zunächst ein fachliches Modell mit Mitteln der UML. Dies geschieht in folgenden Schritten:

- Modellierung der fachlichen Entitäten
- Definition der Datensichten ("Views"), die der Anwender später auf dem Bildschirm sehen wird und manipulieren kann. Zusätzlich werden die Kommandos festgelegt, die der Benutzer auf den Daten auslösen kann. Viele der Kommandos sind bereits fertig implementiert, sodass der Entwickler oft nur eine Auswahl treffen muss.
- Definition der Services, die zur Beschaffung und Verarbeitung der Daten dienen.

Aus dem fachlichen Modell wird dann durch maschinelle Transformation ein technisches Modell und Quellcode für die Zielumgebung erzeugt. Etwa 70% des Codes entstehen heute auf diese Weise maschinell. Bei den restlichen manuell zu erstellenden ca. 30% handelt es sich um komplexe Services. Aber auch für sie werden immer zuerst Modelle gemacht.

In den Modelltransformationen stecken in erheblichem Umfang technisches Knowhow und Fachwissen des Unternehmens, und zwar personenunabhängig in maschineller Form. Die Modelltransformationen macht dieses Wissen wiederverwendbar. Es bleibt dadurch dem Unternehmen auch beim Wechsel von Mitarbeitern erhalten.

Der per Transformation erzeugte Code genügt hohen Qualitätsanforderungen und – jetzt wieder – den Entwurfsprinzipien, die bei den ersten agilen Schritten eine so hohe Bedeutung hatten. Der Wert "Qualität von Architektur und Code" hat seinen Stellenwert zurück.

Modelle sind bei diesem Vorgehen die eigentlichen Ergebnisse der Entwicklung. Sie sind lebende Dokumentation wie der Code selbst. Die modellgetriebene Entwicklung versetzt ein Team aufgrund des hohen Automatisierungsgrads in die Lage, sehr schnell neue Komponenten zu realisieren. Modellgetriebene Entwicklung ist damit für iteratives, inkrementelles Vorgehen geeignet. Modellgetrieben vorzugehen ist zu einem wesentlichen Element unserer Agilität geworden. Deshalb haben wir dem agilen Manifest eine Aussage hinzugefügt: "Modelle stehen über Code" (s. Bild 5). Da modellgetriebene Entwicklung automatisch eine Form der Dokumentation liefert, hat der Punkt "Funktionierende Software über umfassende Dokumentation" für uns an Relevanz verloren.

Bild 5: Agiles Manifest aus Sicht der modellgetriebenen Entwicklung.

Product Owner – ein neues Rollenverständnis

Beim Projektmanagement gab es ebenso tief greifende Veränderungen wie in der Entwicklung. Zwar behielten wir den grundlegenden Ablauf nach Scrum bei: Es finden Release- und Sprintplanungsmeetings statt, die Entwickler sind an der Planung beteiligt, schätzen den Aufwand der User Stories und verteilen die Arbeit unter sich. Die Rolle des Product Owners definierten wir allerdings neu und besetzten sie entsprechend.

Der Product Owner trägt die unternehmerische Verantwortung für die Produktentwicklung. Er ist der interne Auftraggeber und Mitglied der Geschäftsführung von microTOOL.

Um diese Verantwortung wahrnehmen zu können, ist er heute täglich im Team und damit immer ansprechbar. Im Durchschnitt fallen etwa 25 % seiner Arbeitszeit auf die Präsenz im Team. Die Erfahrung zeigt: Die Teammitglieder nutzen diese hohe Verfügbarkeit für die Klärung von Fragen zu den anstehenden Stories, für Reviews mitten im Sprint, und zwar nicht nur der sichtbaren Produktfunktionalität, sondern vor allem der zugrunde liegenden Modelle sowie neuer Transformationen für Code-Erzeugung.

Kostenrelevante Entscheidungen trifft der Product Owner

In drei Bereichen trifft der Product Owner Entscheidungen, die nach Scrum eigentlich beim Team liegen sollten. Diese drei Bereiche haben große Auswirkungen auf die Projektkosten, sie beeinflussen damit den Business Case und fallen deshalb in die Zuständigkeit des Product Owners.

Der erste Bereich betrifft grundlegende Architekturentscheidungen. Verschiedene Lösungswege sind mit unterschiedlichen Kosten verbunden. Hier entscheidet in letzter Instanz der Product Owner.

Der zweite Bereich ist die Verteilung der Arbeit. In der Regel nimmt der Product Owner keinen Einfluss darauf, wer welche Stories bearbeitet, obwohl er die Autorität hat, Ressourcenentscheidungen zu treffen. Es hat sich gezeigt, dass das Team dazu neigt, Aufgaben jeweils nach dem Kriterium zu verteilen: "Wer hat es schon einmal gemacht und ist am schnellsten?" Um Spezialistentum zu verhindern, eine breite Wissensbasis zu schaffen, noch ungeübte Entwickler an neue Aufgaben heran zu führen sowie Aus- und Weiterbildung zu fördern, greift der Product Owner bei Bedarf in die Autonomie des Teams ein. Denn auch hierbei geht es wieder um Investitionsentscheidungen.

Der dritte Bereich ist die Entscheidung über einen Stillstand im Projekt. Bis 2010 wurde Stillstand nach der Regel gehandhabt: "Löst das Problem gemeinsam. Fangt bis dahin nichts Neues an." Dieses Vorgehen hat sich aus Kostensicht als untragbar erwiesen: Ein zweiwöchiger Sprint eines Teams aus acht Personen kostet uns bei ortsüblichen Gehältern rund 42.000 Euro Vollkosten. Diese Kosten zahlt kein Auftraggeber. Ob und in welchem Umfang ein Sprint bei grundlegenden Problemen angehalten wird, wer welche geplanten Arbeiten ruhen lässt, um an der Problembeseitigung mitzuwirken, entscheidet angesichts dieser Kosten der Product Owner.

Wie die Rolle des Product Owner hat sich auch die des Scrum Masters verändert. Anders als im ersten Scrum-Projekt wurde der Scrum Master nicht "ernannt". Ein Teammitglied, das sich durch eine starke Persönlichkeit auszeichnet und dem alle anderen Vertrauen entgegenbringen, ist in diese Rolle hineingewachsen und wird darin akzeptiert.

Die Lehre aus zehn Jahren agil

Was haben wir in zehn Jahren agiler Entwicklung gelernt? Die lehrbuchartige Umsetzung agiler Konzepte und Abläufe hat uns dazu verleitet, vor allem zwei unserer Werte zu vernachlässigen:

1. die Qualität von Architektur und Code
2. die breite Streuung von Wissen und Erfahrung sowie die Sicherung von Wissen in personenunabhängiger Form.

Dafür haben wir im wahrsten Sinne des Worts Lehrgeld bezahlt.

Wir haben gelernt, Scrum als einen Handlungsrahmen zu verstehen, der ausgefüllt werden muss. Wir haben begonnen, unser Vorgehen konsequent an unseren Werten auszurichten. Wir sind bereit, uns dafür auch an der einen oder anderen Stelle über "gängige Lehrmeinung" hinwegzusetzen.

