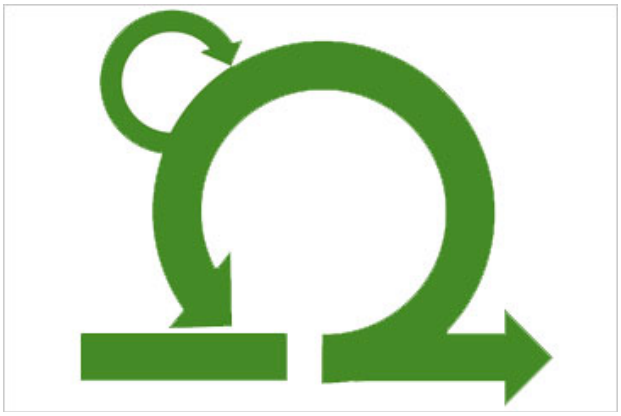
ProjektMagazin
Das Fachportal für Projektmanagement

Spotlight

Wie entwickle ich Software mit Scrum?



Eine themenspezifische Zusammenstellung von Fachartikeln
aus dem Projekt Magazin

www.projektmagazin.de
Machtlfinger Str. 21, 81379 München
Tel: +49 89 2420798-0
Fax: +49 89 2420798-8

Wie entwickle ich Software mit Scrum?

Scrum ist mittlerweile die bekannteste agile Methode zur Software-Entwicklung. In diesem Spotlight lernen Sie zum einen die Grundlagen von Scrum kennen, zum anderen, wie Sie Scrum im Unternehmen einführen und in einem traditionellen Umfeld erfolgreich einsetzen. Wenn Sie gerade erste Erfahrungen mit Scrum gesammelt haben, werden Sie aus den Lessons Learned anderer Scrum-Anwender Nutzen ziehen. Scrum-Profis zeigt dieses Spotlight, wie sie Scrum mit Kanban oder Critical-Chain-Projektmanagement kombinieren können.

Inhalt

So funktioniert Scrum

1. Agiles Projektmanagement. Scrum – eine Einführung ..... Seite 4
2. Wie führe ich als ScrumMaster? Führungsstile für die Praxis ..... Seite 13
3. Scrum-Projekte planen mit System
   Teil 1: Besonderheiten und Voraussetzungen ..... Seite 21
4. Scrum-Projekte planen mit System
   Teil 2: Vorgehen in der Praxis ..... Seite 28

Scrum und traditionelle Unternehmenskultur

5. Scrum im Unternehmen einführen
   Teil 1: Einführung "von oben" ..... Seite 38
6. Scrum im Unternehmen einführen
   Teil 2: Einführung "von unten" ..... Seite 52
7. Risiken der Agilen Software-Entwicklung reduzieren
   Ein bisschen Wasserfall muss sein ..... Seite 64
8. Agile Methoden im traditionellen Projektmanagement-Umfeld einsetzen ..... Seite 78
9. Hybrides Vorgehensmodell
   Agile und klassische Methoden im Projekt passend kombinieren ..... Seite 88

Scrum im Praxiseinsatz

10. Agiles Projektmanagement
    Häufige Stolperfallen in Scrum ..... Seite 99
11. Agile Softwareentwicklung mit Scrum und User Stories ..... Seite 112
12. Anforderungsmanagement in IT-Projekten
    So vermeiden Sie Stolpersteine bei User Stories ..... Seite 121
13. Agile Aufwandsschätzung in Scrum
    Planning Poker – Techniken, Erfahrungen und Empfehlungen ..... Seite 134

© 2011 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Scrum für Fortgeschrittene

14. Prozesse verschlanken, Leistung steigern
Scrum und Kanban sinnvoll kombinieren ..... Seite 144

15. Das Beste aus zwei Welten kombinieren
Scrum + Critical Chain = Reliable Scrum ..... Seite 155

16. Agiles Projektmanagement
Schneller geht's nicht – Ultimate Scrum ..... Seite 170

Anhang: Arbeitshilfen

"Reliable Scrum Portfolio.xlsm", Excel-Tabellenblatt ..... Artikel S.155

"Reliable Scrum WIP and Execution Control.xlsm", Excel-Tabellenblatt ..... Artikel S.155

© 2011 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Agiles Projektmanagement

Scrum – eine Einführung

Eine Gruppe von Investoren will ein neues Internet-Portal für die Stellenvermittlung im High Professional-Umfeld entwickeln. Die Investoren analysieren die existierenden Konkurrenzprodukte und erstellen ein Grobkonzept, das ihre Vision eines web-basierten Job-Portals beschreibt. Dieses Grobkonzept wird an ein Softwarehaus gegeben, das ein Angebot erstellt, bestehend aus Anforderungsanalyse, Spezifikation, Entwurf, Programmierung, Test und Integration. Diese sechs Phasen sollen sequentiell durchlaufen werden; der Plan sieht vor, das Portal in acht Monaten zu veröffentlichen.

Das Softwarehaus erhält den Zuschlag und die Arbeit beginnt. Besonderen Wert wird dabei auf die Ausarbeitung der schriftlichen Spezifikation gelegt, die von sämtlichen Stakeholdern in mehreren Durchläufen gegengelesen und überarbeitet wird. Man will in dieser Phase nichts falsch machen und möglichst jede potenzielle Anforderung von vornherein berücksichtigen. Jeder weiß, dass Änderungen teuer sind, so dass es besser ist, von Anfang an alles richtig zu machen. Die Investoren sind von dem Plan und der Erfahrung des beauftragten Softwarehauses überzeugt und gehen davon aus, dass das geplante Job-Portal in spätestens acht Monaten online sein wird.

Autor

Ralf Wirdemann

Zertifizierter Scrum Master und zertifizierter Scrum Practitioner, Coach für agiles Projektmanagement

Kontakt: business@ralfwirdemann.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in der Rubrik:

› Agiles Projektmanagement

Schwierige Planung

Kommt Ihnen das beschriebene Szenario bekannt vor? Wenn alles gut läuft, und die Software wie geplant entwickelt und geliefert werden kann, dann bekommen die Investoren im besten Fall das, was zu Beginn des Projekts schriftlich spezifiziert wurde. Aus Erfahrung ist jedoch bekannt, dass dies nur selten das ist, was sich der Kunde wirklich vorgestellt hat. Zum einen ist es nur schwer möglich, Software so genau zu spezifizieren, dass die Spezifikation genau das Bild der Software beim Programmierer erzeugt, das der Produktmanager vor Augen hat. Und zum anderen ändern sich die Rahmenbedingungen während der Laufzeit eines Softwareprojekts, was zwangsläufig zu Anpassungen und neuen Anforderungen führt.

Beispielsweise sind während der acht Monate Entwicklungszeit zwei weitere Konkurrenten in den Markt eingetreten. Einer der beiden besitzt ein "Killer-Feature", das die Investoren während ihrer Planungsphase noch nicht kannten, jetzt aber unbedingt auch für ihr eigenes Produkt wollen. Es ist bereits klar: Ohne die-ses Feature kann man am Markt nicht bestehen. Änderungswünsche werden mit dem beauftragten Soft-warehaus zäh verhandelt und resultieren in zusätzlichen Kosten und Terminverschiebungen.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Außerdem läuft die Test- und Integrationsphase längst nicht so glatt, wie geplant. Je näher das Projektende rückt, desto größer wird die Anzahl der Probleme. Projektleitung und Entwicklungsteam geraten unter Druck, Mehrarbeit wird zur Regel, Motivation und Qualität sinken. Irgendwie kriegen die Entwickler es mit einer zweimonatigen Verspätung aber doch noch hin. Allerdings entspricht das Ergebnis nicht wirklich dem, was sich die Investoren vorgestellt haben. Die entwickelten Features sind längst nicht mehr auf der Höhe der Zeit und teilweise schlechter umgesetzt, als in den Produkten der Konkurrenz. Zudem kommt das Portal zu spät auf den Markt, so dass ein Großteil der Zielnutzer bereits an eines der Konkurrenzportale gebunden ist.

Agile Softwareentwicklung als Gegenentwurf

Das beschriebene Szenario skizziert das klassische "Wasserfallvorgehen" bei der Durchführung von Softwareprojekten: Zu Beginn des Projekts wird ein genauer Plan erstellt, dessen Einhaltung höhere Priorität genießt, als die Entwicklung von Software im Sinne des Kunden. Die typischen Probleme dieses Vorgehens sind spätes Feedback aufgrund seltener und zu später Releases, mangelnde Offenheit für Änderun-gen sowie die Entwicklung zu vieler Features, die der Markt nicht braucht. Als Gegenentwurf zum Wasser-fallmodell hat sich in den letzten Jahren die agile Softwareentwicklung etabliert. Der zurzeit wohl populärste Vertreter agiler Entwicklungsprozesse ist Scrum.

Dieser Beitrag richtet sich an Projektmanager mit Erfahrung in der klassischen Softwareentwicklung. Der Artikel stellt das agile Projektmanagement als Alternative zur klassischen Softwareentwicklung am Beispiel von Scrum vor und ermöglicht Ihnen zu entscheiden, ob sich der agile Managementansatz für Ihre jeweilige Projektsituation eignet.

Was ist Scrum?

Scrum ist ein Framework für das Management komplexer Softwareprojekte und entstammt der Familie agiler Softwareentwicklungsprozesse. Agile Softwareentwicklung versucht den Softwareentwicklungspro-zess flexibel und schlank zu gestalten und begegnet damit den eingangs beschriebenen Problemen. Ein wesentliches und zentrales Merkmal der agilen Softwareentwicklung ist Feedback. Kurze Feedback-Schleifen, stellen sicher, dass der Auftraggeber das bekommt, was er haben will. Änderungen sind keine unerwünschten Hindernisse mehr, sondern selbstverständlicher Teil des Prozesses.

Der Begriff "Scrum" tauchte 1986 zum ersten Mal auf und wurde von Hirotaka Takeuchi und Ikujiro Nonaka als Methode zur Produktentwicklung beschrieben. Fast zehn Jahre später griffen Ken Schwaber und Jeff Sutherland das Thema auf und entwickelten gemeinsam Scrum zu einer Methode für das Management agiler Softwareprojekte weiter. Mittlerweile zählt Scrum zu den etabliertesten agilen Managementmethoden, für die sich Projektleiter, Produktmanager und Softwareentwickler von zertifizierten Trainern der Scrum Alliance ausbilden und zertifizieren lassen können (siehe http://www.scrumalliance.org).

Viel Feedback und Transparenz

Neben kurzen Feedback-Schleifen ist Transparenz ein weiteres wichtiges Element jedes Scrum-Projekts. Scrum ist darauf ausgelegt, sämtliche Aspekte unmittelbar sichtbar zu machen, die den Prozess oder dessen Ergebnisse

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

beeinflussen. Verschiedene Reports, tägliche Standup-Meetings und ein Review-Meeting am Ende jedes Sprints zeigen dem Team zu jedem Zeitpunkt, wo es gerade steht. Transparenz hilft nicht dabei, Probleme zu vermeiden, sondern sie so früh wie möglich sichtbar zu machen. Es ist ein entscheidender Unterschied, ob entwickelte Software nach sechs Monaten oder am Ende eines Sprints, also spätestens nach vier Wochen, das erste Mal integriert wird. In beiden Fällen wird es Probleme geben, allerdings mit dem Unterschied, dass die Probleme nach vier Wochen deutlich weniger komplex sind als nach sechs Monaten

Die wichtigsten Scrum-Begriffe im Überblick	
Scrum Master	Hauptverantwortlich für die Implementierung von Scrum und das Sicherstellen von optimalen Arbeitsbedingungen für das Team.
Product Owner	Hauptverantwortlich für den geschäftlichen Erfolg des Projekts. Der Product Owner definiert die Anforderungen und pflegt das Product Backlog.
Team	Hauptverantwortlich für die selbstorganisierte Umsetzung der Anforderungen im Sprint.
Vision	Die Vision beschreibt die Idee des zu entwickelnden Systems und benennt klar, welche Kundenbedürfnisse das System befriedigt.
Product Backlog	Priorisierte Liste aller Anforderungen des Projekts.
Selected Backlog	Die für den aktuellen Sprint ausgewählten Anforderungen.
Sprint	Ein- bis vierwöchige Entwicklungsphase, in der das Team eigenverantwortlich und selbstorganisiert an der Umsetzung der Anforderungen des Sprints arbeitet.
Review	Abschlussmeeting eines Sprints, in dem das Team seine erzielten Arbeitsergebnisse den Stakeholdern präsentiert.
Retrospektive	Ein Meeting im Anschluss an jeden Sprint, in dem das Team den Entwicklungsprozess und seine Zusammenarbeit im letzten Sprint analysiert und bespricht und gemeinsam Vorschläge für Verbesserungen erarbeitet.

Tabelle 1: Die wichtigsten Scrum-Begriffe.

Für wen ist Scrum geeignet?

Scrum funktioniert optimal für Teams von fünf bis neun Personen, die alle am selben Ort, oder besser noch, im selben Raum zusammenarbeiten. Für größere Projekte, die mehr als neun Entwickler erfordern, hat sich eine als "Scrum of Scrums" bezeichnete Organisationsform bewährt. Bei diesem Modell werden mehrere Scrum-Teams gebildet, die jeweils eigenständig nach den Scrum-Regeln arbeiten. Zusätzlich entsendet jedes Team im Anschluss an sein tägliches Standup-Meeting einen Vertreter in das Scrum of Scrums-Meeting, in dem sich die Teams hinsichtlich teamübergreifender Themen synchronisieren.

Überblick und Ablauf von Scrum

Bild 1 gibt einen Überblick über die Elemente und Abläufe von Scrum. Im Zentrum des Geschehens steht das Scrum-Team. Es besteht aus dem eigentlichen Entwicklungs-Team, dem Scrum Master und dem Product Owner. Diese drei Rollen und die dahinter stehenden Personen sind für die Entwicklung und Lieferung des Produkts verantwortlich. Eine Rolle ist keine Position im Sinne einer Linienorganisation, stattdessen umfasst sie verschiedene Verantwortlichkeiten. Die Verantwortlichkeiten der einzelnen Rollen überschneiden sich nicht und sorgen für eine ausgewogene Kräfteverteilung innerhalb des Scrum-Teams

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

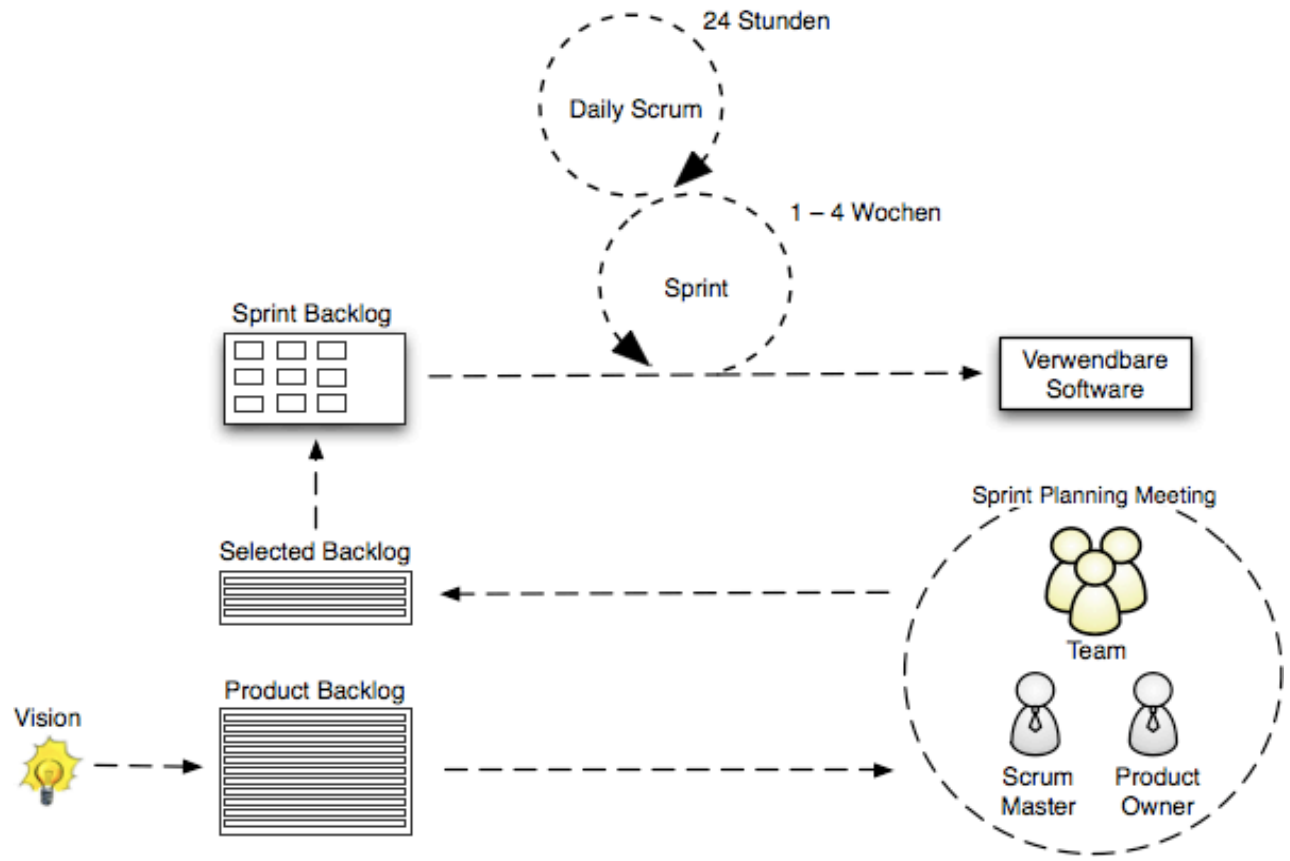


Bild 1: Elemente und Abläufe von Scrum.

Das Team

Das Team eines Scrum-Projekts entwickelt die Software und ist für den Erfolg jedes einzelnen Sprints verantwortlich. Teams bestehen aus fünf bis neun Personen und sind cross-funktional besetzt; das heißt, ein Team besteht nicht nur aus reinen Softwareentwicklern, sondern auch aus Testern, Designern, Dokumentatoren usw. Generell gilt, dass das Team über alle notwendigen Kompetenzen verfügen sollte, die für das Erreichen des Sprint-Ziels erforderlich sind. Das Team arbeitet selbstorganisiert und entschei-det selbst, mit welchen Techniken und Werkzeugen es seine Arbeit erledigt. Eine häufig anzutreffende Variante ist Scrum kombiniert mit Praktiken des Extreme Programmings, wie z.B. Pair-Programming oder testgetriebene Entwicklung.

Der Scrum Master

Der Scrum Master hat zwei Hauptverantwortlichkeiten: Erstens muss er dafür sorgen, dass Scrum funktioniert und die zugrunde liegenden Regeln und Prinzipien eingehalten werden. Zweitens muss er optimale Arbeitsbedingungen für das Team bereitstellen. Der Scrum Master steht hinter dem Team, schützt es vor negativen Einflüssen von außen und beseitigt alle Hindernisse, die sich dem Team in den Weg stellen. So bewahrt er beispielsweise das Team

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

vor neuen Anforderungen, die der Product Owner dem Team während der aktuellen Iteration aufdrücken will. Das ist nicht erlaubt und muss vom Scrum Master unterbunden werden. Oder er wehrt Aufgaben ab, die an das Team aus anderen Teilen der Organisation herangetragen werden, aber nichts mit dem Projekt zu tun haben.

Der Product Owner

Der Product Owner ist für den wirtschaftlichen Erfolg des Projekts verantwortlich. Zusammen mit dem Kun-den – in unserem Beispiel also mit den Investoren – treibt er das Projekt im Sinne des Geschäftsmodells voran. Er re-präsentiert sämtliche Stakeholder, die in irgendeiner Form Interesse am Projekt und Einfluss auf die Anforderun-gen haben. Alle Anforderungen laufen beim Product Owner zusammen und werden von ihm im Product Backlog gesammelt und priorisiert.

Für das Team repräsentiert der Product Owner den Kunden. Der Product Owner muss dabei eng und re-gelmäßig mit dem wirklichen Kunden zusammenarbeiten und dessen Anforderungen soweit aufnehmen und ver-stehen, dass er sie seinem Team kommunizieren kann. Die gleiche Form der engen Zusammenarbeit gilt für sei-ne Arbeit mit dem Team. Idealerweise steht der Product Owner dem Team einen Großteil seiner Arbeitszeit für Fragen und Diskussionen zur Verfügung. Am besten funktioniert dies, wenn der Product Owner seinen Arbeits-platz in unmittelbarer Nähe der Teammitglieder hat.

Die Vision

Am Anfang eines Scrum-Projekts steht dessen Vision. Eine Vision beschreibt in wenigen klaren Worten, worum es in diesem Projekt geht und was entwickelt werden soll. Die Vision holt alle Beteiligten ins Boot und definiert den roten Faden des Projekts. Sie ist keine umfassende Produktbeschreibung, sondern eher eine Art Bild, das jedem klar macht, worum es in dem Projekt geht. Eine griffige Vision für unser Job-Portal ist beispielsweise der Satz: "Wir wollen das einzige Job-Portal für High Professionals in Deutschland bauen."

Das Product Backlog

Ausgehend von der Vision des Projekts erstellt der Product Owner das Product Backlog. Dies ist eine priorisierte Liste mit allen Anforderungen, die umgesetzt werden müssen, um die Vision in ein benutzbares Produkt zu über-führen. Typische Einträge im Product Backlog eines Job-Portals sind:

- Recruiter können Stellenangebote posten
- Bewerber können ihr Profil hinterlegen
- Bewerber können nach Angeboten suchen
- Firmen können sich als Arbeitgeber präsentieren

Ein Eintrag im Product Backlog wird als Backlog Item bezeichnet. Eine verbreitete Variante zur Beschrei-bung von Backlog Items sind User Stories. Eine User Story ist eine kurze, prägnante Aussage darüber, was ein Nutzer

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

des Systems benötigt. Zum Beispiel: "Als Bewerber will ich automatisch benachrichtigt werden, wenn neue, auf mein Profil passende Angebote eingestellt wurden, so dass ich keine Angebote mehr verpasse". Der Trick ist, mit wenigen Worten möglichst viel zu sagen: Wer will etwas? Der Bewerber. Was will derjenige? Automatisch informiert werden. Warum will er das? Um keine interessanten Stellenangebote mehr zu verpassen.

Die Sprint Planung

Steht das Product Backlog, kann das Team in seinen ersten Sprint starten. Sprints sind Entwicklungspha-sen mit einer festen Dauer, in denen das Team die Software entwickelt. Jeder Sprint beginnt mit einem Sprint Planning Meeting, in dem der Product Owner dem Team die Backlog Items in der Reihenfolge ihrer Wichtigkeit vorstellt und erklärt. Das Team entscheidet, ob es das vorgestellte Item im anstehenden Sprint entwickeln kann. Passt das Item aus Sicht des Teams noch in den Sprint, setzt der Product Owner die Vorstellung mit dem nächsten Item fort.

Wichtig ist, dass einzig und allein das Team entscheidet, wie viele Items es in den anstehenden Sprint nehmen will. Der Product Owner gibt ausschließlich die Reihenfolge vor, nicht aber die Anzahl. Das Team legt sich auf eine Anzahl an Items fest und verspricht dem Product Owner, diese Items zum Ende des Sprints zu liefern. Die Backlog Items des aktuellen Sprints werden als Selected Backlog und das Versprechen des Teams als Commitment bezeichnet.

Der Sprint als "geschützter Raum"

Der Product Owner und das Team vereinbaren, dass die Items des Selected Backlogs fix sind. Während des Sprints darf der Product Owner keine weiteren Items in den Sprint nehmen oder beschlossene Items wieder herausnehmen. Der Sprint ist eine Art "geschützter Raum", in dem das Team die Möglichkeit erhält, einmal Beschlossenes zu Ende zu entwickeln, ohne dabei vom Product Owner mit neuen oder geänderten Anforderungen konfrontiert zu werden. Im Gegenzug erhält der Product Owner mindestens alle vier Wochen eine Eingriffsmöglichkeit.

Bis auf die Items des Selected Backlogs darf der Product Owner das Product Backlog jederzeit ändern und so für jeden Sprint neu bestimmen, was zum jeweiligen Sprint-Start wichtig ist. Auf diese Art wird ein Gleichgewicht zwischen Stabilität und Flexibilität geschaffen, das dem Team Sicherheit für eine bestimmte Zeit und dem Product Owner Steuerungsmöglichkeiten im Hinblick auf sich ändernde Anforderungen bietet.

Der Sprint

Nach dem Sprint Planning Meeting beginnt das Team mit der Entwicklung der Software. Das Team synchronisiert und koordiniert sich in einem täglichen Standup-Meeting, dem sogenannten Daily Scrum. In diesem Meeting berichten die Teammitglieder reihum, was sie gestern getan haben, was sie heute vorha-ben und welche Probleme es aktuell gibt. Der Scrum Master sorgt dafür, dass das Meeting nicht länger als 15 Minuten dauert und greift die berichteten Probleme auf. Sein Ziel ist es, für das Team möglichst optimale Arbeitsbedingungen zu schaffen und die berichteten Probleme bis zum nächsten Daily Scrum zu beseitigen.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Während des Sprints arbeitet das Team die Backlog Items nacheinander ab. Idealerweise arbeitet das gesamte Team an einem einzigen Item, stellt es vollständig fertig und setzt dann seine Arbeit mit dem nächsten Item fort. Da die Items im Selected Backlog genau wie im Product Backlog nach Wichtigkeit sortiert sind, wird durch diese Arbeitsweise sichergestellt, dass auch innerhalb des Sprints zuerst an den wichtigsten Items gearbeitet wird. Außerdem wird der Grad der Parallelarbeit reduziert, so dass am Ende des Sprints nicht vier Items halb fertig sind, sondern im schlechtesten Fall immer nur ein Item zwar angefangen, aber nicht abgeschlossen wurde. Halb fertige Arbeit ist nichts wert, zumindest nicht für den Kunden.

Sprint-Review und Sprint-Retrospektive

Das wichtigste Ziel eines Sprints ist die Lieferung funktionierender und benutzbarer Software. Im Abschlussmeeting des Sprints, dem sogenannten Sprint-Review, wird die Software sämtlichen Stakeholdern präsentiert. PowerPoint-Präsentationen sind verboten, und es wird ausschließlich laufende Software gezeigt. So lässt sich vermeiden, dass sich das Team speziell auf das Meeting vorbereitet. Wenn Vorbereitungen nicht erlaubt sind, muss zwangsläufig das gezeigt werden, was im Sprint geleistet wurde – und das ist am Ende immer funktionierende Software.

Anhand der präsentierten Software können die Beteiligten und speziell der Kunde entscheiden, ob die Ergebnisse den Erwartungen entsprechen und welche Richtung für den nächsten Sprint einzuschlagen ist. Das Sprint-Review ist die Zeit für Kurskorrekturen. Hier hätten die Investoren des Job-Portals sehr früh feststellen können, dass die implementierten Suchfilter für Recruiter völlig ungeeignet sind, um potenzielle Kandidaten effizient heraus zu filtern. Das Resultat hätte ein neues Backlog Item sein können, welches die Überarbeitung der Filter enthält und vom Product Owner entsprechend hoch priorisiert wird.

An das Review schließt sich die Sprint-Retrospektive an, eine Art "Meta-Diskussion" über den eigentlichen Entwicklungsprozess und die Zusammenarbeit des Teams während des Sprints. Das Ziel von Retrospektiven ist die kontinuierliche Verbesserung des Entwicklungsprozesses und damit der Produktivität des Teams.

Zusammenfassung und Fazit

Scrum ist ein Framework für die agile Softwareentwicklung, das auf Transparenz und Feedback basiert. Transparenz und Feedback beruhen auf dem Prinzip der Beobachtung und Anpassung. Dieses Prinzip findet sich in den einzelnen Ebenen und Elementen von Scrum wieder. Z.B. wird das Ergebnis der Entwicklung mindestens einmal im Monat im Sprint-Review präsentiert. Entspricht es nicht den Erwartungen oder haben sich die äußeren Bedingungen geändert, wird das Product Backlog angepasst und verändert. Stellen die Investoren des Job-Portals beispielsweise fest, dass die Konkurrenz an der Entwicklung eines bis-her nicht berücksichtigten Features "Kandidatenbewertung" arbeitet, dann kann der Product Owner seinen ursprünglichen Plan ändern und dieses Feature für einen der nächsten Sprints einplanen.

Scrum schafft eine Umgebung, in der gleichzeitig auf Veränderungen reagiert werden kann, in der aber auch ausreichend Sicherheit für das Team existiert, die angefangenen Funktionen fertigzustellen, bevor mit der Arbeit an

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

etwas Neuem begonnen wird. Das Team erhält die Chance zu liefern, ohne dass der Product Owner die Möglichkeit zur Steuerung verliert.

Eines der Ziele von Scrum ist die Maximierung des Geschäftswerts. Das lässt sich erreichen, indem die wichtigsten Features zuerst entwickelt und so früh wie möglich geliefert werden. Die Features müssen grundsätzlich benutzbar sein; dadurch ist es möglich, das Job-Portal nicht erst nach acht Monaten, sondern als Basisversion vielleicht schon nach zwei Monaten zu liefern. Auf diese Art kommen die Investoren ihren Konkurrenten zuvor und binden frühzeitig eine große Zahl von Nutzern an ihr eigenes Portal. Außer-dem können die Investoren nicht erst nach acht Monaten Geld verdienen, sondern schon wesentlich früher.

Voraussetzungen

Wenn Scrum so gut funktioniert, zufriedene und produktive Teams entstehen lässt und Sprint um Sprint genau das liefert, was sich der Kunde wünscht – spricht denn überhaupt irgendetwas gegen den Einsatz von Scrum? Gegen die vollständige Verbannung von Scrum spricht wenig, allerdings gibt es einige Aspek-te, die den Einsatz von Scrum deutlich erschweren. Ein wichtiger Punkt ist, dass man sich bewusst sein sollte, dass Scrum gerade zu Anfang und insbesondere für Teams ohne Scrum-Erfahrung sehr anstren-gend sein kann. Ständig gilt es Probleme zu lösen, da diese ja nicht erst gebündelt zum Projektende sicht-bar werden, sondern kontinuierlich ans Tageslicht gelangen und gelöst werden müssen.

Außerdem stellt Scrum hohe Anforderungen an das Team und dessen Fähigkeit zur Selbstorganisation. Dazu gehören nicht nur sehr gute Programmiererfahrungen, sondern auch eine hohe soziale Kompetenz. Diese ermöglicht es den einzelnen Teammitgliedern, eng und effektiv zusammenzuarbeiten, Führung für bestimmte Aufgaben zu übernehmen und für andere wieder abzugeben und dabei auftretende Probleme und Konflikte auf einer sachlichen Ebene zu lösen. Nicht jedem Team fällt es leicht, plötzlich keinen Pro-jektleiter mehr zu haben, der genau sagt, wo es lang geht und welche Aufgaben es im Detail zu erledigen gilt.

