Hans-Peter Korn

Scrum ohne murcS: Agile Projektmanagementpraxis

(Vorabdruck aus dem Buch http://www.symposion.de/?cmsinfo/0003435)

Scrum verbreitet sich nun auch in großen Firmen rasant. Oft aber wird es als „Scrumade" praktiziert: Die gewohnten Praktiken leben weiter hinter Scrum als formaler Fassade. Zudem wird Scrum für alle Arten von Projekten, für Betrieb, Support und sogar als generelle Managementmethode zu nutzen versucht.

In diesem Beitrag erfahren Sie,

- *wann Agilität und Scrum angemessen sind – und wann nicht,*
- *wie Scrum funktioniert,*
- *wie Scrum in großen Unternehmen inkrementell eingeführt werden kann.*

Sie überlegen, in Ihrem Unternehmen oder Ihrer Abteilung Scrum einzuführen?

Dann berücksichtigen Sie bitte zuallererst Folgendes:

- Scrum ist ein auf das Minimum reduziertes Framework für das agile Vorgehen bei der Produktentwicklung unabhängig von der Art des Produkts. Es kann daher – z. B. im IT-Bereich – nicht anstelle aller bisher praktizierten Methoden und Techniken der Software(SW)-Entwicklung eingeführt werden, sondern dient als „Container" und muss um die weiterhin nötigen Methoden und Techniken unternehmensspezifisch ergänzt werden. Problematisch wird eine Scrum-Einführung dann, wenn bisher keine Methoden und Techniken der SW-Entwicklung professionell genug praktiziert wurden und von Scrum allein Professionalität erwartet wird.
- Scrum ist keine Methode zur Steigerung der Effizienz der SW-Entwicklung. Scrum unterstützt ein iterativ-inkrementelles Vorgehen, mit dem die betrieblich relevantesten SW-Funktionen zuerst geliefert werden, um sie auch als Erste zu nutzen und rasches Feedback dazu von den Nutzern zu erhalten. Damit erhöht Scrum primär die Effektivität und unterstützt die auf der Nutzerseite betrieblich nötige Flexibilität und Dynamik.
- Scrum ist keine Legitimation dafür, endlich „offiziell abgesegnet" SW-Funktionalität schnell und improvisiert zu liefern, ohne vorher die für das Planen und Designen des Planbaren und Designbaren nötige Zeit zu investieren.
- Scrum ist kein Freibrief für die Anwender, ihre Anforderungen ohne möglichst gründliche Vorüberlegungen und nach dem Prinzip „Wunschkonzert" zu formulieren und jederzeit beliebig zu ändern.
- Scrum beruht auf wenigen, aber klar definierten Verantwortlichkeiten („Rollen") für die Produktentwicklung und ist kein missverstandenes partizipatives Führungsmodell, bei dem der Rahmen für die Selbstorganisation des Teams durch das sich selbst organisierende Team festgelegt wird. Scrum ist zudem ein Rahmenwerk für die Produktentwicklung, kein Führungsmodell nur auf Basis der „Scrum-Rollen". Die personelle Führung bleibt zusätzlich erhalten.

Wenn eine dieser fünf Aussagen dem widerspricht, was mit der Einführung von Scrum bezweckt wird, dann bitte: *Hände weg von Scrum!*

Wenn Sie diesen fünf Aussagen zustimmen können, dann geht es weiter in die „nächste Runde":

Inwieweit „passt“ die Agilität zu den Projekten und Produkten des Unternehmens?

Das kann anhand dieser „Agilitätsschieberegler“ (Abb. 1) grob beurteilt werden:

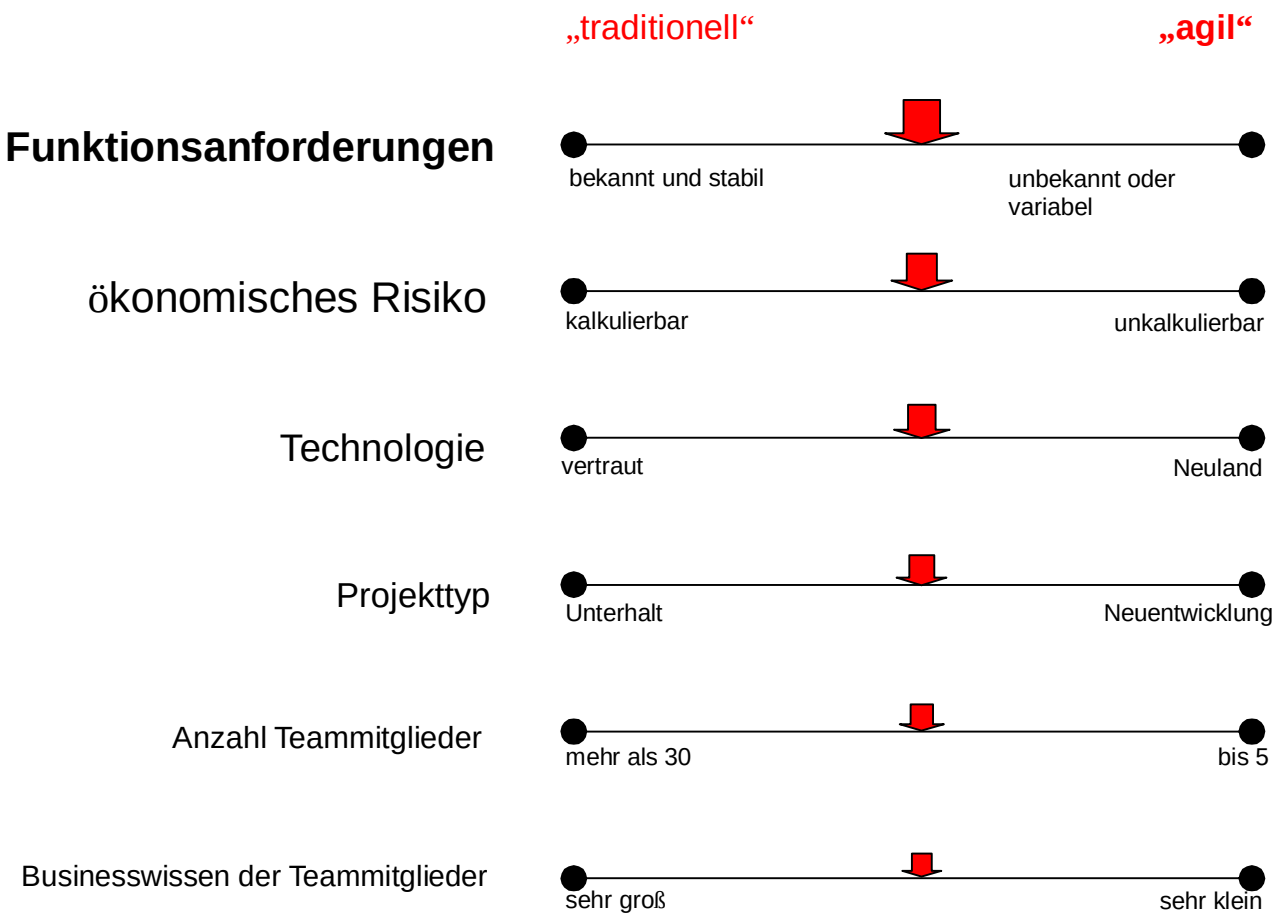

Abb. 1: *Agilitätsschieberegler*

Je weiter die Schieberegler rechts stehen, umso eher ist das agile Vorgehen angemessen. Die Größe der Beschriftung und der Pfeile steht dabei für die Bedeutung des jeweiligen Kriteriums. Je mehr die Schieberegler – insbesondere bei den dominanten Kriterien – links stehen, umso eher ist ein traditionelles Projektvorgehen angemessen.