Gerade bei der Neudefinition der Befugnisse des Product Owners könnte die kritische Frage gestellt werden: "Haben diese Eingriffe in die Selbstorganisation des Teams nicht doch wieder Züge von klassischem Command & Control?" Unsere Haltung dazu ist: Was hier passiert, ist die Wahrnehmung unternehmerischer Verantwortung. Selbstorganisation des Teams – ja. Verantwortung für die Qualität von Lösungen – ja, auch sie gehört selbstverständlich ins Team. Aber unternehmerische Entscheidungen und Investitionsentscheidungen gehören in den Verantwortungsbereich der Geschäftsführung bzw. der von ihr beauftragten Rollen.

In diesem Sinne machen wir agil weiter. Dabei ist uns bewusst, dass das Vorgehen immer wieder an einen sich ändernden Markt, neue Menschen und sich wandelnde Werte angepasst werden muss.

Literatur

- Beck, Kent: Extreme Programming Explained: Embrace Change, Addison-Wesley Longman, 1999, (aktuelle Ausgabe: 2. Aufl. November 2004)
- Gloger, Boris: Scrum: Produkte zuverlässig und schnell entwickeln, 4.Aufl. 2013
- Derby, Esther und Larsen, Diana: Agile Retrospectives: Making Good Teams Great, Pragmatic Programmers, 2006
- Martin, Robert C.: Agile Software Development. Principles, Patterns, and Practices, Prentice Hall, 2003, (Neuauflage: Pearson, International ed., März 2011)
- Beck, Kent u.a.: Manifesto for Agile Software Development, 2001, http://agilemanifesto.org/

Fachbeitrag

Risiken der Agilen Software-Entwicklung reduzieren

Ein bisschen Wasserfall muss sein

Agile Vorgehensweisen erfreuen sich zunehmender Popularität. Sie gelten als das richtige Mittel, um in einem dynamischen Umfeld qualitativ hochwertige Software zu entwickeln, die die Anforderungen der Anwender erfüllt. Traditionelle Vorgehensweisen wie das Wasserfallmodell gelten hingegen als unflexibel, "von gestern" und den Erfordernissen moderner Softwareprojekte nicht mehr gewachsen. Wir meinen allerdings, dass in traditionellen Vorgehensweisen bewährte Mechanismen zur Einhaltung von Zeit und Budget entwickelt wurden, die auch bei agilen Vorgehensweisen vorteilhafte Anwendung finden können.

Wir geben zunächst einen kurzen Überblick über die charakteristischen Eigenschaften der Vorgehensweisen nach dem Wasserfallmodell und nach Scrum. Anschließend analysieren wir anhand eines einfachen Fallbeispiels mögliche Auswirkungen der beiden Vorgehensweisen auf Zeit, Budget und Qualität in einem Projekt. Schließlich betrachten wir die möglichen Risiken detaillierter und geben Handlungsempfehlungen, wie sich diese vermindern lassen.

Wasserfallmodell: In Phasen vom Lastenheft zum fertigen Produkt

Die Umsetzung eines Projekts nach dem Wasserfallmodell erfolgt in einzelnen Phasen, wie sie in Bild 1 beispielhaft dargestellt sind. Jede Phase kann erst begonnen werden, nachdem die vorherige erfolgreich abgeschlossen wurde.

Spezifikation als Vertragsgrundlage

Zu Beginn des Projekts legen Auftraggeber und Auftragnehmer den Liefergegenstand in einer Spezifikation fest, die typischerweise Bestandteil eines Werkvertrags ist. Die gemeinsam von Auftraggeber und Auftragnehmer akzeptierte Spezifikation bietet eine gute Grundlage für die im Projekt zu erbringenden Leistungen. Der Auftragnehmer muss alle in der Spezifikation enthaltenen Anforderungen innerhalb der Vorgaben von Zeit, Budget und Qualität erfüllen.

Autoren

Dorian Gloski

Dipl.-Physiker, freiberuflicher Software-Architekt

Kontakt: dorian@gloski.de

Eva Maria Schielein

Geschäftsführerin aestimat GmbH, langjährige, branchenübergreifende und internationale Erfahrung als Managementberaterin und Coach.

Kontakt: eva.schielein@aestimat.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in den Rubriken:

› Agiles Projektmanagement
› Mit Standards arbeiten
› IT-Projekte
› Informations- und Kommunikationstechnologie

Service-Links

› Agiles Projektmanagement
› PM-Einführung und -optimierung
› IT-Projektmanagement

Umsetzung: Entwurf, Realisierung, Test, Inbetriebnahme

Nachdem die Spezifikation von beiden Seiten abgenommen wurde, wird das IT-System entworfen, erst dann beginnt die Realisierung. Anschließend erfolgen Test und Installation in die Betriebsumgebung, d.h. die Inbetriebnahme. Der Auftraggeber bekommt die von ihm beauftragte Software erst sehr spät zu sehen, normalerweise in der Testphase. Dadurch kann er erst zu einem sehr späten Projektzeitpunkt erkennen, ob die gelieferte Software seinen Anforderungen entspricht.

Sind die Anforderungen in der Spezifikation hinreichend genau beschrieben und nur wenigen Änderungen unterworfen, so lassen sich Projekte nach dem Wasserfallmodell erfolgreich gemäß den Vorgaben umsetzen.

Regelmäßig ergibt sich während der Projektdurchführung jedoch Änderungsbedarf an der Spezifikation. Möglicherweise wurden Anforderungen übersehen, wie z.B. die Schnittstelle zu einem anderen System. Auch kommt es vor, dass Anforderungen zu Widersprüchen führen, die erst bei der Implementierung deutlich werden: Wenn zum Beispiel für eine Benutzerinteraktion sowohl eine extrem kurze Antwortzeit als auch eine aufwendige Plausibilitätsprüfung gefordert wird. Bei der Implementierung stellt sich dann möglicherweise heraus, dass die Berechnung der Plausibilitätsprüfung so lange dauert, dass die geforderte Antwortzeit nicht eingehalten werden kann.

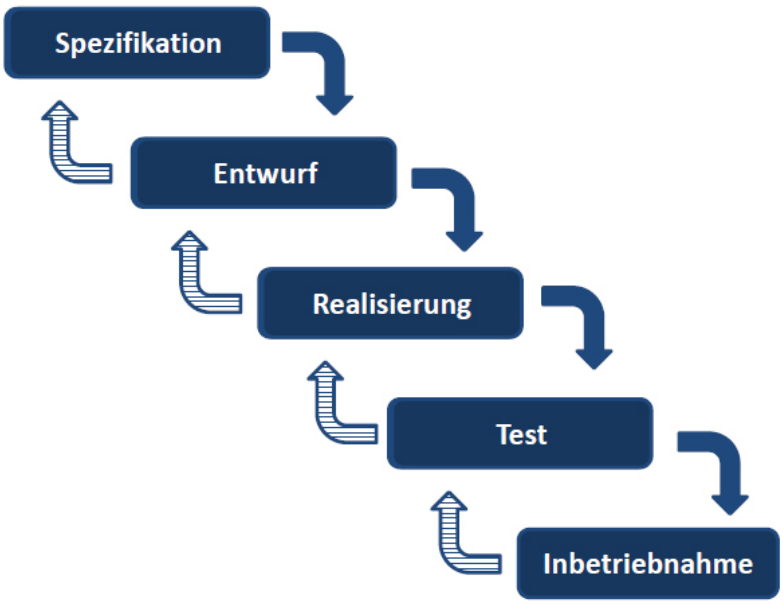

Bild 1: Schematische Darstellung des Wasserfallmodells. In der Regel erfolgt der Übergang von einer Phase in die nächste von oben nach unten. Der Weg zurück ist nur in Ausnahmefällen möglich.