Eine weitere wichtige Bedingung für den Erfolg von Scrum ist die Einhaltung der Teamgröße sowie die Lokalisierung des Teams an einem Ort. Wird das Team zu groß oder befinden sich nicht alle Mitglieder am selben Ort, stellen sich neue Herausforderungen. Diese können zwar mit Technik, wie Videokonferenz-Systemen und webbasierten Product Backlogs oder speziellen Organisationsformen des Teams gelöst werden; die Implementierung von Scrum ist so jedoch deutlich schwieriger gegenüber einem Team, das sich an einem einzigen Ort befindet.

Eingeschränkte Planbarkeit

Eine weitere Herausforderung in Scrum ist die eingeschränkte Planbarkeit. Für ein Projekt, das Sprint für Sprint Ergebnisse liefert, deren Inhalte aber vom Product Owner alle ein bis vier Wochen neu bestimmt werden können, ist es offensichtlich schwierig voraus zu sagen, wann welches Feature kommt, wie lange das Projekt dauert und was es kostet. Sind Festpreisprojekte mit Scrum überhaupt möglich? Die Antwort ist einfach: Festpreisprojekte sind mit Scrum genauso gut oder schlecht möglich, wie mit "Wasserfall-Projekten". Ausgehend von der Reihenfolge

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

der Items im Product Backlog sowie einer Item-bezogenen Aufwandsschätzung lässt sich relativ einfach vorhersagen, in welchem Sprint welches Feature an der Reihe ist und wie viele Sprints insgesamt benötigt werden.

Der entscheidende Unterschied zwischen Scrum und klassisch gemanagten Projekten ist der, dass Scrum-Teams nicht zwingend an einem einmal aufgestellten Plan festhalten. Änderungen sind erwünscht und letztendlich nur eine Reaktion auf sich ändernde Umstände und neu gewonnenes Wissen. Änderungen der Anforderungen führen zwangsläufig zu Änderungen des Plans und damit zu Termin- oder Feature-Verschiebungen. Aber letztendlich geschieht dies alles im Sinne des Kunden und ist allemal besser, als an einem Plan festzuhalten, pünktlich zu liefern, aber am Ende doch an den Wünschen des Kunden vorbei zu entwickeln.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Wie führe ich als ScrumMaster? Führungsstile für die Praxis

Scrum ist derzeit das wohl am weitesten verbreitete agile Management-Rahmenwerk. Es definiert in seinen Grundzügen eine sehr überschaubare Anzahl an Rollen, Artefakten und Meetings. Eine Kernrolle, die Scrum definiert, ist die des "ScrumMasters". In dem vorliegenden Beitrag möchte ich die wesentlichen Merkmale dieser Rolle beleuchten und zwei Führungsstile vorstellen, die sich nach meiner Erfahrung für die Ausübung dieser Rolle besonders gut eignen. Zudem möchte ich zeigen, welche wesentlichen Unterschiede zwischen dem ScrumMaster und dem "traditionellen" Projektmanager bestehen und auf was man beim Wechsel in die Rolle eines agilen Managers besonders achten sollte.

Autor

Susanne Bartel

Dipl.-Ing. (BA), PM-Beraterin und Coach, Certified ScrumMaster, zertifiziert nach PMI und Prince2

Kontakt:
susanne.bartel@yourprojectzone.com

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Scrum – eine Einführung

sowie in den Rubriken:

› Teamführung

› Agiles Projektmanagement

Der ScrumMaster – ein kurzer Überblick

Der im Jahr 2011 aktualisierte "Scrum Guide" (Schwaber; Sutherland, 2011) beschreibt die Rolle des ScrumMasters wie folgt: "Der ScrumMaster ist verantwortlich, sicherzustellen, dass Scrum verstanden und umgesetzt wird. ScrumMaster erreichen dies indem sie sicherstellen, dass das Scrum-Team die Scrum-Theorie, -Praktiken und -Regeln einhält. Der ScrumMaster ist ein "Servant Leader" für das Scrum-Team. Er hilft denjenigen außerhalb des Teams zu verstehen, welche ihrer Interaktionen hilfreich sind, und welche nicht. Der ScrumMaster unterstützt alle Beteiligten, diese Interaktionen zu verändern, um den durch das Scrum-Team erzeugten Wert zu maximieren."

Der ScrumMaster unterstützt also zuallererst das Entwicklungsteam. Dabei ist ein wesentlicher Schwerpunkt seiner Arbeit, das Team auf dem Weg zur Selbstorganisation und zur abteilungsübergreifenden Zusammenarbeit zu unterstützen. Interessant dabei ist: Ein guter ScrumMaster macht sich über kurz oder lang in der Rolle des Organisators "überflüssig", da ein sehr gut eingespieltes und reifes Scrum-Team die Organisation selbst übernehmen wird. Dann wird der ScrumMaster das Team eher in der Rolle eines Coaches begleiten und es bei seiner weiteren kontinuierlichen Verbesserung unterstützen.

Weiter ist der ScrumMaster dafür verantwortlich, Hindernisse aus dem Weg zu räumen. Dazu schreibt der Scrum-Experte Ralf Wirdemann: "Der ScrumMaster steht hinter dem Team, schützt es vor negativen Einflüssen von außen und beseitigt alle Hindernisse, die sich dem Team in den Weg stellen. So bewahrt er beispielsweise das Team vor neuen Anforderungen, die der Product Owner dem Team während der aktuellen Iteration aufdrücken will. Das ist nicht erlaubt und muss vom ScrumMaster unterbunden werden. Oder er wehrt Aufgaben ab, die an

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

das Team aus anderen Teilen der Organisation herangetragen werden, aber nichts mit dem Projekt zu tun haben". (Scrum – eine Einführung, Projekt Magazin 21/2009)

Veränderungen begleiten

Selbstverständlich läuft die Arbeit eines Scrum-Teams nicht ohne Konflikte ab. Gerade in der Einführungsphase von Scrum wird meist sogar ein wesentlich höheres Konfliktpotential vorhanden sein – neue Rollen, Werte und Verhaltensweisen schüren Konflikte, z.B. bezüglich der Aufteilung der Arbeit unter den Teammitgliedern oder dem Druck auf Spezialisten, ihr Wissen zu teilen. Diese Konflikte können dabei sowohl innerhalb des Teams als auch zwischen dem Team und seiner Umgebung (z.B. dem nicht-agilen Rest der Firma) auftreten. Dies ist ein wichtiges Betätigungsfeld für den ScrumMaster.

! An dieser Stelle gilt: Der ScrumMaster unterstützt das Team im Erkennen und Lösen von Konflikten in der Team-Zusammenarbeit – für das Team löst er diese jedoch nicht. So wird er typischerweise das Team auf unterschwellige Konflikte aufmerksam machen ("den Finger in die Wunde legen") und es dabei unterstützen, eigene Kompetenzen und Spielregeln zur Konfliktlösung aufzubauen.

Kenntnisse im Bereich des Konflikt- und Beziehungsmanagements sind für den ScrumMaster hilfreich, die bei Bedarf auch durch Weiterbildungen ausgebaut werden sollten. Außerdem ist in diesem Zusammenhang wichtig, dass der ScrumMaster in seiner Rolle glaubhaft und integer ist und ihm die Teammitglieder Vertrauen entgegenbringen.

Genau dafür möchte ich nachfolgend zwei Führungsstile vorstellen, die sich nach meiner Erfahrung für die Rolle des ScrumMasters besonders gut eignen: Zum einen der Führungsstil "Servant Leader – Führen durch Dienen" sowie der Führungsstil "Creative Manager – Kreativer Manager".

Servant Leadership – Führen durch Dienen

In den meisten Beschreibungen der Rolle des ScrumMasters wird dieser als ein "Servant Leader" bezeichnet. Doch was bedeutet dies konkret? Der Stil "Führen durch Dienen" (Servant Leadership) basiert ursprünglich auf der Arbeit von Robert Greenleaf aus den siebziger und achtziger Jahren, die von vielen Akademikern aufgegriffen und weitergeführt wurde. Unter den "klassischen" Führungsstilen ist er am ehesten dem kooperativen Führungsstil zuzuordnen.

Greenleaf selbst meinte, dass die Überzeugungskraft den Servant Leader am deutlichsten von den anderen Führungsstilen unterscheidet. Dies ist auch im Falle des ScrumMasters ein wesentlicher Faktor, da es keine hierarchische Führung gibt, sondern der ScrumMaster als Teil des Teams "auf Augenhöhe" agiert (laterales Führen). Damit verfügt der ScrumMaster über keine Führungsbefugnis (z.B. durch eine vorgesetzte Position) und muss sein Team durch Überzeugungskraft dazu bewegen, ihm zu folgen.

In der ursprünglichen Literatur sind die dem Servant Leader eigenen Kompetenzen und Eigenschaften nur lose und nicht konsistent definiert. Eine von Russel und Stone veröffentlichte Studie (Russel, Stone, 2002) analysiert und systematisiert diese in neun funktionale Charaktereigenschaften und elf begleitende Eigenschaften. Diese neun Eigenschaften, die den ScrumMaster bei seiner Arbeit unterstützen können, stelle ich nachfolgend mit Beispielen vor:

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- **Vision** – die Fähigkeit, die Zukunft voraussehen und zu entwerfen.

  Während die Produktvision die Domäne des Product Owners ist, kann der ScrumMaster eine klare Vision für das Team entwickeln und transportieren.

- **Aufrichtigkeit und Integrität** – Aufrichtigkeit und Offenheit sind wesentliche Charaktereigenschaften, um als Servant Leader glaubhaft zu sein. Integrität bedeutet insbesondere, ethisch und nach geltenden moralischen Werten zu handeln.

  Nur aus einer starken persönlichen Position heraus wird ein ScrumMaster in der Lage sein, auch unbequeme Punkte offen und ehrlich anzusprechen. Durch das offene Kommunizieren wird er Vertrauen bei seinem Team und dem Management erwerben.

- **Vertrauen** – Servant Leader wecken das Vertrauen anderer in ihre Werte und Fähigkeiten. Als vertrauenswürdig angesehene Leader wird man bei Entscheidungsfindungen mehr Zuversicht entgegenbringen. Um Vertrauen aufzubauen, muss ein Leader sich um seine Mitarbeiter sorgen und stets integer agieren.

  Nur wenn der ScrumMaster Vertrauen in der Organisation genießt, werden ihm (und dem Team) eher die nötigen Freiräume eingeräumt und autonome Entscheidungen ermöglicht.

- **Service** – diese Eigenschaft liegt im Kern von Servant Leadership und basiert darauf, anderen und der Sache zu dienen (im Gegensatz zu Egoismus und Eigennutz).

  Der ScrumMaster stellt sich voll in den Dienst des Teams (Service), beispielsweise indem er sich aller Hindernisse annimmt. Er wird sein Handeln stets daran orientieren, sein Team voranzubringen, ohne sich selbst in den Vordergrund zu stellen.

- **Vorbildwirkung (Modeling)** – mit gutem Beispiel vorangehen, wobei Taten mehr zählen als Worte.

  Die Adaption eines agilen, selbstorientierten Arbeitsansatzes durch das Team kann nur gelingen, wenn der ScrumMaster zum einen angestrebte, neue Verhaltensweisen selbst vorlebt und gleichzeitig die nötige Glaubwürdigkeit und Integrität besitzt, andere "auf diese Reise mitzunehmen".

- **Wegbereitung (Pioneering)** – Servant Leaders müssen außergewöhnliche "Change Agents" sein. "Alle Führungspersönlichkeiten hinterfragen den Prozess. Sie sind Pioniere – Leute die willig sind, Neuland zu betreten. Sie sind bereit, Risiken in Kauf zu nehmen, zu erfinden und zu experimentieren, um neue und bessere Wege zu finden, bestimmte Dinge zu tun." (Russel und Stone, 2002).

  Verbesserungen in Scrum werden durch "Inspect and Adapt" angetrieben, also durch eine kontinuierliche Reflektion und Anpassung bzw. Weiterentwicklung. Dafür benötigt der ScrumMaster u.a. eine gewisse Risikobereitschaft (zu der er auch das Team ermutigen muss), um ausgetretene Pfade zu verlassen und neue, mitunter radikale Dinge zu versuchen.

- **Wertschätzung anderer** – Servant Leaders wertschätzen, ermutigen und umsorgen ihnen anvertraute Mitarbeiter deutlich und sichtbar. Die Beziehungen zu anderen und die Sorge um deren Interessen hat hohe Priorität.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

ScrumMaster müssen für ihre Arbeit eine starke Wertschätzung des Einzelnen sowie des Teams aufbringen. Dann kann der ScrumMaster auch das nötige Vertrauen aufbauen, um Freiräume für die Selbstorganisation zu schaffen.

- **Bevollmächtigung (Empowerment)** – Empowerment ist ein zentrales Element von Servant Leadership und betont die Teamarbeit.

  Indem der ScrumMaster das Team in seiner Selbstorganisation stärkt, werden die einzelnen Teammitglieder bevollmächtigt. Auf diese Weise führt der ScrumMaster seine Teammitglieder in die agile Welt ein. Dieser Multiplikator-Effekt wird auch erfolgreich bei der Einführung von Scrum in Organisationen genutzt.

Die elf identifizierten "begleitenden" Eigenschaften ergänzen und verstärken die neun funktionalen Merkmale. Sie sind keinesfalls zweitrangig, sondern oft nötige Voraussetzungen: Diese Eigenschaften sind im Einzelnen: Kommunikation, Glaubwürdigkeit, Kompetenz, (Produkt-)Verantwortung, Sichtbarkeit, Einfluss, Überzeugungskraft, Ermutigung, Zuhören, das Lehren bzw. Coachen sowie das Delegieren.

Creative Manager – der Kreative Manager

Ein weiterer relevanter Führungsstil in diesem Kontext ist der des "Creative Managers". Dieser Führungsstil wird aktuellen Entwicklungen in Organisationen gerecht, da Manager heute fähig sein müssen, in schnelllebigen Umgebungen zu arbeiten, über die sie deutlich weniger Kontrolle haben. Weiter müssen sie in der Lage sein, das Beste aus anderen herauszuholen. Die Rolle des kreativen Managers lässt sich als eine überwiegend moderierende definieren (Morgen et al., 2010).

Kreative Manager

- fördern und unterstützen eine offene, kollaborative Arbeitsatmosphäre und
- sehen einen Teil ihres Jobs darin, das Team vor unnötigen Unterbrechungen zu schützen.

Dies weist deutliche Parallelen zur Definition der ScrumMaster-Rolle auf. Dafür benötigen Kreative Manager folgende Kompetenzen und Eigenschaften:

- **Auffassungsgabe** – beispielsweise um Situationen aus verschiedenen Perspektiven betrachten zu können, was Flexibilität im Denken benötigt. Dies bedeutet auch, die Logik und Belange die sich hinter den Positionen verschiedener Interessenvertreter befinden, nachzuvollziehen und entsprechend darauf eingehen zu können.

  Eine gute Auffassungsgabe wird dem ScrumMaster beispielsweise beim Lösen von Hindernissen hilfreich sein, wo oftmals Dinge sehr fachlich bzw. technisch diskutiert werden. Außerdem kann er dadurch Interessen gut ausgleichen, z.B. in einer Sprintplanung zwischen dem Product Owner und dem Team vermitteln.

- **Unterstützung und Entwicklung der Mitarbeiter** – der Kreative Manager ist für die Koordinierung sowie Unterstützung der kreativen Bestrebungen seines Teams zuständig und muss dem Team mehr Freiräume und Selbstbestimmtheit ermöglichen können. Außerdem sollte er interessante Entwicklungsmöglichkeiten für seine Mitarbeiter schaffen.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Für den ScrumMaster bedeutet dies in erster Linie Unterstützung der Selbstorganisation und das Schaffen von Freiräumen.

- **Eigeninitiative** – sowohl für die Weiterentwicklung des Teams insgesamt und seiner Mitglieder als Individuen als auch bezüglich des Erkennens und Nutzens zukünftiger Möglichkeiten (z.B. mittels Szenarios).

  Der ScrumMaster wird in der Regel selber die Initiative ergreifen müssen, wenn er Dinge im Interesse des Teams umsetzen möchte.

- **Vernetzung** – Förderung der Vernetzung des Teams intern und extern. Dies dient der Etablierung von Feedbackschleifen und erweitert den Wissensraum über "Communities of Practice".

  Der ScrumMaster wird das Team dabei unterstützen, den Horizont zu erweitern und auch extern den Austausch zu suchen, sowohl zu fachlichen Aspekten als auch zu agilen Methoden.

Nutzen dieser Führungsstile für den ScrumMaster

Beide Stile haben gemeinsame Elemente, jedoch verschiedene Schwerpunkte und sind somit als Ergänzung zueinander zu sehen. Zusammenfassend kann man feststellen, dass der Servant-Leader-Stil für den ScrumMaster besonders in seiner Funktion als "Treiber der Veränderung" sehr hilfreich sein wird. Damit kann er aus einer starken persönlichen Position heraus (Offenheit, Integrität, Glaubwürdigkeit) andere für gemeinsame Ziele begeistern und sie motivieren, aktiv daran mitzuarbeiten.

Die Eigenschaften des Kreativen Managers hingegen werden den ScrumMaster vor allem dabei unterstützen, das kreative und innovative Potential seines Teams zu nutzen und auszubauen. Zum anderen werden sie ihm helfen, neue und unkonventionelle Wege zu finden, Gegebenheiten zu ändern, identifizierte Hindernisse zu beseitigen und kreativ mit Widerständen gegen Veränderungen umzugehen.

Für verschiedene Situationen der passende Führungsstil

Der Servant-Leader-Stil ist somit gerade in der Einführungsphase von Scrum sehr gut geeignet, da es gerade in dieser Phase die größten Veränderungen zum Teil gegen starke Widerstände anzustoßen und umzusetzen gilt. Für die tägliche Arbeit des Teams können die Kompetenzen des Kreativen Managers sehr gut genutzt werden, um das Team in seiner Arbeit erfolgreicher zu machen. Je nach Situation lassen sich also verschiedene Schwerpunkte setzen.

Führungsstile müssen immer situativ angemessen sein. Wie bei den meisten kooperativen Führungsstilen werden auch hier durch die Einbeziehung des Teams Entscheidungsprozesse etwas länger dauern; dafür ergeben sich in der Regel qualitativ bessere Entscheidungen. In akuten Krisensituationen, in denen schnelles Handeln nötig ist, sollten Leader stärker autokratisch agieren.

Voraussetzungen für diese Art der Teamführung

Für die beiden vorgestellten Führungsstile muss eine geeignete Umgebung vorhanden sein – oder geschaffen werden. Dabei spielen die Kultur in der Organisation, die Hierarchie oder das Geschäftsfeld eine wesentliche Rolle.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Konflikte können sich schnell abzeichnen, oder diese Art von Führung kann sogar scheitern, wenn bestimmte Merkmale in einer Organisationsumgebung nicht vorhanden sind. Geeignete Umgebungen zeichnen sich wie folgt aus:

- Eher flache statt hierarchische Organisationsform
- Eine offene Unternehmenskultur
- Hochqualifizierte Teams in wissensintensiven Branchen (in diesem Fall wissen die Mitarbeiter selbst, was zu tun ist)
- Ein starkes Vertrauensverhältnis zwischen Management und den Mitarbeitern
- Teams müssen in einem gegebenen Rahmen autonom handeln und entscheiden dürfen (Selbstorganisation)
- Mitarbeiter sollten möglichst in einem hohen Grad ermächtigt sein (Empowerment)
- Kein Mikromanagement
- Relativ kleine Teamgrößen bis ca. zehn Personen (da ansonsten Schwierigkeiten insbesondere für die Kommunikation entstehen)
- Räumliche Nähe des Teams (verteilte Umgebungen sind natürlich möglich, aber erhöhen den Grad der Komplexität)

Neben den Voraussetzungen in der Organisation müssen auch die beteiligten Personen entsprechende Fähigkeiten / Eignungen mitbringen bzw. diese müssen gezielt entwickelt werden. Die in den obigen Abschnitten zum Servant Leader und Creative Manager genannten Kompetenzen bieten dazu Orientierungspunkte.

Wenn der traditionelle Projektmanager agil wird

Im Zuge von Diskussionen über die ScrumMaster-Rolle, insbesondere in "traditionellen" Projektmanagement-Kreisen, kommt häufig die erstaunt-verängstigte Frage: "Brauchen wir nun keine Projektmanager mehr?" Die Antwort darauf hat meines Erachtens verschiedene Aspekte.

Zum einen hat der Projektmanager in traditionellen Projekten natürlich noch seine Berechtigung. Die derzeit vorherrschende Ansicht ist, dass es diese auf lange Sicht noch geben wird, da ein agiles Vorgehen weder heute noch in Zukunft für alle Projekte angemessen sein wird (Stichwort Kraftwerks- oder Brückenbau). Allgemein ist jedoch der Trend zu beobachten, dass agile Ansätze immer mehr aus der Software-Domäne in anderen Bereichen übernommen werden, z.B. bei Veränderungsinitiativen oder bei der Entwicklung von Produkten, die nichts mit Software zu tun haben.

Zum anderen ergeben sich durch die agilen Ansätze neue, spannende Entwicklungsmöglichkeiten für Projektmanager – z.B. in der Rolle des ScrumMasters oder des Product Owners. Viele der erlernten und gesammelten Fähigkeiten und Erfahrungen, die der "traditionelle" Projektmanager gemacht hat, werden sich weiterhin auszahlen. Dazu gehören beispielsweise Soft Skills wie kommunikative Fähigkeiten oder die Arbeit mit Teams und verschiedensten Interessenvertretern; als Product Owner natürlich auch die Erfahrung in bestimmten Fachgebieten.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Stolperfallen beim Wechsel

Wenn der "traditionelle" Projektmanager in die ScrumMaster-Rolle schlüpft, wird der Wechsel vom Macher zum Unterstützer eine große Veränderung bedeuten. Eine zu starke Positionierung kann leicht in einer Bevormundung des Teams oder einer unangemessenen Einflussnahme enden, was die Selbstorganisation des Teams behindern würde. Auch der Wechsel des Planungshorizonts (von einer sehr langfristigen) Planung hin zur kurzen Sprintplanung wird neu sein. Allgemein wird es viel mehr "lose Enden" geben und die Tätigkeit mehr Flexibilität erfordern. Weiterhin von hohem Nutzen werden jedoch "Kümmerer-Kompetenzen" sein, besonders wenn es um das Entfernen von Hindernissen geht.

Wer die Abgabe der Verantwortlichkeit scheut, für den könnte die Rolle des Product Owners die geeignetere Alternative sein – besonders wenn durch die Arbeit als Projektmanager bereits umfangreiches Kunden- und Fachwissen in der entsprechenden Domäne aufgebaut wurde.

Konfliktbereiche im "traditionellen" Unternehmensumfeld

Eine stärkere Orientierung an den beschriebenen Führungsstilen bedeutet in vielen Unternehmen einen erheblichen Grad an Veränderung. In den seltensten Fällen verläuft die Einführung agiler Ansätze konfliktfrei. Was überraschenderweise vielleicht auf den ersten Blick ein großes Hindernis darstellt, ist die Stärkung der Selbstorganisation: Nach meiner Erfahrung und der anderer agiler Coaches sind wir besonders durch hierarchische Strukturen in Unternehmen so in "Command and Control" verhaftet, dass wir selbstorganisiertes und verantwortliches Handeln erst wieder erlernen müssen.

Der ScrumMaster hält in diesem Prozess dem Team ab und an den Spiegel vor, damit es diese eingefleischten Verhaltensweisen erkennt. Als hauptsächliches Instrument stehen ihm dazu die Retrospektiven zur Verfügung, in denen er das Team beispielsweise bestimmte Aspekte reflektieren lassen kann. Die nötige Glaubhaftigkeit in dieser Rolle bedingt natürlich, dass er oder sie selbst genau diese ungewollten Verhaltensmuster nicht mehr zeigt. Gerade wir als Projektmanager sind das "Machen" gewohnt und müssen in der Rolle des ScrumMasters das "Machen lassen" bewusst (wiederer-)lernen. Dies schließt auch das Zulassen von Fehlern und Lernerfahrungen des Teams ein!

In deutschen Unternehmen sieht die Realität überwiegend doch noch anders aus. Selbst in jungen, vermeintlich "hippen" Organisationen ist oft ein erstaunlicher Grad an Hierarchiedenken und fehlender Selbstbestimmung zu finden. Manager der mittleren Ebenen fühlen sich oft von Veränderungen dieser Art bedroht. Allerdings vertrete ich die Meinung, dass nicht die Position an sich bedroht ist, sondern eine Verschiebung der Kompetenzen nötig wird hin zu mehr:

- Ausrichten des Teams auf ein gemeinsames Ziel
- Einfordern von Verpflichtungen (Commitment)
- Eingehen auf Ideen aus dem Team und dieses dabei unterstützen
- Voraussehen von Änderungen in der Umgebung und schnelles Reagieren darauf

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Einladung zur Diskussion

- Natürlich konnte in diesem Artikel das Thema nicht vollständig mit all seinen Aspekten betrachtet werden. Falls Sie Interesse an einem weiteren Austausch dazu haben, lade ich Sie ein, sich im Forum des Projektmagazins an der zugehörigen Diskussion zu beteiligen.

Literatur

- Greenleaf, Center for Servant Leadership unter http://www.greenleaf.org/whatissl/ (Stand: 05.02.2012)
- Morgen et. al in: B822 Creativity, Innovation and Change, Book 1, The Open University Business School, Milton Keynes, 2010
- Schwaber, Ken; Sutherland, Jeff: The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game, 2011
- Wirdemann, Ralf: Agiles Projektmanagement. Scrum – eine Einführung, Projekt Magazin 21/2009
- Robert F. Russell; A. Gregory Stone: A review of servant leadership attributes: developing a practical model, Leadership & Organization Development Journal , Volume 23 (3): 13, Emerald Publishing, May 1, 2002

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fachartikel

Scrum-Projekte planen mit System

Teil 1: Besonderheiten und Voraussetzungen

Scrum wird in immer mehr Software-Projekten jeder Größe eingesetzt. Dabei ist natürlich auch die Planung wesentlich, allerdings verschiebt sich im Vergleich zu traditionellen Ansätzen der Zeitpunkt dafür – denn anstatt einer langen, vorgelagerten Planungsphase erfolgt in Scrum die Planung hauptsächlich rollierend und parallel zur Entwicklung. Leider messen aber selbst erfahrene Scrum-Anwender der Planung oft zu wenig Bedeutung bei, denn häufig hört man Aussagen wie: "Wir sind agil, wir brauchen keine Planung!", oder: "Das Agile Manifest sagt, dass wir auf Veränderungen reagieren und keinem Plan folgen müssen!"

Beide Aussagen sind falsch und führen im schlimmsten Fall dazu, dass die unzureichend geplanten Projekte am Ende scheitern. Typische Fehler bei der Planung sind, dass z.B. ohne klares Ziel gestartet wird oder keinerlei Aufwandsschätzung erfolgt. Ohne Aufwandsschätzung lässt sich jedoch die Velocity, also die Geschwindigkeit des Teams, nicht ermitteln und man erreicht in der Folge auch keine Prognosesicherheit. Kommt dann die Nachfrage des Geschäftsführers, wie gut sich das Team macht, und ob die angestrebte Produktivitätssteigerung erreicht wurde, ist wieder Unschuldslächeln und Achselzucken angesagt, da man ja keine Vorstellung davon hat, wie aufwändig die drei umgesetzten Features relativ zu den noch ausstehende sieben waren.

Autor

Dominik Maximini

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

Kontakt:
dominik.maximini@novatec-gmbh.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Agile Softwareentwicklung mit Scrum und User Stories

sowie in den Rubriken:

› Ablauf/Termine
› Agiles Projektmanagement
› IT-Projekte

In diesem zweiteiligen Artikel zeige ich, wie sich die Planung in Scrum strukturiert durchführen lässt und welche Besonderheiten es dabei zu berücksichtigen gilt. Weiter möchte ich auf typische Fallstricke eingehen, denen man in der Praxis oft begegnet. Ziel ist, die Komponente "Planung" für fortgeschrittene Scrum-Anwender näher zu beleuchten. Der erste Teil stellt kurz das Vorgehen vor und beleuchtet Besonderheiten bzw. Fehler, die häufig in der Praxis gemacht werden. Der zweite Teil beschreibt ausführlich das Vorgehen an einem praxisnahen Beispiel.

Neun Planungsschritte in Scrum

Scrum sieht neun Planungsschritte vor, die sich je nach Zielsetzung in strategische und operative Planung unterteilen lassen (zur näheren Erläuterung s. Abschnitt "Operative und strategische Planung"). Nachfolgend stelle ich die Schritte kurz vor; im zweiten Teil werde ich diese an einem Praxisbeispiel näher erläutern. Die Planungsschritte sind im Einzelnen:

1. Die Geschäftsführung definiert mit dem Product Owner die Produktvision (strategische Planung).
2. Der Product Owner definiert die Releaseziele zusammen mit den Stakeholdern (strategische Planung).

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

3. Der Product Owner definiert die Sprintziele zur Erreichung des aktuell anstehenden Releaseziels (strategische Planung).
4. Der Product Owner trägt die benötigten Features im Product Backlog ein (strategische Planung).
5. In der ersten Hälfte des Sprint Plannings stellt der Product Owner dem Team diese Ziele vor. Das Team wählt sich eine Untermenge an Features aus dem Product Backlog aus, die es umsetzen will (operative Planung). Ggf. werden die Sprintziele nochmals angepasst oder Features im Product Backlog geändert.
6. In der zweiten Hälfte des Sprint Plannings plant das Team präzise auf Taskebene, wie es sein Sprintziel erreichen möchte (operative Planung).
7. Die Planung wird täglich, spätestens während des Daily Scrums, überprüft. Abweichungen werden an den ScrumMaster adressiert und an den Product Owner kommuniziert (operative Planung).
8. Am Ende des Sprints wird im Sprint Review das Produkt inspiziert. Hier planen Product Owner, Entwicklungsteam und Stakeholder gemeinsam, welche Veränderungen im Product Backlog nötig sind, um das Releaseziel zu erreichen (strategische Planung).
9. In der Retrospektive wird ermittelt, wie das Team noch effizienter arbeiten kann (strategische Planung).

Wesentliche Voraussetzungen für die Planung

Alle Planungsschritte wiederholen sich regelmäßig in jedem Sprint. Die Anzahl der notwendigen Änderungen variiert allerdings stark: Produktvision und Releaseziel ändern sich wesentlich seltener als die Sprintziele. Trotzdem wird ihre Gültigkeit ständig durch den Product Owner überprüft. Die operativen Planungsartefakte werden jeden Sprint komplett neu erstellt und können täglich im Daily Scrum geändert werden.