Bei einer agilen Projekt- und Produktcharakteristik können Sie nun anhand der „Kultur-Schieberegler“ (Abb. 2) überlegen, ob ein eher management- und individualitätsorientiertes agiles Verfahren oder ein eher emergenz- und teamorientiertes agiles Verfahren passend ist:

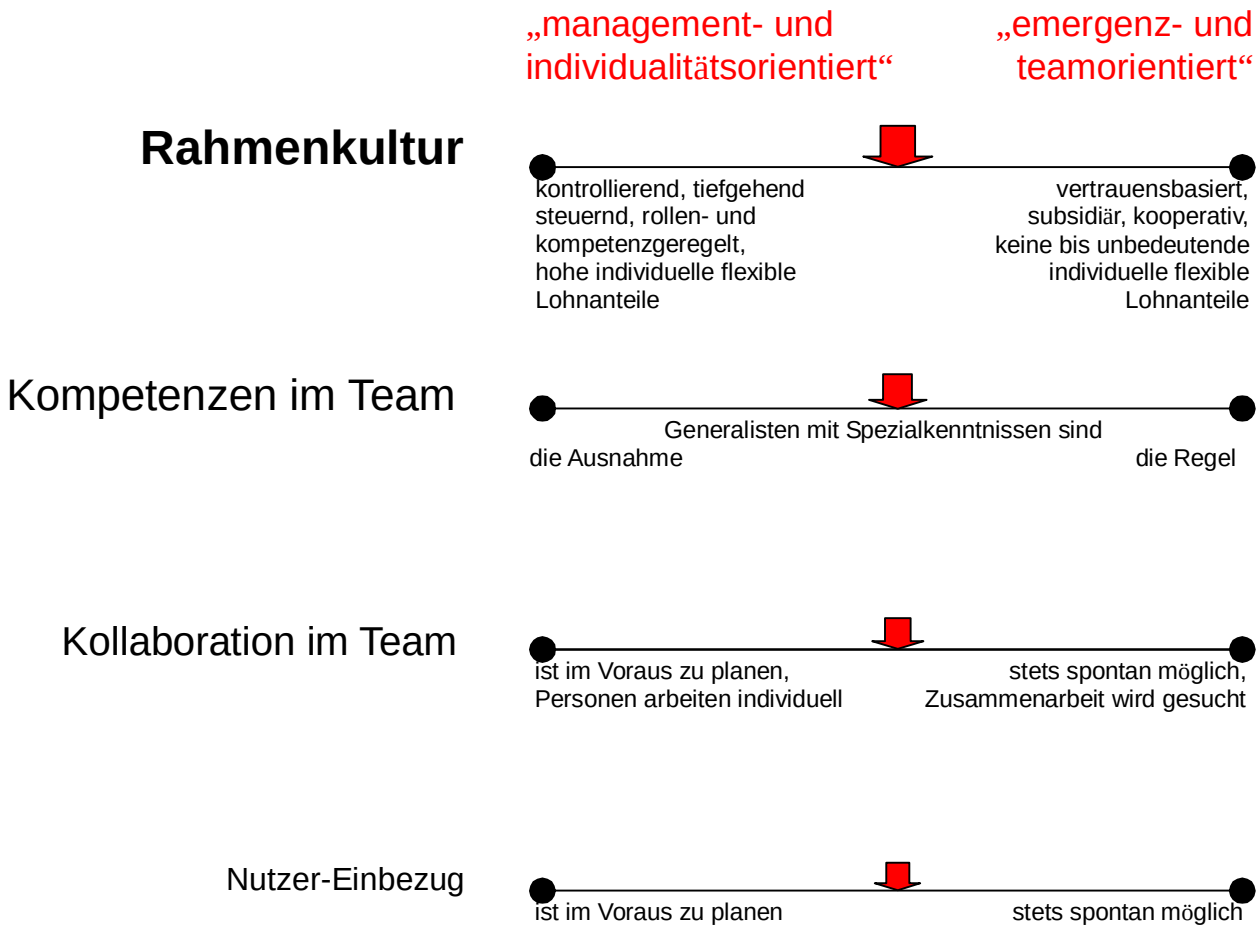

Abb. 2: *Kultur-Schieberegler*

Wenn die Schieberegler vor allem der dominanten Kriterien eher rechts stehen, ist Scrum eine passende Option. Stehen sie jedoch eher oder deutlich links, dann sollte Agilität besser auf Basis z. B. eines entsprechend „zurechtgeschneiderten“ Rational Unified Process (agile RUP) oder KANBAN praktiziert werden.

Mehr zu den Hintergründen dieser Selbsteinschätzung erfahren Sie in den folgenden Kapiteln.

Wann und warum überhaupt „agil“ vorgehen?

Dass der Managementprozess aus den Phasen Zielsetzung – Planung – Realisierung – Kontrolle besteht, entspricht der weit verbreiteten Sichtweise. Und dass auch Projekte auf dieser Basis zu managen sind, findet ebenso breite Zustimmung. Aus unserer Alltagserfahrung wissen wir, dass genau dieses Vorgehen erfolgreich ist. Deshalb wenden wir es auch immer wieder als „Option der ersten Wahl“ an. Ein Beispiel:

Wenn ich mit dem Zug von Mantua nach Erlangen fahren möchte, dann ist das keine ganz triviale Angelegenheit. Ich werde das also anhand des Fahrplans im Detail planen. Vor einigen Jahren noch hätte ich dazu dicke Bücher verschiedener Bahnbetreiber wälzen müssen, heute erhalte ich innerhalb von Sekunden mit dem Onlinefahrplan z. B. der SBB diese Auskunft:

Mantua	ab	12:05	IC 621
Modena	ab	13:40	IC 9814
Milano	ab	16:10	S-Bahn
Bellinzona	ab	18:06	IR 2192
Basel	ab	22:07	CNL40478
Frankfurt	ab	04:44	ICE 827
Nürnberg	ab	07:01	ICE 1612
Erlangen	an	07:18	

Ich entscheide mich für den gewünschten Abfahrtstag und kann dann via Internet oder am Bahnschalter die Fahrkarten kaufen und die Reise realisieren. Zwischendurch werde ich immer wieder mal kontrollieren, wie pünktlich der Zug ist, um sicher zu sein, dass die Anschlüsse klappen. Und möglicherweise habe ich auch bereits alternative Verbindungen notiert, sollte ein Anschluss nicht funktionieren.

Die ganze Reise ist also gut vorausplanbar und aus Erfahrung weiß ich, dass das mit großer Wahrscheinlichkeit auch wie geplant funktionieren wird.

Nun stellt sich aber heraus, dass am Abfahrtstag die Bahnen in Italien bestreikt werden. Sicher ist, dass keine Lokalbahnen und S-Bahnen verkehren. Die IC-Züge fahren „vermutlich“. Und dummerweise sind auch noch heftige Schneefälle auf der Alpennordseite und für Süddeutschland vorausgesagt. Zwischen Bellinzona und Frankfurt sind die Anschlüsse also sehr unsicher.
Was tun? Aufs Auto umsteigen? Bei diesem Wetter – nee. Und die Flüge sind ausgebucht. Und wetterbedingt ohnehin stark verspätet.

Also: Probieren wir es etappenweise „Richtung Nord“. Je nach Streiksituation gibt es vielleicht doch noch einen Zug. Oder einen Bus. Bis Milano werden wir schon irgendwie kommen. Und dort schauen wir weiter. Vielleicht sind die Schneefälle für den Zugverkehr nicht so schlimm wie befürchtet.

Wir reisen also Etappe für Etappe nach dem Prinzip „Probieren – Beobachten – Anpassen“ Richtung Erlangen. Nicht chaotisch, sondern „mit System“ und „agil“.

In dieser von Unsicherheit und Unplanbarkeit geprägten Situation haben wir den vertrauten Managementprozess „Zielsetzung – Planung – Realisierung – Kontrolle“ verlassen und durch das agile „Probieren – Beobachten – Anpassen“ ersetzt.

Übertragen wir das nun auf das Managen von Projekten:

Der vertraute Projektablauf umfasst Phasen wie

- Initialisierung
- Voranalyse
- Konzept
- Realisierung
- Einführung
- Abschluss

Im Detail betrachtet treffen wir sehr häufig auf Situationen dieser Art: (Abb. 3)