Durch das starre Prozessmuster sind Projekte nach dem Wasserfallmodell im Anforderungsmanagement wenig flexibel. Änderungen des Leistungsumfangs können nur über formale Änderungsanträge (Request for Change = RFC) berücksichtigt werden. Es ist nicht unüblich, dass die Änderungsrate eines durchschnittlichen Projekts ca. 25% des Gesamtumfangs der Spezifikation beträgt. Bei einem traditionellen Vorgehen ist deshalb ein entsprechender Aufwand für die Erstellung und Bearbeitung von RFCs einzuplanen.

Durch die verhältnismäßig langen Phasen und die späte Auslieferung der Software wird Änderungsbedarf oft erst sehr spät erkannt. Damit ergeben sich häufig Konflikte mit Zeit- und Budgetzielen. Möglicherweise sinkt auch die Akzeptanz der Software bei den Anwendern, wenn sich erst bei der Inbetriebnahme herausstellt, dass die umgesetzten Anforderungen nicht ihren tatsächlichen Bedürfnissen entsprechen.

Agile Software-Entwicklung: Anforderungen iterativ umsetzen

Agile Vorgehensweisen haben aus den Schwächen der traditionellen Vorgehensweisen gelernt. Sie versuchen möglichst frühzeitig im Projekt Änderungsbedarf zu erkennen und flexibel darauf zu reagieren. Die Software wird in Iterationen (die z.B. bei Scrum "Sprint" heißen) entwickelt, deren Dauer typischerweise einige Wochen beträgt. Zu Beginn jeder Iteration wird verbindlich festgelegt, welche Anforderungen umgesetzt werden. Mit jeder Iteration erhält der Auftraggeber Software, welche die bisher umgesetzten Anforderungen enthält. Diese Software kann von Anwendern getestet werden, die sofort Änderungswünsche zur Verbesserung der Akzeptanz der Software geben können. Während der gesamten Projektlaufzeit kann der Auftraggeber Anforderungen ändern, streichen, neu erstellen und umpriorisieren.

Die Erreichung des Sprintziels, also die erfolgreiche Umsetzung aller für einen Sprint geplanten Anforderungen, hat bei agilen Projekten eine sehr hohe Priorität. Um diese nicht zu gefährden, dürfen Anforderungen während des Sprints nur in Ausnahmefällen geändert werden. Hier verhält sich ein Sprint wie ein Projekt nach dem Wasserfallmodell. Die Kontrolle von Zeit und Budget für das Gesamtprojekt spielt eine geringere Rolle, da sich das sog. Product Backlog ständig ändern kann.

Eine detaillierte Beschreibung von Scrum findet sich im Artikel von Wirdemann: "Agiles Projektmanagement. Scrum – eine Einführung" (Wirdemann, Projekt Magazin 21/2009).

Rollen und Budgets

Projekte nach traditionellem Vorgehen werden in der Regel als Festpreisprojekte umgesetzt. D. h. es steht ein fixes Budget zur Verfügung. Der Auftragnehmer stellt typischerweise nach jeder erfolgreich durchgeführten Projektphase einen Teil seiner Leistung in Rechnung. Den größten Teil wird er nach erfolgreicher Inbetriebnahme der Software berechnen. Der Auftragnehmer trägt das Verlustrisiko bei höheren Aufwänden. Das kaufmännische Interesse des Auftragnehmers besteht daher darin, die Spezifikation mit möglichst geringen Aufwänden umzusetzen.

Bei Scrum ist der Auftragnehmer bestrebt, seine Aufwände dem Auftraggeber nach entstandenem Aufwand in Rechnung zu stellen. Sind die Gesamtaufwände geringer als das ursprünglich veranschlagte Budget, spart der Auftraggeber Geld. Sind sie höher, so vergrößert der Auftraggeber das Budget – zumindest in der Theorie. In der Praxis wird ein Auftraggeber sein Budget nicht nach Belieben steigern können. Oft werden Anforderungen, die am Ende des Budgets noch offen sind, nicht mehr umgesetzt. Zu Konflikten kommt es regelmäßig dann, wenn Anforderungen nicht umgesetzt wurden, die für die Inbetriebnahme der Software essenziell sind.

Für unser Beispiel sind folgende Rollen des Scrum Prozesses relevant:

- Der Product Owner ist für die Definition des Softwareprodukts aus Kundensicht verantwortlich. Er spezifiziert und priorisiert die Anforderungen (in Form von User Stories) im Product Backlog.
- Das Entwicklungsteam ist für die Umsetzung der Anforderungen verantwortlich. Das Team arbeitet eigenständig und trägt die Verantwortung für die gesamte technische Umsetzung der Anforderungen inklusive Einhaltung von Qualitätsstandards und Erstellung einer adäquaten Architektur. Damit übernimmt das Team die Aufgaben des technischen Projektleiters (s.u.) in traditionellen Projekten.

- Der Scrum Master ist für den reibungslosen Ablauf des Scrum Prozesses verantwortlich. Er hat die Aufgabe, dem Team und dem Product Owner optimale Bedingungen für die Ausführung ihrer Tätigkeiten zu schaffen.

Im Wasserfallmodell gibt es typischerweise folgende Rollen:

- Der Projektleiter trägt die Verantwortung für das gesamte Projekt. Dies betrifft die Umsetzung der Spezifikation sowie die Einhaltung von Zeit und Budget.
- Der technische Projektleiter (bzw. Teilprojektleiter, Arbeitspaketverantwortliche oder Leiter des Entwicklungsteams) trägt die Verantwortung für die technische Umsetzung des Projekts. Dazu gehören die Konzeption des IT-Systems und die Gewährleistung der Qualität. Er arbeitet sehr eng mit dem Projektleiter zusammen.
- Das Team trägt die Verantwortung für die Umsetzung von Anforderungen gemäß den Vorgaben der Projektleiter. Damit hat es deutlich weniger Verantwortung für das Projekt als ein agiles Team.

Ein Beispiel: Relaunch eines Webportals

Um die Risiken traditioneller und agiler Projekte zu untersuchen, betrachten wir als Beispiel den Relaunch eines Webportals für einen Buchverlag. Dieses Beispiel ist fiktiv und stark vereinfacht, jedoch greift es typische Situationen auf, die wir selbst in agilen Projekten mehrfach beobachtet haben.

Nehmen wir an, dass zwei Buchverlage ihre Webportale relaunchen möchten. Der Verlag *Agile Books* entscheidet sich für ein agiles Vorgehen nach Scrum, der Verlag *Waterfall Literature* für ein traditionelles Vorgehen nach dem Wasserfallmodell.

Die Rahmenbedingungen für beide Projekte sollen, abgesehen von der Vorgehensweise, identisch sein:

- Über einen Onlineshop sollen Kunden weltweit Bücher kaufen können. Die Abwicklung des Einkaufs erfolgt über ein Backend-System (z.B. Warenwirtschaftssystem). Einkaufsrelevante Daten werden auch im Portal selbst gespeichert, um dem Anwender eine Liste seiner Einkäufe zu zeigen.
- Über einen Shopfinder kann der Anwender herausfinden, in welchen Geschäften er die Bücher des Verlags kaufen kann.
- Die Projekte werden zum Festpreis abgewickelt.
- Für die Produktivsetzung der Portale ist ein fester Termin vorgegeben.
- Die Entwicklung erfolgt auf Basis von Java Enterprise Edition (JEE) unter der Verwendung von Java Server Faces (JSF) und eines Applikations-Servers.