Jeder muss seine Rolle ausführen können

Übergreifend gilt für alle oben beschriebenen Schritte, dass für eine erfolgreiche Planung die richtigen Fähigkeiten an den richtigen Stellen vorhanden sein müssen. Das bedeutet, dass die Rollen mit dafür geeigneten Personen besetzt und diese zur Erfüllung ihrer Aufgaben auch ermächtigt werden müssen. Nur wenn ein Product Owner beispielsweise Entscheidungen treffen darf, kann er seine Rolle korrekt ausfüllen. Ähnlich verhält es sich mit dem Entwicklungsteam: Nur wenn es ohne äußere Einflussnahme wählen kann und darf, wie viele Features es im anstehenden Sprint umsetzen möchte, ist es fähig, nach Scrum zu arbeiten.

Scrum lernt

Scrum ist ein empirisches Modell – es lernt aus der Vergangenheit. Das bedeutet auch, dass mit der Zeit die Prognosesicherheit steigen wird. Ein eingespieltes, erfahrenes Team, das an einem Produkt arbeitet, das es kennt, liefert letztendlich auch sehr verlässliche Vorhersagen darüber, wie viele Features es in einem Sprint schaffen wird. Kennt sich das Team jedoch nicht und ist auch das Produkt für die Entwickler neu, so ist eine Vorhersage in der Regel äußerst ungenau. Optimieren Sie Ihre Prognosesicherheit, indem Sie ein erfolgreiches Entwicklungsteam zusammen lassen und auch das Produkt, an dem dieses Team arbeitet, nur dann ändern, wenn dies unter Berücksichtigung aller Faktoren unbedingt notwendig ist.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Schnelle und direkte Kommunikation – kein "Tooling"

Auch sollte man sich bewusst sein, dass Kommunikation wichtiger ist als "Tooling". Sich schnell verändernde Umgebungen, wie sie bei der Softwareentwicklung die Regel sind, erfordern zeitnahe (mindestens tägliche) und zielgerichtete (persönliche und fokussierte) Kommunikation. Kein Werkzeug kann diese ersetzen. Zwar können die richtigen Werkzeuge den Planungsprozess unterstützen, jedoch kann nur das, was auch mit Papier funktioniert, in einem Tool klappen. In einigen Fällen enthalten Tools sogar Beschränkungen, die es Ihnen verbieten, die Planung so durchzuführen, wie Sie es für richtig halten. Achten Sie daher darauf, grundsätzlich nur flexible und von Ihren Mitarbeitern anpassbare Werkzeuge einzusetzen.

Dazu ein Beispiel aus der Praxis: Ein Unternehmen beschäftigte sich mehrere Monate damit, ein sehr mächtiges Tool zur Modellierung und Ausführung der Workflows einzuführen. Nach der Einführung, stellte das Unternehmen fest, dass die Mitarbeiter diese Workflows nicht nutzten. Eine Analyse zeigte, dass wichtige Faktoren nicht berücksichtigt worden waren, insbesondere Ausnahmeregelungen und bewusste Prozessverletzungen ließen sich nicht abbilden. Also begann man damit, die Prozesse auf Papier zu zeichnen und durch physische Artefakte wie Karten und Boards zu unterstützen. Davon ausgehend optimierte das Unternehmen seine Prozesse und schon bald erfreute sich dieses Vorgehen großer Beliebtheit. Das Workflow-Tool wurde nie wieder eingesetzt, da man feststellte, dass es zu vielen Restriktionen unterlag; man stritt sich mehr um die Möglichkeiten und Unmöglichkeiten der Software als um die produktive Arbeit.

Besonderheiten der Planung in Scrum

Scrum verzichtet auf eine lange und vollständig vorgelagerte Planungsphase. Während beim traditionellen Vorgehen durchaus mehrere Monate – manchmal sogar Jahre – geplant wird, bevor die erste Zeile Code geschrieben werden darf, legt Scrum den Schwerpunkt auf den frühen Beginn der Implementierung. Dazu muss zwar auch in einem gewissen Maß vorab geplant werden, allerdings reicht ein Sprint bereits als Vorlaufzeit aus – dieser lässt sich auch als "Sprint 0" bezeichnen.

Die Dauer dieses Sprints ist abhängig von den noch ausstehenden Aufgaben und beträgt oft nur zwei, maximal vier Wochen. Product Owner, ScrumMaster und Entwicklungsteam nutzen ihn dafür, die Infrastruktur vorzubereiten und die strategische Planung durchzuführen. Am Ende von Sprint 0 sind alle Beteiligten arbeitsfähig – oft bewiesen durch die Implementierung eines ersten winzigen Features – und verfügen über Releaseziele, Sprintziele und ein geschätztes initiales Product Backlog. Der Sprint 0 endet mit der operativen Planung des ersten "richtigen" Sprints.

Während der gesamten Projektlaufzeit erfolgt die Planung im Unterschied zu traditionellen Planungsmethoden parallel zur Umsetzung, wobei man zwischen strategischer und operativer Planung unterscheiden kann. Hier ist auch zu beachten, dass die Planungsverantwortung nicht bei einer Person liegt, sondern auf mehrere Rollen verteilt wird (s. dazu den Abschnitt "Operative und strategische Planung").

Richtiger Zeitpunkt, richtiger Umfang

Scrum erfordert nicht, dass die Planung vollständig abgeschlossen ist, sondern lediglich, dass sie zu jedem Zeitpunkt "gut genug" ist. Das bedeutet, dass entsprechend dem Projektkontext im Voraus geplant werden muss, wobei

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Zeitpunkt und Umfang der Planung entscheidend sind. Denn setzt die Planung zu spät ein, verfällt das Scrum-Team in Chaos, da es dann von unzureichend definierten Anforderungen überschwemmt wird, und kein klares Ziel mehr vor Augen hat. Anders ausgedrückt: Das Team muss etwas produzieren, weiß allerdings nicht, was genau. Setzt die Planung jedoch zu früh ein, muss sie zu oft verändert werden und es entstehen unnötige Kosten. Die gleichen Folgen hat ein unangemessener Umfang der Planung. Wie kann man diesen Problemen begegnen?

Operative und strategische Planung

Um die Planung in Scrum jeweils zum richtigen Zeitpunkt und im geeigneten Umfang durchzuführen, lohnt es sich, die zwei Dimensionen der Planung in Scrum zu betrachten: Auf der einen Seite gibt es die strategische Planung, zu der die Produktvision, die Releaseplanung und die Sprintziele gehören. Auf der anderen Seite steht die operative Planung, zu der die Sprintplanung sowie die tägliche Anpassung dieser Planung an die aktuellen Gegebenheiten zählen. Die Verantwortung für die strategische Planung liegt beim Product Owner, die Verantwortung für die operative Planung beim Entwicklungsteam. Das bedeutet, dass der Product Owner die zukünftigen Sprints und Releases plant, während das Entwicklungsteam sich zeitgleich um die Identifikation und Abschätzung der Tasks für den aktuellen Sprint kümmert.

Faustregeln für die Planung

Das nächste Release

Der Product Owner plant auf strategischer Ebene das nächste Release komplett durch. Das bedeutet, dass er zunächst den Releasezeitpunkt bestimmt und die Anzahl der Sprints bis zu diesem Termin berechnet. Anschließend legt er Release- und Sprintziele fest. Diese ergänzt er durch möglichst konkrete fachliche Anforderungen, oft in Form von User Stories. Falls diese noch nicht geschätzt wurden, lässt er sie von seinem Team schätzen. Liegen diese Informationen vor, gleicht der Product Owner seine Planung mit der Geschwindigkeit seines Teams, der Velocity, ab und passt die Planung falls nötig an.

Vorausplanung	Planungsumfang
Release 2 (Das nächste Release)	Anforderungen vollständig geplant
Release 3 (Das übernächste Release)	Anforderungen grob geplant
Release 4 und 5 (Die darauffolgenden Releases)	Grobe Ziele geplant

Tabelle 1: Beispiel für die Vorausplanung während Release 1.

Die Planung des nächsten Releases erfolgt parallel zum bereits laufenden Release und kann insgesamt mehrere Tage in Anspruch nehmen – allerdings nicht am Stück, sondern meist verteilt über mehrere Wochen, da die Zwischenergebnisse mit Kunden und Stakeholdern diskutiert werden müssen. Für die Schätzungen benötigt der Product Owner etwa 6-10% der Kapazität seines Teams.

Häufig plant der Product Owner auf dieser Detailebene rollierend im Voraus. Er wartet also nicht auf einen speziellen Zeitpunkt, sondern plant immer eine Releaselänge im Voraus. Bei einer Releasedauer von drei Monaten

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

sorgt er also z.B. dafür, dass immer die nächsten drei Monate auf der Detailstufe geplant sind. So bleibt seine Belastung für die Planung konstant.

Das übernächste Release

Während der Product Owner das nächste Release plant, wird er weitere Sprintziele sowie Anforderungen identifizieren. Diese nimmt er in sein Product Backlog auf und ordnet sie, falls sinnvoll, den passenden Releases zu. Um seinen Stakeholdern und dem Team einen Ausblick geben zu können, stellt der Product Owner sicher, dass er mit dem Abschluss der Planung des nächsten Releases bereits Release- und Sprintziele für das übernächste Release präsentieren kann. Er plant dieses jedoch nur grob und verzichtet meist weitgehend auf konkrete Anforderungen. Zusätzliche Planungszeit benötigt er dafür in der Regel nicht.

Die darauffolgenden Releases

Für die beiden Releases, die auf das übernächste folgen, muss mindestens eine diffuse Vorstellung in Form von groben Zielen existieren, wobei ein Release nicht länger als drei Monate dauern sollte. Auch dies erfolgt im Zuge der Planung für das nächste Release. In der Regel ergeben sich die kommenden Releaseziele automatisch aus den Diskussionen mit Stakeholdern und werden vom Product Owner mit diesen abgestimmt.

Die Planung des Teams

Die operative Planung des Teams erfolgt immer nur für den aktuellen Sprint und zwar bis auf Task- und Stundenebene hinunter. Dies geschieht initial während der zweiten Hälfte des Sprint Planning Meetings. Die so entstandene Planung wird täglich während des Daily Scrums überprüft und gegebenenfalls korrigiert.

Beispiel

Unter der Annahme, dass ein Release drei Monate und ein Sprint vier Wochen dauert, verfügt ein Product Owner also zu jedem Zeitpunkt über eine präzise Planung bis auf Taskebene für den aktuellen Sprint (Monat 1). Die folgenden Sprints (Monate 2 bis 4 bzw. das nächste Release) sind bis auf die Ebene der Anforderungen geplant. Das übernächste Release (Monate 5 bis 7) ist lediglich mit Sprintzielen versehen. Darüber hinaus (ab Monat 8) gibt es noch zwei Releaseziele, die mit den Stakeholdern abgestimmt sind.

Wie präzise im Voraus geplant wird, hängt von den Anforderungen des konkreten Projekts ab. Je nach Auftraggeber kann es auch notwendig sein, für mehr als einen Monat im Voraus bis auf Taskebene zu planen.

Typische Fallstricke bei der strategischen Planung

In vielen Unternehmen wird nicht in "Produkten", sondern in "(Teil-)Projekten" gedacht. Entsprechend werden auch hauptsächlich (Teil-)Projektziele beachtet, während der übergeordnete Kontext des Produkts aus den Augen verloren wird. In der Folge fühlt sich z.B. der Product Owner eines Teilprojekts nicht für das gesamte Produkt, sondern nur für die im Teilprojekt abzudeckenden Aspekte verantwortlich. Selbstverständlich ist dieses Vorgehen

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

aus Sicht des einzelnen Teilprojekts vollkommen richtig, was in den meisten Fällen jedoch auf lange Sicht zu einem nicht optimalen Return on Invest (RoI) und zu einem höheren Total Cost of Ownership (TCO) führt.

Ein Beispiel aus der Praxis: Der Product Owner eines Teilprojekts erstellte einen eigenen Standalone-Client für seine Features. Ein Kollege, der für dasselbe Produkt entwickelte, erstellte ebenfalls einen eigenen Client. Beide Clients erforderten Eingriffe am selben Code des zugrunde liegenden Serverprodukts. Die Anwender mussten sich in der Folge mit mehreren Clients herumquälen und auftretende Fehler im Code des gemeinsam genutzten Servers wurden im Ping-Pong-Verfahren herumgereicht. Zwei Jahre später war die Erstellung eines neuen, einheitlichen Clients notwendig. Ein unnötiger Schritt, hätte man von Anfang an das Gesamtprodukt in den Fokus gerückt.

Releaseplanung – nicht alle Wünsche realisieren

Agile Entwicklung heißt nicht, alle Wünsche der Kunden unreflektiert umzusetzen! Der gängigste Fehler bei der Releaseplanung ist das vollständige Verlassen der Vogelperspektive, also des Blicks auf das Produkt in einer zeitlich langen Sicht. Es ist natürlich einfacher, alle Anforderungen seiner Stakeholder in mehr oder weniger sinnvolle Einheiten zu gruppieren, anstatt aus strategischer Sicht zu planen, was für das Produkt wirklich benötigt wird – denn diese strategische Planung erfordert Weitsicht und jede Menge Arbeit. Bleibt diese Arbeit jedoch aus, so wird nicht klar, in welche Richtung sich das Projekt bzw. Produkt entwickeln soll und die Gefahr steigt, falsche Entscheidungen zu treffen.

Die Gefahr, dass die Produktentwicklung in eine falsche Richtung geht, besteht auch, wenn keine klare Produktvision vorhanden ist. Da die Entwickler täglich Entscheidungen treffen müssen, die sich u.U. stark auf die Zukunft der Software auswirken, benötigen sie eine klare Vision, die als Grundlage für die richtige Entscheidung dienen kann. Fehlt hingegen eine klar formulierte Produktvision, kann es zu Entscheidungen kommen, die nicht im Sinne der gesetzten Projektziele sind.

Typische Fallstricke bei der operativen Planung

Die operative Planung beschäftigt sich mit dem detaillierten Planen einzelner Aufgaben. Was in der strategischen Planung an Vorbereitung häufig fehlt, wird bei der operativen Planung oft zu viel investiert. Die beiden größten Fehler sind dabei das Planen zu lange im Voraus sowie die Mikroplanung.

Langfristige Ziele lassen sich nur grob planen

In manchen Organisationen wird Druck auf die Projektbeteiligten ausgeübt, nach der Identifizierung aller Tasks eine scheinbar präzise Vorhersage des Fertigstellungstermins abzugeben. Scrum sieht das nur für den jeweils aktuellen Sprint vor. Weitet man diese Vorgabe allerdings auf Releases oder sogar ganze Projekte aus, so wird eine Menge Aufwand in die eine Planung gesteckt, die mit sehr hoher Wahrscheinlichkeit bereits nach kurzer Zeit hinfällig ist. Versucht man dann auch noch auf Biegen und Brechen diesen Plan einzuhalten, ist der Vorteil der Agilität verloren.

Absolute Planungssicherheit ist in Softwareprojekten nur selten zu erreichen (in solchen Fällen ist Scrum auch ungeeignet). In Scrum müssen alle akzeptieren, dass langfristig nur grobe Ziele geplant werden, deren genaue Ausgestaltung im Detail noch nicht festgelegt ist. Kurzfristig plant man hingegen präzise. So weiß der Kunde,

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

welche Themen behandelt, aber nicht exakt, welche Features geliefert werden. Ein solches Vorgehen berücksichtigt, dass Vorhersagen in der Softwareentwicklung nicht sinnvoll möglich sind und ermöglicht es dem Kunden, auch während der Projektlaufzeit ständig neue Anforderungen einzubringen, die dann anstelle früherer Wünsche den noch ungeplanten Releases zugewiesen werden können.

Keine Mikroplanung

Selbst wenn durch die Organisation und den Product Owner keine Fehler gemacht wurden, kann das Entwicklungsteam immer noch unnötige Kosten verursachen. Beliebt ist z.B. die Mikroplanung: Statt Aufgaben auf Tagesbasis zu planen, werden hier bis auf die Minutenebene Einzelschritte extrahiert. Dabei kostet die Planung mancher Schritte oft mehr Zeit, als die Ausführung der Aufgabe selbst beanspruchen würde.

Ähnlich kritisch ist es, wenn das Entwicklungsteam die Aufgaben zu große Schritte aufteilt: Dauert ein Task länger als einen Tag, ist es schwierig festzustellen, ob das Team noch im Plan liegt oder ob es Handlungsbedarf gibt. Die Transparenz geht verloren, eine sinnvolle tägliche Neuplanung wird unmöglich.

Bewährt hat sich, nur Tasks zu planen, die mehr als eine Stunde und weniger als einen Tag dauern. Kleinere Tasks werden zusammengefasst. Größere müssen in kleinere Teilaufgaben heruntergebrochen werden. Auf diese Weise ist einerseits sichergestellt, dass Mikroplanung vermieden wird, andererseits erhält der Product Owner täglich einen transparenten Projektstatus.

Ausblick

Der zweite Teil beschreibt detailliert die neun Planungsschritte und zeigt, worauf man bei den einzelnen Schritten achten sollte. Anhand eines realen Praxisbeispiels erfahren Sie, wie auf diese Weise die Planung in Scrum bei einem Unternehmen aus der Automobilbranche durchgeführt wurde.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Scrum-Projekte planen mit System

Teil 2: Vorgehen in der Praxis

Der erste Teil beschrieb kurz die neun Planungsschritte in Scrum und zeigte die wesentlichen Faktoren auf, die es dabei zu beachten gilt. Außerdem beleuchtete der Beitrag die Aspekte der strategischen und operativen Planung und ging darauf ein, welche Fehler bei der Planung häufig gemacht werden. Dieser zweite und abschließende Teil stellt die einzelnen Planungsschritte nun detailliert an einem Praxisbeispiel vor.

Autor

Dominik Maximini

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

Kontakt:
dominik.maximini@novatec-gmbh.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Agile Softwareentwicklung mit Scrum und User Stories

sowie in den Rubriken:

› Ablauf/Termine
› Agiles Projektmanagement
› IT-Projekte

Neun Planungsschritte in Scrum

Die neun Planungsschritte in Scrum sind:

1. Die Geschäftsführung definiert mit dem Product Owner die Produktvision (strategische Planung).
2. Der Product Owner definiert die Releaseziele zusammen mit den Stakeholdern (strategische Planung).
3. Der Product Owner definiert die Sprintziele zur Erreichung des aktuell anstehenden Releaseziels (strategische Planung).
4. Der Product Owner trägt die benötigten Features im Product Backlog ein (strategische Planung).
5. In der ersten Hälfte des Sprint Plannings stellt der Product Owner dem Team diese Ziele vor. Das Team wählt sich eine Untermenge an Features aus dem Product Backlog aus, die es umsetzen will (operative Planung). Ggf. werden die Sprintziele nochmals angepasst oder Features im Product Backlog geändert.
6. In der zweiten Hälfte des Sprint Plannings plant das Team präzise auf Taskebene, wie es sein Sprintziel erreichen möchte (operative Planung).
7. Die Planung wird täglich, spätestens während des Daily Scrums, überprüft. Abweichungen werden an den ScrumMaster adressiert und an den Product Owner kommuniziert (operative Planung).
8. Am Ende des Sprints wird im Sprint Review das Produkt inspiziert. Hier planen Product Owner, Entwicklungsteam und Stakeholder gemeinsam, welche Veränderungen im Product Backlog nötig sind, um das Releaseziel zu erreichen (strategische Planung).
9. In der Retrospektive wird ermittelt, wie das Team noch effizienter arbeiten kann (strategische Planung).

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

	Planungsartefakte	Verantwortlich
Strategische Planung	Produktvision Releaseplanung Sprintziele	Product Owner
Operative Planung	Sprintplanung tägliche Anpassung dieser Planung an die aktuellen Gegebenheiten	Entwicklungsteam

Tabelle 1: Ergebnisse und Verantwortlichkeiten der strategischen und operativen Planung.

Wie bereits im ersten Teil erwähnt, wiederholen sich diese Schritte regelmäßig für jeden Sprint. Allerdings müssen dabei nicht immer alle Planungsartefakte von Grund auf neu erstellt werden. Zum Beispiel ändern sich Produktvision und Releaseziel wesentlich seltener als die Sprintziele. Dennoch überprüft der Product Owner regelmäßig in jedem Sprint ihre Gültigkeit.

Praxisbeispiel

Das Projekt der "Beispiel AG" stammt aus dem Automobilbereich und beschäftigt sich mit der integrierten Softwareentwicklung (also Software, die Hardware steuert). Dabei arbeiten zehn Scrum-Teams mit insgesamt ca. siebzig Entwicklern in einem fortlaufenden Projekt. Die Budgets werden jeweils für ein Release freigegeben, die Releasezyklen betragen drei Monate. Das Produkt besteht aus Server-, Client-, Datenbank- sowie Endgerätebestandteilen und gliedert sich in fünf Teilprodukte auf, die jeweils von einem Product Owner und zwei Entwicklungsteams betreut werden. Diese fünf Product Owner kümmern sich gemeinsam mit weiteren Produktmanagern um die strategische Planung. Das Unternehmen ist noch in der Scrum-Einführungsphase und lernt jeden Sprint dazu.

Die Planungsschritte im Detail

Die oben aufgeführten Schritte sehen wir uns nun etwas genauer an. Neben dem eigentlichen Ziel jedes einzelnen Planungsschritts und der Anwendung bei der Beispiel AG betrachten wir auch die gängigsten Stolperfallen inklusive der Lösungsvorschläge.

1. Die Geschäftsführung definiert mit dem Product Owner die Produktvision

Dies geschieht normalerweise zu Projektbeginn oder die Vision ist in Weiterentwicklungsprojekten bereits vorhanden. Im Regelfall bringt die Geschäftsführung alle relevanten Stakeholder inklusive des Product Owners an einen Tisch und erarbeitet gemeinsam mit diesen eine Produktvision. Dafür müssen selbstverständlich umfassende Analyseaufgaben vorangehen, um den Vergleich mit dem Wettbewerb, Marktanforderungen, SWOT-Diagramme und ähnliches vorliegen zu haben. Diese Daten bereitzustellen ist Aufgabe des Product Owners.

Der recht hohe Aufwand für die Erstellung der Produktvision führt zu einem im Umfang kleinen, aber sehr wertvollen Ergebnis: Ein paar Zeilen Text, die eine "Produktvision für Kopf und Herz" bieten. So gibt es bei der Beispiel AG ein Teilprojekt, welches sich mit der Innenbeleuchtung der Fahrgastzelle beschäftigt. Für dieses Teilprojekt

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

könnte die Produktvision beispielsweise lauten: "Wir erleuchten unsere Kunden." Kurz, prägnant und trotzdem ein starker Bezug zum Produkt inklusive einer emotionalen Komponente.

Problematisch ist es, wenn – wie im Beispielprojekt – keine Produktvision existiert. Ebenso schädlich ist es, wenn ohne ausführliche Recherche eine Vision "aus dem Bauch heraus" definiert wird – diese ist zwar oft wohlklingend, aber leider nicht fundiert. Die gesamte strategische Planung bezieht sich dann möglicherweise auf ein falsches Ziel – bzw. bei einer fehlenden Produktvision auf gar kein Ziel. Da dieses Risiko erst langfristig eintritt, fällt es auch anfangs nicht auf. Bleiben wir beim obigen Beispiel: Wenn die Marktanalyse fehlt, entgeht der Beispiel AG möglicherweise, dass die Kunden gar nicht "erleuchtet" werden wollen und der Wettbewerb ganz andere Schwerpunkte setzt, z.B. die Konzentration auf akustische Faktoren in der Fahrgastzelle. So schön die Vision dann auch sein mag – am Ende der Entwicklung wird der Kunde das Produkt vermutlich ablehnen.

Es gibt nur einen Product Owner

Ein weiteres Problem entsteht, wenn es mehrere Product Owner für ein Produkt gibt. Das steht zwar im Widerspruch zu Scrum, ist in der Praxis aber leider häufig anzutreffen. Hier ist es wichtig, dass alle Product Owner involviert werden, d.h. an der Erstellung der Vision gleichberechtigt beteiligt sind. Im Falle eines Konflikts muss die Geschäftsleitung steuernd eingreifen.

Bei der Beispiel AG wäre das richtige Vorgehen, zunächst klare, in sich geschlossene Teilprodukte zu identifizieren und bestimmten Product Ownern zuzuweisen. Anschließend hätten alle Product Owner gemeinsam mit der Geschäftsleitung eine gemeinsame übergeordnete Produktvision und in der Folge aufeinander abgestimmte Teilproduktvisionen erarbeiten müssen. Lässt sich ein Produkt nicht in mehrere Teilprodukte aufteilen, darf auch nur ein einziger Product Owner benannt werden.

2. Der Product Owner definiert die Releaseziele zusammen mit den Stakeholdern

Ist die Produktvision vorhanden, lassen sich die Releaseziele daraus ableiten. Anders als bei der Produktvision ist hier nicht die Geschäftsführung verantwortlich, sondern der Product Owner. Er sammelt die Anforderungen seiner Stakeholder und fasst die Vorschläge zusammen. Auch bewertet er – oft gemeinsam mit den Stakeholdern – den Geschäftswert jeder Anforderung und stellt sie in den Kontext der Produktvision. In der Regel sind die Wünsche der Stakeholder zu diesem Zeitpunkt noch sehr vage und haben "Überschriftencharakter", z.B. "Innenraumregelung" oder "Kundenprofile auf der Webseite". Genau das ist es, was für Releaseziele benötigt wird.

Die Beispiel AG bringt alle Produktmanager, einige ausgewählte Kunden und sonstige Stakeholder jährlich in einem eintägigen Workshop zusammen. Darin werden die bereits während des Jahres gesammelten Anforderungen und Analysen diskutiert und am Ende eine Releasestrategie verabschiedet. Selbstverständlich haben Product Owner und Geschäftsführung schon vor diesem Workshop einen Vorschlag erstellt, der aber in der Regel während des Workshops gemeinsam verbessert wird. Das Ergebnis eines solchen Workshops kann z.B. aussehen wie in Tabelle 2.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Releaseziel (3 Monate)	Sprintziele (4 Wochen)
Innenraumregelung Stimmungsleuchten	Manuelle Helligkeitsregelung durch den Fahrer
	Automatisches Abschalten der Innenraumbeleuchtung ab 20 km/h
	Automatische Helligkeitsregelung der Innenraumbeleuchtung in Abhängigkeit vom vorhandenen Tageslicht
Regelung Sitzheizung	Manuelle Regelung der Temperatur durch den Fahrer
	Unabhängige manuelle Regelung aller beheizten Sitze
	Automatikmodus für unabhängige Regelung aller beheizten Sitze in Abhängigkeit von Nutzerpräferenzen
Regelung Belüftung	...
Scheinwerfereinstellungen	...

Tabelle 2: Release- und Sprintziele eines Workshops.

Auffällig ist, dass zwar Releaseziele für das ganze Jahr definiert wurden, nicht aber vollständige Sprintziele. Der Grund hierfür liegt darin, dass die Einigung der Stakeholder im Vordergrund steht, nicht die vollständige Definition aller Details. Der Product Owner muss wissen, wohin der Weg führt – dafür reichen die Releaseziele prinzipiell aus.

Das größte Problem bei der Erstellung der Releaseziele ist das Mikromanagement. Gute Ziele sollten konkret erreich-, aber trotzdem verhandelbar sein. Das bedeutet, dass der Weg zur Erreichung des Ziels sowie die exakte Ausgestaltung nicht durch den Product Owner vorgegeben, sondern mit dem Team zusammen erarbeitet werden sollte. Wenn man sich bei der strategischen Planung von der Vogelperspektive auf die Mikroebene begibt, sind diese Kriterien schnell nicht mehr erfüllt. Das verleitet dazu, konkrete Spezifikationen statt sinnvoller Strategien zu definieren und führt in der Folge zu einem Kollaps des gesamten nachfolgenden Planungsprozesses.

3. Der Product Owner definiert die Sprintziele zur Erreichung des aktuell anstehenden Releaseziels

Sind die Releaseziele definiert und abgestimmt, legt der Product Owner die Sprintziele für das kommende Release, manchmal auch für mehrere Releases, fest. Normalerweise werden diese Sprintziele nicht mehr mit allen Stakeholdern abgestimmt, aber zumindest an diese kommuniziert, um die größtmögliche Transparenz zu gewährleisten. Die Beispiel AG hat dreimonatige Releasezyklen und legt vor jedem Release die Sprintziele für das gesamte Release in Zusammenarbeit mit allen Product Ownern fest. Die Scrum-Teams haben unterschiedliche Sprintlängen, daher besteht ein Release aus drei bis sechs Sprints mit ihren jeweiligen Zielen.

Für Sprintziele gilt genau wie für Releaseziele, dass sie erreichbar und verhandelbar sein müssen. Allerdings sind sie wesentlich spezifischer als die übergeordneten Releaseziele. Wichtig ist, dass die Sprintziele auch dann erreichbar sind, wenn nicht alle ursprünglich eingeplanten Anforderungen im Sprint umgesetzt werden können. Bei der Beispiel AG könnte beispielsweise das Sprintziel "Manuelle Helligkeitsregelung durch den Fahrer" auch dann erreicht werden,

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

wenn die Regelung noch nicht über ein zentrales Display (über das auch die sonstigen Funktionen des Fahrzeugs gesteuert werden), sondern durch einen separaten Knopf erfolgt. Begeht man den Fehler, als Sprintziel nur die Summe der geplanten Anforderungen zu sehen, so verliert man leicht das übergeordnete Ziel aus den Augen und verschwendet die Energien seines Teams auf Arbeiten, die nur einen geringen Mehrwert zum aktuellen Zeitpunkt bieten.

4. Der Product Owner trägt die benötigten Features im Product Backlog ein

Der Product Owner trägt fortlaufend alle Anforderungen in sein Product Backlog ein. Die Anforderungen entwickelt er dann weiter, lässt sie durch das Team schätzen und ordnet sie gemäß ihrer Priorität an. Die Beispiel AG hat dazu "Estimation Meetings" eingeführt, die ausreichend lange vor jedem Sprint Planning Meeting stattfinden, um während der Schätzung auftretende Fragen noch bis zur Planung klären zu können. Diese Estimation Meetings dauern zwei bis vier Stunden. Alle Entwickler des jeweiligen Teams – also diejenigen, die später auch die Arbeit ausführen müssen – diskutieren darin mit dem Product Owner über die Anforderungen und schätzen diese relativ zueinander. Bei der Beispiel AG werden jeden Sprint genug Anforderungen diskutiert und geschätzt, um einen weiteren Sprint in die Zukunft planen zu können.

Die Anforderungen werden dabei noch nicht in kleinteilige Aufgaben (Tasks) zerlegt, sondern übergeordnet betrachtet. Ausgehend von diesen Schätzungen und seiner persönlichen Einschätzung des Geschäftswertes ordnet der Product Owner dann die Anforderungen in Eigenregie neu an. Wenn die Möglichkeit besteht, ist es am besten, den Geschäftswert durch die Stakeholder selbst bestimmen zu lassen (z.B. mit Hilfe von Entscheidungsmatrizen o.ä.).