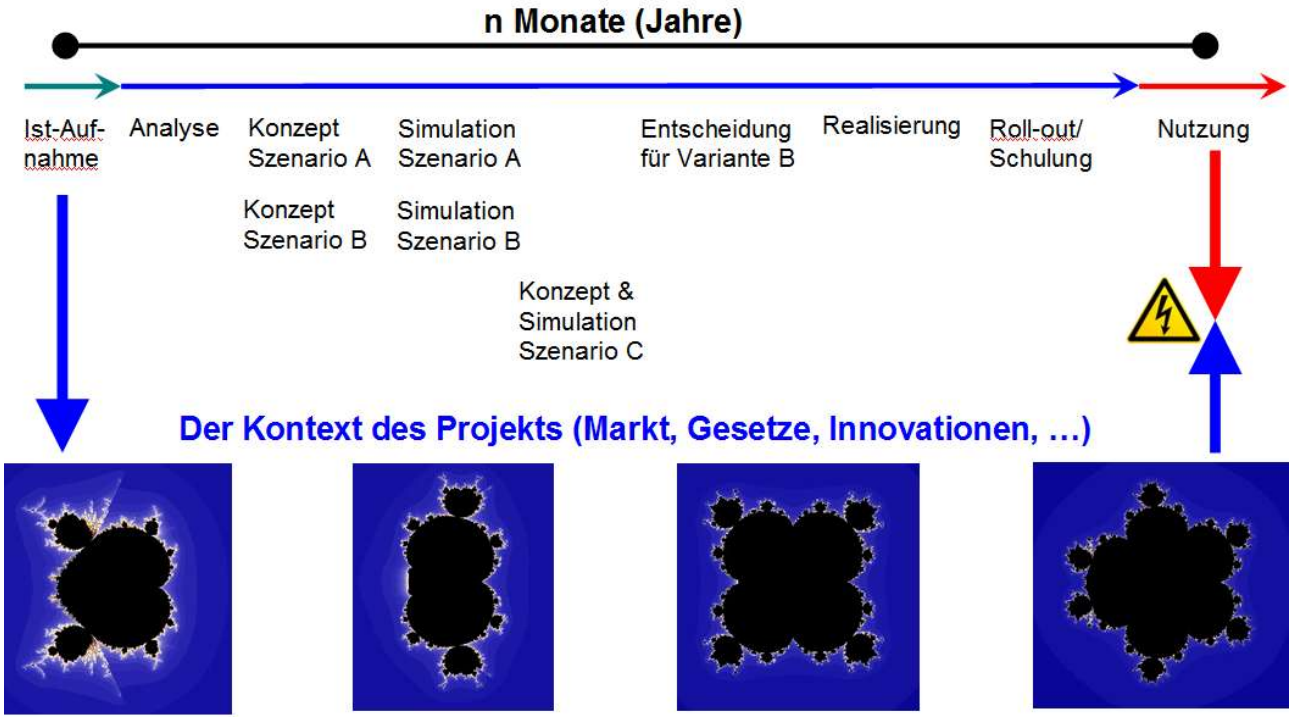

Abb. 3: *Vertrauter Projektablauf – im Detail betrachtet*

In der Voranalyse untersuchen wir sehr genau die Rahmenbedingungen und Anforderungen. Diese sind dann Basis für Konzepte und Konzeptvarianten, die im Detail überprüft werden. Die am besten geeignete Variante wird gewählt, realisiert, ausgerollt und genutzt. All das dauert etliche Monate bis Jahre. In dieser Zeit hat sich aber unter Umständen der Kontext des Projekts verändert. Die zu Projektbeginn erkannten Rahmenbedingungen der und Anforderungen an die Lösung sind jetzt teilweise überholt. Die jetzt – endlich – fertiggestellte Lösung passt nicht mehr.

Möglicherweise werden Sie jetzt einwenden: Im Projektverlauf muss eben immer wieder auch der veränderte Projektkontext beobachtet und analysiert werden und die Konzeptionen sind fortlaufend zu aktualisieren. Damit würden jedoch nahezu alle Projekte in der Konzeptionsphase stecken bleiben. Und wir alle wissen, dass es genau solche blockierten Projektsituationen recht häufig gibt. Deshalb muss es bei diesem Vorgehen in „sequentiellen Phasen“ immer so etwas wie einen „design freeze“ oder „Gates“ geben – mit dem Preis, dass man sich vom Projektkontext „abkoppelt“ und zum Zeitpunkt des Nutzungsbeginns eine nicht mehr passende Lösung vorliegt.

Im Verlauf eines Projekts gibt es zusätzlich dazu auch noch eine Reihe anderer möglicher Effekte:

- Bereits die Kenntnis – oder das Gerücht –, dass es so ein Projekt gibt, verändert den Markt (Kaufverhalten der Kunden, Produktentwicklungen der Konkurrenten, …).
- Im Verlauf der Konzeption und Realisierung entstehen neue Ideen und Sichtweisen.
- Veränderungen innerbetrieblicher Macht- und Organisationsstrukturen (neuer Chef mit neuen Ideen, Bereichszusammnenlegungen etc.) verändern die Kostenstrukturen … und damit die Wirtschaftlichkeit einzelner Lösungskomponenten.
- Ein Keyplayer im Projektteam wird für eine zweiwöchige „Feuerwehrübung“ in ein anderes Projekt abkommandiert … bleibt dort aber vier Monate.
- Etc.

Unplanbarkeit bei Komplexität

Ist die Antwort auf die genannten Unwägbarkeiten eines Projekts, noch umfassendere Analysen und Konzeptionen auszuarbeiten sowie noch rigideres Risiko- und Projektmanagement zu betreiben? Nein. Nicht unprofessionelle Arbeit ist „schuld“ an vielen Projektproblemen, sondern eine der Komplexität unangemessene Vorgehensweise.

Dave Snowden bietet mit seinem „Cynefin Sense Making Framework“ [1] eine gute Orientierungshilfe. Er unterscheidet fünf Domänen [2].

„Die ersten vier davon sind:

- *Simple*, in der die Beziehung zwischen Ursache und Wirkung für alle offensichtlich ist. Die Herangehensweise ist hier *Sense – Categorise – Respond*, und wir können bewährte Praktiken (best practice) anwenden.
- *Complicated*, in der die Beziehung zwischen Ursache und Wirkung Analyse oder eine andere Form der Prüfung erfordert und/oder die Anwendung von Fachwissen. Hier geht man mittels *Sense – Analyze – Respond* heran, und man kann *gute* Praktiken (good practice) anwenden.
- *Complex*, in der die Beziehung zwischen Ursache und Wirkung nur im Nachhinein wahrgenommen werden kann, aber nicht im Voraus. Hier ist der Ansatz *Probe – Sense – Respond*, und wir können *emergente* Praktiken (emergent practice) feststellen.
- *Chaotic*, in der es keine Beziehung zwischen Ursache und Wirkung auf Systemebene gibt. Hier ist der Ansatz *Act – Sense – Respond*, und wir können *innovative* Praktiken entdecken.

Die fünfte Domäne ist *Disorder*, der Zustand des Nicht-Wissens, welche Art von Kausalität besteht. In diesem Zustand gehen die Leute in ihre eigene Komfort-Zone zurück, wenn sie eine Entscheidung fällen.“

IT-Systeme, als rein technische Systeme betrachtet, können wir als komplizierte und nicht als komplexe Systeme betrachten. Auch wenn sie noch so undurchsichtig erscheinen – mit genügend Analyseaufwand und Expertenwissen sind sie derart analysierbar, dass deren künftiges Verhalten kontrollierbar ist. Deshalb vertrauen wir auch den Kontoauszügen unserer Bank. Mit dieser „Maschinensicht“ haben wir große Erfolge erzielt – bis hin zur Schulmedizin bei der Behandlung auch schwerster Krankheiten. Und wir konnten damit auf dem Mond landen.

Daraus folgt: Komplexe Projekte können – längerfristig – nicht vorausgeplant werden. Die vernünftig im Voraus planbare Zeitspanne hängt dabei von vielen Faktoren ab:

- Umfang und Art der Produkt- und Projekt-Stakeholder
- Dynamik der Marktsituation (Lieferanten und Nutzer)
- Dynamik der (technischen) Innovationen
- Größe und Zusammensetzung des Projektteams
- Umfang, Vielfalt und Dynamik der funktionalen Anforderungen
- Dynamik und „Spirit“ der Organisation etc.

O. k. – und was jetzt?

Bei Unplanbarkeit „adaptiv“ vorgehen

Die Lösung besteht im Kern darin: „Probe – Sense – Respond“ statt „Sense – Analyze – Respond“. Das heißt:

- In kleinen Schritten so vorgehen, sodass mit jedem Schritt ein weiteres – kleines – Stück der Lösung erzeugt wird.
- Alle diese „Stücke“ können sofort genutzt werden – zumindest als „Piloten“ in der realen Umgebung.
- Die damit gemachten Erfahrungen werden rasch zurückgemeldet.
- Aus diesen Erfahrungen werden so schnell wie möglich neue oder geänderte Anforderungen für einen der nächsten Schritte abgeleitet.

Ab den 1990er-Jahren entstanden in der Informatik auf dieser Basis diverse „alternative“ Formen des Projektmanagements – gekennzeichnet durch

- iterativ-inkrementelles Erstellen einzelner kleiner, bereits brauchbarer Lösungsteile,
- permanente und enge Zusammenarbeit und persönliche Kommunikation mit den Nutznießern sowie
- Detailplanung und Erarbeitung der Lösung durch innerhalb klarer Grenzen sich selbst organisierender Projektteams.

Adaptiv dank „Agilität“

2001 haben 17 Protagonisten dieser Vorgehensweisen bei einem Treffen in Utah als Essenz das „Agile Manifest“ formuliert. Zwei davon, Ken Schwaber und Jeff Sutherland, sind die Begründer von Scrum.