Die Verteilung der Rollen und die Bezahlung des Auftragnehmers erfolgt für die unterschiedlichen Vorgehensweisen wie im vorigen Kapitel beschrieben.

Im Lauf der Entwicklung treten nun verschiedene für Software-Projekte typische Ereignisse ein, die wir jeweils für das *Waterfall Literature*- und das *Agile Books*-Projekt durchspielen.

Fall 1: Umstellung der Entwicklungsplattform

Während des Projekts kommen beide Entwicklungsteams zu der Überzeugung, dass sich die Anforderungen mit dem Framework "Ruby on Rails" viel effizienter umsetzen lassen.

Das Team bei *Agile Books* setzt die Entscheidung für in Ruby on Rails schnell in die Tat um und kann damit die Entwicklungszeit deutlich reduzieren.

Die Entwickler bei *Waterfall Literature* können eine solche Entscheidung nicht alleine treffen, da in der Spezifikation JEE als Laufzeitumgebung festgelegt wurde. Vor einem Umstieg ist ein RFC notwendig, der bezüglich Aufwand und Risiko geprüft werden muss. Die Bearbeitung des RFC wird sicherlich einige Zeit in Anspruch nehmen. Entwickelt das Team währenddessen bereits Anforderungen in JEE, so müssten diese bei einem erfolgreichen RFC mit Ruby on Rails erneut implementiert werden. Durch diesen Mehraufwand steigt das Risiko, den Releasetermin nicht zu halten. Der RFC hätte wenig Aussicht auf Erfolg.

Der agile Prozess scheint in diesem Szenario dem traditionellen Vorgehen überlegen zu sein. Was aber, wenn das Rechenzentrum von *Agile Books* Ruby on Rails als Laufzeitumgebung nicht unterstützt? Vor dem Produktivschalten der Software müsste das Rechenzentrum die Voraussetzungen für eine Bereitstellung einer entsprechenden Laufzeitumgebung prüfen. Terminverzögerungen und Kostensteigerungen wären die Folge. Im schlimmsten Fall würde das Rechenzentrum die Unterstützung für Ruby on Rails verweigern – sei es weil entsprechendes Know-how nicht vorhanden ist, oder weil bestehende Wartungsverträge diese Laufzeitumgebung nicht abdecken. Dann müsste das Portal sogar komplett neu entwickelt werden.

Eigentlich sollten das Team und der Scrum Master von *Agile Books* diese Implikationen vor einem Wechsel der Entwicklungsumgebung beachten. Für das Team von *Agile Books* steht jedoch eine möglichst effiziente Arbeitsweise und die erfolgreiche Umsetzung der einzelnen Sprints im Vordergrund. Risiken, die die Produktivumgebung betreffen, können dabei leicht vernachlässigt werden. Das Fehlen einer dedizierten Rolle, die die Verantwortung für das Zusammenspiel von Software und Produktivumgebung trägt, führt unserer Erfahrung nach bei agilen Projekten häufig zu Projektrisiken.

Bei *Waterfall Literature* hingegen liegt die Verantwortung hierfür beim technischen Projektleiter. Dieser wird nach Rücksprache mit allen Beteiligten entscheiden, ob er das Risiko einer Änderung der Laufzeitumgebung während der Projektlaufzeit eingeht.

Fall 2: Aufnahme neuer Anforderungen während der Entwicklung

Kurz nach Projektstart stellen die Verlage fest, dass Apps eine immer größere Verbreitung finden. Daher soll das Einkaufen im Onlineshop auch über eine App möglich sein.

Bei *Agile Books* werden die Anforderungen vom Product Owner umgehend formuliert und mit höchster Priorität im Product Backlog versehen. So kann innerhalb weniger Sprints eine App angeboten werden, die Anwendern das Anschauen und Kaufen von Büchern mit ihrem Smartphones ermöglicht.

Allerdings wurden durch die hohe Priorisierung der App andere Anforderungen im Product Backlog nach hinten verschoben und können eventuell nicht mehr innerhalb des geplanten Zeitrahmens und des Budgets umgesetzt werden. Diese Verschiebung erfolgt automatisch und häufig ohne einen dedizierten Prozess, so dass dem Auftraggeber die Auswirkungen der Umsetzung der App nicht bewusst werden.

Bei *Waterfall Literature* ist man unflexibler. Der Projektleiter erstellt in Zusammenarbeit mit dem Kunden einen RFC. Dieser wird hinsichtlich Kosten, Risiken und Zeit bewertet. Beim Wasserfallmodell ersetzt man nur ungern bestehende Anforderungen durch neue, da dies die Vertragsgrundlage zwischen Auftraggeber und Auftragnehmer ändert. Für die App würde der Projektleiter daher zusätzlich Zeit und Budget beantragen müssen. Selbst wenn die Freigabe des Budgets relativ schnell erfolgen sollte, wird er bei *Waterfall Literature* kaum das Risiko eingehen, mehr Anforderungen in der gegebenen Zeit umzusetzen. Die Anwender des Portals von *Waterfall Literature* müssen daher im ersten Release auf eine App verzichten.

Offenbar hat *Agile Books* das bessere Vorgehensmodell gewählt, konnte schneller auf die Anforderungen des Markts reagieren, und mit der App lange vor *Waterfall Literature* seinen Umsatz erhöhen.

Allerdings mussten andere Anforderungen der App weichen. Handelt es sich dabei um Anforderungen, die für das Portal nicht essenziell sind, wie z.B. den Shopfinder, kann der geplante Releasetermin eingehalten werden. Die Anforderungen für den Shopfinder werden dann entweder fallen gelassen oder in einem Folgerelease umgesetzt. Da für den RFC keine umfangreiche Analyse der möglichen direkten und indirekten Auswirkungen durchgeführt wurde, kann es aber auch sein, dass durch die Erstellung der App eine essenzielle Anforderung nach hinten priorisiert wurde, z.B. die Anbindung an das Backend-System. In diesem Fall kann das Portal erst produktiv gehen, wenn auch diese Anforderung umgesetzt wurde. Damit hätte die Erstellung der App Dauer und Budget des Projekts erhöht, ohne dass dies dem Auftraggeber beim Priorisieren der App bewusst war. In diesem Fall könnte *Waterfall Literature* bereits Umsatz mit seinem neuen Onlineshop machen, während *Agile Books* noch am Relaunch seines Portals arbeiten würde.

Das Risiko für das Eintreten eines solchen Szenarios wird bei *Agile Books* dadurch begünstigt, dass der Product Owner Anforderungen vor allem aus Anwendersicht erstellt und priorisiert und das Team seinen Fokus vorrangig auf den jeweiligen Sprint richtet. Sprintübergreifende technische und nicht funktionale Anforderungen können dadurch aus dem Blickfeld geraten.

Fall 3: Komplexe Anforderungen umsetzen

Betrachten wir schließlich, wie sich die Anforderungen der Internationalität und der Anbindung von Fremdsystemen auf die Portale auswirken.

Das Team von *Agile Books* geht davon aus, dass sich alle Anforderungen des Product Backlogs, bis auf die des aktuellen Sprints, ändern können. Daher wählt es die Systemarchitektur nur so komplex, dass sie den jeweils aktuellen Anforderungen genügt. Mit jedem weiteren Sprint muss die Architektur entsprechend angepasst werden.