Bei diesem Schritt ist der häufigsten Fehler – dessen Auswirkungen dann oft auch noch unterschätzt werden –, die Anforderungen weder hinsichtlich des **Aufwands** (Wie viel kostet uns das Feature?) noch hinsichtlich des **Wertes** (Wie viel bringt uns das Feature?) zu bewerten. Die Folge ist zum einen eine falsche Erwartungshaltung bei allen Projektbeteiligten im Hinblick darauf, was wann geliefert wird. Zum anderen müssen Prioritäten häufig geändert werden. Während begründete und transparente Änderungen das Wesen der Agilität darstellen, sind chaotische, nicht nachvollziehbare Sprünge für alle Beteiligten demotivierend. Das wiederum führt zu Unzufriedenheit, die im schlimmsten Fall darin mündet, dass die Stakeholder dem Projektteam das Vertrauen entziehen oder die Entwickler sogar kündigen. Natürlich kündigen nicht alle – nur die besten.

5. Der Product Owner stellt im Sprint Planning dem Team die Sprintziele vor

Das Sprint Planning besteht aus zwei Teilen: Im ersten Teil stellt der Product Owner das vor, was er in diesem Sprint umgesetzt haben möchte, im zweiten Teil erfolgt die operative Planung des Teams. Zu Beginn des Sprint Plannings rückt der Product Owner die Produktvision sowie das Releaseziel in den Fokus, um den größeren Kontext ins Bewusstsein der Entwickler zu bringen. Dies mag auf den ersten Blick übertrieben erscheinen, schließlich haben die Entwickler das Gesagte schon mehrfach gehört. Die Praxis zeigt allerdings, dass das Tagesgeschäft die Vision schnell hinter sich lässt und es deshalb sinnvoll ist, noch einmal die Vogelperspektive bewusst einzunehmen. Der Product Owner sollte nur dann darauf verzichten, wenn das Team ihn ausdrücklich darum bittet.

Nach der Präsentation der Produkt- und Releaseziele stellt er die Features vor, die er am Ende des Sprints geliefert bekommen möchte. Dabei orientiert er sich an der Velocity (Geschwindigkeit) des Teams. Unter Umständen kann es in einem solchen Planungsworkshop vorkommen, dass einzelne Anforderungen oder sogar das gesamte

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Sprintziel abgeändert werden müssen. Unabhängig davon endet die erste Hälfte des Planungsmeetings mit einer Vorhersage (Forecast) des Teams, was es glaubt, in diesem Sprint liefern zu können.

! Der Product Owner ist nicht dazu berechtigt, den Forecast des Teams zu verändern – er muss die Entscheidung der Experten akzeptieren.

Die Beispiel AG hält sich vollständig an das beschriebene Schema, jedoch gibt es keine klare Trennung zwischen den beiden Phasen des Sprint Plannings. So wird der Forecast erst am Ende der gesamten Besprechung durch das Team abgegeben. Das hat den Nachteil, dass der Product Owner während des kompletten Meetings anwesend sein muss und sich nicht anderen Tätigkeiten widmen kann. Die Qualität des Ergebnisses leidet dadurch aber nicht.

Zudem kommt es bei der Beispiel AG vor, dass eine Anforderung bereits in kleine Tasks aufgebrochen wird, während man sie noch bespricht. Der Grund dafür ist, dass viele Teammitglieder zu externen Lieferanten gehören, häufig wechseln und entsprechend unerfahren in der Applikation sind. Daher haben Sie keine Vorstellung von der Komplexität bestimmter Aufgaben, bevor sie diese in kleinere Pakete zerteilt haben. In einem solchen Fall ist diese Zerteilung für die Teammitglieder notwendig, um überhaupt eine Vorhersage abgeben zu können. Mit zunehmender Erfahrung der Teammitglieder und der Reduktion der Fremdarbeiterquote wird sich die Prognosefähigkeit zusammen mit der Produktivität weiter erhöhen.

Keinen Druck ausüben

Der größte Fehler, den der Product Owner in der ersten Hälfte des Sprint Plannings begehen kann, besteht darin, Druck auf das Entwicklungsteam auszuüben, um eine höhere Anzahl Features "versprochen" zu bekommen. Oft geschieht das, indem der Product Owner seine Enttäuschung über die niedrige Vorhersage zum Ausdruck bringt oder sogar einen Wutanfall vortäuscht. Zwar kann durch solche Maßnahmen u.U. die Menge der zugesagten Anforderungen steigen, jedoch sinkt zum einen das Vertrauen in den Product Owner und zum anderen sinkt die Motivation des Teams.

Letzten Endes ist ein solches Vorgehen kontraproduktiv: Denn eine höhere Zusage von Features führt selbstverständlich nicht dazu, dass auch mehr geliefert wird. Entsprechend verliert der Product Owner über kurz oder lang den Rückhalt bei seinen Stakeholdern, denen er mehr verspricht, als das Team halten kann. Das Vertrauensverhältnis zwischen Product Owner und Entwicklungsteam nimmt so bleibenden Schaden. Um dieses Problem nicht heraufzubeschwören, sollte der Product Owner den Aussagen der Entwickler vertrauen. Schließlich wissen sie sehr genau, was zu tun ist und wie lange sie dafür brauchen. Ohne Vertrauen funktioniert Scrum nicht!

6. Das Team plant im Sprint Planning, wie es sein Sprintziel erreichen wird

Im zweiten Teil des Sprint Plannings beginnt die operative Planung: Die Anforderungen des Product Owners, die das Team für den Sprint ausgewählt hat und die die Erreichung des Sprintziels gewährleisten sollen, werden im Detail analysiert. Das Entwicklungsteam teilt alle Anforderungen in kleine Arbeitspakete auf und schätzt diese in Stunden. Dabei ist es normal, dass auch ein Blick in den Quellcode, Spezifikationen oder andere Dokumente geworfen wird. Jeder Task sollte innerhalb eines Arbeitstags von einer Person erledigt werden können. Die so geplanten Aufgaben werden zusammen mit den vorhergesagten Anforderungen im Sprint Backlog festgehalten.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Dieses Sprint Backlog gehört allein dem Entwicklungsteam. Es ist daher auch so formuliert, dass die Entwickler, nicht aber zwingend der Product Owner oder andere Personen, verstehen, was gemeint ist. Niemand außer dem Entwicklungsteam darf Änderungen am Sprint Backlog vornehmen. Selbst der ScrumMaster fasst es in der Regel nicht an – auch nicht, um Inhalte nachzupflegen. Das Team hat nicht nur das Recht, sondern auch die Verpflichtung, sich selbst um sein Planungswerkzeug zu kümmern.

Die Beispiel AG schätzt die Tasks auf "Tagesbasis". Dieses Vorgehen stammt noch aus einer Zeit, in der auch Schätzungen von fünf Tagen und mehr für einzelne Aufgaben normal waren. Mittlerweile hat das Unternehmen seine Risiken allerdings minimiert, die Schätzungen bewegen sich im Bereich von einem Vierteltag bis hin zu zwei Tagen (in Ausnahmefällen).

Den größten Fehler, den das Team hier machen kann, ist, dass es die Aufgaben zu groß oder zu klein schneidet. Plant das Team mit zu kleinen Aufgaben, verliert es sich in einer Vielzahl von "Mini-Aufgaben" und wird einen erheblichen Teil seiner Zeit für die administrative Bearbeitung dieser Tasks aufbringen müssen. Plant das Team hingegen zu große Aufgabenpakete, geht die Transparenz verloren. So ist es bei großen Aufgaben (z.B. fünf Tage) schwer zu erkennen, dass ein Problem eingetreten ist, wenn der Bearbeiter der Aufgabe dies nicht selbst kommuniziert und die übrigen Teammitglieder erst nach Ablauf der geplanten Bearbeitungszeit nachfragen. Folglich nimmt die Fähigkeit des Teams stark ab, auf Probleme entsprechend zu reagieren. Es hat sich daher bewährt, im Sprint nur Aufgaben mit einem Umfang von zwei bis acht Stunden einzuplanen.

7. Tägliche Überprüfung der Planung

Das im Sprint Planning erstellte Sprint Backlog ist das Planungsartefakt für das Entwicklungsteam. In der Regel wird es gut sichtbar in Papierform an die Wand gehängt, damit es für alle Teammitglieder ständig im Fokus bleibt. Sobald sich Veränderungen in der operativen Planung ergeben, wird das Sprint Backlog aktualisiert.

Täglich überprüfen die Entwickler ihre Planung spätestens während des Daily Scrums, das etwa fünfzehn Minuten dauert. Der ScrumMaster unterstützt das Team dabei durch gezielte Fragen oder mit Moderationsmethoden. Zwar wird in diesem Meeting auch der Status jedes einzelnen Teammitglieds abgefragt, das Ziel ist aber die Anpassung der operativen Planung. Führt das Entwicklungsteam das Daily Scrum korrekt und täglich durch, so hat der Product Owner jeden Tag die aktuelle Information, ob die anvisierten Ziele erreicht werden können oder nicht. Treten Probleme auf, so werden diese ebenfalls spätestens im Daily Scrum identifiziert und durch den ScrumMaster gelöst.

Die Beispiel AG führt in jedem Team täglich ein fünfzehnminütiges Daily Scrum durch. Anschließend sind in jedem Team weitere fünfzehn Minuten geblockt, um den eventuell im Daily Scrum auftretenden weiteren Diskussionsbedarf zu decken. Etwa in vierzig Prozent der Fälle wird diese Zeit auch in Anspruch genommen. Zur Koordination aller Teams gibt es darüber hinaus ein zweimal wöchentlich stattfindendes "Scrum of Scrums". Vertreter aller Teams kommen dabei zusammen und berichten einander von den Bereichen, in denen sie aktuell arbeiten und von Problemen, die dabei aufgetreten sind. Soweit möglich, werden diese auch direkt gelöst.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Das Daily Scrum ist Pflicht!

Des Öfteren wird von Scrum-unerfahrenen Budgetverantwortlichen, aber auch von Entwicklern oder Product Ownern, verlangt, den "Overhead" des Daily Scrums abzuschaffen und dieses Meeting beispielsweise nur einmal pro Woche stattfinden zu lassen. Dieser Wunsch ist darauf zurückzuführen, dass das Daily Scrum häufig grundlegend missverstanden wird: Denn führt man diese Besprechung nicht durch, geht ein Großteil der Transparenz verloren und Probleme werden im schlimmsten Fall erst nach einer Woche identifiziert. Außerdem kann sich das Team nicht mehr kurzfristig abstimmen, wodurch schnell Missverständnisse und ein höherer Kommunikationsaufwand entstehen. Das "Einsparen" der Daily Scrums kostet daher unterm Strich mehr, als die tägliche Durchführung!

8. Am Ende des Sprints wird im Sprint Review das Produkt inspiziert

Das Review wird häufig als "Demo" missverstanden. Das eigentliche Ziel des Reviews sollte jedoch sein, anhand des aktuellen Produktstands weitere Verbesserungen zu planen; denn häufig weiß der Kunde erst was er will, wenn er das Produkt selbst ausprobiert hat. Das Sprint Review gibt ihm dazu die Möglichkeit und er kann zudem Verbesserungswünsche gegenüber dem Product Owner äußern. Dieser nimmt die dafür notwendigen Veränderungen im Product Backlog vor und kann so seine strategische Planung entsprechend anpassen. Deshalb ist es auch sinnvoll, dass der Product Owner zum Review das Ergebnis des Sprints bereits vorab kennt.

Die Beispiel AG hatte den Sinn des Sprint Reviews zunächst nicht verstanden und es in der Einladung sogar als "Demo" bezeichnet. Die Entwickler präsentierten dort ihre Sprintergebnisse, noch dazu häufig mit PowerPoint oder anhand von Testfällen – nicht aber an der laufenden Software. Entsprechend überflüssig war die Besprechung, die Stakeholder blieben ihr schnell fern. Kunden wurden sowieso nicht eingeladen. Ein Umdenken setzte langsam ein und erst mit der Zeit wurde den Stakeholdern bei den Reviews die laufende Software präsentiert, was letztendlich dazu führte, dass das Review produktiv zur Planung der weiteren Produktentwicklung genutzt werden konnte.

Nur fertige Features präsentieren

Neben den oben beschriebenen Fehlern kann es auch vorkommen, dass im Sprint Review unfertige Features gezeigt werden – häufig auf Wunsch der Entwickler, die Stolz auf ihre Arbeit sind. Das Problem dabei ist jedoch, dass die Review-Teilnehmer annehmen, dass alles was sie gesehen haben, auch tatsächlich "fertig" ist. Selbst wenn bei der Präsentation gesagt wird, dass es sich nur um einen Entwicklungsstand handelt, bleibt in der Regel der Eindruck haften, dass das Feature ja "eigentlich fertig" sei. Diese Annahme mündet nicht selten in großer Enttäuschung, wenn klar wird, dass das Team doch noch einen Sprint benötigt, bevor es die Features endgültig liefern kann. Unterm Strich scheint so die Entwicklungsgeschwindigkeit des Teams niedriger.

Lösen lässt sich dieses Dilemma, indem im Review wirklich nur fertige Features gezeigt werden. Möchte man dennoch unfertige Features präsentieren, empfiehlt es sich, unbedingt den Raum zu wechseln (z.B. zum Arbeitsplatz des Entwicklers), das Feature aus einer Entwicklungsumgebung zu starten und klar zu kommunizieren, dass es sich nur um einen Entwicklungsstand handelt. Noch besser ist es natürlich, auf diese verfrühte Präsentation ganz zu verzichten, was sich aber nicht immer realisieren lässt, da die Rückmeldungen der Stakeholder möglicherweise benötigt werden.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

9. In der Retrospektive wird ermittelt, wie das Team noch effizienter arbeiten kann

Auch die Sprint-Retrospektive ist ein Planungsmeeting. Sie ist ein Instrument zur kontinuierlichen Verbesserung der Zusammenarbeit im Team sowie der Rahmenbedingungen. Es geht hier also nicht um das Produkt, sondern um die Produktivität des Entwicklungsteams. Gemeinsam wird durch das gesamte Scrum-Team, also Entwickler, ScrumMaster und Product Owner, ermittelt und geplant, welche Maßnahmen bis zum Ende des nächsten Sprints getroffen werden können, um noch produktiver zu arbeiten. Diese Maßnahmen erfordern in aller Regel einen gewissen Aufwand, der wiederum vom Product Owner bei der Planung seiner Strategie berücksichtigt werden muss. Zudem fließen diese Maßnahmen auch direkt in die operative Planung des nächsten Sprints ein, was das Entwicklungsteam berücksichtigen muss. Beispielsweise könnte das Team beschließen, zur Verbesserung der Geschwindigkeit des Builds einen zusätzlichen Buildserver einzusetzen. Neben dem notwendigen Budget muss der Product Owner dann auch die Zeit für die Einrichtung eines solchen Servers durch das Team bereitstellen.

Die Beispiel AG hat dieses Werkzeug vorbildlich implementiert. Am Ende eines jeden Sprints setzen sich in allen Teams der Product Owner, die Entwickler und der ScrumMaster zusammen und diskutieren gemeinsam, welche Verhaltensweisen beibehalten und welche geändert werden sollten. Probleme, die nicht durch die Teams gelöst werden können, trägt der ScrumMaster in die Organisation.

Problematisch ist häufig, wenn die Teilnehmer der Retrospektive damit beginnen, sich gegenseitig Schuld zuzuweisen. Dieses Verhalten deutet darauf hin, dass man es eher mit einer Arbeitsgruppe und nicht mit einem echten Team zu tun hat – denn ein Team steht gemeinsam für den Erfolg seiner Arbeit ein. Lösen muss dieses Problem der ScrumMaster. Die "Prime Directive" für Retrospektiven muss durch ihn immer transparent gemacht werden: "Unabhängig davon, was wir im Verlauf der Retrospektive entdecken, wissen und glauben wir ernsthaft, dass jeder sein Bestes gegeben hat, unter der Berücksichtigung seines damaligen Wissensstands, seiner Fähigkeiten, der verfügbaren Ressourcen und der konkreten Situation."

Schlussbetrachtungen und Fazit

Planung ist wichtig und muss sorgfältig sowie mit Fachkenntnis durchgeführt werden. Geschieht das nicht, so entstehen unnötige Kosten in erheblicher Höhe und das Projekt droht zu scheitern. Sorgen Sie dafür, dass Sie entsprechend qualifizierte Spezialisten an Bord haben – sowohl Scrum-Experten als auch Entwickler und Product Owner, die ihr Handwerk verstehen. Nur so können Sie garantieren, dass Ihr agiles Projekt auch ein Erfolg wird.

Scrum ermöglicht es dem Projektteam, die Kundenwünsche noch besser zu befriedigen, da man spätestens zum Sprintwechsel auf die unvermeidlich eintretenden Veränderungen reagieren kann. Der Kunde selbst kann jederzeit mit dem Product Owner über seine veränderten Bedürfnisse sprechen. Das ist es auch, was das Agile Manifest mit "Reagieren auf Veränderungen über dem Folgen eines festgelegten Plans" meint: Der Plan ist wichtig, aber ein agiles Team ist bereit, diesen zum nächsten Sprint zu ändern, falls es erforderlich sein sollte. Auf diese Weise lässt sich Verschwendung im Bereich der Planung vermeiden, da man normalerweise nichts plant, was nicht auch umgesetzt werden soll. In traditionellen Projekten muss hingegen oft ein erheblicher Anteil der ursprünglichen Planung im Verlaufe des Projekts verändert und erneut erstellt werden.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Die jeden Sprint wiederkehrenden Planungselemente ermöglichen es, ständig ein aktualisiertes Bild an alle Stakeholder zu kommunizieren. Verschiebt sich ein Ziel oder kommen neue Prioritäten ins Spiel, so lassen sich die Auswirkungen unmittelbar und transparent vermitteln. Nutzen Sie diese Transparenz, um die Zufriedenheit Ihrer Stakeholder zu erhöhen!

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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,

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

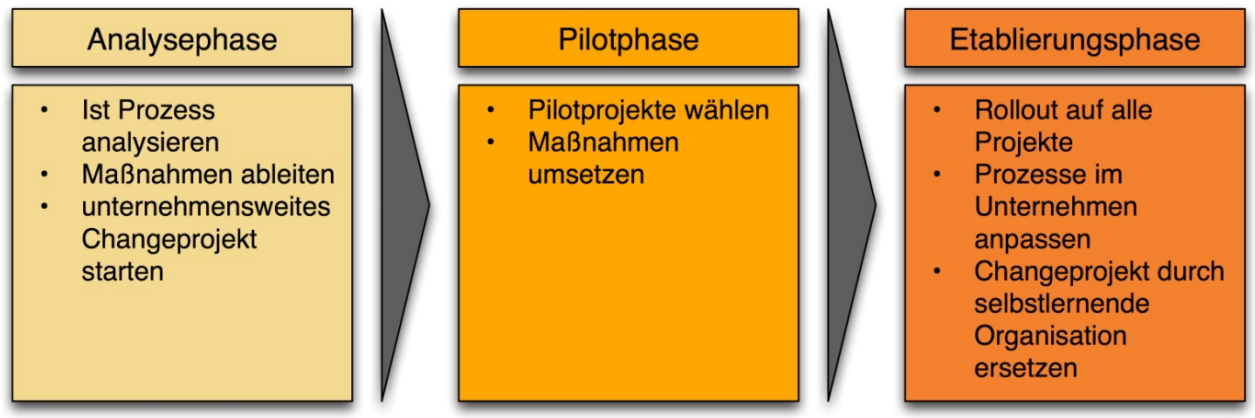


Bild 1: Phasen bei der "Einführung von oben".

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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, 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

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

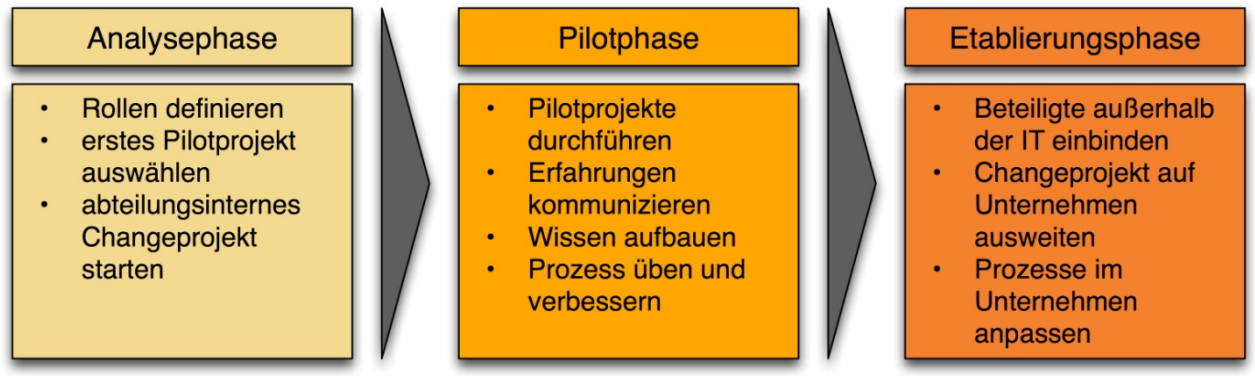


Bild 1: Phasen bei der Einführung des Scrum-Prozesses "von unten".

- **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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- 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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

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

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

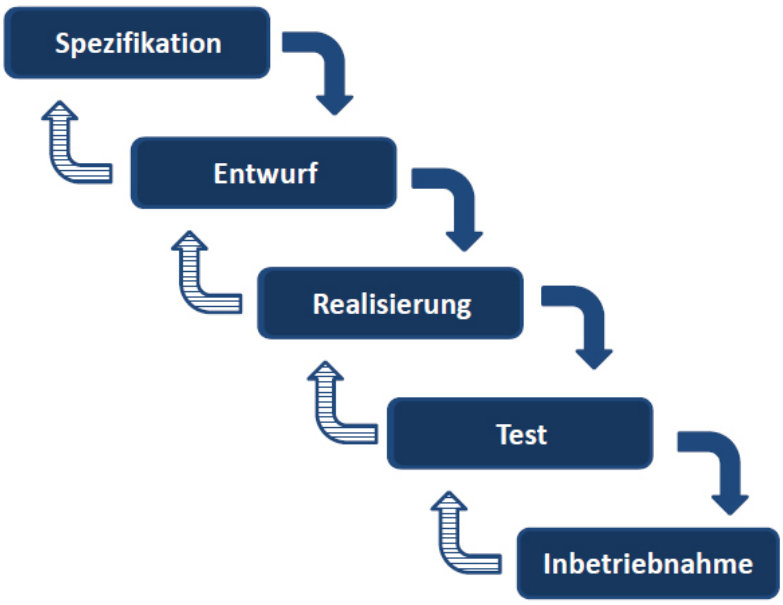


Bild 1: Schematische Darstellung des Wasserfallmodells. In der Regel erfolgt der Übergang von einer Phase in die nächste von oben nach unten. Der Weg zurück ist nur in Ausnahmefällen möglich.

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- 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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

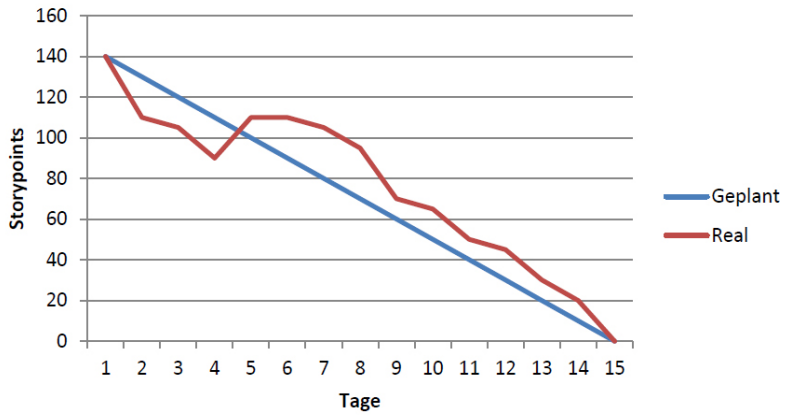


Bild 2: Typischer Verlauf eines Burndown-Charts für einen Sprint von 15 Tagen Dauer.

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

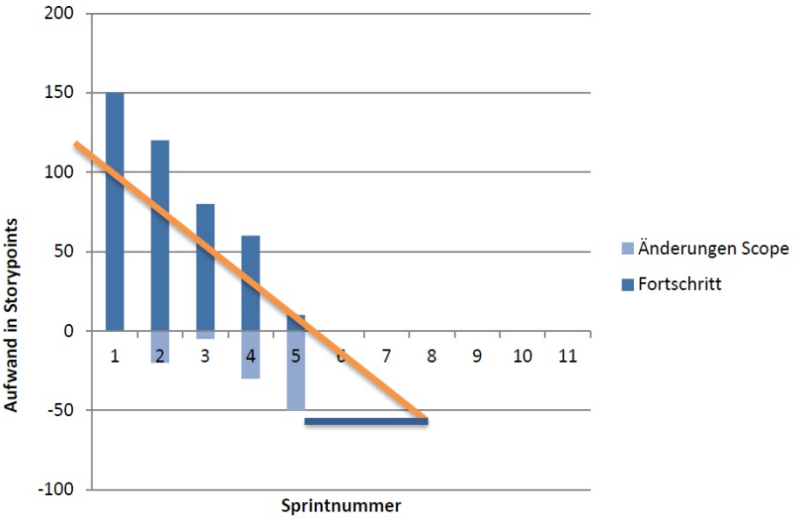


Bild 3: Angepasstes Release Burn Down Chart nach Mike Cohn.

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

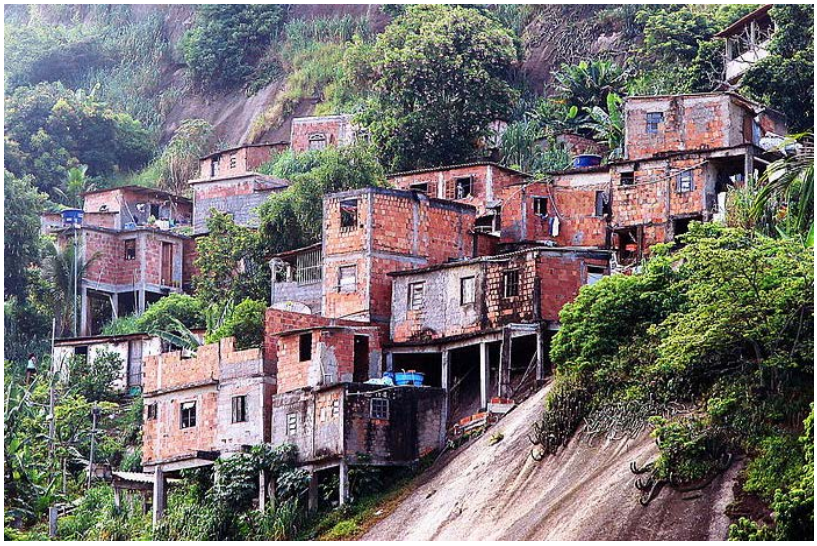


Bild 4: Die Architektur dieses Bauwerks zeigt die Vor- und Nachteile eines agilen Vorgehens: Es wurde kein Stein zu viel verbaut, die Statik setzt zukünftigen Erweiterungen allerdings enge Grenzen (Quelle: Eurico Zimbres, http://de.wikipedia.org/wiki/Datei:Favela-Niteroi.JPG)

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

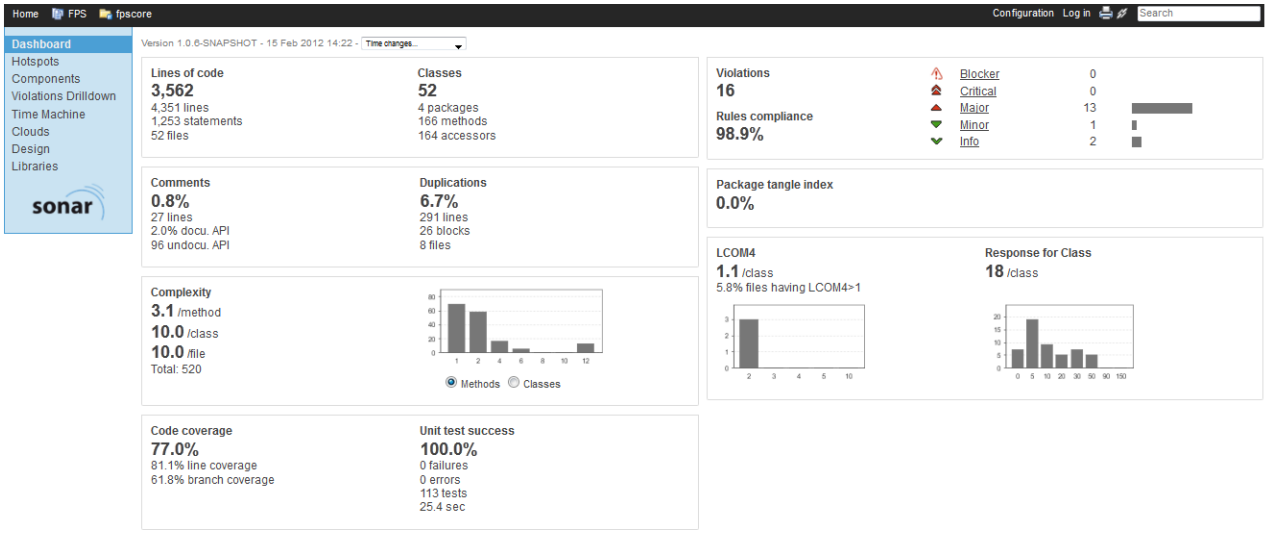


Bild 5: Sonar Dashboard zur Visualisierung von Softwarequalität. Auch Nichtprogrammierer können hier auf einen Blick erkennen, wie sich die Softwarequalität entwickelt hat.

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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).