Das Manifest umfasst vier „Werte“ und zwölf „Prinzipien“. Die vier Werte lauten: [3]

„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 Vertragsverhandlung
- *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.“

Agilität ist nicht immer angemessen

Derzeit verbreiten sich in der IT agile Vorgehensweisen und insbesondere Scrum auch in großen und traditionellen Firmen rasant, nachdem sie bis etwa 2005 auf sehr große Skepsis stießen. Und auch innerhalb der etablierten Vorgehensweisen – wie das V-Modell oder (in der Schweiz) HERMES – wird versucht, sie in Richtung Agilität zu erweitern. Diese Popularität hat aber auch ihre Schattenseiten. Sehr oft ist es wie beim „alten Wein in neuen Schläuchen“: Hinter der „Fassade der Agilität“ wird versucht, die gewohnten Verantwortlichkeiten und Usancen aufrechtzuerhalten. Zum Teil wird auch die Auffassung vertreten, dass Agilität oder Scrum flächendeckend für alle Arten von Projekten, auch für Betrieb/Support und sogar als allgemeine Managementmethode zu nutzen sei. Das widerspricht aber dem Prinzip des professionellen Projektmanagements, wonach die für die jeweilige Art des Projekts angemessene Vorgehensweise zu wählen ist und innerhalb großer

Projekte durchaus auch unterschiedliche Vorgehensweisen opportun sein können. In der modernen Architektur kennen wir ja auch das Prinzip „form follows function“ im Gegensatz zu Schaugiebeln und Blendfenstern.

Eine differenzierte Anwendungspraxis agilen Vorgehens wird auch vom Project Management Institute (PMI) vertreten. Im Dokument „PMI Agile Certified Practitioner – FAQ“ [4] ist zu lesen:

- „Agile principles and practices are suited to projects which require a nimble response to change and continual communication to customers.
- Waterfall principles and practices are suited to projects where little to no change in requirements is expected, and where requirements are clear and well-understood by all team members.
- Agile principles and practices can be adapted to suit organizations and industries that follow waterfall principles and practices. Agile and Waterfall principles and practices are not mutually exclusive – some organizations apply elements of both principles and practices. However, it is important to know that not all organizations lend themselves to Agile principles and practices.“

Der Begriff „Wasserfall“ beruht übrigens auf einem unausrottbaren Missverständnis eines Artikels von Winston W. Royce aus dem Jahr 1970, vgl. [5]. Das dort von ihm in „Figure 2. Implementation steps to develop a large computer program for delivery to a customer“ gezeigte „Wasserfallvorgehen“ ist nur der Ausgangspunkt seiner Überlegungen. Aus diesem simplen Vorgehen heraus entwickelt er ein hoch iteratives Vorgehensmodell und fasst das so zusammen: „In my experience, however, the simpler method has never worked on large software development efforts and the costs to recover far exceeded those required to finance the five-step process listed.” Dennoch ist dieses „Wasserfallmodell“ auch heute, nach über 40 Jahren, immer noch in unseren Köpfen.

Eine noch differenziertere Positionierung des agilen Vorgehens formuliert Robert K. Wysocki [6]:

Ziel klar, Lösung klar:	traditionelles Projektmanagement
Ziel klar, Lösung unklar:	agiles Projektmanagement
Ziel unklar, Lösung unklar:	„extremes“ Projektmanagement
Ziel unkar, Lösung klar:	„verkehrtes“ extremes Projektmanagement

Scrum wird auch von Wysocki als eine der agilen Vorgehensweisen genannt. Unter „extrem“ versteht er ein im Vergleich dazu noch schneller getaktetes Vorgehen mit noch viel höherer Änderungsbereitschaft und Unsicherheit ohne Garantie, dass das Projekt erfolgreich ist. Diese Situation ist typisch z. B. für Forschung und Entwicklung. „Verkehrtes“ extremes Projektmanagement liegt dann vor, wenn z. B. für eine technologische Innovation Anwendungschancen zu finden sind.

Scrum – ein Überblick

Scrum ist keine SW-Entwicklungsmethode wie z. B. der Rational Unified Process (RUP), sondern ein sehr offenes und auf „das Minimum reduziertes“ Vorgehens-Framework der *Produkt*entwicklung – unabhängig davon, ob es ein SW-Produkt ist. Scrum kann somit nicht „pfannenfertig“ angewendet werden, sondern muss je nach Art des Produkts und Projekts mit den dazu nötigen spezifischen Methoden und Techniken – z. B. zur Ableitung und Formulierung von funktionalen Anforderungen, zur Schätzung des Aufwandes oder der Techniken der Testautomatisierung oder der kontinuierlichen Integration – ergänzt werden. Sehr verbreitet beim Einsatz von Scrum für die SW-Entwicklung ist die Kombination mit Methoden und Techniken des Extreme Programmings. Es können aber auch etliche mit den Intentionen von Scrum und der Agilität zu vereinbarende Methoden, Techniken und Templates – z. B. aus RUP und UML – im Rahmen von Scrum genutzt werden. Und zwar überall dort, wo Planungen und Entwürfe sinnvoll machbar sind: *„Plane das Planbare – höre jedoch auf, das Unplanbare ebenfalls planen zu wollen.“* Das ist pragmatisch verstandene Agilität.

Jeff Sutherland schuf den Scrum Process 1993. Mit dem Wort „Scrum“ bezog er sich auf eine 1986 im Harvard Business Review publizierte Studie von Takeuchi und Nonaka, in der „high-performing, cross-functional teams“ mit der Scrum-Formation im Rugby verglichen wurden. Ken Schwaber fomalisierte den Prozess für die Softwareindustrie in einem Artikel für die OOPSLA 1995.

Scrum ist bereits in einer Vielzahl von Büchern und Artikeln beschrieben worden. Stellvertretend für alle empfehle ich den „Scrum Guide“ von Ken Schwaber [7]. Dort ist Scrum kurz und bündig so dargestellt, wie es von ihm als einem seiner Väter gedacht ist. Alles, was an weiteren „Scrum-Regeln“ an diversen Orten zu lesen ist, sind zusätzliche „good practices“ diverser Anwender und Trainer, die durchaus auch widersprüchlich sein können. Dazu gehört auch mein hier vorliegender Artikel. Eine Darstellung von Scrum auf nur einer Seite findet sich in http://www.scrumalliance.org/learn_about_scrum.

Scrum kennt drei Rollen:

- Der *Product Owner (PO)* ist verantwortlich für den „Business Value“ des Projekts. Er legt die Anforderungen an das Produkt (das „Was“) fest, priorisiert die Teillieferungen und nimmt die realisierten Teillieferungen ab. Er ist weder Teamleiter noch Vorgesetzter der Teammitglieder.
- Das Entwickler*team* erledigt die Arbeit, organisiert sich dabei innerhalb des gesetzten Rahmens für das „Was“ und das „Wie“ selbst und ist „cross-functional“ zusammengesetzt, sodass es seine Arbeit weitestgehend autonom erledigen kann. Es kann dabei – entgegen einer landläufigen Scrum-Interpretation – auch aus sich einander ergänzenden Spezialisten bestehen.
- Der *ScrumMaster (SM)* unterstützt den PO methodisch, stellt sicher, dass das Entwicklerteam arbeitsfähig und produktiv ist und sorgt für den dazu nötigen Rahmen, also das „Wie“ der Ergebniserbringung. Er treibt zusätzlich auch die Verbreitung von Scrum im Unternehmen voran. Er ist „dienende Führungskraft“ des Entwicklerteams als Ganzes, aber weder Teamleiter noch Vorgesetzter der Teammitglieder.

PO, Team und SM zusammen bilden das *„Scrum Team“.* Die Rolle des klassischen Projektleiters ist auf diese drei Rollen aufgeteilt. Weitere Produkt- und Projekt-Stakeholder (Nutzervertreter, Architekten, Produktmanager, ...) können situativ beigezogen werden.