Bei *Agile Books* werden die Anforderungen nach Internationalisierung daher erst bei ihrer Umsetzung in der Systemarchitektur berücksichtigt. Bei *Waterfall Literature* hingegen müssen diese schon in der Entwurfsphase beachtet werden.

Die Anforderung nach Internationalisierung enthält unter anderem die Unterstützung von unterschiedlichen Zeitzonen. Damit gehört sie zu der Art von Anforderungen, deren Aufwand und Risiko steigt, je später sie im Projekt umgesetzt werden. So muss z.B. jede bestehende Codestelle im System betrachtet werden, die sich mit Datumswerten befasst und bestehende Daten müssen evtl. konvertiert werden. Wird diese Anforderung bereits beim Design des Systems beachtet, so sind Aufwand und Risiko minimal.

Der technische Projektleiter von *Waterfall Literature* hat während der Spezifikation mit dem Auftraggeber geklärt, dass die Speicherung der einkaufsrelevanten Daten im Portal und die Kommunikation mit dem Backend-System innerhalb einer Transaktion ablaufen sollen. Deshalb wurde für *Waterfall Literature* eine Systemarchitektur entworfen, die Transaktionen mit dem sog. "Zwei-Phasen-Commit" unterstützt: Zunächst wird bei allen Systemen geprüft, ob sie in der Lage sind, ihre Transaktion erfolgreich abzuschließen. Ist dies der Fall, werden die Transaktionen für alle Systeme abgeschlossen. Falls ein System nicht bereit ist, die Transaktion abzuschließen, weil sie z.B. zur Verletzung von Constraints (Gültigkeitsbedingungen) führen würde, wird die Transaktion für alle Systeme abgebrochen.

Das Team von *Agile Books* hat sich, im Sinne einer möglichst einfachen Systemarchitektur, über Transaktionen wenig Gedanken gemacht. Die Anbindung an das Backend-System wurde über eine sog. "REST-Schnittstelle" realisiert, die keine Transaktionen unterstützt, sondern lediglich Informationsstände übermittelt.

Im laufenden Betrieb kommt es bei *Agile Books* zu inkonsistenten Datenbeständen: Teilweise sehen Anwender in der Liste ihrer letzten Einkäufe Bücher, die nicht geliefert wurden, teilweise fehlen Bücher, die sie jedoch tatsächlich gekauft haben.

Eine aufwendige Analyse zeigt die Ursachen: Es kommt vor, dass Buchungen im Backend-System erfolgreich sind, das Speichern der Daten im Portal jedoch fehlschlägt, weil dies z.B. Constraints in der Portaldatenbank verletzen würde. Dann hat ein Anwender zwar einen erfolgreichen Einkauf getätigt, sieht diesen aber nicht in der Liste seiner Einkäufe. Wenn andererseits Buchungen im Backend-System nicht erfolgreich durchgeführt werden können, kann das umgekehrte Verhalten auftreten: Der Anwender sieht einen Einkauf, der im Backend-System nie angekommen ist.

Agile Books ist mit dem Verhalten im Produktivsystem äußerst unzufrieden, kann aber keine Nachforderungen stellen, da die Unterstützung von Transaktionen mit Zwei-Phasen-Commit keine explizite Anforderung war. Hier handelt es sich um eine technische Anforderung, deren Auswirkungen der Product Owner erst bemerkte, als Fehler im Produktivbetrieb auftraten. Durch die Anbindung des Backend-Systems über eine REST-Schnittstelle hat das Team von *Agile Books* eine Architektur gewählt, die Transaktionen mit Zwei-Phasen-Commit nicht unterstützt.

Um eine Neuentwicklung des Portals zu vermeiden, kann das hier beschriebene Verhalten nur durch Workarounds korrigiert werden. Z.B. könnten alle fehlgeschlagenen Transaktionen protokolliert und die Systeme dann manuell synchronisiert werden. Dies bedingt jedoch zusätzlichen Aufwand bei der Wartung des Portals.

Risiken für "Zeit und Budget"

Agile Projekte sind normalerweise auf die Umsetzung der Anforderungen des jeweils aktuellen Sprints fokussiert. Anforderungsänderungen während des Sprints werden dabei möglichst vermieden. Der Fortschritt der Umsetzung wird fortwährend beobachtet, um frühzeitig zu erkennen, ob die Erreichung des Sprintziels in Gefahr ist.

Ein probates Mittel der Überwachung ist das sogenannte Burndown-Chart, das verdeutlicht, in wie weit sich ein Sprint im Plan befindet.

Das Chart (Bild 2) zeigt an, wie viele Storypoints an einem bestimmten Tag noch offen sind. Die blaue Linie zeigt den idealen Verlauf bei konstanter Abarbeitungsgeschwindigkeit an. Am ersten Tag sind 140 Storypoints umzusetzen und am 15. Tag 0. Die rote Linie zeigt den Ist-Stand der Umsetzung. Diese Linie wird an jedem Tag des Sprints fortgeführt. Arbeitet das Team zu langsam oder zu schnell, so kommt es zu erheblichen Abweichungen zwischen den beiden Linien, und der Scrum Master muss gegensteuern.

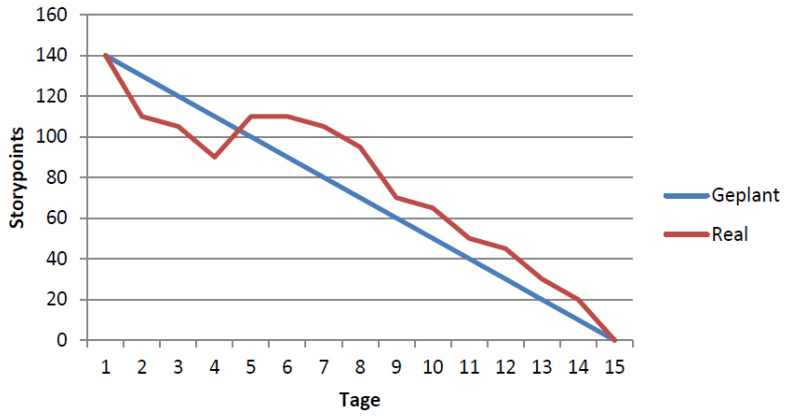

Bild 2: Typischer Verlauf eines Burndown-Charts für einen Sprint von 15 Tagen Dauer.

Durch die Konzentration auf den aktuellen Sprint können Zeit- und Budgetvorgaben für das Gesamtprojekt aus dem Blick geraten. Denn selbst wenn jede einzelne Iteration in Zeit und Budget zur Zufriedenheit des Kunden abgeschlossen wird, können am Ende der ursprünglich geplanten Laufzeit noch essenzielle Anforderungen übrigbleiben. In unserem Beispiel führte dies dazu, dass selbst die Realisierung des vermeintlich minimal notwendigen Leistungsumfangs für die App-Programmierung eine Überschreitung von Zeit und Budget nach sich zog.

Das Risiko der Zeitüberschreitung erhöht sich, wenn Software nicht nach jeder Iteration produktiv geschaltet wird. Dies betrifft Neuentwicklungen wie das Webportal in unserem Beispiel stärker als Projekte, die eine bereits in Betrieb befindliche Software weiterentwickeln.