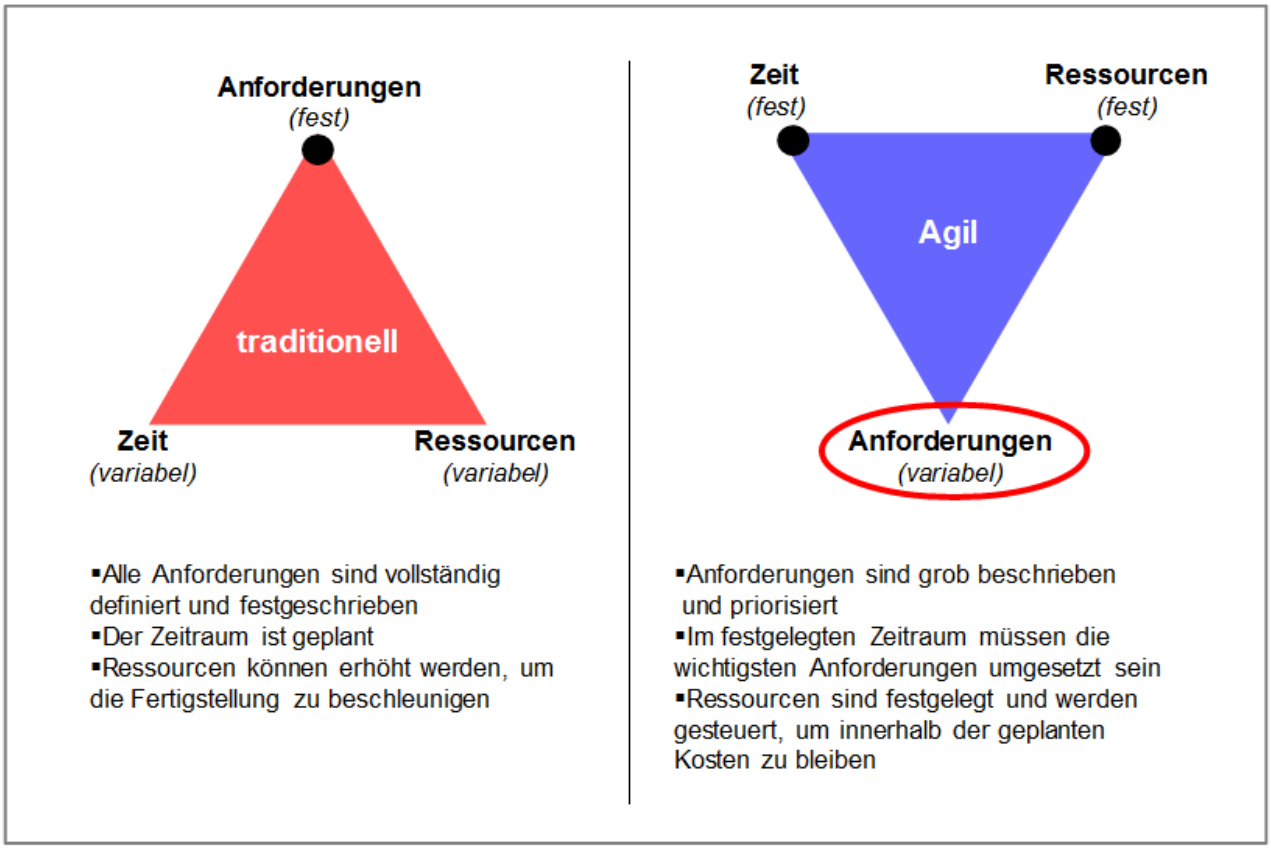


Bild 1: Das "Dreieck des Projektmanagements" für traditionelle und agile Verfahren.

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

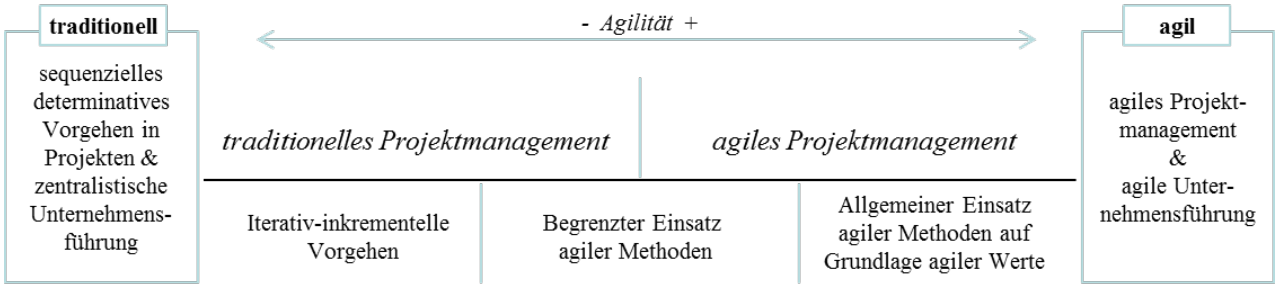


Bild 2: Auch in traditionellen Umgebungen können agile Methoden eingesetzt werden.

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

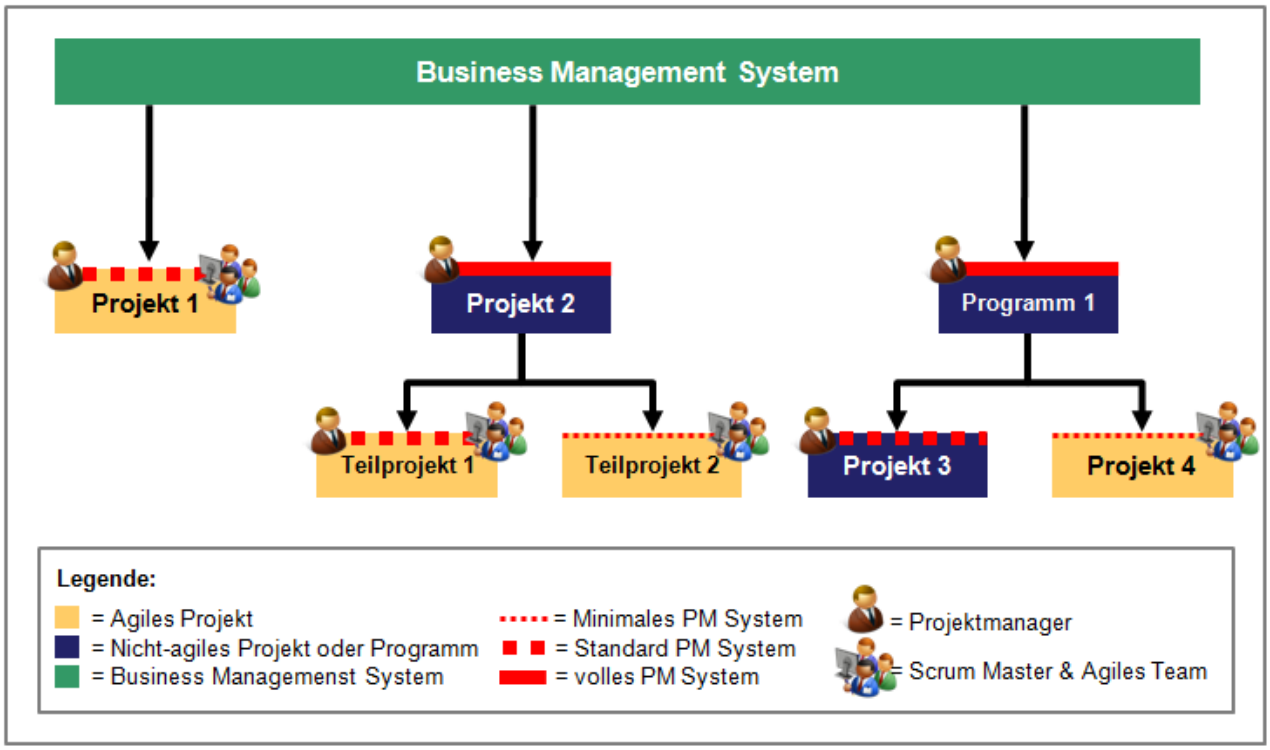


Bild 3: Koexistenz verschiedener Projektvorgehensweisen.

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

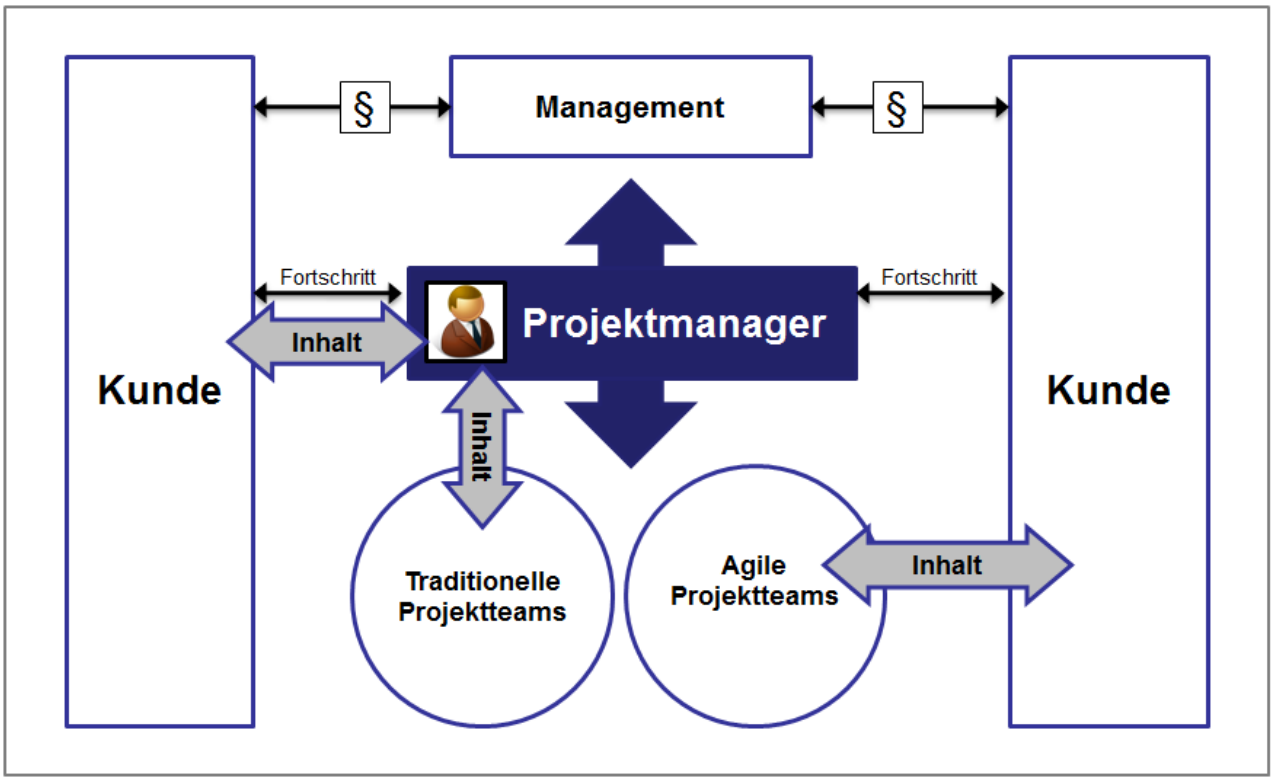


Bild 4: Der traditionelle Projektmanager ist nach wie vor die zentrale Instanz im Projektumfeld.

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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 Steuerungsmechanismen 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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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 detailliert 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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

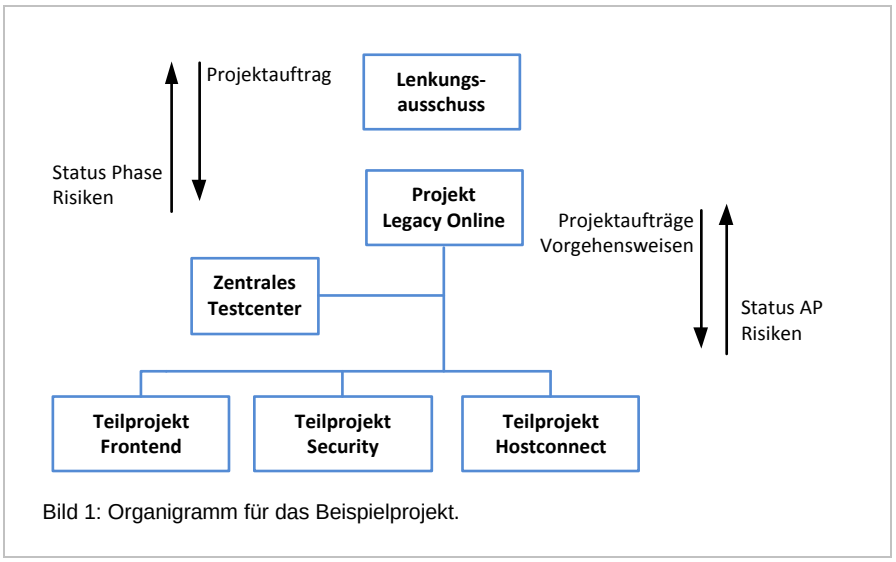


Bild 1: Organigramm für das Beispielprojekt.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

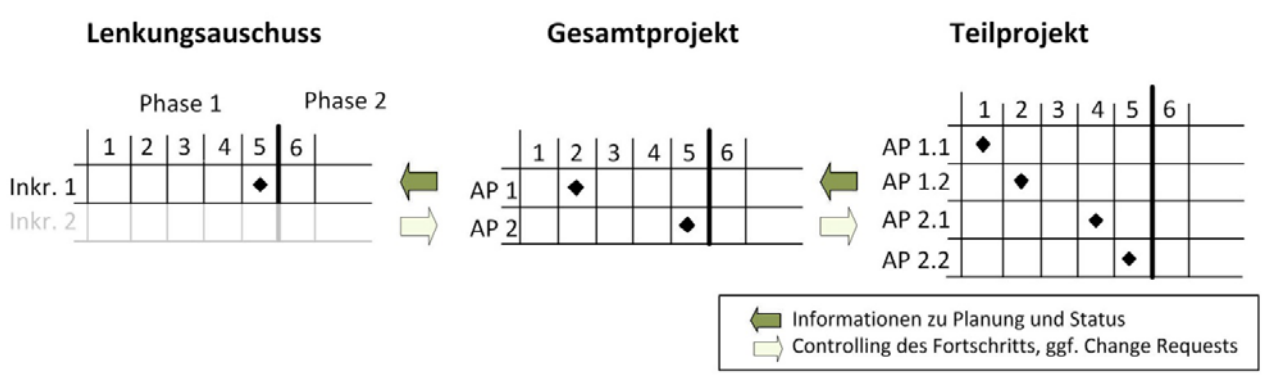


Bild 2: Verfeinerung der Aufgaben in der mehrstufigen Projekthierarchie.

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"

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

	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-

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- 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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Agiles Projektmanagement

Häufige Stolperfallen in Scrum

Scrum ist ein agiles Projektmanagement-Framework, das nur wenige Rollen Besprechungen und Berichte verpflichtend vorgibt. Anstatt viele Prozesse und Dokumente im Detail vorzuschreiben, setzen agile Methoden darauf, das richtige Maß an Prozess und Formalität zwischen allen Beteiligten iterativ auszuhandeln: Der Schwerpunkt liegt dabei auf der Kommunikation.

Bild 1 zeigt das Prozessbild von Scrum. Eine detaillierte Beschreibung der Rollen und Abläufe in Scrum finden Sie in dem Artikel "Agiles Projektmanagement. Scrum – eine Einführung" von Ralf Wirdemann (Projekt Magazin 21/2009).

Was ein erfolgreiches Scrum-Team ausmacht

Auf den ersten Blick scheint Scrum als Software-Entwicklungsprozess bestechend einfach: Es gibt nur einige wenige Rollen, eine überschaubare Menge an Regeln und so gut wie keine zu erstellenden Dokumente. In der Praxis zeigt sich jedoch immer wieder, dass eine erfolgreiche Umsetzung von Scrum nicht trivial ist. Gerade weil die Methode als solche wenig formell ist und damit kaum Vorgaben macht, hängt der Erfolg wesentlich von den beteiligten Personen ab.

Das perfekte Team ...

Ein erfolgreiches Team zeichnet sich dabei vor allem durch eine hohe Lernbereitschaft, das Bestreben kontinuierlich besser werden zu wollen und eine offene Feedback-Kultur aus. Einem solchen Team gelingt es, am Ende jeder Iteration eine potentiell releasefähige Version des Produkts auszuliefern. Dieses Zwischenergebnis, das sogenannte "Produkt-Inkrement", ist dabei implementiert, integriert, dokumentiert und erfolgreich getestet. Ein perfektes Team ist außerdem eingebettet in eine Umgebung, die den Prozess unterstützt, fördert und die damit verbundenen agilen Werte lebt.

... und die Wirklichkeit

Doch leider ist nicht jedes Team erfolgreich und die Ursachen sind häufig auch erst auf den zweiten Blick zu erkennen. Bild 2 ergänzt das Scrum-Prozessbild um einige der häufigsten Probleme. Diese Stolperfallen werden im Folgenden beschrieben, die Ursachen analysiert und Wege zur Verbesserung der Anwendung von Scrum ge-

Autor

Thomas Lieder

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

Kontakt: thomas.lieder@setzwein.com

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 der Rubrik:

› Agiles Projektmanagement

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

zeigt. Natürlich ersetzt dies nicht die detaillierte Analyse der jeweils konkret vorliegenden Situation, kann aber erste Hinweise auf potentielle Probleme und ihre Lösung geben.

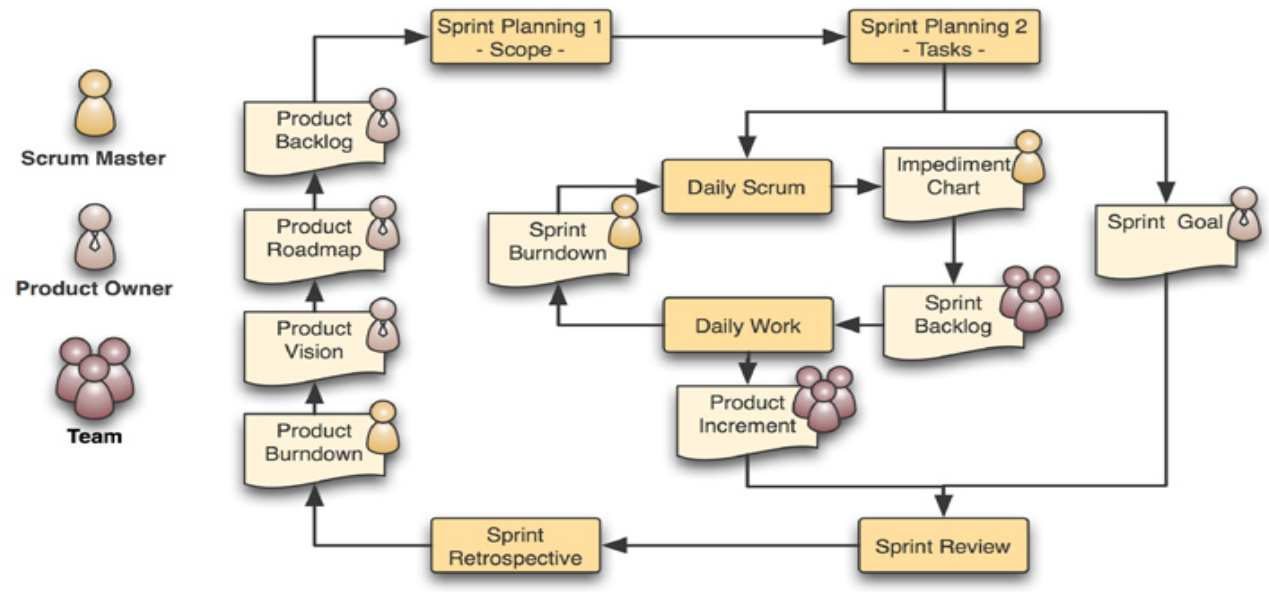


Bild 1: Scrum-Prozessbild.

Das Rollenverständnis des Scrum Masters

In einem Scrum-Team ist der Scrum Master für die Einhaltung des Prozesses verantwortlich. Er stellt sicher, dass sich das Team ungestört auf die Erstellung des Produkt-Inkrements konzentrieren kann. Nicht selten gibt es Probleme an dieser zentralen Stelle: Zum einen mit dem Rollenverständnis des Scrum Masters selbst, zum anderen mit den Aufgaben, die er aufgrund seiner Rolle durchführen muss.

Direkte Verstöße gegen die Rollendefinition – z.B. das Einmischen in den Sprint durch die Verteilung neuer, für den Sprint nicht eingeplanter Aufgaben – lassen sich in der Regel einfach feststellen und durch entsprechende Schulungen beheben. Es gibt aber auch Verhaltensweisen eines Scrum Masters, deren negative Wirkung sich eher indirekt zeigt, da er sich vordergründig entsprechend der Rollen-Definition verhält. Sie äußern sich z.B. darin, dass das Team in der Retrospektive wiederholt dieselben Probleme schildert oder dass aufgetretene Störungen, sog. "Impediments", vom Scrum Master nicht beseitigt werden. Das "Impediment Chart", d.h. die Störungsliste, wird infolge dessen immer länger.

Diese Symptome sind unter anderem dann zu beobachten, wenn der Scrum Master gegenüber der Organisation nicht durchsetzungsfähig ist. Sind die Führungsqualitäten des Scrum Masters nicht ausreichend ausgeprägt, kann er z.B. die in der Organisation liegenden Störungen nicht beseitigen. In einem solchen Fall hilft ein Coaching des Scrum Masters, sowohl bezüglich seiner Rolle als auch bezogen auf Mittel und Wege, Dritte zur Mitarbeit und damit zur Beseitigung der Impediments zu bewegen. Im einfachsten Fall mag es reichen, konstruktive Vorschläge

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

zur Lösung eines Problems zu machen, z.B. feste Sprechzeiten vorzuschlagen, wenn ein Mitglied des Scrum-Teams auch als Know-how-Träger für andere benötigt wird. In anderen Fällen, z.B. wenn zugesagte Ressourcen nicht zur Verfügung gestellt werden, ist vielleicht eine konstruktive Eskalation im Lenkungsausschuss notwendig.

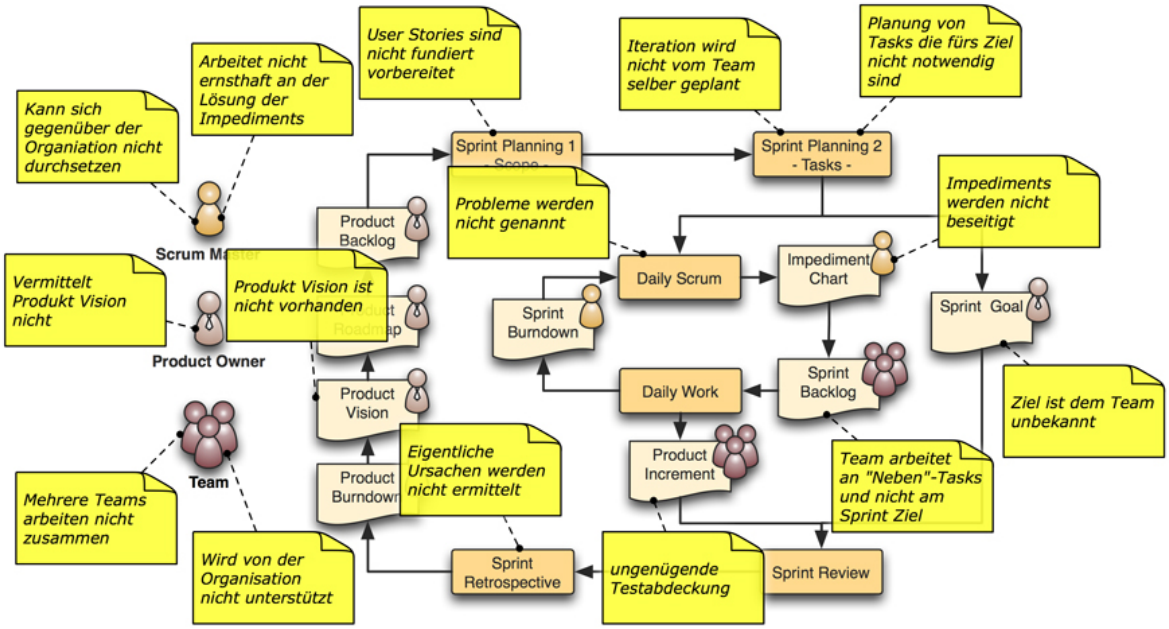


Bild 2: Einige der typischen Probleme bei der Umsetzung von Scrum.

Die ganze Organisation muss dahinter stehen

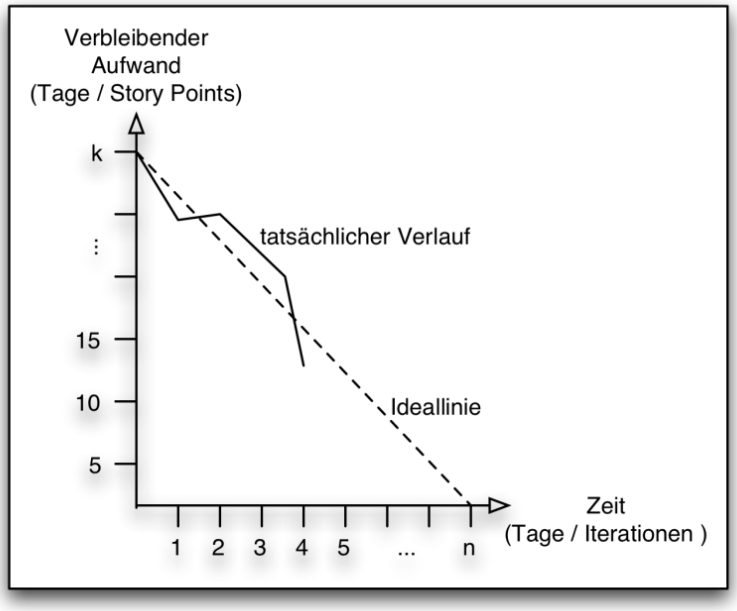
Sollte die Organisation, in der das Projekt durchgeführt wird, insgesamt jedoch nicht hinter der Methode Scrum stehen und die Einführung eher kritisch sehen, lassen sich die Impediments alleine durch den Scrum Master häufig nicht lösen. Hier gilt es, wie bei jedem Change-Projekt, Aufklärungsarbeit zu leisten, die Organisation von dem Vorgehen Scrum zu überzeugen und eine für Scrum funktionierende Umgebung zu schaffen.

Diese Aufklärungsarbeit sollte dabei mindestens von der Managementebene ausgehen, die das Vorgehen für das Projekt ausgewählt hat. Je höher in der Hierarchie die Förderer angesiedelt sind, desto höher ist die Wahrscheinlichkeit, Neuerungen in der Organisation anstoßen zu können. Es gilt natürlich wie bei jeder Änderung der Organisation, dass im Zweifelsfall auch Rückendeckung durch ein Mitglied des oberen Managements oder der Geschäftsleitung notwendig ist.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Die Charts im Auge behalten

Doch auch, wenn die Organisation dem Vorgehen positiv gegenübersteht, kann es zu einem Anwachsen des Impediment Charts kommen. Dies ist insbesondere dann der Fall, wenn der Scrum Master die in Retrospektive und Daily Scrum genannten Störungen für nicht so wichtig hält und sich nicht ernsthaft um deren Beseitigung kümmert. Dies geschieht häufig dann, wenn der Scrum Master auch noch andere Aufgaben wahrnimmt, etwa wenn er zur Hälfte Scrum Master und zur anderen Hälfte Entwickler ist. Hier gilt es, das Rollenverständnis des Scrum Masters zu schärfen. Nur so lassen sich Impediments auch wirklich beseitigen und das Vertrauen in den Scrum Master und die Methode aufrecht erhalten.



Burndown Charts

Burndown Charts zeigen den Restaufwand für einen Sprint (Sprint Burndown Chart) oder ein Release (Release Burndown Chart) im Zeitverlauf.

Sprint Burndown Chart

Die vertikale Achse zeigt den Restaufwand gemessen in Tagen oder Stunden im aktuellen Sprint. Die horizontale Achse stellt die Tage des Sprints dar.

Release Burndown Chart

Die vertikale Achse stellt den Restaufwand gemessen in Story Points zu Beginn jedes Sprints dar. Die horizontale Achse zeigt die Sprints eines Release.

Jeder Scrum Master sollte als höchstes Ziel verfolgen, eine ideale Arbeitsumgebung für das Team herzustellen. Ein anwachsendes Impediment Chart oder wiederkehrende Impediments sind ein deutliches Zeichen dafür, dass der Scrum Master seiner Rolle nicht gerecht wird. Ob dies am Scrum Master selbst oder an äußeren Einflussfaktoren liegt, lässt sich nur in direkten Gesprächen oder durch einen externen Scrum Coach in einem Prozess-Review ermitteln. Um solche Probleme zu erkennen, ist es sinnvoll, im Statusbericht eines agilen Projekts neben dem Burndown Chart (s. Kasten) auch immer das Impediment Chart und die gerade dringendsten Probleme zu betrachten.

Mit fundierten Retrospektiven Problemursachen beheben

Gleichermaßen problematisch ist eine nicht fundiert durchgeführte Retrospektive. In einer solchen schildert das Team Störungen, die nur Symptome des eigentlichen Problems sind. Die wahren Ursachen werden vom Scrum Master jedoch nicht herausgearbeitet. Ein Beispiel hierfür wäre das wiederholte Verfehlen des Sprint-Ziels auf-

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

grund von nicht-geschätztem Mehraufwand: Zur Lösung dieses Problems nimmt sich das Team dann vor, beim nächsten Mal besser zu schätzen, um so ein realistisches Commitment abgeben zu können. Die wirklichen Ursachen für das Verfehlen des Sprint-Ziels, beispielsweise Probleme im Umgang mit der Architektur oder fehlendes Fachwissen, werden aber nicht erkannt und daher auch nicht gelöst.

Somit werden lediglich die Symptome kuriert. Die eigentlichen Probleme verlagern sich möglicherweise, werden aber nicht beseitigt. Zur Lösung dieser Situation bietet sich ein Coaching des Scrum Masters in Bezug auf Moderations- und Fragetechniken an. Durch den gezielten Aufbau der entsprechenden Soft Skills erkennt der Scrum Master die wirklichen Ursachen eher und kann die Probleme dann auch lösen. Ein Training in der Methode Scrum alleine hilft in diesem Fall hingegen nicht.

Herausforderung "Commitment im Team"

Scrum basiert auf der Idee, dass sich ein Team pro Iteration auf die Umsetzung einer Reihe von Anforderungen verpflichtet. Das Team arbeitet dabei selbstorganisiert: Es plant sein Arbeitsvolumen für jede Iteration selbst und ist daher auch für die Erreichung des selbstgesteckten Ziels verantwortlich.

Ein fehlendes Commitment im Team in Bezug auf das Sprint-Ziel äußert sich u.a. darin, dass sich das Team ablenken lässt und sich nicht auf die Bearbeitung der Aufgaben auf dem Taskboard fokussiert. "Nebenkriegsschauplätze" sind wichtiger als die Realisierung dessen, was angefordert wurde. Architektur- bzw. Technologiefragen haben in solchen Situationen zumindest für Teile des Teams eine größere Bedeutung als die Umsetzung der Anforderungen, auf die sich das Team im Sprint Planning verpflichtet hat. Das KISS-Prinzip ("Keep it small and simple") wird dabei gerne ignoriert, sodass folglich Zeit mit Lösungen verbracht wird, die so nicht angefordert waren. Dies alles führt in Summe dazu, dass das Sprint Burndown Chart keinen wirklichen Fortschritt zeigt und sich der Restaufwand nicht weiter verringert.

Die Produkt-Vision im Team kommunizieren

Die Ursachen für ein fehlendes Commitment im Team sind vielschichtig. Möglicherweise gelingt es dem Product Owner nicht, die Produktvision zu verkaufen. Das Team identifiziert sich dann nicht mit dem Produkt und arbeitet wenig begeistert an den anstehenden Aufgaben. Die Erfahrung zeigt, dass ein Team dann besonders motiviert ist, wenn ihm zum einen die Vision bekannt ist und zum anderen transparent wird, wie die anstehenden User Stories zum Erreichen der Visionsziele beitragen. Dies ist in erster Linie eine kommunikative Aufgabe des Product Owners. Sollte er sich nicht in der Lage sehen, die Vision zu vermitteln, etwa weil eine solche nicht explizit formuliert wurde, muss dies nachgeholt werden. Ansonsten wäre es denkbar, dass jedes Teammitglied seine eigene Vorstellung von dem entwickelt, was umgesetzt werden soll.