Scrum kennt fünf „Ereignisse“. Alle sind als streng einzuhaltende „Zeitfenster“ zu verstehen:

- Der *Sprint* ist der zentrale Pulsschlag von Scrum. Er ist ein während der Produktentwicklung stets gleich langes Zeitfenster von einer bis vier Wochen, währenddessen ein nutzbares und potenziell auslieferbares Produktinkrement fertiggestellt wird. Alle Sprints folgen unmittelbar aufeinander.
- Beim *Sprint Planning* zu Beginn des Sprints trifft sich das Entwicklerteam mit dem PO, formuliert das „Sprint-Ziel“ – und verpflichtet sich darauf – und macht eine „Vorhersage“, welche nutzbaren Ergebnisse es als Team gemeinsam am Ende des Sprints hergestellt haben wird, um das Sprintziel zu erreichen. Die Ergebnisse wählt das Entwicklerteam aus einer nach der Bearbeitungsreihenfolge geordneten Liste aller nötigen Ergebnisse („Product Backlog“) aus, die der PO verwaltet.
- Beim *Daily Scrum* trifft sich das Entwicklerteam für nur 15 Minuten täglich stets zur selben Uhrzeit, um sich gegenseitig über die laufenden Arbeiten, das Gelernte und Erkannte und über zusätzlich nötige oder zu ändernde Arbeitsvoraussetzungen zu informieren.
- Beim *Sprint Review* am Ende des Sprints demonstriert das Entwicklerteam dem PO und anderen Stakeholdern und Nutzern die im Sprint erarbeiteten und ab jetzt (potenziell) konkret nutzbaren Ergebnisse. Der PO nimmt sie ab – oder gibt sie zur Überarbeitung zurück.
- In der *Sprint Retrospective* nach dem Review reflektieren das Entwicklerteam, der PO und der SM die Qualität ihrer Ergebnisse und ihre Vorgehensweise, um Erfolgsmuster zu stärken und ggf. Verbesserungsmaßnahmen zu ergreifen.

Scrum kennt drei Artefakte:

- Das *Product Backlog* ist eine möglichst vollständige und nach der Bearbeitungsreihenfolge geordnete Liste aller gewünschten Projektergebnisse/Features und ist Eigentum des PO. Inhalte und Reihenfolge des Product Backlog werden vom PO (und nur von ihm) fortlaufend aktualisiert.
- Das *Sprint Backlog* ist Eigentum des Entwicklerteams und besteht aus jenen Projektergebnissen/Features des Product Backlogs, die das Entwicklerteam im Sprint erledigen will, unterteilt in die vom Entwicklerteam festgelegten Teilaufgaben. Für die Dauer des Sprints kann das Entwicklerteam das Sprint-Backlog nur innerhalb des vereinbarten „Sprintziels“ verändern.
- Das *Inkrement* ist die Summe aller im Sprint und in allen vorherigen Sprints fertiggestellten (d. h. nutzbaren und potenziell auslieferbaren) Ergebnisse auf Basis des Product Backlogs.

Und so spielt alles zusammen (vgl. Abbildung 4):

- Der „Product Owner“ erstellt zu Projektbeginn eine erste Version einer geordneten Liste der gewünschten Ergebnisse („Product Backlog“) des gesamten Projekts, soweit sie zu diesem Zeitpunkt bereits einigermaßen bekannt sind. Das Product Backlog kann im Verlauf des Projekts jederzeit geändert und ergänzt werden.
- Das Projekt besteht aus einer Abfolge stets gleich langer „Sprints“ (meistens zwei Wochen).
- Das Entwicklerteam entnimmt zu Beginn eines Sprints vom Product Backlog von „oben“ jene Ergebnisse, die es im kommenden Sprint realisieren will, um das „Sprintziel“ zu erreichen („Sprint Backlog“).
- Das Entwicklerteam als Ganzes vereinbart, wie die Ergebnisse (unter Beachtung der im Unternehmen oder vom Kunden festgelegten technischen Rahmenbedingungen) zu realisieren sind.

- Während des Sprints organisiert sich das Entwicklerteam selbst: Es definiert die Teilaufgaben, teilt sie untereinander auf und sorgt für die notwendige Detailplanung, Kommunikation und Kooperation. Das Entwicklerteam trifft sich täglich zu einem 15-minütigen Daily Scrum. Wann immer möglich, sitzen alle Teammitglieder im selben Raum und arbeiten gemeinsam nur an diesem einen Projekt / Produkt.
- Auf Prozessebene wird das Entwicklerteam und der PO vom „ScrumMaster“ unterstützt. Er stellt auch die notwendige Arbeitsumgebung sicher.
- Die Ergebnisse („Product Increment“) sind sofort nach Sprintende – potenziell – nutzbar.

Der Ablauf (vgl. Abbildung 4) zeigt, dass das Entwicklerteam im ersten Sprint das Ergebnis B wegen seines Umfangs nicht auswählte, sondern sich zusammen mit dem PO für A, C und D entschied – was auch als Inkrement geliefert werden konnte. Im zweiten Sprint plante es dann B und E, konnte aber nur B fertigstellen. Das Inkrement umfasst zu diesem Zeitpunkt A B C D. Zu Beginn des dritten Sprints hatte der PO aufgrund neuester Erkenntnisse H höher geordnet und zusätzlich noch R eingefügt. Das Team plante danach H, F und R und konnte diese im dritten Sprint auch fertigstellen. Das Inkrement nach dem dritten Sprint umfasst A B C D F H R.

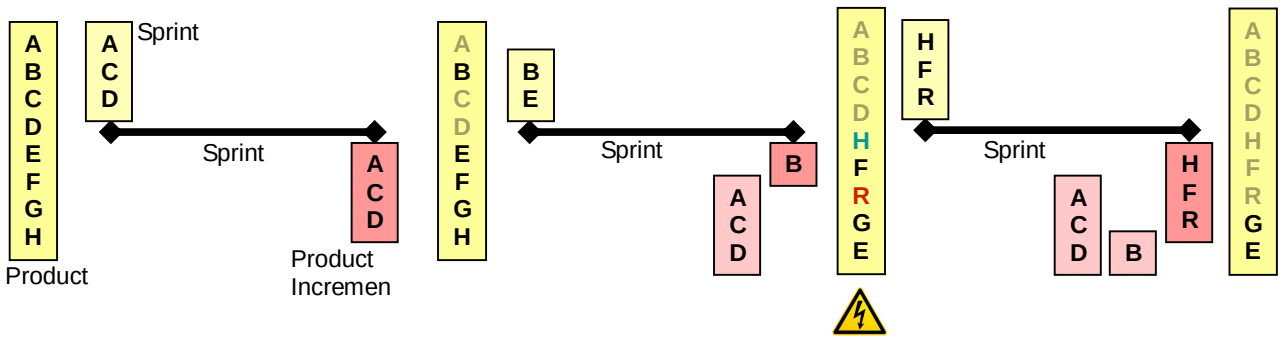

Abb. 4: *Scrum-Ablauf*

All das kann den Eindruck erwecken, dass diese drei Rollen, fünf Zeremonien und drei Artfakte genügen, um ein Projekt abzuwickeln. Das wäre ein fatales Missverständnis. Wie eingangs gesagt: Scrum ist keine SW-Entwicklungsmethode, sondern ein sehr offenes und auf „das Minimum reduziertes“ Vorgehens-Framework der Produktentwicklung und muss je nach Kontext und Art des Produkts spezifisch ergänzt werden.
So kann etwa der Product Owner die initiale Version des Product Backlog nicht allein und in ein paar Tagen „aus dem Ärmel schütteln“ und das Projekt kann nicht unmittelbar mit dem ersten Sprint starten. Das Product Backlog ist das Ergebnis einer angemessen langen „ramp-up“-Phase unter Einbezug von Business-Analysten in enger Zusammenarbeit mit dem Product Management mit Einsatz etlicher der üblichen Methoden und Techniken des Requirement Engineerings. Zusätzlich zum Product Backlog gibt es auch weiterhin Konzeptionen und Designs – etwa für die betriebliche und technische Architektur –, soweit sie zu Beginn bereits festegelegt werden müssen und können (nicht aber detaillierter!). Erst dann kann der erste Sprint sinnvoll starten. Und im Team gibt es nicht nur Generalisten, sondern durchaus auch GUI-Spezialisten oder Test-Engineers – jedoch innerhalb des Teams und nicht in einer Spezialabteilung, und zwar so, dass diese Spezialisten fähig und bereit sind, auch etwas anderes als nur ihre Spezialität zu leisten.

Es würde den Rahmen dieses Beitrags sprengen, detaillierter auf die Rollen, Artefakte und Zeremonien von Scrum einzugehen. Eine kompakte Darstellung finden Sie im „Scrum Guide“ von Ken Schwaber unter http://www.scrum.org/scrumguides. Dieser „Scrum Guide“ ist jedoch, wie

gesagt, ein relativ breiter Rahmen den es je nach Unternehmen und Projekt spezifisch zu ergänzen gilt. Einige Anregungen dazu finden Sie in www.korn.ch/scrum-ohne-murcs

Scrum „out of the book“ oder „Scrum-artig“ – oder etwas ganz anderes statt Scrum?