Bei einem agilen Vorgehen werden bei der Erfassung neuer Anforderungen im Product Backlog bestehende Anforderungen oft nicht gestrichen, sondern – implizit – nach hinten priorisiert. Dem Auftraggeber wird erst spät bewusst, welche seiner Anforderungen nicht mehr im Rahmen des verfügbaren Budgets bzw. bis zum gesetzten Termin umgesetzt werden können.

Anders beim traditionellen Vorgehen: Bei einem RFC werden dessen Auswirkungen auf Termin und Budget des Projekts bereits im Vorfeld untersucht. Hier muss sich der Auftraggeber explizit entscheiden, welche RFCs er mit seinen vorhandenen Mitteln umsetzen möchte.

Wie kann man bei agilen Vorgehensweisen ein stärkeres Bewusstsein bezüglich der Einhaltung von Zeit und Budget erreichen? Ähnlich wie die Erfüllung des Sprintziels mit Hilfe des Burndown Charts geplant und kontrolliert wird, sollte auch für das gesamte Release vorgegangen werden. Die Projektbeteiligten sollten jederzeit erfahren können, ob sich das Projekt insgesamt noch in Zeit und Budget befindet.

Diese Information kann das relativ unbekannte Release Burndown Chart liefern. Mike Cohn hat dieses Chart so angepasst, dass auch Änderungen gegenüber dem initialen Product Backlog und deren Auswirkungen auf die Projektlaufzeit berücksichtigt werden (Cohn: Alternative Scrum Release Burndown Chart).

Je nach Größe des Projekts kann es sinnvoll sein die Anforderungen in Blöcken zusammenzufassen und für jeden Block ein Budget einzuplanen (in unserem Beispiel also für Shopfinder, Backend-System, Apps). Zum Controlling des Budgets für jeden dieser Blöcke sollte dann jeweils ein eigenes Burndown Chart erstellt werden.

Die Balken des Release Burn Down Chart (Bild 3) zeigen die pro Sprint noch offenen Anforderungen in Storypoints an. Die orange Linie zeigt die pro Sprint abgearbeiteten Storypoints bei gemittelter Abarbeitungsgeschwindigkeit an. Eine Änderung der Anforderungen wird jeweils am unteren Ende der Balken zugefügt bzw. gelöscht (im Chart der hellblaue Anteil der Balken unterhalb der Abszisse). Dies bewirkt dann ein Verschieben der Grundlinie. Das voraussichtliche Releasedatum ergibt sich aus dem Schnittpunkt der aktuellen Grundlinie (blaue Linie) und der orangen Linie. Im Beispiel-Chart werden voraussichtlich alle Anforderungen erst mit Sprint 7 umgesetzt.

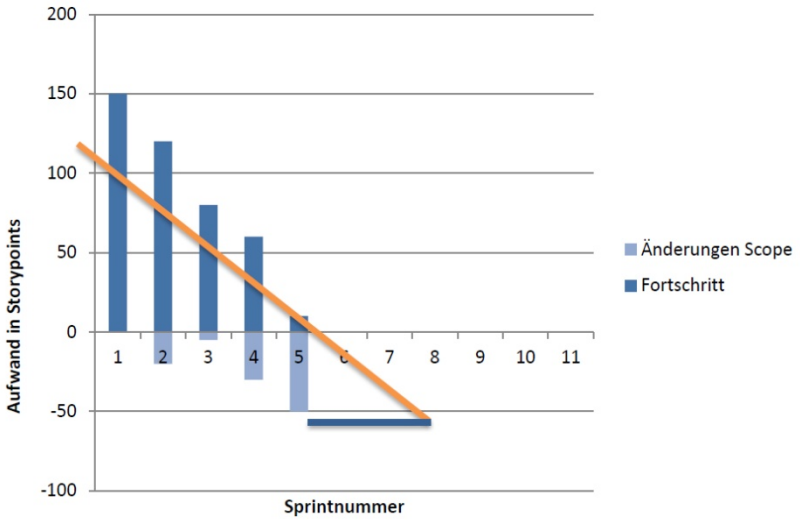

Bild 3: Angepasstes Release Burn Down Chart nach Mike Cohn.

Ein aktuelles Release Burndown Chart sollte bei jedem Sprintreview präsentiert werden. Auf diese Weise wird auf einen Blick deutlich, ob sich das Projekt in Zeit und Budget befindet. Bei drohenden Überschreitungen können rechtzeitig Gegenmaßnahmen getroffen werden.

Risiken für die Qualität agil entwickelter Software

Die Optimierung der Vorgehensweise auf die unverzügliche, flexible Umsetzung von Anforderungen wirkt sich auch auf die Qualität der gelieferten Software aus. Im Idealfall wurde eine schlanke Architektur entwickelt, die weder unnötige Komplexität noch überflüssige Features enthält. Im schlimmsten Fall kann aber Software entstehen, die den an sie gestellten Anforderungen nicht mehr gewachsen ist. In unserem Beispiel wurde vom Team von *Agile Books* mit Ruby on Rails ein Entwicklungsframework gewählt, mit dem sich sehr viele Anforderungen

sehr effizient umsetzen ließen, das aber die Anbindung an das Backend-System nicht in der gewünschten Art und Weise leisten kann.

Um eine tragfähige Software-Architektur zu gewährleisten, ist es wichtig zu Beginn eines Projekts die bekannten Anforderungen bzgl. ihrer Auswirkungen auf die Architektur zu bewerten und die Architektur dann entsprechend zu definieren. In unserem Beispiel hätte man bei *Agile Books* schon zu Beginn des Projekts eine Architektur wählen müssen, die Zwei-Phasen-Commits unterstützt.

Agile Software hat ein hohes "Alterungsrisiko"

Das ideale Scrum Team macht bei der Qualität der erstellten Software keinerlei Abstriche. So die Theorie. In der Praxis gibt es bei agilen Projekten, genauso wie bei traditionellen, kurz vor Auslieferung der Software die Tendenz, eine Anforderung lieber "mehr schlecht als recht" umzusetzen anstatt sie zurückzustellen. Als Entwickler ist es befriedigender dem Anwender vermeintlich fertige Funktionen zu präsentieren, als erklären zu müssen, warum die entsprechenden Anforderungen nicht umgesetzt werden konnten. Hinzu kommt der Erwartungsdruck des Auftraggebers, der in der Regel nervös wird, wenn wiederholt nicht alle Anforderungen eines Sprints umgesetzt wurden und somit absehbar ist, dass die gewünschte Funktionalität nicht innerhalb des veranschlagten Budgets umgesetzt werden kann. In der Praxis kann man daher häufig beobachten, dass die Qualität von Software mit jedem erstellten Release sinkt.