Eine gute Produktvision grenzt dabei das Produkt ab, ohne es zu spezifizieren. Ein Beispiel: "Das zu entwickelnde Produkt ist eine Plattform für den Online-Aktienhandel, die den An- und Verkauf von Aktien ermöglicht. Mindestens 100 Benutzer sollen gleichzeitig über die Plattform arbeiten können." Mit dieser Vision weiß jedes Teammitglied, worum es geht. Jeder kann abschätzen, was für die Entwicklung relevant ist, im Beispiel also der An- und Verkauf von Aktien. Gleichermaßen gibt die Vision vor, was nicht wichtig ist, im Beispiel etwa der An- und Verkauf

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

von Fonds. Ohne eine solche Vision und damit ohne ein konkretes Ziel vor Augen sollte kein Projekt – und damit auch kein agiles Projekt – durchgeführt werden.

Iterations-Planung – nicht ohne das Team

In der Iterations-Planung definieren die Beteiligten den Umfang der Iterationen und gliedern die User Stories in Tasks. Geben Scrum Master oder Product Owner die Planung vor, ist eine Identifikation des Teams mit der Iteration ebenfalls unwahrscheinlich. Die Motivation des Teams leidet in diesem Fall, da es nicht in die Planung mit einbezogen wurde. Infolge dessen wird sich das Team als vorrangiges Ziel die Bearbeitung der vorgegebenen Tasks setzen, sich jedoch nicht einer qualitativ akzeptablen Umsetzung der zugrunde liegenden User Stories verbunden fühlen. Im schlimmsten Fall sind dem Team die eigentlichen Ziele unbekannt, so dass es Fehler oder Auslassungen in den Tasks nicht einmal erkennen könnte.

In einer solchen Konstellation lässt man das Wissen und das Engagement des Teams mindestens teilweise ungenutzt. Vorrangig ist deshalb dafür zu sorgen, dass das Team die Iterations-Planung eigenverantwortlich durchführt. Ist dies nicht der Fall, weil sich Scrum Master oder der Product Owner nicht an ihre Rollenvorgabe gehalten haben, bieten sich hier entsprechende Schulungen mit hohem Praxisanteil an, in denen insbesondere die Scrum-Rollen und deren Aufgaben geübt werden.

Das Team als Ganzes einschwören

Es kommt auch vor, dass einzelne Team-Mitglieder ihre eigenen Interessen verfolgen und sich daher nicht der Umsetzung der vom Product Owner vorgestellten Anforderungen verpflichtet fühlen. Es besteht dann die große Gefahr, dass Aufgaben im Sprint Planning identifiziert werden, die zwar das Teammitglied, nicht aber das Projekt weiterbringen. Ein Beispiel hierfür wäre die Umsetzung einer Anforderung mit einer neuen Technologie, obwohl die Umsetzung auch mit den vorhandenen Bordmitteln möglich wäre. In einem solchen Fall kann es ratsam sein, Maßnahmen zur Teambildung durchzuführen, um das Team als Ganzes auf das Projekt und die Vision einzuschwören. Dabei empfiehlt es sich, die gewohnte Arbeitsumgebung zu verlassen.

In der Praxis hat es sich beispielsweise bewährt, das Team zunächst spielerisch auf die Vision einzustimmen, etwa durch das gemeinsame Malen eines Bildes zur Visualisierung der Vision. Gleichermaßen sollte ein solcher Workshop Elemente integrieren, die etwaigen Differenzen im Team entgegenwirken. Hier könnte man unter anderem mit einem paradoxen Ansatz arbeiten und dem Team die Frage stellen, was geschehen müsste, damit die Situation noch schlimmer wird. Auf diese Weise gelingt es die Atmosphäre zu lockern, um dann konstruktiv Verbesserungsmaßnahmen abzuleiten. Bleibt der Erfolg dieser teambildenden Maßnahmen aus, sollte man das Team verstärken oder einige Teammitglieder austauschen.

Unterstützung durch das Umfeld

Wird einem Team die erforderliche Unterstützung seitens der Organisation verweigert, fühlt es sich im Stich gelassen und es lässt sich ebenfalls kein echtes Commitment erzielen. Damit ein Team motiviert arbeiten kann,

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

muss es die Unterstützung seines Umfeldes erfahren. Führt jedoch jeder Wunsch zu Blockaden und kann kein Impediment gelöst werden, entsteht beim Team der Eindruck, dass das Projekt und seine Ziele unwichtig sind. Wenn wir davon ausgehen, dass kein Unternehmen unwichtige Projekte durchführt, handelt es sich hierbei also in erster Linie um eine Aufgabe des Scrum Masters: Er muss gegenüber der Organisation verdeutlichen, wie wichtig es ist, die Impediments zu beseitigen.

Das Team muss außerdem die Unternehmensstrategie kennen und das Projekt als einen Teil davon betrachten, um motiviert arbeiten zu können. Es ist daher empfehlenswert, dem Team von vornherein die Bedeutung des Projekts auch im Hinblick auf die Unternehmensstrategie klar zu machen. Aus der Einbettung des Projekts in die Strategie folgt im Umkehrschluss, dass das Team entsprechende Unterstützung durch die Organisation tatsächlich erfahren muss: Wichtige Impediments müssen auch tatsächlich aufgelöst werden.

Fehlerhafte Schätzung mit Planning Poker

Eine besondere Herausforderung in Scrum sind die Schätz-Workshops, in denen das Team mit dem sogenannten Planning Poker die Größe von User Stories schätzt. Sicheres Zeichen für eine nicht-optimale Scrum-Umsetzung ist unter anderem eine lange Dauer solcher Planungssitzungen. Denn eine gut vorbereitete Sitzung zeichnet sich dadurch aus, dass die User Stories hinreichend detailliert vorgestellt werden und dem Team die Gesamtfachlichkeit und die Systemarchitektur bewusst sind. Nachfragen beziehen sich nur auf Details und können schnell geklärt werden. Entstehen aber in der Planungssitzung lang andauernde Diskussionen um Inhalt oder Umsetzung einer User Story, so ist die Story entweder schlecht vorbereitet oder das Wissen über das System im Team nicht gleichmäßig verteilt. Beides sollte in einem funktionierenden Scrum-Umfeld nicht vorkommen.

Fehlerhafte Schätzungen aufgrund falsch eingeschätzter Komplexität sind ebenso ein Symptom für eine schlecht durchgeführte Schätzklausur. Die Velocity, also die Geschwindigkeit des Teams, ist definiert durch die Anzahl der Story Points, die ein Team pro Sprint umsetzen kann (näheres zu Velocity und Story Points erfahren Sie in dem Artikel "Agile Softwareentwicklung mit Scrum und User Stories", Projekt Magazin 2/2010). Weichen erwartete und tatsächliche Velocity voneinander ab, ist das ein weiterer Hinweis darauf, dass das Team die Komplexität der Anforderungen nicht richtig eingeschätzt hat.

Planning Poker

Mit Planning Poker wird die Größe von User Stories relativ zueinander in Story Points geschätzt.

Die Regeln

Jedes Teammitglied bekommt einen Kartenstapel, wobei die Kartenwerte einer unreinen Fibonacci-Folge (0, 1, 2, 4, 8, 13, 20, 40, 100) entsprechen. Zu Beginn wählen die Spieler die User Story aus, die eine der kleinsten aus dem Product Backlog ist und weisen dieser 2 Story Points zu. Diese User Story ist fortan die Referenz-User-Story und ist bei jeder Schätzung zu berücksichtigen. Anschließend werden die übrigen User Stories nacheinander vorgestellt. Die Teilnehmer schätzen die Größe durch Auswahl der Karte, die dem Aufwand entspricht, der in dieser User Story relativ zur Referenz-Story vermutet wird. Die Karten werden dabei solange verdeckt gehalten bis sich jeder Spieler für eine Karte entschieden hat. Nach dem Offenlegen der Karten erläutern die Spieler mit der höchsten und der niedrigsten Schätzung ihre Wahl. Nach einer kurzen Diskussion findet eine erneute Schätzrunde statt. Es folgen weitere Diskussions- und Schätzrunden bis ein Konsens gefunden ist.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Story Points richtig verstehen

Die Schwierigkeiten mit dem Planning Poker haben einerseits technische Ursachen, können aber andererseits auch fachlicher Natur sein. So ist bei der Durchführung des Planning Pokers unbedingt darauf zu achten, dass die Maßeinheit Story Point richtig verstanden wird. Mit dieser Einheit schätzt man die Größe der User Stories und zwar nicht absolut, sondern relativ zueinander. Außerdem muss jeder Schätzende die gemeinsame Referenz-Story kennen und bei seiner Schätzung berücksichtigen.

Aus fachlicher Sicht scheitert das Planning Poker unter anderem dann, wenn es keine Teammitglieder gibt, die sich im Fachgebiet auskennen und somit die Komplexität der zu schätzenden Anforderungen zuverlässig beurteilen können. Infolge dessen wird die Schätzung zwangsläufig fehlerhaft sein.

Erfolgreiches Schätzen nur mit guter Vorbereitung

Aber auch wenn Fach-Experten im Team sind, hängt der Erfolg einer Planungssitzung von der Vorbereitung durch den Product Owner ab. Dieser muss die User Stories so formulieren, dass sie spezifisch, messbar, attraktiv, rational und testbar sind (in Anlehnung an das SMART-Prinzip). Zur Sicherstellung der Testbarkeit bietet sich die Formulierung von Akzeptanzkriterien an. Diese beschreiben anhand konkreter Beispiele, unter welchen Bedingungen eine User Story als erfolgreich umgesetzt gilt.

Es ist nicht zielführend, mit einer nur vagen Idee eines Features in diese Sitzung zu gehen und dort die Details gemeinsam zu erarbeiten. Das Team kann die Komplexität einer User Story besser einschätzen, wenn die Story vom Product Owner bereits zusammen mit Abnahmekriterien und Testfällen präsentiert wird. Dies hilft auch, um die Auswirkungen einer Änderung auf andere betroffene Systemteile zu analysieren. Workshops zur detaillierten Ausarbeitung von Features sind durchaus sinnvoll, sollten aber unabhängig von und vor der Planning-Poker-Sitzung durchgeführt werden.

Fachwissen ist unerlässlich

Um den Planungs-Workshop technisch korrekt durchzuführen, bietet sich ein Training sowohl des Product Owners als auch des Teams an. Hat ein neu zusammengestelltes Team noch kein Fachwissen aufgebaut und muss es eine Referenz-User-Story erst noch definieren, bietet es sich an, eine Exploration durchzuführen. In einer solchen Phase werden zur Klärung grundlegender Fragen einige wenige sehr kurze Iterationen durchgeführt. Die Exploration schließt mit einer ausführlichen Retrospektive, um eine Basis für dann folgende reguläre Iterationen zu bilden. Im Rahmen einer Exploration könnte z.B. eine Funktion, die vom neuen System bis in das anzubindende Altsystem reicht, implementiert werden.

In manchen Fällen, z.B. wenn das Team insgesamt unerfahren in der anstehenden Fachlichkeit ist, kann es auch hilfreich sein, externe Fachexperten hinzuzuziehen. Im Falle einer veränderten Team-Zusammensetzung sollte die Referenz-User-Story neu definiert werden. Hier kann es sinnvoll sein, diese von einem Kern-Team, das aus bereits erfahrenen Entwicklern besteht, bestimmen zu lassen. Sie kennen sich im Fachgebiet bereits aus und

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

müssten die Komplexität der Anforderungen einschätzen können. Die so definierte User Story mit den daraus abgeleiteten Tasks muss aber allen Teammitgliedern bekannt und die damit verbundene Komplexität verstanden sein.

Fehlende Testabdeckung und Testautomatisierung

Will man Scrum erfolgreich anwenden, muss man auf eine ausreichend hohe Testabdeckung achten. Nur so lassen sich die kontinuierlichen Änderungen und Erweiterungen am System bei einer gleichbleibend hohen Qualität durchführen.

Es ist denkbar, dass eine bereits lauffähige Funktionalität durch Weiterentwicklungen wieder fehlerhaft wird. Bei ungenügender Testabdeckung kann es dann passieren, dass diese Fehler nicht rechtzeitig erkannt werden und in den produktiven Betrieb gehen. Um nach jeder Iteration eine releasefähige Version des Produkts bereitstellen zu können, sollte die Testabdeckung kontinuierlich überprüft und insbesondere die bereits lauffähige Funktionalität früherer Iterationen durch entsprechende Tests abgesichert werden.

Scheiternde Reviews als Warnung verstehen

Ein weiteres Signal dafür, dass in die Testabdeckung investiert werden sollte, sind regelmäßig scheiternde Reviews, in denen eigentlich die umgesetzten Anforderungen abgenommen werden sollen. Denn nur mit einer guten Testabdeckung lässt sich sicherstellen, dass auch kurz vor der Abnahme angepasster Code lauffähig ist. Dabei ist sowohl auf eine ausreichend hohe Abdeckung der durch die Entwickler erstellten Unit-Tests zu achten, als auch auf eine kontinuierliche Weiterentwicklung einer automatisierten Regressions-Testsuite. Unter einer Regressions-Testsuite versteht man in der Softwareentwicklung die regelmäßige Wiederholung aller oder einiger Testfälle, um durch Code-Korrekturen entstandene Fehler in bereits getesteten Softwareteilen zu entdecken.

Eine kontinuierlich abnehmende Velocity des Teams weist darauf hin, dass es Probleme mit der Wartbarkeit des produzierten Codes gibt. Um eine gute Velocity zu erhalten, ist es erforderlich, die Software kontinuierlich zu refaktorisieren (Restrukturierung des Quellcodes). Ein Team wird sich vorzugsweise dann trauen, Überarbeitungen und Änderungen durchzuführen, wenn die Funktionalität mit automatisierten Tests abgesichert ist. Eine ungenügende Automatisierung der Tests kann auch Inkonsistenzen in der Versionsverwaltung nach sich ziehen, da Fehler nicht schnell genug auffallen. Oft ist es dann nicht mehr möglich, einen fehlerfreien Code aus der Versionsverwaltung zu exportieren und eine lauffähige Version der Software zu erstellen.

Automatische Tests anstelle überflüssiger Features

Die einfachste Ursache einer mangelnden Testabdeckung ist das Fehlen einer Verpflichtung zur Erstellung von Tests. Die Lösung ist in diesem Fall, dass das Scrum-Team die Definition of Done anpasst. Dabei handelt es sich in der agilen Softwareentwicklung um eine Checkliste, die definiert, wann fertig wirklich fertig ist. Dazu gehört typischerweise nicht nur die lauffähige Software, sondern auch die Erstellung der entsprechenden Dokumentation sowie die verpflichtende Erstellung von Tests. Diese Definition of Done erstellt das Team für sich selbst und wird daher auch vom Team gepflegt. Auch wenn mehrere Teams parallel arbeiten, sollten teamspezifische Definitionen die Ausnahme sein.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Eine Ursache dafür, dass automatische Tests vielfach nicht vorhanden sind, ist der in vielen Projekten gelebte hohe Featuredruck. Die Zeit für die Erstellung von Tests wird eingespart, um möglichst viel neue Funktionalität zu erstellen. Mit Scrum versucht man jedoch, sich auf das wirklich Wichtige zu konzentrieren und Verschwendung in Form unnötiger Funktionen zu vermeiden. Bezogen auf den hohen Featuredruck heißt das, dass die Notwendigkeit von Features geprüft und nur wirklich wichtige umgesetzt werden sollten. Eine Priorisierung der Anforderungen durch den Product Owner ist also notwendig, um neben der Umsetzung von Funktionen auch Zeit für die Erstellung von Tests zu haben und damit qualitativ hochwertige Software produzieren zu können.

Eigene Testsysteme lohnen sich

Können Tests nur auf einem Entwicklungssystem durchgeführt werden, kann häufig keine wirklich verlässliche Aussage über die Qualität der Software getroffen werden, da die verwendeten Testdaten nicht repräsentativ sind. Standardisierte Testdaten und dedizierte Testsysteme sind jedoch eine wichtige Voraussetzung für zuverlässige Tests. Die Erfahrung zeigt, dass eine vielleicht anfänglich abschreckend hohe Investition in den Aufbau von Testsystemen mit für die Testfälle ausreichenden Testdaten sich mittelfristig lohnt. Durch die Möglichkeit der Testautomatisierung werden mehr Tests erstellt und durchgeführt, was einen sehr positiven Einfluss auf die Qualität und Wartbarkeit der erstellten Software hat.

Mangelnde Zusammenarbeit mehrerer Teams

Die optimale Größe eines Scrum-Teams liegt bei ca. 7 bis 12 Personen. Werden mehr Personen zur Realisierung eines Projekts benötigt, ist es notwendig, mehr als ein Scrum-Team parallel arbeiten zu lassen. Modelle für die Skalierung von Scrum sind schon häufig beschrieben worden und sollen hier nicht im Fokus stehen. Neben dem auch bei Scrum höheren Verwaltungsaufwand für ein größeres Projekt, z.B. durch ein größeres Product Backlog und mehr Stakeholder, ergeben sich durch mehrere Teams auch typische Probleme im Prozess.

Direkt beobachtbar werden Unstimmigkeiten zwischen Teams spätestens dann, wenn sich Teams übereinander beschweren. Nicht so offensichtlich ist es hingegen, wenn nur die Qualität des Codes abnimmt. Häufig kommt es zu Inkompatibilitäten und Mehrfach-Entwicklungen und die teamübergreifende Integration des Quellcodes wird zunehmend schwieriger. Letztendlich nimmt die Velocity der einzelnen Teams ab. Dies kann so weit gehen, dass immer häufiger das Sprintziel nicht erreicht wird.

Architectual Board – regelmäßige Abstimmungstreffen der Teams

Begründet sind alle diese Probleme aber nicht in der fehlenden Skalierbarkeit von Scrum an sich, sondern in einer unzureichenden Zusammenarbeit der Teams. Wie diese verbessert werden kann, hängt wiederum von den jeweiligen Ursachen ab. Fehlt z.B. eine Plattform, in der sich die Teams über die Gesamtarchitektur austauschen können, sollte ein entsprechendes "Architectural Board" eingerichtet werden. Dies empfiehlt sich auch, um eine gegebenenfalls vorhandene Unternehmensarchitektur in das Projekt zu tragen.

Das Board dient der regelmäßigen Abstimmung der Teams untereinander auf einer technischen Ebene. Es geht also nicht in erster Linie darum, welche Fachlichkeit von welchem Team realisiert wird, sondern welche Änderun-

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

gen am System vorgenommen werden und wie dies ggf. andere Teams beeinflusst. Ebenso kann das Board genutzt werden, um offene Architekturentscheidungen gemeinsam zu treffen.

Die Organisation eines Architectural Boards muss den Gegebenheiten des jeweiligen Projekts angepasst werden: Sind viele hochqualifizierte Entwickler in allen Teams vorhanden, kann jedes Team zu jedem Treffen jeweils einen anderen Entwickler ins Board entsenden. Will man auf der anderen Seite nicht immer wieder die gleichen Diskussionen führen, spricht vieles dafür, eine feste Zuordnung vorzunehmen. Dabei besteht wiederum die Gefahr, dass damit die Homogenität innerhalb eines Scrum-Teams aufgebrochen wird und sich implizite Hierarchien bilden.

Ebenso wie die Zusammensetzung muss auch die Häufigkeit der Abstimmungstreffen im Projektkontext entschieden werden. Sinnvoll ist sicherlich ein Treffen im Zusammenhang mit der Sprint-Planung, um die sich daraus ergebenden Fragen zu klären. Ob auch innerhalb einer Iteration Treffen sinnvoll sind, hängt sehr davon ab, wie stark die Arbeiten der Teams einander beeinflussen.

Scrum of Product Owner – Iterationen teamübergreifend planen

Probleme anderer Art treten auf, wenn nicht nur mehrere Teams sondern auch mehrere Product Owner eingesetzt werden. Es besteht dann die Gefahr, dass die Teams einander widersprechende Anforderungen umsetzen sollen und es zu Konflikten bei der Integration kommt. So kann es zum Beispiel vorkommen, dass durch die beauftragten User Stories Änderungen am zugrunde liegende Objektmodell vorgenommen werden, die untereinander nicht kompatibel sind.

Ebenso kommt es vor, dass der Product Owner Zuarbeiten des eigenen Teams für ein anderes als nicht so wichtig einstuft. Dies hat zur Folge, dass die Arbeit des betroffenen Teams behindert wird. Lösen lassen sich solche Widersprüche nur durch eine integrierte, teamübergreifende Iterationsplanung. Hierbei werden die Inhalte der Backlogs der verschiedenen Teams und auch die jeweiligen Iterationsziele aufeinander abgestimmt. Diese Iterationsplanung wird am besten gepaart mit einem "Scrum of Product Owner"-Meeting, in dem die tägliche Arbeit aufeinander abgestimmt wird.

Scrum of Scrums – Tägliche Abstimmung der Scrum Master

Arbeiten mehrere Teams parallel, ist es nur natürlich, dass mehrere Teams durch die gleichen Hindernisse gestört werden. So werden alle Teams von denselben Infrastrukturproblemen betroffen sein, wenn eine Testumgebung für die kontinuierliche Integration fehlt. Um solche Störungen effizient zu beseitigen und den Teams eine Gelegenheit für die organisatorische Abstimmung zu geben, sollte ein "Scrum of Scrums", also ein tägliches Scrum-Meeting der Scrum Master eingesetzt werden.

Nicht eindeutig beantworten lässt sich hingegen die Frage, wie denn ein Projekt auf mehrere Teams aufgeteilt werden sollte. Nahe liegend ist sicherlich die Aufteilung entlang von Systemgrenzen, also z.B. getrennte Teams für Frontend und Backend. Ebenso legen standortübergreifende Projekte eine Aufteilung der Teams je Standort

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

nahe. Eine solche Aufteilung führt aber in der Regel zur Verstärkung einer schon vorhandenen Spezialisierung mit dem Ergebnis, dass sich jedes Team nur noch in seinem Bereich auskennt.

Dies wiederum widerspricht fundamental der Idee der Collective Code Ownership, also dem Prinzip, dass jedes Teammitglied gleichermaßen für den gesamten Code verantwortlich ist und diesen auch jederzeit verändern kann und darf. Aus diesem Grund sollte bei standortübergreifenden Teams in die Kommunikationsinfrastruktur investiert werden. Dies führt mit hoher Wahrscheinlichkeit zu homogenen Teams und einer konsistenten Plattform.

Fazit

Auch wenn agile Vorgehensweisen auf den ersten Blick bestechend einfach sind, ist eine problemfreie Umsetzung nicht garantiert. Viele der geschilderten Probleme treten jedoch in fast allen Projekten – unabhängig von der verwendeten Methode – auf und auch ihre Lösungen sind nicht spezifisch für agile Vorgehensweisen. Durch die starke Fokussierung auf die Umsetzung einer Aufgabe innerhalb eines festen Zeitrahmens werden die geschilderten Probleme bei Verwendung agiler Methoden aber viel früher deutlich. Die einfachste Möglichkeit, die hier beschriebenen Probleme zu vermeiden, ist die Qualifikation der Mitarbeiter zu verbessern, in Scrum also die Qualifikationen des Product Owner, des Scrum Master und der Teammitglieder. Je vertrauter die Mitarbeiter mit agilen Methoden sind, desto eher fällt ihnen selbst auf, dass ein Problem vorliegt, und desto eher finden sie auch aus dem Projekt heraus eine passende Lösung.

Neben der reinen Lehre spielen im agilen Kontext aber auch die mit dem Vorgehen verbundenen Werte eine entscheidende Rolle. Agile Methoden setzen in besonderem Maße bestimmte Werte voraus wie Offenheit und Vertrauen, vor allem aber Kommunikation: Viele Elemente des Vorgehens beruhen im Kern darauf, dass man miteinander redet. Dieser hohe Anteil an Kommunikation ist nur in einer von Vertrauen und Offenheit geprägten Unternehmenskultur möglich.

Das agile Vorgehen versucht darüber hinaus, durch iteratives Vorgehen Komplexität zu beherrschen und Lernen zu ermöglichen. Lernen setzt aber die Möglichkeit voraus, trotz Unsicherheit Entscheidungen zu fällen und Fehler machen zu dürfen. In einer Unternehmenskultur, die von Angst vor Fehlern geprägt ist, wird sich daher ein agiles Vorgehen nicht erfolgreich umsetzen lassen.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Bild 3: Agile Werte.

Werteunterschiede sind häufig die eigentliche Ursache von Problemen, dies gilt auch für agile Methoden. Viele der in diesem Artikel beschriebenen Probleme sind in ihrem Kern Probleme der Kommunikation: Kommunikation untereinander, innerhalb einer Organisation und übereinander. Eine Atmosphäre, in der eine offene und vertrauensvolle Kommunikation stattfinden kann, lässt sich jedoch nicht per Dekret herbeiführen. Sie setzt vielmehr ein Bekenntnis zu den agilen Werten voraus, wie in Bild 3 dargestellt. In diesem Sinne ist der agile Wertekanon auch der Schlüssel zur Lösung der hier geschilderten Stolperfallen: In dem Maße, wie ein Unternehmen bereit ist, sich ernsthaft auf die den agilen Methoden zugrunde liegenden Werte einzulassen, wird auch die Wahrscheinlichkeit einer erfolgreichen Implementierung agiler Methoden in dem Unternehmen steigen.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Agile Softwareentwicklung mit Scrum und User Stories

Das Vorgehensmodell Scrum erfreut sich durch wenige Regeln und klare Verantwortlichkeiten in der Softwareentwicklung zunehmender Beliebtheit. So erlaubt Scrum ein einfaches und flexibles Vorgehen – wie die Beteiligten den Entwicklungsprozess gestalten und organisieren, bleibt ihnen selbst überlassen. (Siehe auch "Agiles Projektmanagement. Scrum – eine Einführung", Projekt Magazin, Ausgabe 21/2009.)

Allerdings enthält Scrum keine konkreten Vorgaben, wie die Anforderungen an die zu entwickelnde Software erfasst und spezifiziert werden sollen. Die agile Softwareentwicklung verfolgt jedoch den Ansatz, die Anforderungen auf eine Art und Weise zu beschreiben, die es dem Entwicklungsteam ermöglicht, nach jedem Sprint nutzbare Software für den Kunden zu liefern.

Ein etablierter Weg, dies zu tun, ist die Verwendung von User Stories. Dabei handelt es sich um einfache Formulierungen der Anforderungen aus Sicht des späteren Anwenders. Dieser Artikel beschreibt, wie man die Anforderungen eines Scrum-Projekts mit Hilfe von User Stories beschreiben und verwalten kann.

Autor

Ralf Wirdemann

Zertifizierter Scrum Master und zertifizierter Scrum Practitioner, Coach für agiles Projektmanagement

Kontakt: business@ralfwirdemann.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in der Rubrik:

› Agiles Projektmanagement

Praxisbeispiel

Zum besseren Verständnis soll das Prinzip der User Stories innerhalb eines Scrum-Projekts am Beispiel eines webbasierten Job-Portals erläutert werden: Eine Gruppe von Investoren möchte ein Portal für die Stellenvermittlung im High Professional-Umfeld entwickeln, das Firmen und Recruitern Zugriff auf nachgewiesen hoch qualifiziertes Fachpersonal ermöglichen soll. Im Gegenzug bietet das Portal Job-Suchenden die Möglichkeit, Kontakte zu renommierten und interessanten Arbeitgebern aufzubauen.

Ein Auszug aus der zentralen Anforderungsliste des Job-Portals, dem sog. Product Backlog, könnte wie folgt aussehen:

- Datenbankmodell entwerfen
- OR-Mapping-Framework (Mapping zwischen den Objekten des Systems und der unterliegenden relationalen Datenbank) entwickeln
- User Interface für die Job-Suche entwerfen
- Security Framework (Authentifizierung und Autorisierung von Nutzern) implementieren

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

- Suchmaschine entwickeln und integrieren
- ...

Das Problem: Scrum enthält keine Spezifikation der Anforderungen

Das Product Backlog in Scrum ist eine priorisierte Liste mit sämtlichen Anforderungen des Projekts. Die Einträge im Backlog werden als Backlog Items bezeichnet. Für jeden Eintrag liegt eine Schätzung des Aufwands vor; die Priorität des Eintrags bestimmt seine Position in der Liste. Während Scrum zwar mit dem Product Backlog sehr genau vorgibt, wie und womit Anforderungen verwaltet werden, trifft es keinerlei Aussagen darüber, wie die einzelnen Einträge des Backlogs formell genau auszusehen haben. Von "Die Software muss in Java programmiert werden" bis hin zu nicht-funktionalen Anforderungen, wie "Die Applikation muss skalierbar sein" ist alles erlaubt, was irgendwie getan werden muss.

Beide Beispiele sind notwendige Anforderungen, liefern aber keinen offensichtlichen Mehrwert für die Nutzer der Anwendung. Und darum geht es schließlich bei Scrum – der Entwicklung und regelmäßigen Auslieferung von nutzbarer Software für den Kunden. Entsprechend sollte das Product Backlog nur Einträge enthalten, die einen klar ersichtlichen Wert für die späteren Nutzer der Anwendung liefern. Hier hat der Product Owner die Aufgabe, ein Backlog zu erstellen, das diesen Kriterien genügt.

Ein Beispiel für Anforderungen ohne praktische Funktion für den Nutzer ist das oben genannte Backlog Item "OR-Mapping-Framework entwickeln". Die Entwicklung eines solchen OR-Mappers ist sinnvoll und notwendig, nur bietet der OR-Mapper für sich allein genommen keinerlei Mehrwert für die späteren Anwender des Portals. Entsprechend schwierig ist es für den Product Owner, der ja die Sicht des Kunden repräsentiert, dieses Backlog Item im Verhältnis zu den anderen Anforderungen im Sinne des Geschäftswerts zu priorisieren. Der Begriff Geschäftswert beschreibt in diesem Zusammenhang den Wert, den ein neues Feature für das Geschäftsfeld liefert, für das die Software entwickelt werden soll. Dieses kundenorientierte Anforderungsmanagement zielt folglich auf die regelmäßige Produktion und Lieferung von Features mit dem größtmöglichen Geschäftswert ab.

Die Lösung: User Stories im Kontext von Scrum

User Stories sind ein Werkzeug der agilen Softwareentwicklung, mit dem sich Software-Anforderungen kurz und prägnant aus Sicht des Nutzers beschreiben lassen. Viele Teams in der Softwareentwicklung setzen aktuell auf Scrum, so dass sich ein genauerer Blick auf die Verwendung von User Stories im Kontext von Scrum lohnt.