Ich unterstütze die Aussage unter „Fazit“ in [7], dass Scrum, genau so wie im „Scrum Guide“ beschrieben, dann anzuwenden ist, wenn gesagt wird: „Wir machen Scrum.“ Alles andere ist kein „Scrum“ – kann aber im jeweiligen Kontext besser passen als das „reine“ Scrum. Dann sollte es jedoch „Scrum-artig“ genannt werden. „Fußball“ ist schließlich auch kein Fußball, wenn zwei Mannschaften mit je fünf Leuten auf einem 30 mal 50 Meter großen Feld und ohne Abseitsregeln spielen. Dann ist es „Tschutten“, wie man in der Schweiz sagt.

Nicht zu empfehlen ist, statt zunächst das „reine“ Scrum auszuprobieren, sofort „Scrum-artig“ vorzugehen. Das, was im „Scrum Guide“ beschrieben ist, ist sehr offen für projektspezifische Anpassungen. Scrum ist ein „Framework“ und keine vollständige Produkt- oder SW-Entwicklungsmethode. Es lohnt sich also, dieses Framework zunächst einmal unverändert anzuwenden, Erfahrungen zu sammeln und erst dann davon „Scrum-artig“ abzuweichen. Auch in diesem Fall sollten aber die essenziellen Intentionen unangetastet bleiben. Ich empfehle also, bei jeder Abweichung diese Fragen zu stellen:

- Was ist die im „reinen“ Scrum beabsichtigte Wirkung dieser Rolle/dieses Ereignisses/dieses Artefakts?
- Wie stark verändert die beabsichtigte Abweichung diese Wirkung?
- Will und kann (oder muss) ich tatsächlich diese veränderte Wirkung in Kauf nehmen?

Diese drei Pfeiler von Scrum sollten dabei nie geschwächt werden:

- *Transparenz*: Die zu erbringenden Ergebnisse (Product Backlog und Sprint Backlog) und der tagesaktuelle Erledigungsstand im Sprint sind für den PO, alle Teammitglieder, den SM und alle Stakeholder jederzeit sichtbar.
- *Inspektion*: Es gibt in kurzen Abständen regelmäßige Überprüfungen des Projektfortschritts anhand nutzbarer Ergebnisteile und der Vorgehensweise durch das Scrum-Team.
- *Adaption*: Als Ergebnis jeder Inspektion gibt es Maßnahmen zur Festigung und Stärkung der Erfolgsmuster und zur Verbesserung der Schwachstellen.

Einige Abweichung sind – aus meiner Sicht – eher „harmlos“, andere wiederum haben gravierende Folgen. So etwa ist ein „Daily Scrum“ nur jeweils am Montag, Mittwoch und Freitag eine deutlich harmlosere Abweichung als die Übernahme der Rolle des PO und des SM durch ein und dieselbe Person. Letzteres würde ich nicht einmal mehr „Scrum-artig“ nennen, es ist möglicherweise aber immer noch „agil“.

Im Bereich des „agilen“ Vorgehens stehen nebst Scrum auch etliche andere Vorgehens- und Methodenvarianten wie z. B. KANBAN, Lean SW Development, Extreme Programming, Crystal, Adaptive Project Management oder Feature Driven Development zur Verfügung. Besser ist es, eine davon zu wählen, statt Scrum zu „verbiegen“, bis es passt. So ist etwa KANBAN (vgl. [8]) für fortlaufend anfallende Support- und Wartungsaufgaben angemessener als Scrum oder ein anderes auf Projekte fokussiertes Vorgehen.

Wie kann Scrum eingeführt werden?

Bitte blättern Sie zurück zum Kapitel „Sie überlegen sich, in Ihrem Unternehmen oder in Ihrer Abteilung „Scrum“ einzuführen?“ und bedienen Sie nochmals die „Schieberegler“. Nach all dem, was Sie bis jetzt gelesen haben: Möchten Sie die Projekte in Ihrem Verantwortungsbereich weiterhin oder sogar noch überzeugter „agil“ oder auf Basis von Scrum abwickeln?
Wenn ja, dann lesen Sie bitte weiter!

Revolutionären „Big Bang“ vermeiden

Die Einführung eines stark veränderten Vorgehens bei der Produkt- bzw. SW-Entwicklung, verbunden mit völlig neuen Rollen anstelle der allseits bekannten etwa wie der Rolle des Projektleiters, ist ein großes und kritisches organisatorisches Change-Vorhaben. Wie diese Veränderung gestaltet wird, hängt vom Charakter des Unternehmens ab. Die Gestaltung der Veränderung in Richtung „Scrum“ ist umso anspruchsvoller, je mehr „Kultur-Schieberegler“ in der Mitte oder gar eher links stehen.

Simone Inversini zeigte in ihrer Dissertation (siehe [9]) auf, dass es nicht nur „den einen“ Ansatz zur Veränderungsgestaltung gibt, der den Erfolg in wirtschaftlicher, organisationsbezogener und qualifikatorischer Hinsicht garantiert, sondern dass u. a. diese erfolgsrelevanten Zusammenhänge erkennbar sind: [10]

- Traditionell-bürokratische Organisationsstrukturen brauchen eher ein pragmatisches, graduell-anpassendes und langfristiges Vorgehen.
- Flexible Organisationsstrukturen brauchen eher ein konzeptuelles, tief greifendes, temporäres Vorgehen.
- Autonomieorientierte und veränderungskompetente Organisationen brauchen ein eher selbstbeurteilendes Vorgehen.
- Wirtschaftlicher Handlungsdruck (Wettbewerb, Kundenbedürfnisse) verlangt ein eher wenig fremdbeurteilendes, tief greifendes und konzeptuelles Vorgehen, das langfristig angelegt und rollend geplant wird, bei eher hohem Tempo.
- Politisch-rechtlicher Handlungsdruck (neue Gesetzgebung) verlangt ein Vorgehen, das eher linear geplant ist und eine hohe Zielfokussierung aufweist.
- Bei soziokulturellem Handlungsdruck durch Wertewandel der Mitarbeitenden ist bei gemäßigtem Tempo ein eher graduell-anpassendes Vorgehen sinnvoll.

Auf Basis dieser Zusammenhänge und aus all dem, was ich bisher in Veränderungsprojekten erlebt habe, halte ich eine „revolutionäre“ Scrum-Einführung mittels „Big Bang“ für wenig empfehlenswert. Selbst bei wirtschaftlichem Handlungsdruck („wir stehen mit dem Rücken zur Wand“) erscheint ein zwar schnelles, aber dennoch langfristig angelegtes und rollend geplantes Vorgehen besser als eine „Revolution“.

Natürlich ist nachvollziehbar, dass statt einer längeren Übergangsphase eine möglichst rasche Transition auf breiter Front gewünscht wird. Aber: Wenn Scrum in einem oder zwei Projekten mit – notabene dafür sehr offenen – Teams, POs und SMs erfolgreich war und dann innerhalb nur weniger Wochen ein Dutzend oder mehr Teams eines IT-Unternehmens oder des unternehmensinternen IT-Bereichs Scrum einführen – oder besser: angeleitet von „Scrum-Evangelisten“ einzuführen *haben* –, dann entsteht daraus höchstwahrscheinlich eine „Scrumade“, also Scrum als Fassade: Formal gibt es zwar die Rollen, Artefakte und Ereignisse von Scrum, dahinter aber verbergen sich die gewohnten Praktiken. Wird diese „Scrumade“ über mehrere

Sprints hinweg praktiziert, ist die Einsicht klein, daran etwas verändern zu müssen: „Was sollen wir denn ändern? Seht her: Wir machen doch Scrum!“ Ich habe erstaunlich viele Beispiele dafür kennengelernt – mehr als ich ursprünglich dachte. Und das auch in kleinen und mittleren Unternehmen und ohne „revolutionäre“ Einführung.

Besser: inkrementell vorgehen

Besser sind für mich die folgenden zwei inkrementellen Vorgehensformen. Beide beginnen mit etwa drei (nicht nur einem!) Projekten, in denen Scrum „out of the book“ angewendet und immer wieder reflektiert wird. Unterstützt wird das von erfahrenen Projektmanagement-Coaches/-Beratern mit Spezialisierung auf agile Vorgehensweisen und breiten Erfahrungen auch im traditionellen Projektmanagement. Aktuell bietet das PMI eine entsprechende Zertifizierung (PMI Agile Certified Practitioner) als Pilot an. Der aktuelle Stand ist jeweils unter http://www.pmi.org/Certification/New-PMI-Agile-Certification.aspx zu finden. Trainer und Coaches, die Erfahrungen nur im Bereich Scrum haben und sich selbst gerne als „Scrum-Evangelisten“ bezeichnen, stoßen sehr oft auf große Vorbehalte bei den „mit Scrum zu Beglückenden“.