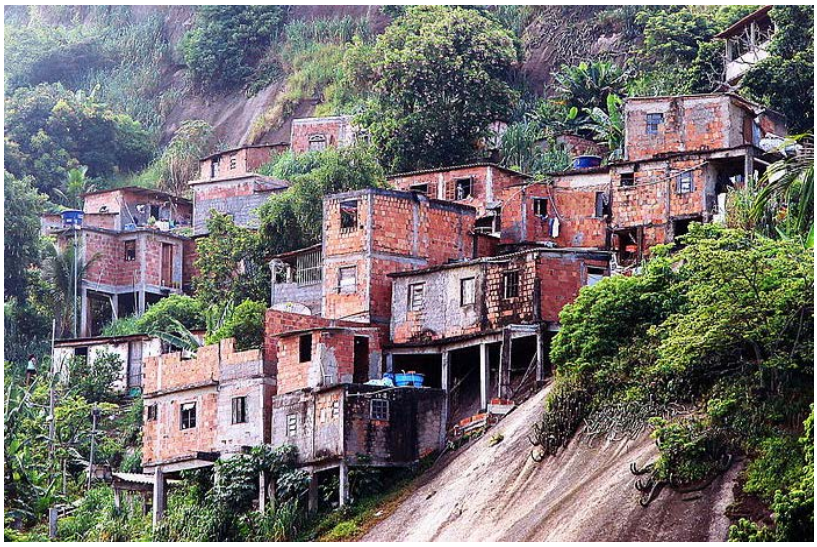

Bild 4: Die Architektur dieses Bauwerks zeigt die Vor- und Nachteile eines agilen Vorgehens: Es wurde kein Stein zu viel verbaut, die Statik setzt zukünftigen Erweiterungen allerdings enge Grenzen (Quelle: Eurico Zimbres, http://de.wikipedia.org/wiki/Datei:Favela-Niteroi.JPG)

Da bei agilen Vorgehensweisen nach jedem Sprint ein lauffähiges Softwarerelease erstellt wird, entstehen mehr Releases als bei traditionellen Vorgehensweisen, was den oben beschriebenen Effekt verstärken kann.

Ob ein Team einen hohen Qualitätsstandard vertritt oder einen eher laxen Umgang mit Softwarequalität pflegt, kann der Auftraggeber den bei einem Sprintreview präsentierten Features in der Regel nicht ansehen.

Deshalb ist es sinnvoll, Tools zur automatisierten Erstellung von Metriken zur Softwarequalität einzusetzen. Neben kommerziellen Tools wie CAST (http://www.castsoftware.com) kommen hierfür auch lizenzfrei verfügbare Programme wie sonar (http://www.sonarsource.org) in Frage. Solche Tools bieten Übersichts-Charts mit Metriken, welche die Einhaltung von Programmierrichtlinien, den Grad der Testabdeckung, die Anzahl doppelter

Codestellen und den Grad der Dokumentation über den Zeitverlauf beschreiben. Diese Charts geben auch dem Laien Hinweise auf Veränderungen der Qualität der Software. Über die Einbindung solcher Tools in Continuous Integration Systeme lassen sich diese Charts regelmäßig generieren, so dass jedes Projektmitglied immer auf einen aktuellen Stand zugreifen kann. Auch diese Charts sollten bei jedem Sprint Review präsentiert werden.

Scrum definiert einen Abnahmeprozess für die in einem Sprint umgesetzten Anforderungen. Nach der Präsentation der umgesetzten Anforderungen legt der Product Owner fest, ob das Sprintziel erreicht wurde oder nicht. Diese Beschränkung auf die umgesetzten Anforderungen greift unserer Meinung nach meist zu kurz. Wichtig wäre, dass der Product Owner bei seiner Bewertung mit einbezieht, ob die geforderte Qualität eingehalten wurde und ob sich das Gesamtprojekt bzgl. Zeit und Budget noch im Rahmen befindet. Die Charts, die für die Überwachung der Qualität notwendig sind, kann das Entwicklungsteam bereitstellen. Die Aktualisierung des angepassten Release Burndown Charts wäre eine Aufgabe für den Scrum Master.

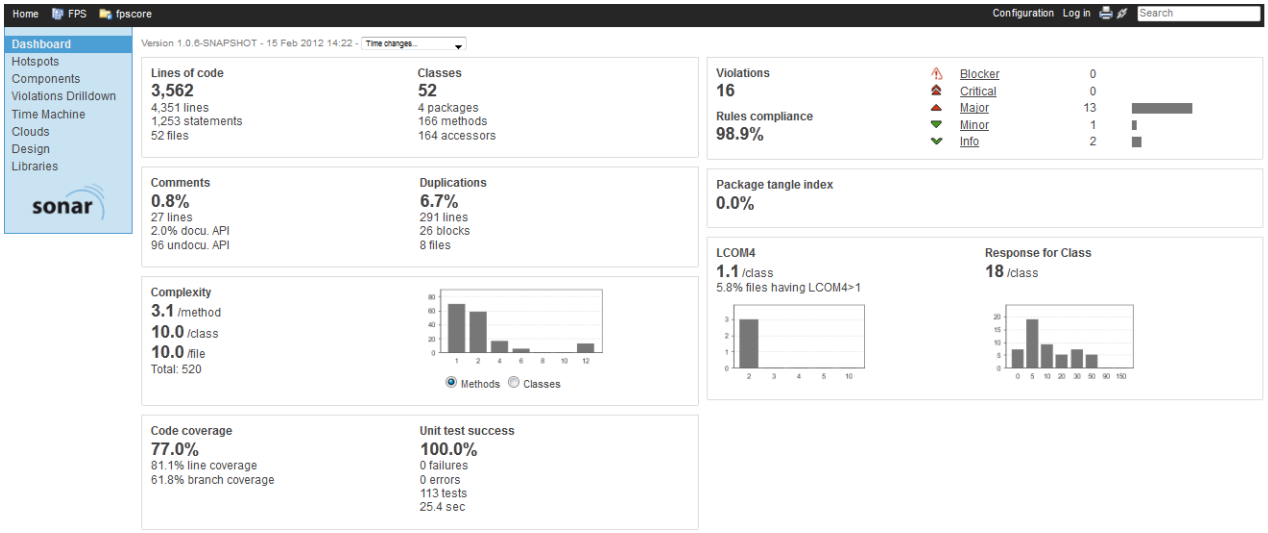

Bild 5: Sonar Dashboard zur Visualisierung von Softwarequalität. Auch Nichtprogrammierer können hier auf einen Blick erkennen, wie sich die Softwarequalität entwickelt hat.

Verwendung neuer Technologien

Bei einem traditionellen Projektvorgehen liegen die Hürden für die Einführung neuer Technologien während des Projekts (im Beispiel Ruby on Rails) relativ hoch, so dass die Entwickler dadurch tendenziell weniger effizient programmieren. Dagegen liegen diese Hürden bei einem agilen Vorgehen vergleichsweise niedrig.

In der Praxis führt die Selbstbestimmung agiler Teams deshalb häufig dazu, dass in jedem Projekt eine andere Technologie eingesetzt wird (z.B. Ruby on Rails, Spring Web MVC, JBoss Seam, Groovy…). So entstehen schnell IT Landschaften, die nur schwer zu warten sind, da für jede Laufzeitumgebung unterschiedliches technisches Know-how vorgehalten werden muss.

Bei der traditionellen Vorgehensweise schafft ein RFC mehr Klarheit, da er alle Projektbeteiligten, von der Fachseite bis hin zum Betrieb, über geplante Änderungen informiert. Auch in agilen Projekten sollte es daher selbstverständlich sein, dass vor einer Entscheidung über die Verwendung neuer Tools und Frameworks möglichst alle Beteiligten über die Vor- und Nachteile sowie die damit verbundenen Risiken informiert werden. Auf diese Weise können Risiken schnell erkannt werden, und der Auftraggeber kann entscheiden, ob er einer Einführung zustimmt.