Ein wichtiges Prinzip von Scrum ist, dass die Auftraggeber regelmäßig in kurzen Abständen die aktuelle Version der Software zu sehen bekommen, um möglichst früh Kurskorrekturen vornehmen zu können. Deshalb muss der Product Owner die wichtigsten Anforderungen so früh wie möglich vom Team umsetzen lassen, um dem Kunden funktionsfähige Software präsentieren zu können. Damit diese relevanten Anforderungen auch hoch priorisiert werden, eignen sich User Stories, da sie für den Product Owner sehr viel greifbarer sind und sich gegeneinander abwiegen lassen.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Im Kontext des Job-Portals könnte eine User Story z.B. das Einstellen oder die Suche nach Stellenangeboten sein. An diesem Beispiel wird außerdem offensichtlich, dass das Einstellen von Angeboten vor einer möglichen Suche stattfinden muss und folglich höher priorisiert werden sollte.

Im Gegensatz zu den vorher exemplarisch genannten Anforderungen zielt eine User Story immer auf einen konkreten Mehrwert für den Kunden ab: Statt der Entwicklung eines OR-Mappers beschreibt eine User Story immer eine konkrete und für den Nutzer sinnvolle Funktion. Technische Details für die Programmierer werden bei dieser Sichtweise nicht ignoriert, sondern zu einer notwendigen Voraussetzung für die Entwicklung einer User Story. Der OR-Mapper liefert für sich genommen keinen Mehrwert, ist aber erforderlich für die Entwicklung eines sinnvollen Features für den Nutzer. Während im Backlog nur die User Story auftaucht, wird das Team den OR-Mapper im Rahmen der Entwicklung aufgreifen und als technisches Detail der Story implementieren.

Was ist eine User Story?

Eine User Story beschreibt eine Anforderung an das zu entwickelnde Softwaresystem. Jede User Story besteht aus drei Teilen: Einer Story Card, ihren Akzeptanzkriterien und die für ihre konkrete Umsetzung erforderliche Konversation zwischen Product Owner und Entwicklungsteam.

Story Card

Der schriftliche Teil einer User Story, die sogenannte Story Card, besteht häufig nur aus einem einzigen Satz, der den Kern der Anforderung auf den Punkt bringt. Beispiele für User Stories sind

- "Als Nutzer kann ich nach Stellenangeboten suchen" oder
- "Als Arbeitgeber kann ich Stellenanzeigen aufgeben".

Beide Beispiele bringen die jeweilige Anforderung auf den Punkt und machen in einem Satz klar, um wen es geht und was derjenige mit dem System machen kann. Der Product Owner kann dadurch den Geschäftswert der jeweiligen Anforderung bestimmen und sie entsprechend priorisieren.

Story Cards lassen sich am einfachsten mit Hilfe von Karteikarten oder großen Post-Its verwalten. Diese lassen sich schnell schreiben und für alle sichtbar an die Wand hängen. Sie sind weder zu groß, noch zu klein und bieten ausreichend Platz für den Titel der Story, vermeiden dabei aber das Schreiben von zu langen Texten. Physische Story Cards funktionieren allerdings nur für zentral arbeitende Teams; verteilt sich das Team über mehrere Orte, bietet sich der Einsatz elektronischer Tools, wie Scrumy, Mingle oder ScrumNinja (siehe Referenzen) an.

Akzeptanztests

Die Story Card ist nur einer von drei Teilen einer User Story. Über die Card hinaus besteht eine User Story aus ihren Akzeptanztests. Dabei handelt es sich um beispielhafte Beschreibungen der Funktionen einer User Story. Mit den Akzeptanztests legt der Product Owner fest, welche konkreten Funktionen die Story liefern muss. Einige Akzeptanztest-Beispiele für die Suche nach Stellenangeboten im Kontext des Jobportals:

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

- Eine Stichwortsuche nach "Java" liefert alle Angebote, die den Begriff "Java" entweder im Titel oder in der Beschreibung des Angebots enthalten.
- Eine Suche nach "Ort" liefert alle Angebote, die sich entweder direkt auf den gesuchten Ort oder auf einen Umkreis von 50 Kilometer außerhalb dieses Orts beziehen.

Akzeptanztests werden vom Product Owner formuliert und sind gleichermaßen Test und exemplarische Spezifikation. Für den Product Owner sind sie Abnahmekriterien, die er nach der Fertigstellung der Story überprüfen muss. Für das Team sind Akzeptanztests während des Sprints eine Richtschnur, was genau zu entwickeln ist und wann die Story aus Geschäftssicht fertig ist.

Ein geeigneter Platz für das Festhalten der Akzeptanztests ist die Rückseite der jeweiligen Story Card. Elektronische Tools bieten i.d.R. gesonderte Eingabebereiche, um die Tests zu notieren.

Konversation zwischen Product Owner und Entwicklungsteam

Der dritte und wichtigste Teil einer User Story ist die Konversation zwischen dem Product Owner und dem Team. Wenn der schriftliche Teil einer User Story nur aus einem einzigen Satz und den zugehörigen Akzeptanztests besteht, wo bleiben dann die Details, d.h. die konkreten Ausprägungen der jeweiligen Story? Ganz einfach: Sie werden vom Product Owner dem Team erzählt und von ihm während des Sprints maßgeblich mitbestimmt. Anstatt die Details der Anforderungen im Vorfeld der Entwicklung sehr genau aufzuschreiben, verlagern User Stories die Anforderungsbeschreibung von der schriftlichen auf die verbale Kommunikation. Dadurch rückt der Product Owner während des Sprints nah an das Team heran, sieht die entwickelte Software sehr früh und erhält so viel bessere Steuerungsmöglichkeiten, die Software im Sinne des Kunden zu entwickeln.

Der typische Ablauf für die Planung und Umsetzung einer Story im Rahmen eines Sprints gestaltet sich wie folgt: Der Product Owner erklärt dem Team im Sprint Planning seine Vorstellungen von der Story "Als Job-Suchender will ich nach Stellenangeboten suchen" und das Team entscheidet die Story in den anstehenden Sprint zu nehmen. Anschließend startet das Team in den Sprint und beginnt mit der konkreten Umsetzung der Story; es arbeitet Aufgabe für Aufgabe (sog. Tasks) ab und entwickelt möglichst schnell eine erste vorzeigbare Version der Story.

Zusammen mit dem Product Owner geht das Team diese erste Version direkt vor dem Rechner durch und diskutiert die konkreten Details ihrer Weiterentwicklung. Die Story wird Schritt für Schritt und unter ständiger und sehr enger Einbeziehung des Product Owners weiter entwickelt. Durch diese Art der engen Zusammenarbeit wird von vornherein sichergestellt, dass das Team etwas entwickelt, was der Product Owner auch haben will.

Der Anforderungsworkshop

Um User Stories zu sammeln, sollte zu Beginn des Projekts ein Anforderungsworkshop durchgeführt werden. Der Product Owner initiiert den Workshop und lädt neben dem Entwicklungsteam alle Stakeholder, wie z.B. Marketing, Vertrieb, Investoren oder auch die Geschäftsführung ein. Jeder, der in irgendeiner Form etwas zu den Anforderungen der anstehenden Produktentwicklung sagen kann, sollte an diesem Workshop teilnehmen.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Der Workshop beginnt damit, dass der Product Owner den Teilnehmern seine zuvor in Zusammenarbeit mit dem Kunden ausgearbeitete Produktvision präsentiert. Die Produktvision ist ein kurzes und knappes Statement, das beschreibt, was entwickelt werden soll. Die Vision bringt den Kern des Produkts auf den Punkt und soll möglichst alle Stakeholder überzeugen und ins Boot holen. Die Produktvision des Job-Portals könnte z.B. folgender Satz sein: "Wir wollen das einzige Job-Portal für die Stellenvermittlung im High Professional-Umfeld in Europa werden."

Wurde die Vision vermittelt und von den Teilnehmern verstanden, ist der nächste Schritt im Workshop die Modellierung der Nutzerrollen. Eine Nutzerrolle ist eine Gruppe von Nutzern mit identischen Zielen in Bezug auf die Nutzung des Systems. Die zentrale Frage bei der Modellierung der Nutzerrolle ist also, welche Bedürfnisse der Anwender der Software hat. Denken Sie beispielsweise an eine der diversen Job-Seiten im Internet, dann drängen sich einige offensichtliche Nutzerrollen unmittelbar auf: Job-Suchende, als Gruppe von Nutzern die über die Plattform neue Stellen finden wollen, oder Job-Anbieter, als Gruppe von Nutzern, die Stellenangebote ausschreiben wollen, um neue Mitarbeiter zu finden.

Mit Nutzerrollen die Anforderungen festlegen

Nutzerrollen, deren Beschreibung und Ziele sind ein wichtiger Ausgangspunkt für das Schreiben der ersten User Stories. Das funktioniert gut, indem sich das Anforderungsteam Rolle für Rolle vornimmt und anhand der jeweiligen Ziele die User Stories überlegt, die für die Zielerreichung notwendig sind. Das Anforderungsteam besteht idealerweise aus Vertretern sämtlicher am Projekt beteiligen Stakeholder, wie dem eigentlichen Scrum-Team, der Marketingabteilung, dem Vertrieb oder auch der Geschäftsführung. Auf diese Art wird sichergestellt, dass alle relevanten Parteien von Anfang an beteiligt sind und den Product Owner beim Schreiben der User Stories unterstützen.

Nehmen wir als Beispiel die Nutzerrolle "Job-Suchender". Das primäre Ziel dieser Rolle ist es, einen neuen Job zu finden. Dazu muss der Job-Suchende den Stellenmarkt nach bestimmten Kriterien durchsuchen können. Enthält die Ergebnisliste interessante Angebote, dann will er die Angebote im Detail betrachten. Interessiert ihn ein Angebot besonders, dann will er sich schließlich darauf bewerben. Ein einfacher Blick auf das Ziel "Job finden" führt zu drei offensichtlichen User Stories:

1. Als Arbeitssuchender will ich die Angebote nach bestimmten Kriterien durchsuchen.
2. Als Arbeitssuchender will ich die Details der gefundenen Angebote sehen.
3. Als Arbeitssuchender will ich mich auf ein Angebot bewerben.

Beim Betrachten dieser User Stories fällt auf, dass die Stories einem Muster folgen: Als <Nutzerrolle> will ich <das Ziel> [, so dass <Grund für das Ziel>].

Die Verwendung dieses Musters hat sich in der Praxis bewährt und ermöglicht es dem Product Owner mit sehr wenigen Worten sehr viel auszudrücken: Für wen ist die Anforderungen gedacht, welches Ziel verfolgt derjenige mit der Anforderung und was ist der Grund für dieses Ziel. Außerdem zwingt das Muster den Product Owner, den Kern der Anforderung und damit deren Wert für den Kunden auf den Punkt zu bringen.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Der letzte Teil des Musters, der "Grund für das Ziel", ist optional und sollte immer dann verwendet werden, wenn das Ziel alleine nicht ausreicht, um den Geschäftswert der Anwendung zu formulieren. Ein Beispiel ist die Story: "Als Job-Anbieter will ich mich einloggen". Die Funktion "Einloggen" stellt keinen offensichtlichen Geschäftswert dar. Erweitert man hingegen die Story um den Grund "so dass kein anderer Anbieter unter meinem Namen Job-Angebote erstellen kann", werden der Wert und damit der Grund fürs Einloggen deutlich.

Ergebnis des Workshops

Das Ergebnis des ersten Anforderungsworkshops ist ein initiales Product Backlog, gefüllt mit den wesentlichen User Stories des Projekts. Die initialen User Stories sind häufig noch zu groß, als dass sie sich für die direkte Umsetzung in einem der ersten Sprints des Projekts eignen würden. Deshalb müssen sie auf die richtige Größe geschnitten werden. Hierbei kommt es darauf an, dass die resultierenden Teil-Stories weiterhin ihren Geschäftswert behalten. Ein Beispiel für eine zu große Story ist die Suche nach Stellenangeboten. Diese kann die Stichwortsuche, die Suche nach Ort, Bezahlung oder Position einbeziehen. Jede dieser Teilfunktionen sind wieder eigenständige User Stories die weiteren Wert für den Kunden schaffen und sich eigenständig umsetzen lassen.

Mit zunehmender Projekterfahrung wird relativ schnell offensichtlich, wann eine Story zu groß ist und wie sie sich am sinnvollsten zerschneiden lässt. Grundsätzlich gilt es, Stories so klein wie möglich zu schneiden, sofern sie weiterhin eine nützliche Funktion für den Anwender liefern. Die natürliche Obergrenze für die Größe einer Story ist dabei immer ein Sprint; schätzt das Team die Entwicklung einer Story auf länger als einen Sprint, ist klar, dass die Story zerschnitten werden muss.

Weitere Anhaltspunkte für zu große User Stories liefern die regelmäßig stattfindenden Estimation-Meetings. In diesen Meetings schätzt das gesamte Team die bis dahin geschriebenen User Stories.

Kommt es dabei zu sehr großen Schätzungen, oder kann der Product Owner die Story dem Team nicht ausreichend weit erklären, so dass das Team nicht in der Lage ist, die Story zu schätzen, ist es offensichtlich, die Story zu verkleinern.

Der erste Schritt vor dem Schätzen und dem daraus ggf. folgendem weiteren Zerschneiden der Stories ist ihre Priorisierung.

User Stories priorisieren und schätzen

Wie eingangs erwähnt, ist es ein wesentliches Scrum-Prinzip, die wichtigsten Dinge zuerst zu erledigen, um dem Kunden früh funktionsfähige Software zu präsentieren. Folglich muss sich der Product Owner für sein Backlog überlegen, welches die wichtigsten Stories sind und sie entsprechend priorisieren. Die wichtigsten Stories wandern im Backlog nach oben und werden damit zu Kandidaten für den nächsten Sprint.

Eine einfache und schnelle Variante der Priorisierung ist die sogenannte "MuSCoW-Priorisierung", die Stories hinsichtlich der folgenden vier Kriterien bewertet:

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Must Have: Diese User Stories sind zwingend erforderlich. Ohne sie würde das System nicht funktionieren.

Should Have: Diese User Stories sind sehr wichtig, das System funktioniert aber auch ohne sie, weil z.B. eine provisorische Lösung (Workaround) existiert.

Could Have: User Stories dieser Kategorie haben eine geringe Bedeutung und werden nur umgesetzt, wenn neben Must Have- und Should Have-Stories noch Kapazitäten zur Verfügung stehen.

Won't have this time: Diese User Stories sind zurzeit nur vorgemerkt, werden aber aktuell nicht umgesetzt.

Nach dem ersten Anforderungsworkshop enthält das Backlog meistens noch viele zu große Stories. Mit dem Prinzip MuSCoW kann der Product Owner eine schnelle und einfache Priorisierung des initialen Product Backlog durchführen und die Stories hinsichtlich ihrer Notwendigkeit für ein Funktionieren des Systems bewerten. Alle Must Have- und Should Have-Stories werden weiterverfolgt und auf konkretere Stories "heruntergebrochen". Die resultierenden Teil-Stories werden ihrerseits nach MuSCoW priorisiert. Auf diese Weise ist es möglich, relativ schnell zu einem sinnvoll priorisierten Product Backlog zu kommen, das vom Team geschätzt und als Grundlage für die Planung des ersten Sprints genutzt werden kann.

Stories schätzen

Nach der Priorisierung steht das Schätzen der Stories auf dem Programm. Zusammen mit dem Team geht der Product Owner das Backlog von oben nach unten durch und lässt das Team gemeinsam den Aufwand der jeweiligen User Story schätzen. Das relative Schätzen findet im Rahmen agiler Vorgehensweisen und damit auch im Rahmen von Scrum zunehmende Verbreitung. Bei dieser Art des Schätzens werden Anforderungen nicht in Personentagen, sondern ausschließlich in ihrer Größe geschätzt.

Der entscheidende Vorteil des relativen Schätzens ist die Trennung von Dauer und Größe. Während man beim traditionellen Schätzen versucht, in Personentagen vorherzusagen, welches Feature wie viel Entwicklungszeit benötigen wird, trifft man beim größenbasierten Schätzen ausschließlich Aussagen über die relative Größe von Features zueinander. So lässt sich sehr viel einfacher bestimmen, dass Feature A kleiner als Feature B ist, als vorherzusagen, dass Feature A genau einen Personentag beansprucht, während Feature B zwei Personentage dauern wird. Das Schätzen in relativen Größen erhöht die Wahrscheinlichkeit, dass die Schätzungen zutreffen, erheblich.

Der offensichtliche Nachteil relativer Schätzungen ist, dass Aussagen über die Entwicklungsdauern der jeweiligen Features fehlen. Schließlich wollen Projektleiter und Geschäftsführung nicht wissen, was wie groß im Verhältnis zueinander ist, sondern wann welches Feature geliefert wird. Ein Punkt, auf den ich später im Artikel zurückkommen werde.

Für die Schätzungen verwendet man "Story Points", die die relativen Größenverhältnisse der Stories zueinander ausdrücken. Eine Story von zwei Punkten ist doppelt so groß, wie eine 1-Punkt-Story. Eine 8-Punkte-Story ist viermal so groß, wie eine 2-Punkte-Story, und so weiter. Größe ist dabei zunächst ein abstrakter Begriff, der nichts darüber aussagt, wie lange die Entwicklung der jeweiligen Story dauern wird. Im Abschnitt "Sprints messen und planen" (s. unten) erkläre ich, wie Story Points für das Messen der Entwicklungsgeschwindigkeit des Teams und so für die Release-Planung genutzt werden können.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

User Stories im Sprint

Hat der Product Owner die Stories priorisiert und liegen ausreichend geschätzte Stories für mindestens einen Sprint vor, dann kann das Team in seinen ersten Sprint starten. Am Anfang jedes Sprints steht das Sprint Planning Meeting, das in zwei Teile gegliedert ist: das "Sprint Planning 1" und das "Sprint Planning 2". Das Sprint Planning 1 ist ein Analyse-Meeting, in dem der Product Owner dem Team die User Stories der Reihe nach vorstellt. Das Team stellt Fragen, diskutiert die Story und versucht die Anforderungen ausreichend weit zu verstehen, um entscheiden zu können, ob es die Story im anstehenden Sprint vollständig umsetzen kann. Sieht das Team anschließend Raum für eine weitere Story, setzt der Product Owner das Vorstellen fort.

Einzig und allein das Team bestimmt, wie viele Stories es in den Sprint aufnimmt. Der Product Owner gibt ausschließlich die Reihenfolge der Stories, jedoch nicht deren Anzahl vor. Diese Regel ist wichtig im Hinblick auf die Teilung der Verantwortlichkeiten in Scrum: Das Team ist für die Software verantwortlich. Wenn das Team bestimmen darf, wie viel Arbeit es in den anstehenden Sprint nimmt, dann übernimmt das Team damit auch die Verantwortung dafür, die Software am Ende des Sprints zu liefern. Die Übernahme von Verantwortung basiert also auf Freiwilligkeit, was nicht gegeben wäre, wenn der Product Owner bestimmen würde, wie viele Stories das Team im anstehenden Sprint zu liefern hat.

Das im Anschluss an das Sprint Planning 1 folgende Sprint Planning 2 ist ein Design-Meeting. In diesem Meeting entwirft das Team das Software-Design der Stories und zerbricht jede Story in die konkret auszuführenden Arbeitsaufgaben, die sogenannten Tasks. Die Task-Liste der Angebotssuche könnte z.B. wie folgt aussehen:

- Domain-Modell entwerfen und implementieren
- Controller-Logik programmieren
- Layout entwerfen und Basis-CSS erstellen
- Suchtechnologie evaluieren
- ...

Stories und Tasks werden auf Karteikarten notiert und ans Taskboard gehängt. Das Taskboard ist die To-Do-Liste des Teams, das sämtliche noch ausstehende Arbeit des Sprints visualisiert.

Während des Sprints arbeitet das Team Story für Story ab. Die Stories werden im Sprint in der Reihenfolge ihrer Prioritäten nacheinander umgesetzt. Alle am Ende des Sprints nicht fertigen User Stories nimmt der Product Owner zurück ins Product Backlog. Ist die Story bei der nächsten Priorisierung noch genauso wichtig wie im gerade abgeschlossenen Sprint, schafft sie es unmittelbar zurück in den nächsten Sprint. Hier hat Scrum einen großen Vorteil: Die noch nicht umgesetzten Anforderungen werden ständig darauf überprüft, ob die ursprüngliche Fassung noch gültig ist.

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Sprints messen und planen

Auch Scrum-Teams reihen nicht Sprint um Sprint aneinander und gucken mal, wann sie fertig sind. Sowohl der Kunde als auch der Product Owner haben ein berechtigtes Interesse an Planung und einer darauf basierenden Kosten- und Fertigstellungsvorhersage. Planung in Scrum basiert auf der sogenannten "Velocity". Diese bezeichnet die Entwicklungsgeschwindigkeit des Teams gemessen an der tatsächlich entwickelten und am Ende des Sprints gelieferten Funktionalität.

Dazu ein Beispiel: Hat das Team im Sprint Planning Meeting Stories mit einer Summe von 20 Story Points angenommen, schafft während des Sprints aber nur drei statt vier Stories, dann beträgt die tatsächliche Velocity dieses Sprints 20 minus der Story Points der vierten Story. Angenommen die vierte Story wurde vom Team auf fünf Punkte geschätzt, dann liefert das Team 15 Story Points ab, was die tatsächliche Velocity dieses Sprints ist. Basierend auf diesem Wert kann der Product Owner eine erste Releaseplanung vornehmen, indem er das Product Backlog in Häppchen zu je 15 Punkten aufteilt. Die sich daraus ergebende Anzahl an Sprints multipliziert mit der ursprünglich geplanten Sprint-Dauer (z.B. vier Wochen) liefert das geschätzte Fertigstellungsdatum. Voraussetzung dafür ist, dass sämtliche der für das anstehende Release geplanten Stories vom Team geschätzt wurden.

Die Velocity wird von Sprint zu Sprint genauer. Während die tatsächliche Velocity nach dem ersten Sprint noch ein sehr grober und ungenauer Wert ist, pendelt sie sich nach einigen Sprints auf einen mittleren und damit für die Planung geeigneten Wert ein. Für eine sichere Planung empfiehlt es sich, mit einer niedrigeren als der mittleren Velocity zu planen, da auch eine mittlere Velocity durchaus in dem einen oder anderen Sprint unterschritten werden kann. Außerdem ist zu beachten, dass ein Product Backlog für Änderungen offen ist und Änderungen am Backlog folglich zu Änderungen am Release-Plan führen.

Fazit

Dieser Beitrag zeigt die Vorteile der Verwendung von User Stories im Rahmen von Scrum. User Stories fokussieren Team und Product Owner auf den Geschäftswert: Jeder Sprint liefert sichtbaren Mehrwert im Sinne des Kunden. Das Product Backlog enthält mit den User Stories ausschließlich werthaltige Items und kann vom Product Owner entsprechend priorisiert werden. User Stories verlagern den Fokus von der schriftlichen auf die verbale Kommunikation und erzwingen die Vollzeit-Verfügbarkeit des Product Owner. Statt genau aufzuschreiben, was das Team zu liefern hat, bleibt er eng an der Entwicklung dran und kann die Entwicklung des Teams sehr zeitnah und in kurzen Feedback-Schleifen steuern. Auf diese Art wird sichergestellt, dass am Ende jedes Sprints etwas herauskommt, was der Product Owner wirklich haben will, und nicht etwas, was er vor zwei Monaten aufgeschrieben hat.

Referenzen

- http://scrumy.com
- http://www.thoughtworks-studios.com/mingle-agile-project-management
- http://scrumninja.com/scrum_software
- Wirdemann, Ralf: Scrum mit User Stories, Hanser 2009

© 2013 Projekt Magazin Vervielfältigung, auch auszugsweise, mir mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Anforderungsmanagement in IT-Projekten

So vermeiden Sie Stolpersteine bei User Stories

Autor

Steffen Thols

Senior Consultant,
codecentric AG,
Projektmanager IPMA
Level D, Certified Scrum Master

Kontakt: steffen.thols@codecentric.de

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Agile Softwareentwicklung mit Scrum und User Stories

sowie in den Rubriken:

› Anforderungsmanagement

› Agiles Projektmanagement

› IT-Projekte

› Best Practices

Mit der Verbreitung agiler Vorgehensweisen in IT-Projekten stellt sich immer häufiger auch die Frage, auf welche Weise Anforderungen am besten beschrieben werden sollten. Vor allem die sog. "User Stories", in welchen Anforderungen aus Sicht des Nutzers beschrieben werden, erfreuen sich dabei steigender Beliebtheit: Sie haben einen klaren, einfachen Aufbau und konzentrieren sich auf die wesentlichen Elemente einer Anforderung.

Eine User Story darf inhaltlich nur so komplex sein, dass sie innerhalb eines Sprints umgesetzt werden kann und am Ende des Sprints dem Kunden – oder dem Product Owner, der die Kundensicht einnimmt – ein abnahmefähiges Teilprodukt gezeigt werden kann.

Im Artikel "Agile Softwareentwicklung mit Scrum und User Stories" (Projekt Magazin 02/2010) haben Sie erfahren, wie User Stories grundsätzlich aufgebaut sind und wie sich die Anforderungen eines Scrum-Projekts durch User Stories beschreiben, priorisieren und abarbeiten lassen.

Ergänzend hierzu sollten Sie bei der Konzeption von User Stories einiges beachten. Auch müssen die entsprechenden Rahmenbedingungen im Unternehmen geschaffen werden, damit die Software-Entwicklung mit dieser Vorgehensweise erfolgreich ist.

In diesem Artikel erfahren Sie,

- was eine User Story ist,
- warum es sinnvoll ist, mit User Stories zu arbeiten,
- welche formalen und strukturellen "Stolpersteine" es bei der Konzeption von User Stories gibt und
- wie Sie diese umgehen können.

Was ist eine User Story?

Ende der 90er Jahre arbeiteten die Software-Entwickler Ron Jeffries und Kent Beck in einem Projekt zur Entwicklung einer Gehaltsabrechnungssoftware bei Chrysler. Für die Softwareentwicklung nutzten sie die Methode XP (Extreme Programming) und verwendeten erstmals für die Beschreibung der Anforderungen User Stories, die Funktionalitäten der Software aus Anwendersicht und in einfachen Worten darstellten. So stellten sie sicher, dass die entwickelte Software auch wirklich den Anforderungen der Anwender entsprach.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Die Realisierung einer User Story besteht aus folgenden drei Elementen:

1. Die Anforderung wird in einfacher Form auf einer Indexkarte festgehalten (Card):

   Als <Nutzerrolle> will ich <das Ziel>, sodass <Grund für das Ziel>.

   Beispiel:

   "Als Kunde eines Online-Shops kann ich meine Bankverbindung in meinem Profil abspeichern, sodass ich diese bei einer weiteren Bestellung nicht erneut eingeben muss."

2. Das Entwicklungsteam stimmt sich direkt mit dem Anforderungssteller – oder dem Vertreter des Anforderungsstellers, d.h. dem Product Owner – ab. Die Details einer User Story werden in einer gemeinsamen Diskussion kurz vor Beginn der nächsten Iteration, d.h. des nächsten Entwicklungsabschnitts, geklärt (Conversation).

3. Jede User Story beinhaltet Testfälle, die sog. "Akzeptanzkriterien", die genau beschreiben, wie die Funktionalität nach der Umsetzung der User Story durch die Entwickler funktionieren soll (Confirmation).

   Beispiel:

   "Wenn die Bankleitzahl weniger als neun Ziffern enthält, führende Nullen oder Buchstaben eingegeben werden, gibt das System eine Fehlermeldung aus.

   Die Kontonummer kann bis zu zehn Stellen umfassen und darf nur Ziffern beinhalten; führende Nullen sind erlaubt …"

Warum User Stories sinnvoll sind

Die Firma cyclespeed24 konnte in der Vergangenheit ihre Software-Entwicklungsprojekte nicht immer erfolgreich abschließen. Sie wurden zwar häufig "in time" und "in budget" beendet, aber die entwickelten Features entsprachen nicht den Erwartungen der Anwender in den Fachbereichen. Dies erschwerte die Akzeptanz dieser Anwendungen durch die Fachbereiche und mit dem Image der IT-Abteilung war es innerhalb des Unternehmens auch nicht zum Besten bestellt.

Da die interne Entwicklungsmannschaft die größeren Projekte häufig nicht alleine umsetzen konnte, wurden mit einem externen Dienstleister Werkverträge zu Festpreisen geschlossen. Die Anforderungen so zu beschreiben, dass sie eine juristisch verwertbare Grundlage für den Werkvertrag bildeten, bedeutete erhebliche Aufwände für die anfordernden Fachbereiche, war aber Voraussetzung dafür, dass ein Projekt überhaupt genehmigt und ausgeschrieben werden konnte.

Die erstellten Lastenhefte nahmen schon einmal 200 Seiten und mehr ein. Sie enthielten alle denkbaren Anforderungen an das System, ohne dass jeweils eine Aufwand-Nutzen-Betrachtung erfolgt wäre, und damit auch jede Menge unnötige Funktionalitäten. Die Begeisterung (und in der Folge das Engagement) der Mitarbeiter, die sich in der Vor-Projekt-Phase in der korrekten Abwicklung ihres Tagesgeschäfts beeinträchtigt sahen, fiel entsprechend gering aus.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Hinzu kam, dass im Falle der Vergabe einzelner Projektphasen an den externen Dienstleister eine firmenübergreifende Informationsweitergabe erforderlich war. Diese führte, wie bei dem bekannten Kinderspiel "Flüsterpost", zum Verlust von Informationen und zu Fehlinterpretationen, wenn eine Anforderung nicht exakt genug beschrieben worden war.

Aus diesem Grund entschied die Geschäftsführung, dass künftig jede Software mit Hilfe von Scrum entwickelt werden sollte und zur Beschreibung der Anforderungen User Stories verwendet werden sollten. Die User Stories sollten in einem Collaboration Tool erfasst werden, um die Transparenz zu verbessern.

Mit dieser Änderung im Softwareentwicklungsprozess waren folgende Hoffnungen verbunden:

- Da die Anforderungen nun aus Sicht des Anwenders formuliert wurden, erwartete die Geschäftsführung, dass die Hemmschwelle für nicht-IT-affine Mitarbeiter aus den Fachbereichen, die jedoch Experten in ihrem Fachgebiet waren, wesentlich geringer sein würde, sich im Projekt mit ihrer Expertise einzubringen. Dabei sollten passgenauere Funktionalitäten herauskommen.
- In jedem Fachbereich sollte künftig derjenige Verantwortliche die User Stories erfassen und aktualisieren, der auch bisher bei Projekten als wichtiger Ansprechpartner fungierte.
- Die Visualisierung aller User Stories in einem Collaboration Tool sollte den Entwicklungsstand der Features transparent machen. Alle Mitarbeiter der Firma erhielten Lesezugriff auf dieses Tool, so dass sie sich jederzeit auch einen Überblick darüber verschaffen konnten, welche User Stories für die nächsten Entwicklungsiterationen vorgesehen waren.
- Durch den direkten Austausch wollte man zwischen Vertretern der anfordernden Fachbereiche und Mitarbeitern des Entwicklungsteams Fehlinterpretationen bei Anforderungen vermeiden. Dies sollte aber auch dazu führen, dass Anforderungen zu einem frühen Zeitpunkt abgelehnt würden, wenn die Aufwand-Nutzen-Betrachtung entsprechend negativ ausfiele.
- Die Aufwände für die Vertreter der Fachbereiche würden sich verringern, da nicht mehr alle Anforderungen "bis ins Letzte" spezifiziert wurden, sondern nur jeweils diejenigen, bei denen die Aufwand-Nutzen-Betrachtung positiv ausfiel und die entsprechend hoch priorisiert wurden.