Wichtiges Ergebnis dieser drei mit Scrum abgewickelten Entwicklungsprojekte ist die Einsicht, inwieweit Scrum im Unternehmen oder Unternehmsteil funktionieren könnte und mit welchen – bereits bewährten – Methoden und Techniken des Business- und SW-Engineerings es kombiniert werden soll. Oder ob statt Scrum in einer weit weniger „revolutionären“ Art nach KANBAN [12] vorgegangen werden soll.

All das wird von einem spätestens jetzt einzurichtenden *„Process Support Office“* gesammelt und koordiniert. Unterstützt wird es dabei von einem *„Process Improvement Board“*, in dem im Fall eines unternehmensinternen IT-Bereichs auch die nutzenden Fachbereiche vertreten sind.
Wenn weiterhin Scrum und nicht KANBAN oder ein anderes agiles Vorgehen angestrebt wird folgen zwei alternative Vorgehensweisen:

Alternative 1: Inkrementelle Einführung quer über alle für Scrum geeigneten Entwicklungsprojekte

1. Inkrement:

Jedem (Projekt)-Team wird ein PO im Sinne von Scrum und ein Teamleiter im bekannten Sinne zugeordnet. Begleitet von einem Scrum-erfahrenen Projektmanagement-Coach/-Berater erstellen die PO ihre Product (bzw. Project) Backlogs. Das ist ab jetzt die alleinige Basis für die Arbeit in den Teams mit ihren Teamleitern. Jedes Projekt plant Releases mindestens alle drei Monate.

2. Inkrement für jene Projekte, die noch mehr in Richtung Scrum arbeiten möchten:

Die Entwicklungsarbeit wird für alle Teams innerhalb der Releases in Sprints (zwei bis allenfalls vier Wochen) unterteilt. Planning, Review und Retrospective sowie Sprint Backlog werden eingeführt. Die Teamleiter verändern ihre Rolle weg vom Erteiler von Detailaufträgen und Detailkoordinator hin zum Rahmensetzer und Unterstützer („Servant Leadership“). Sie geben dem Team möglichst viel Eigenverantwortung und unterstützen die dazu nötige Selbstorganisation.

3. Inkrement für jene Projekte, die vollständig auf Basis von Scrum arbeiten möchten:

Die Teamleiter geben ihre Rolle als Führungspersonen ab. Die Entwickler aller Teams gehören neu zu „Entwicklerpools“ mit je rund 10 bis 25 Personen pro Produktfamilie und jeweils einem personellen Vorgesetzten. Die ehemaligen Teamleiter übernehmen die Rolle der ScrumMaster in einem *anderen* als ihrem bisherigen Team. Von einem Scrum-erfahrenen Projektmanagement-Coach/-Berater werden sie in ihrer neuen Rolle als ScrumMaster begleitet.

Der Vorteil dieses Vorgehens ist, dass jedes Inkrement eine funktionierende Ausprägung des agilen Vorgehens darstellt und jedes Projekt bzw. Team so viele Elemente von Scrum praktizieren kann, wie es für seinen Kontext angemessen ist. Das Risiko besteht darin, dass vielleicht schon nach dem ersten Inkrement die Veränderung dank einer klaren PO-Rolle und eines geordneten Product Backlogs in Verbindung mit quartalsweisen Releases als derart wirksam erlebt wird, dass gar nicht erst versucht wird, noch weiter in Richtung Scrum zu gehen. Dann bleibt insbesondere die Chance ungenutzt, selbstverantwortliche Teams entstehen zu lassen.

Alternative 2: Inkrementelle Einführung für jeweils einige der für Scrum geeigneten Entwicklungsprojekte

Jene Teams werden identifiziert, deren relevanteste „Agilitäts-Schieberegler“ (vgl. Abbildung 1) am weitesten rechts stehen. Zwei bis drei dieser Teams stellen in einem ersten Inkrement vollständig in diesen Schritten auf Scrum um:

- Nominierung der PO für diese Teams/Projekte, Erarbeiten der Product Backlogs, begleitet von einem Scrum-erfahrenen Projektmanagement-Coach/-Berater
- Nominierung der SM für diese Teams, Einrichten der Teamräume (idealerweise je ein großer Raum pro Team)
- Die Entwickler dieser Teams gehören neu zu einem „Entwicklerpool“ mit rund 10 bis 25 Personen pro Produktfamilie und einem personellen Vorgesetzten. Das ist weder einer der PO noch der SM.
- Gemeinsame Scrum-Schulung der Teams, der PO und SM unter Benutzung der realen Product Backlogs
- Pro Team Releaseplanung und erstes Sprint Planning Meeting und dann Start der Sprints

All das wird von einem Scrum-erfahrenen Projektmanagement-Coach/-Berater begleitet.

Danach werden die nächsten zwei bis drei Teams identifiziert, die vollständig auf Scrum in den genannten Schritten umstellen. Und danach die nächsten Teams …

Vorteil dieser zweiten Alternative ist, dass einzelne Teams vollständig auf Scrum umstellen, ohne sich an Situationen zwischen „bisher“ und „Scrum“ zu gewöhnen, die ihnen dann bereits „gut genug“ erscheinen könnten.

UPscaled Scrum für große Unternehmen

Die Mehrzahl der Publikationen zu Scrum behandelt ein „Scrum in der Nussschale“: Ein Entwicklerteam arbeitet nur für ein bestimmtes Entwicklungsprojekt, hat kaum Wartungsarbeiten für Altprodukte und der PO kann als Person allein die funktionalen und nichtfunktionalen Anforderungen aller Stakholdergruppen unter einen Hut bringen. Allenfalls arbeitet höchstens eine

Handvoll weiterer Teams auch für dieses und nur für dieses Projekt. Das entspricht in keiner Weise der Situation in großen Unternehmen.
Der Scrum Guide [7] beschreibt nur den Team-Layer und bietet keinerlei Rahmen für den Programm- und Portfolio-Layer. Scrum muss daher in großen Enterprises im *agilen* Sinn entsprechend erweitert werden, wie es z. B. Dean Leffingwell beschreibt [11].
Geschieht das nicht, gerät Scrum unweigerlich in Konflikt mit den zumindest informell vorhandenen und nicht gleichzeitig agil angepassten Strukturen und Prozessen des Programm- und Portfolio-Layers.

Wenn mit Scrum im IT-Bereich eines Unternehmens gestartet wird, muss gleichzeitig auch das Vorgehen in der IT-übergeordneten Produktentwicklung agil umgestaltet werden. Sonst wird Scrum auf einen vom Produktmanagement strikt vorausgeplanten, schrittweisen und nicht agilen Lieferprozess reduziert und verliert damit seine Berechtigung.

Die sich selbst organisierenden und für ihre Ergebnisse *insgesamt* verantwortlichen Teams (nicht die Teammitglieder als Einzelpersonen tragen die Verantwortung) und die Personalführung der Teammitglieder durch die von der Team-Tagesarbeit weitgehend entkoppelten „Pool-Leiter“ erfordern spezielle und tief greifende Veränderungen der HR-Instrumente. Bei großen Unternehmen mit ihrem Anspruch auf Einheitlichkeit quer über alle Unternehmensteile kann das zu erheblichen Widersprüchen zum „Geist“ von Scrum führen und seine Anwendung massiv behindern.

Top-down oder Bottom-up?

Wie erfolgreich ist es, wenn einzelne Teams mit Unterstützung ihrer unmittelbaren Vorgesetzten beschließen, auf Scrum umzustellen? Und von „außen“ und „weiter oben“ betrachtet alles weiterhin so aussehen lassen wie bisher? Nur bedingt erfolgreich. Denn Scrum ist ja keine SW-Entwicklungsmethode, sondern ein Framework für die Produktentwicklung. Es erfordert also, Sprint für Sprint, den intensiven Kontakt mit den Nutzern der Ergebnisse. Wenn das „versteckt“ wird, bleibt der Hauptnutzen von Scrum auf der Strecke.