Technische Entscheidungen, die die IT Landschaft eines Auftraggebers beeinflussen, sollten stets auch mit dem Betrieb des Auftraggebers abgestimmt werden. Hier ist es durchaus sinnvoll, sich an Prozesse des Betriebs zu halten. Auch wenn diese auf den ersten Blick "verkrustet" und nicht agil erscheinen, so haben sie sich in der Regel bewährt und sollten nur behutsam geändert werden.

Erstellung einer adäquaten Software-Architektur

Beim Entwurf von Software-Architektur bewegt man sich zwischen zwei Extremen: Versucht man zu Beginn eines Projekts alle nur denkbaren zukünftigen Anforderungen zu berücksichtigen, so entsteht ein System, das zwar in der Theorie für jegliche Anforderung bereit ist, in der Praxis aber nicht gewartet werden kann. In unserem Beispiel könnte ein solcher Fall eintreten, wenn beim Entwurf der Architektur bereits versucht wird, neben Büchern den Verkauf weiterer Produkte im Onlineshop zu berücksichtigen. Auf der anderen Seite kann eine zu einfache Architektur dazu führen, dass bestehende Anforderungen nur mit Mühe oder gar nicht umgesetzt werden können.

In unserem Beispiel betrifft dies die Unterstützung unterschiedlicher Zeitzonen, deren frühzeitige Berücksichtigung den Aufwand und das Risiko bei der Umsetzung minimiert hätte so wie die – nicht explizit ausgesprochene – Anforderung nach Unterstützung von Transaktionen mit Zwei-Phasen-Commit. Ziel sollte es sein, die Software-Architektur so einfach wie möglich zu entwerfen – aber nicht einfacher.

Bild 6: Der "Koloss von Prora" auf Rügen ist ein Beispiel für eine Architektur nach klassischem Vorgehen mit unrealistisch hohen Anforderungen. Das Seebad wurde nie fertig gestellt und verfällt seitdem - die Welt hatte sich durch Ausbruch des 2. Weltkrieges zu sehr verändert (Quelle: Steffen Löwe, http://commons.wikimedia.org/wiki/File:ProraLandseite.jpg)

Um auch in agilen Projekten eine tragfähige Architektur zu gewährleisten, sollte der Auftragnehmer zu Beginn eines Softwareprojekts alle bekannten Anforderungen möglichst durch erfahrene Mitarbeiter bezüglich ihrer Auswirkungen auf das Systemdesign bewerten lassen. Typische Beispiele für solche Anforderungen sind Internationalisierung, Anbindung von Systemen und nicht funktionale Anforderungen wie

Performance, Ausfallsicherheit oder Skalierbarkeit. In Zusammenarbeit mit dem Auftraggeber sollten diese Anforderungen soweit präzisiert werden, dass eine adäquate Systemarchitektur erstellt werden kann (also z.B. für die Anbindung an das Backend-System über Zwei-Phasen-Commit). Dieses Vorgehen kann zwar nicht gewährleisten, dass völlig neue Anforderungen von der Architektur erfüllt werden können, stellt aber zumindest sicher, dass sie bestehende Anforderungen erfüllt.

Fazit

Wir haben anhand eines Beispiels zwei Vorgehensweisen gegenübergestellt, die auf unterschiedliche Aspekte optimiert sind.

Das traditionelle Vorgehen verspricht Sicherheit bei der Einhaltung von Zeit und Budget sowie die adäquate Umsetzung der spezifizierten Anforderungen. Beim Erkennen und Reagieren auf Anforderungsänderungen während der Projektlaufzeit hat dieses Vorgehen in seiner reinen Form hingegen deutliche Schwächen.

Das agile Vorgehen ist auf das schnelle Erkennen von Anforderungsänderungen und deren Umsetzung optimiert. Es verspricht Flexibilität, die allerdings zu Lasten der Einhaltung von Zeit, Budget und Qualität gehen kann.

Jedoch ist es für die erfolgreiche Umsetzung von Projekten sowohl wichtig die Rahmenbedingungen von Zeit, Budget und Qualität einzuhalten, als auch die Bedürfnisse der Anwender optimal zu erfüllen.

Um dies zu erreichen, sollte die Flexibilität eines agilen Projekts an einigen Stellen eingeschränkt werden. Wenn Anforderungen die Komplexität der Software-Architektur beeinflussen oder andere Projekte betreffen (z.B. bei der Definition von Schnittstellen), dann sollten sie besser traditionell umgesetzt, d. h. frühzeitig spezifiziert und eingeplant werden. Die grundlegende Änderung solcher Anforderungen kann den Projekterfolg gefährden und sollte daher nur mit Bedacht erfolgen. Anforderungen, die nur geringe Auswirkungen auf die Software-Architektur haben und keine anderen Projekte betreffen, können dagegen agil umgesetzt werden.

Da sich grundlegende Anforderungen, etwa die Anbindung an andere Systeme oder die Internationalisierung, während eines Projekts selten ändern, stellt diese Forderung in der Regel keine große Einschränkung dar. Auf der anderen Seite hilft sie bei der Einhaltung von Zeit und Budget und kann dadurch auch Auftraggeber beruhigen, die agilen Methoden skeptisch gegenüber stehen.

Die Einhaltung der Softwarequalität wird bei beiden Vorgehensweisen nicht explizit adressiert. Die Annahme, dass agile Teams per Definition eine hohe Qualität abliefern, ist so naiv wie die Annahme, dass es im laufenden Projekt keine Änderungen an der Spezifikation geben wird. Der Unterschied zwischen den beiden Vorgehensweisen besteht allerdings darin, dass bei einem traditionellen Vorgehen der Auftragnehmer das Risiko für auftretende Fehler trägt, da er diese im Rahmen des festen Budgets beheben muss. Bei einem agilen Vorgehen trägt dieses Risiko erfahrungsgemäß eher der Auftraggeber. Da es häufig keine klare Trennung zwischen Fehlerbehebung und Feature-Entwicklung gibt, zahlt er den Aufwand für Fehlerbehebung, ohne dass ihm das bewusst ist.

Erfahrene Teams werden sowohl mit einem traditionellen als auch mit einem agilen Vorgehen erfolgreich Projekte umsetzen. So können auch in einem Projekt nach dem Wasserfallmodell Anforderungen ohne RFC ausgetauscht

werden (in unserem Beispiel der Shopfinder durch die App), wenn ein entsprechendes Vertrauensverhältnis zwischen Auftraggeber und Auftragnehmer herrscht. Ein erfahrenes agiles Team wird bei der Entscheidung für eine Systemarchitektur alle architekturkritischen Anforderungen berücksichtigen und vor dem Einsatz neuer Technologien mit allen betroffenen Abteilungen und Personen Rücksprache halten.

Bei aller Begeisterung für agile Vorgehensweisen ist es gerade für unerfahrene Teams wichtig zu verstehen, dass bewährte Prozesse zur Einhaltung von Zeit und Budget ökonomisch sinnvoll sind. So kann ein agiles Projekt mit "etwas Wasserfall" nicht nur zur Zufriedenheit der Anwender, sondern auch in Zeit-, Budget- und Qualitätszielen umgesetzt werden.

Literatur

- Wirdemann, Ralf: Agiles Projektmanagement. Scrum – eine Einführung, Projekt Magazin, 21/2009
- Cohn, Mike: Alternative Scrum Release Burndown Chart, http://www.mountaingoatsoftware.com/pages/19-an-alternative-release-burndown-chart