Wie erfolgreich ist es, Scrum von oben zu „verordnen“? Nicht sehr erfolgreich. Simone Inversini schreibt dazu:
Es zeigt sich, „dass in den erfolgreicheren Projekten durchwegs sämtliche Mitarbeitende einbezogen wurden. Die Form des Einbezugs reichte von der Basisbeteiligung bis hin zur Konsultation. Es scheint jedoch weniger die Art der Partizipation eine Rolle zu spielen als deren klar kommunizierte Festlegung.“ [9]

Essenziell ist die Unterstützung durch die Entscheidungsträger – bis hinauf zum Top Management auf Geschäftsleitungsebene. Simone Inversini schreibt dazu:
„Es erweist sich als von zentraler Bedeutung, die Entscheidungsträgerinnen und -träger der Organisation verbindlich in die Veränderung einzubinden. Diese Ergebnisse widersprechen klar der von frühen Vertreterinnen und Vertretern der OE (Organisationsentwicklung) aus der gruppendynamischen Tradition geübten Kritik an der Hierarchie an sich, mit welcher rein partizipative Ansätze legitimiert wurden. Die Ergebnisse veranschaulichen, dass eine Veränderung mit mangelhaft agierenden bzw. ohne Machtpromotorinnen und -promotoren wenig Chancen hat.“ [9]

Ohne klare Unterstützung – auch bei gelegentlichen Rückschlägen und Interessenkonflikten – durch das für die IT verantwortliche Geschäftsleitungsmitglied haben Agilität und Scrum keine längerfristige Überlebenschance.

Wenn Scrum – dann alles mit Scrum?

Professionelles Projektmanagement ist bestrebt, für jede Projektsituation die angemessene Vorgehensweise einzusetzen. Das können sogar innerhalb eines Projekts unterschiedliche Formen sein. Wann Scrum bzw. das agile Vorgehen passen, habe ich im Abschnitt „Agilität ist nicht immer angemessen“ beschrieben. Wenn in IT-Firmen oder IT-Abteilungen das Projektvorgehen für alle Arten von Projekten einheitlich zu sein hat, dann werden immer einzelne Projekte suboptimal abgewickelt werden, egal ob Scrum oder ein „traditionelles“ Vorgehen vorgeschrieben ist.

Es gibt auch andere Wege. So etwa existiert in der Informatik der Credit Suisse ein Verfahren zur Beurteilung der für ein Projekt geeigneten Vorgehensweise von „agil“ bis „traditionell“ (vgl. Abbildung 5). In die Beurteilung fließen die rechts in der Abbildung genannten Kriterien ein. Das Evaluationsverfahren führt durch eine Reihe von Fragen. Die Antworten werden ausgewertet und das Resultat ist eine Aussage, wie „agil“ oder „traditionell“ die zu diesem Projekt passende SW-Entwicklungsmethodik sein sollte. Zusätzlich gibt es bei der Credit Suisse noch ein „Agile Framework“, eine Kombination der besten Praktiken aus den agilen Methoden zur SW-Entwicklung. Damit kann der für ein Projekt am besten geeignete Satz von Methoden bestimmt werden.

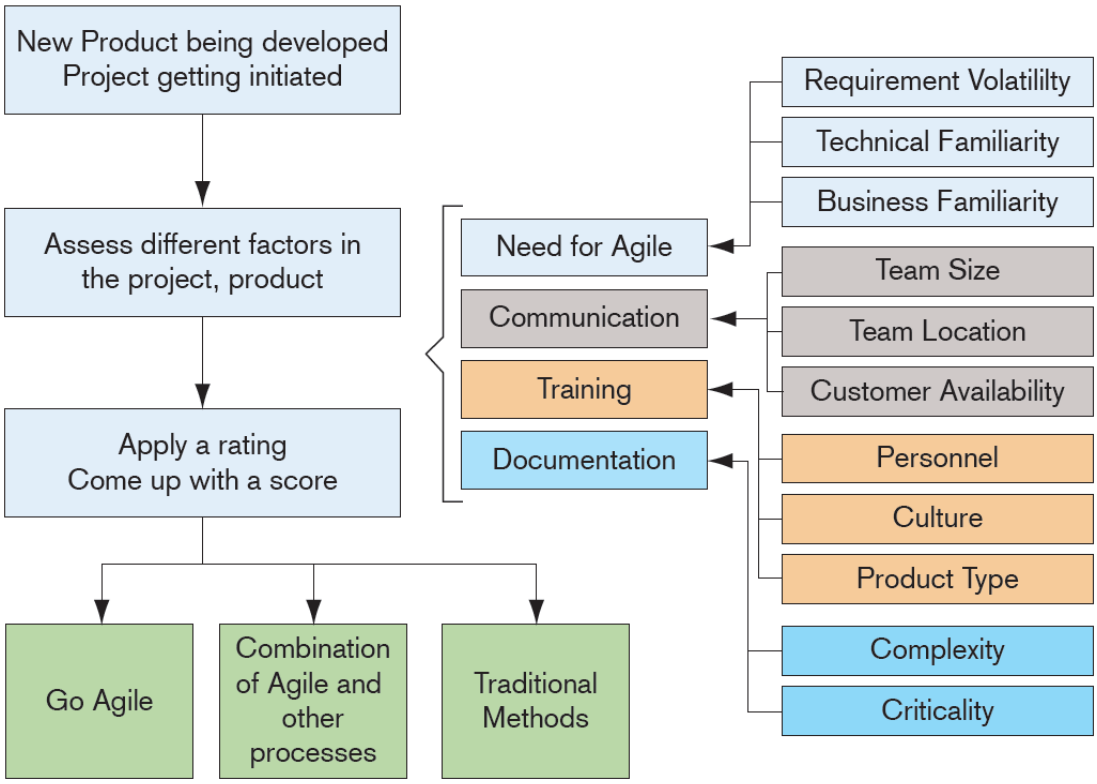

Abb. 5: *Projekt-Assessment der Credit Suisse [28]*

Zusammenfassung

In der SW-Entwicklung erlebt das „agile Vorgehen“ und insbesondere Scrum als das derzeit wohl bekannteste Vorgehensframework der Produktentwicklung ein steigendes Interesse auch in mittelgroßen und großen IT-Unternehmen und unternehmensinternen IT-Bereichen. Scrum sieht, auf wenigen Seiten dargestellt, sehr „simpel“ aus – ist in seiner konkreten Umsetzung jedoch alles andere als einfach. Und endet leicht bei einem Scrum als Fassade, einer „Scrumade“. Damit wird nichts oder gar das Gegenteil der erhofften Verbesserungen erreicht, also eher ein von hinten nach vorne zu lesendes Scrum, ein „murcS“.

In diesem Artikel habe ich den Schwerpunkt der Betrachtung darauf gelegt, unter welchen Voraussetzungen agile Vorgehensweisen und insbesondere Scrum angemessen sind und wie sie statt mit einem „Big Bang“ in mittleren und größeren IT-Organisationen schrittweise eingeführt werden können. Die Rollen, Ereignisse und Artefakte von Scrum sowie die spezifischen Aspekte des Multiteam- und Multiprojekt-Managements werden in meinem umfassenden Artikel unter www.korn.ch/scrum-ohne-murcs vertieft behandelt.

Literatur- und Quellenverzechnis

[1] Kurtz, Cynthia; Snowden, Dave: The New Dynamics of Strategy: Sense-making in a Complex-Complicated World. IBM Systems Journal. Vol. 42, no. 3, 2003, S. 462–83

[2] Quelle: http://de.wikipedia.org/wiki/Cynefin-Framework

[3] Quelle: http://agilemanifesto.org/iso/de/

[4] Quelle : http://www.pmi.org

[5] Royce, Winston W.: Managing the Development of Large Software Systems, IEEE, Wescon, August 1970, S. 1–9 und in http://de.wikipedia.org/wiki/Wasserfallmodell

[6] Wysocki, Robert K.: Effective Project Management: Traditional, Agile, Extreme. New York: John Wiley & Sons, 2009

[7] Schwaber, Ken: Scrum Guide, http://www.scrum.org/scrumguides/

[8] Kniberg, Henrik; Skarin, Mattias: Kanban and Scrum – making the most of both, http://www.infoq.com/minibooks/kanban-scrum-minibook (auch in Deutsch verfügbar)

[9] Inversini, Simone: Wirkungsvolles Change Management in Abhängigkeit von situativen Anforderungen, Humanwissenschaftlichen Fakultät der Universität Potsdam, 2005, http://opus.kobv.de/ubp/volltexte/2005/549/pdf/inversini.pdf; Kurzfassung: http://www.iafob.ch/deu/10j/downloads/Teil_3_6.pdf

[10] Quelle: Inversini, Simone: Referat an der Jahrestagung des OE-Forums Schweiz, 13.01.2006

[11] Leffingwell, Dean: Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development). Upper Saddle River/München: Addison Wesley, 2010

[12] Anderson, David J., übersetzt von Arne Rook und Hening Wolf: Kanban, Paderborn, Media Print dpunkt.verlag, 2011