Soweit die Theorie. Bei der Arbeit mit den User Stories stellte sich jedoch heraus, dass es doch noch einige formale und strukturelle "Stolpersteine" gab, die es aus dem Weg zu räumen galt.

Formale "Stolpersteine"

Fachliche Abhängigkeiten nicht bedacht

Problem

Da die Firma cyclespeed24 durch den Einsatz neuer Technologien die bestehende Software um neue Funktionalitäten erweitern und neue Schnittstellen zu Fremdsystemen implementieren wollte, um die Attraktivität der Software für die Kunden zu steigern, starteten viele Produktideen gleichzeitig. Dies erschwerte es dem Product Owner, der die Anforderungen verwaltete und priorisierte, den Überblick über die wachsende Menge an Ideen und damit User Stories zu behalten.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Die Folge war, dass das Entwicklungsteam zwar mit konstanter Geschwindigkeit User Stories in guter Qualität in den Sprints umsetzte, die User Stories aber nicht gemeinsam nutzbar waren. Der Product Owner hatte die fachliche Abhängigkeiten nicht erkannt und deshalb die User Stories nicht immer richtig priorisiert.

Lösung

In den Retrospektiven nach den ersten Sprints diskutierten der Product Owner, Scrum Master und die Entwickler diese Schwierigkeiten und suchten gemeinsam nach einer Lösung. Nach einigen Überlegungen entschlossen sie sich, die Erfahrungen der Anforderungsspezialisten in den Fachabteilungen hinsichtlich UML-Modellierung zu nutzen und für die Darstellung des gesamten zu entwickelnden Systems mehrere Kontextdiagramme zu verwenden. In diesen Kontextdiagrammen sind die Anforderungen, ihre Beziehungen zueinander und zu den Nutzern des Systems in grober Granularität dargestellt (Bild 1).

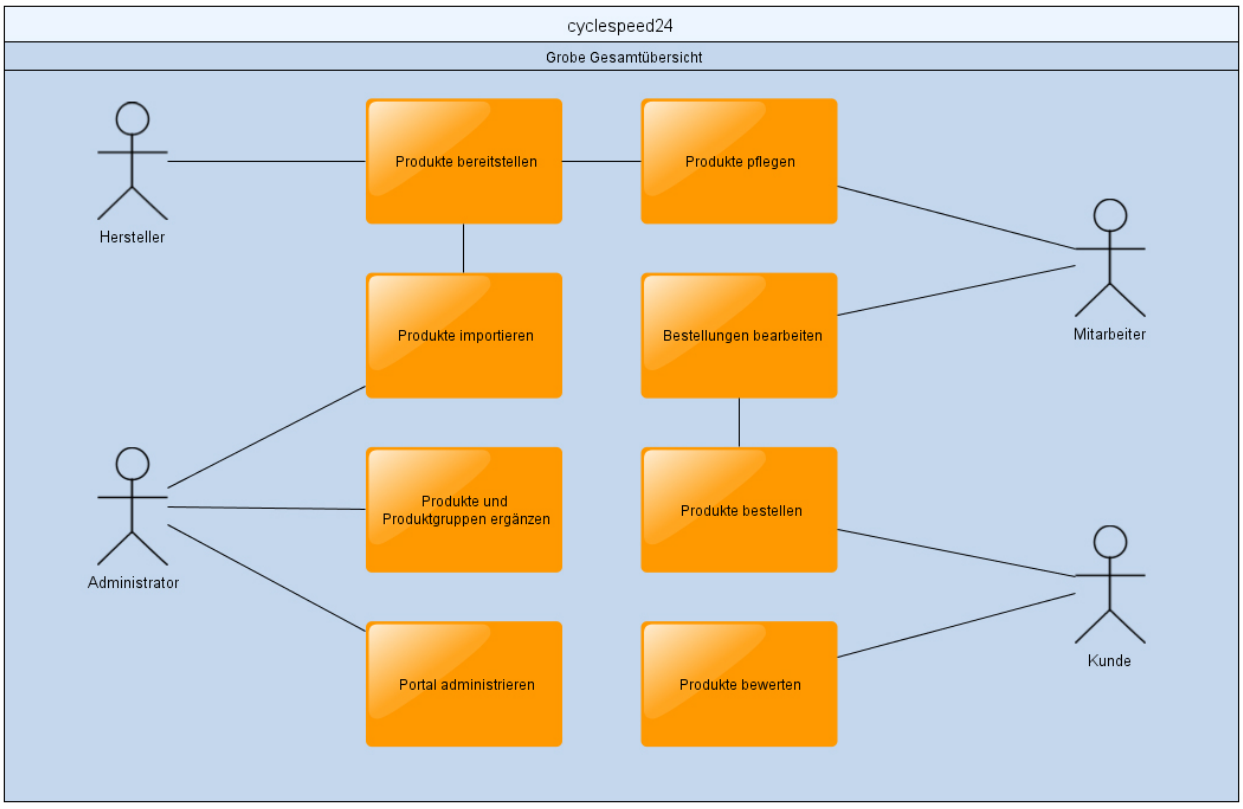


Bild 1: Grobe Gesamtübersicht der Anforderungen, ihrer Beziehungen zueinander und der jeweiligen Nutzer (Kontextdiagramm).

Durch dieses Vorgehen gewannen Anforderer und Softwareentwickler ein gemeinsames Verständnis von der gewünschten Funktionalität und möglichen Abhängigkeiten von anderen Funktionalitäten. Auch erlaubte es bereits Rückschlüsse auf die erforderliche Systemarchitektur. So sollte es z.B. Herstellern künftig möglich sein, ihre Produktinformationen für das Portal so bereitzustellen, dass ein Administrator von cyclespeed24 diese würde

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

importieren und ggf. ergänzen können. Für die Importschnittstelle konnten daher die benötigten Felder definiert und eine sinnvolle Technik ausgewählt werden.

User Stories zu groß gewählt

Problem

Die Entwicklung nach Scrum mit dem regelmäßigen Feedback im Sprint Review war für alle Softwareentwickler von cyclespeed24 neu. Um eine lösungsorientierte Projektkultur mit häufigen Plan-do-check-act-Zyklen zu etablieren, entschied sich das Scrum-Team für eine Sprintlänge von zwei Wochen.

Ganz im Sinne des Agilen Manifests hatte sich das Entwicklungsteam gleich zu Projektbeginn darauf geeinigt, dass der Projektfortschritt nur anhand der lauffähigen Software gemessen werden sollte. Nach den ersten Sprints stellte das Team allerdings fest, dass es in einem Sprint nur ein bis drei Stories vollständig umsetzen konnte. Der Product Owner musste häufig unvollendete User Stories wieder in das Product Backlog zurücknehmen. Es wurde deutlich, dass die User Stories kleiner werden mussten, damit das Entwicklungsteam in der Lage war, sechs bis zehn Stories in einem Sprint fertigzustellen.

Lösung

Bei cyclespeed24 wendeten die Anforderer aus den Fachbereichen in Zusammenarbeit mit den Entwicklungsteams verschiedene Methoden an, um die User Stories auf eine handhabbare Größe zuzuschneiden:

- Aufteilen in Workflow-Schritte
- Festlegen von Durchführungsregeln
- Aufteilen nach Eingabearten
- Untergliedern nach Datenoperationen

Diese Methoden werden im Folgenden näher ausgeführt.

Aufteilen in Workflow-Schritte

Folgende User Story scheint auf den ersten Blick recht einfach umsetzbar zu sein:

"Als Administrator kann ich Produktinformationen auf der Homepage veröffentlichen, damit diese den Interessenten und Kunden für eine Kaufentscheidung zur Verfügung stehen."

Im Gespräch zwischen dem Anforderungssteller und dem Entwicklungsteam stellte sich jedoch heraus, dass diese Produktinformationen nicht ohne vorherige Prüfung veröffentlicht werden sollten. Es bedurfte eines Freigabeprozesses, während dessen es möglich sein sollte, die Produktinformationen durch Mitarbeiter des Fachbereichs prüfen, Änderungen vornehmen und die Freigabe nach dem Vier-Augen-Prinzip genehmigen zu lassen.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Die Umsetzung aller damit verbundenen Anforderungen hätte nicht in einen Sprint gepasst. Die Anforderung wurde daraufhin anhand der Schritte im Workflow in mehrere User Stories aufgeteilt:

User Story 1:

"Als Administrator

… kann ich Produktinformationen direkt auf der Homepage veröffentlichen."

Dies entspricht der absoluten Mindestfunktionalität, die erforderlich wäre, damit die formulierte Anforderung als erfüllt gelten kann. In diesem Fall reichte dies natürlich nicht aus. Deshalb muss diese User Story noch ergänzt werden um:

… kann ich Produktinformationen zum Review durch den Fachbereich ablegen."

User Story 2:

"Als Mitarbeiter des Fachbereiches

… kann ich Produktinformationen ändern.

… kann ich zum Review abgelegte Produktinformationen nach dem Vier-Augen-Prinzip freigeben."

User Story 3:

"Als Administrator

… kann ich freigegebene Produktinformationen für die produktive Seite publizieren, damit diese den Interessenten und Kunden für eine Kaufentscheidung zur Verfügung stehen."

Festlegen von Durchführungsregeln

Häufig wird eine Anforderung so formuliert, dass für den Entwickler ein Interpretationsspielraum bleibt, wie das Ziel erreicht werden kann:

"Als Interessent kann ich ein Produkt mit Hilfe einer Suchfunktion finden."

In der Diskussion der Anforderung und durch das Ausformulieren der Akzeptanzkriterien, anhand derer die Anforderung als erfüllt gilt, wird meist schnell klar, wie dieses Ziel, hier das Auffinden eines Produkts über eine Suchfunktion, erreicht werden kann:

"Als Interessent

… kann ich ein Produkt durch die Eingabe eines einzigen Suchbegriffs finden." (User Story 1)

… kann ich ein Produkt durch Eingabe eines alternativen Suchbegriffs finden." (User Story 2)

... kann ich ein Produkt durch die gleichzeitige Eingabe von mehreren Suchbegriffen finden." (User Story 3)

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

In diesem Fall kann also die Suchfunktion in mehreren Ausprägungen entwickelt werden: von einer eher einfachen Suche bis hin zu komplexen Suchanfragen inklusive alternativer Suchbegriffe. Jede Ausprägung stellt dann eine eigene User Story dar.

Aufteilen nach Eingabearten

Bei Anforderungen, welche die Benutzeroberfläche betreffen, kann eine Aufteilung in verschiedene User Stories über die verschiedenen Eingabearten erfolgen. In der Regel wird in einer ersten User Story eine einfache Art der Eingabe entwickelt, welche im späteren Verlauf in einer weiteren User Story komplexer gestaltet werden kann.

User Story 1:

"Als Kunde kann ich ein Wunschdatum für die Auslieferung des von mir bestellten Artikels eingeben."

Akzeptanzkriterium: System bietet einfaches Eingabefeld an.

User Story 2:

"Als Kunde kann ich ein Wunschdatum für die Auslieferung des von mir bestellten Artikels per Kalenderauswahl auswählen. Die Feiertage sind im Kalender ausgegraut dargestellt."

Akzeptanzkriterien: System bietet Kalenderauswahl an. System zeigt in der Kalenderauswahl die Feiertage an. Die Feiertage sind nicht auswählbar.

Untergliedern in Datenoperationen

Die vierte Möglichkeit, Anforderungen in unterschiedliche User Stories zu unterteilen, stellt die Aufteilung nach den Datenoperationen "**C**reate", "**R**ead", "**U**pdate" und "**D**elete" (CRUD) dar. Wenn man in Anforderungen die Schlüsselwörter "pflegen" oder "verwalten" antrifft, verbergen sich dahinter meistens viele separat umsetzbare User Stories.

"Als Kunde kann ich meinen Account pflegen, um meine Daten immer aktuell zu halten."

Wird z.B. zu:

"Als Interessent

… kann ich einen Account anlegen, um Kunde zu werden."

"Als Kunde

… kann ich die Daten in meinem Account bearbeiten, um meine Daten immer aktuell zu halten."

… kann ich meinen Account löschen." (oder: "… kann ich beantragen, dass mein Account gelöscht wird.")

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Strukturelle "Stolpersteine"

Feste Ansprechpartner in den Fachbereichen fehlen

Problem

In den bisherigen Projekten der Firma cyclespeed24 wurden die unterschiedlichen Projektphasen immer von verschiedenen Fachbereichen durchgeführt. So waren die Anforderungsspezialisten verantwortlich für die Phase der Lastenhefterstellung, die nach erfolgreichem Erreichen des Meilensteins "Lastenheft erstellt" das fertige Lastenheft an die Entwicklung übergaben. Eine Zusammenarbeit zwischen den Anforderungsspezialisten und den Entwicklern fand nur bei kritischen Punkten statt, z.B. wenn die Gefahr bestand, dass ein Meilenstein nicht erreicht werden konnte, da die Anforderungsspezialisten im Sinne einer bestmöglichen Auslastung bereits für ein neues Projekt verplant waren.

Agile Vorgehensweisen betonen jedoch das regelmäßige und enge Zusammenarbeiten aller Personen, die für die Realisierung des Vorhabens notwendig sind. Kollaborative Elemente waren aber zu Projektstart nicht in der Organisation der Firma verankert. So war z.B. die regelmäßige Mitarbeit der Fachbereiche an IT-Vorhaben nicht etabliert.

Aus diesen Gründen traten im Projektverlauf folgende, aufeinander aufbauende negative Effekte auf:

- Für die Mitarbeiter der anfordernden Fachbereiche hatte es gemäß ihrer persönlichen Zielvereinbarungen höchste Priorität, ihr Tagesgeschäft abzuwickeln. Deshalb meinten sie, keine Zeit dafür zu haben, auf Abruf für das Projekt als Ansprechpartner zur Klärung von Anforderungen zur Verfügung zu stehen.
- Dadurch hatte das Entwicklungsteam keine festen Ansprechpartner für die jeweiligen Fachbereiche, welche befugt gewesen wären, über Anforderungen zu entscheiden.
- Für die Klärung offener Fragen mit den Fachbereichen musste das Entwicklungsteam einen Mitarbeiter in Vollzeit abstellen, der für die eigentliche Entwicklungsarbeit nicht mehr zur Verfügung stand.
- Der aktuelle Stand der Anforderungen war nicht für alle Mitarbeiter transparent, da keine etablierten Kommunikationswege zur Verbreitung dieser Informationen innerhalb der Fachbereiche existierten und auch keine fachbereichsübergreifenden.
- Dadurch hegten manche Mitarbeiter in den Fachbereichen falsche Erwartungen bezüglich der umgesetzten Anforderungen.
- Da ihre Erwartungen nicht erfüllt wurden, kam bei diesen Mitarbeitern Frust auf.
- Diese Frustrationen führten zur Verstimmungen: Die Fachbereiche fühlten sich nicht abgeholt; die IT hat wieder einmal das Falsche entwickelt, also hatte sich also im Grunde mit der Einführung agiler Vorgehensweisen nichts verbessert.

Lösung

Damit die Situation, die das Entwicklungsteam und der Scrum Master dem Lenkungsausschuss berichtete, nicht weiter eskalierte, vereinbarte die Unternehmensführung mit den Führungskräften der Mitarbeiter, die für das Projekt benötigt wurden, dass deren Zielvereinbarungen entsprechend anzupassen seien. Des Weiteren wurden im Rah-

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

men einer (nachgeholten) umfangreichen Umfeld- und Stakeholderanalyse die Schwachpunkte in der derzeitigen Kommunikationsstruktur innerhalb des Unternehmens ermittelt und entsprechende Gegenmaßnahmen ergriffen:

- eine einmalige Informationsveranstaltung, die von der Geschäftsführung für alle Mitarbeiter zum Projektstatus und zum aktuellen Stand der Anforderungen initiiert wurde
- regelmäßige Updates des Product Owners zum Projektstand für alle Mitarbeiter im Intranet
- regelmäßige persönliche Information der betroffenen Fachbereiche über den Projektverlauf durch die Führungskräfte
- Ernennung eines Ansprechpartners aus jedem Fachbereich und aus dem Management, der für Fragen zum Vorhaben allen Mitarbeitern zur Verfügung stand

Unterschiedliches Verständnis vom Inhalt einer Anforderung

Problem

Bei der Zusammenarbeit zwischen den Entwicklern und den Mitarbeitern aus den anfordernden Fachbereichen (bzw. dem Product Owner) bestand immer wieder eine grundsätzliche Herausforderung darin, dass alle Beteiligten dasselbe Verständnis von einer Anforderung und ihrer Realisierbarkeit erhielten (Bild 2).

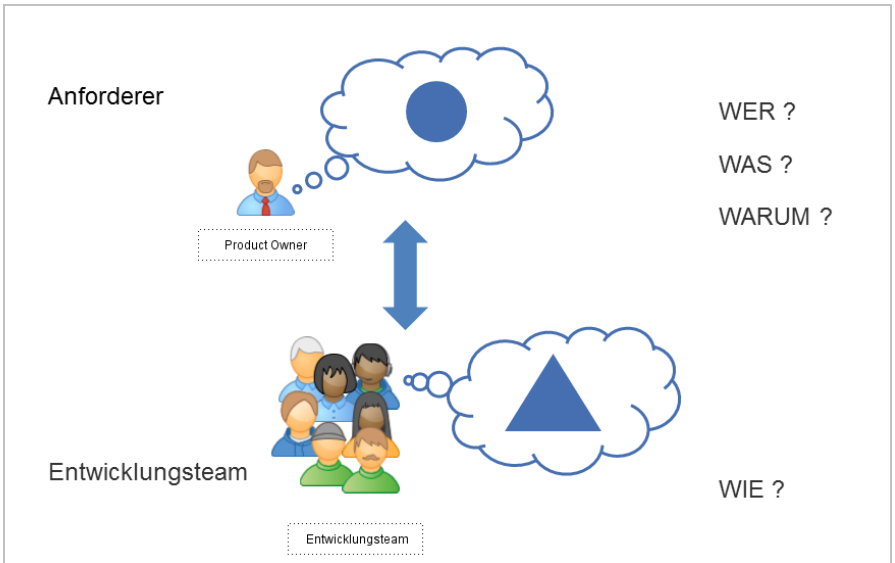


Bild 2: Zuständigkeiten von Anforderer und Entwicklungsteam bei der Anforderungsdefinition.

Warum die regelmäßige persönliche Abstimmung zwischen beiden Seiten so wichtig ist, lässt sich anhand eines Alltagsbeispiels veranschaulichen:

"Wie der Bundespräsident der Ukraine erklären lässt, dass er nicht nach Jalta fahren wird."

Angenommen, ein derartiger Satz – dieses Beispiel ist aus einer Tageszeitung entnommen – wäre Inhalt einer Anforderungsbeschreibung. Er ist nur teilweise verständlich und der logische Zusammenhang zwischen den Teilsätzen ist für den Leser nicht ersichtlich. Eine Klärung dieses Sachverhalts könnte natürlich in einem E-Mail-Austausch erfolgen. Wenn es sich aber um einen inhaltlich noch komplexeren Sachverhalt handelt, ist eine persönliche Abstimmung zwischen den Beteiligten effektiver, da dort weniger Raum für Interpretationen bleibt.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Lösung

Aufgrund schlechter Erfahrungen aus früheren Projekten wurde bei cyclespeed24 eine kontinuierliche persönliche Zusammenarbeit zwischen dem Product Owner, der die Anforderungen aus den Fachabteilungen bündelte, und dem Entwicklungsteam fest im Projektvorgehen verankert. Im sog. "Grooming", das alle zwei Wochen stattfand, wurde der Anforderungskatalog (Product Backlog) gemeinsam gepflegt (Bild 3).

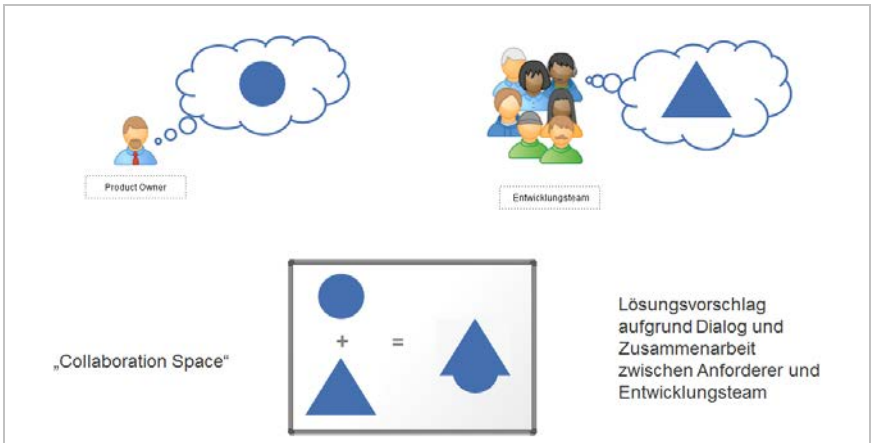


Bild 3: Gemeinsame Anforderungsklärung zwischen Product Owner und Entwicklungsteam.

Anforderer gibt technische Lösung vor

Problem

Ein wichtiges Merkmal einer User Story ist, wie bereits erwähnt, die Formulierung der Anforderung aus Anwendersicht. Der Anforderer macht sich also über das *Wer*, *Was* und *Warum* Gedanken. Das Entwicklungsteam ist gefordert, die geeignetste technische Lösung für die gestellte Anforderung auszuwählen und umzusetzen.

Allerdings kam es immer wieder vor, dass in der Anforderung bereits eine technische Lösung vorweg genommen wurde. Der Anforderer schrieb dem Entwicklungsteam also das *Wie* vor und nahm ihm den Entscheidungsspielraum, wie die User Story technisch umgesetzt werden könnte.

Vor allem bei Features, die architektonisch aufeinander aufbauen, kann es große Probleme verursachen, wenn der Anforderer nicht gewillt ist, der Argumentation des Entwicklungsteams zu folgen und die technische Lösung zu akzeptieren, die das Entwicklungsteam aufgrund seiner Fachkenntnis für die beste hält.

Lösung

Um zu vermeiden, dass der Anwender bei der Formulierung einer User Story die technische Lösung bereits vorwegnimmt, bietet es sich an, das Format "*Als <Anwender> will ich … <Ziel>, um … <Nutzen> zu erzielen*" zu verwenden. Dadurch wird der Fokus des Anwenders auf die für eine User Story wichtigen Aspekte gerichtet und das Abschweifen in technische Lösungsbeschreibungen vermieden.

Der letzte Teil der User Story, also der eigentliche Grund für diese Anforderung, wird bei der Formulierung der User Story gerne weggelassen. Um dies zu vermeiden, kann eine User Story auch dieses Format haben: "*Um … <Nutzen> zu erzielen, will ich als <Anwender> <Ziel> …* " Diese Art der Formulierung ist zugegebenermaßen etwas holprig, erfüllt aber den Zweck, den Nutzen einer User Story zu betonen, und erleichtert es, die User Stories nach Geschäftswert zu sortieren.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Ineffiziente Sprint Reviews

Problem

Die ersten Sprint Reviews von cyclespeed24 liefen meistens nach einem ähnlichen Muster ab: Das Entwicklungsteam brannte darauf, dem Product Owner auch bereits zu 90% fertiggestellte Funktionalitäten vorzuführen, die auf Basis der User Stories erstellt worden waren. Von den eingeladenen Mitarbeitern aus den Fachbereichen kam nur ein Teil zum Review, da sich das Projekt noch in der Entwicklung befand und die Führungskräfte der jeweiligen Mitarbeiter nicht wollten, dass diese regelmäßig zu diesen Terminen im Projekt gebunden waren.

Dies führte bei den ersten Sprint Reviews zu folgendem Ergebnis:

Der Product Owner konnte die vorgeführten Features nicht als "fertig" kennzeichnen, da

- sie nur in der lokalen Entwicklungsumgebung einwandfrei funktionierten.
- nicht alle notwendigen Vertreter aus den Fachbereichen anwesend waren, um die Features abzunehmen.
- die anwesenden Mitarbeiter unsicher waren, ob die Features die in den User Stories beschriebenen Anforderungen auch vollständig erfüllten.

So verursachten die ersten Sprint Reviews viel Enttäuschung im Entwicklungsteam, aber auch auf Seiten der anfordernden Fachbereiche wurden erste Stimmen laut, dass die IT "mal wieder am Bedarf vorbei" entwickeln würde.

Lösung

Um diesem Negativtrend entgegenzuwirken, wurden folgende Maßnahmen ergriffen:

1. Jede User Story erhielt genaue Akzeptanzkriterien, anhand derer möglichst automatisiert getestet werden konnte, ob die in der jeweiligen User Story enthaltenen Anforderungen auch erfüllt wurden.
2. Die Entwicklungsteams vereinbarten darüber hinaus mit den Fachbereichen eine "Definition of Done", also eine Definition, wann eine User Story als fertig gilt. Darin waren u.a. folgende Punkte enthalten:
    - Lauffähigkeit der User Story auf einer Integrationsumgebung (nicht der lokalen Entwicklungsumgebung)
    - Erfüllen der vorab definierten Akzeptanzkriterien
    - Fortlaufendes Schreiben eines Benutzerhandbuchs für Anwender und Administratoren
    - Erfüllen der übergreifend geltenden sog. "Nicht-Funktionalen-Anforderungen", z.B. Sicherheitsaspekte, Usability oder Performance
3. Die involvierten Mitarbeiter aus den Fachbereichen erhielten von ihren Vorgesetzten die Möglichkeit, an den Sprint Reviews teilzunehmen und sich vorher dafür vorzubereiten.
4. Alle beteiligten Mitarbeiter bekamen vor dem Sprint Review eine Information über die im Sprint enthaltenen User Stories inklusive der Akzeptanzkriterien.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Um die reibungslose Umsetzung der User Stories in einem Sprint zu unterstützen, wurde ein weiteres Instrument genutzt: die "Definition of Ready". Mit der Definition of Ready gibt das Entwicklungsteam dem Product Owner das Signal, dass eine User Story die erforderliche Reife besitzt, dass sie im Sprint Planning berücksichtigt werden kann.

Die Merkmale der Definition of Ready sind:

- Die User Story ist für alle Mitglieder des Entwicklungsteams klar verständlich.
- Die User Story ist testbar, d.h. es wurden entsprechende Akzeptanzkriterien festgelegt.
- Die User Story ist klein genug, dass sie neben weiteren User Stories in einem Sprint umgesetzt werden kann.

Nach den ersten Sprints wurde die Definition of Ready derartig erweitert, dass die Anforderung möglichst automatisiert testbar sein musste.

Unvorhersehbare Veränderungen im Projektumfang

Problem

Bei der Formulierung von Anforderungen an ein neues Softwaresystem wird vom Anforderer ein großes Abstraktionsvermögen abverlangt. Eine Software kann ja nicht angefasst werden; bestenfalls kann mittels Prototypen ein Überblick gegeben werden, wie die Software aussehen und sich verhalten könnte. Aber auch nur, wenn Geld für derartige "Wegwerfprodukte" ausgegeben werden darf.

In dem Projekt von cyclespeed24 wurden neue Features erstellt, welche aufgrund der veralteten Systemarchitektur bislang gar nicht entwickelt werden konnten. Die Anforderungen an das neue System waren daher auch für einen längeren Zeitraum in die Zukunft nicht immer bekannt, sondern entstanden quasi "auf der Strecke". Dies hatte zur Folge, dass Anforderungen an das Entwicklungsteam herangetragen wurden, die im Widerspruch zu bereits umgesetzten User Stories standen oder vorhandene Features wieder veränderten.

Lösung

Bei der Verwendung von agilen Vorgehensweisen müssen sich alle Beteiligten klar darüber sein, dass die Möglichkeit zum nahezu ständigen Einbringen von Anforderungen eine große Disziplin innerhalb der Fachbereiche erfordert, damit kein Anforderungschaos entsteht. Die gängigen Vorurteile gegenüber agilen Vorgehensweisen ("Ich bekomme immer weniger zu einem späteren Zeitpunkt und das zu einem höheren Preis") können nur dann abgebaut werden, wenn ein starker Product Owner ein klares Bild vom Produkt-/ Projektvorhaben hat und nichtzielführende Anforderungen konsequent ablehnt.

Dazu muss er neben dem weitreichenden Überblick über die Systemlandschaft aus fachlicher Sicht auch ein technisches Grundverständnis mitbringen. Auch sollte der Product Owner in der Lage sein, aufgrund der vorhandenen Entwicklungsgeschwindigkeit eine längerfristige Projektplanung durchzuführen. Da bei agilen Vorgehensweisen die Details zu Anforderungen zu einem möglichst späten Zeitpunkt festgelegt werden, muss die Geschäftsleitung den Product Owner mit den erforderlichen Befugnissen ausstatten, um auch entsprechend vorgehen zu können, ohne dafür vorher mit allen Stakeholdern Rücksprache halten zu müssen.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fazit

Die Verwendung von User Stories im Rahmen von agilen Vorgehensweisen ist kein "Selbstläufer". Auf den ersten Blick scheint das Vorgehen sehr einfach zu sein. Doch die Krux liegt hier im Detail.

Das Umdenken aller am Projekt beteiligten Abteilungen (Fachbereiche und IT) von einem "Lagerdenken" hin zu einer zielorientierten, pragmatischen und vor allem kollaborativen Zusammenarbeit erfordert zudem einen Kulturwandel und stellt für manche Unternehmen sicherlich eine große Herausforderung dar.

Dabei ist es zwingend notwendig, dass alle Führungskräfte der relevanten Fachbereiche diese Vorgehensweise unterstützen und für ihre Mitarbeiter die erforderlichen zeitlichen Freiräume zur regelmäßigen Zusammenarbeit mit dem Product Owner und dem Entwicklungsteam schaffen. Beschränkt sich die Arbeit mit User Stories auf die IT-Abteilung, ist dieses Vorgehen von vornherein zum Scheitern verurteilt, da die Entwickler im Projektverlauf immer wieder auf den Input der Fachbereiche angewiesen sind.

Letztendlich bietet die Arbeit mit User Stories neben der verbesserten fachübergreifenden Zusammenarbeit eine gleichmäßigere Auslastung der Fachbereiche, da hohe Arbeitsspitzen zur Erreichung von Meilensteinen vermieden werden. Dies wiederum ermöglicht eine bessere Einplanbarkeit der Mitarbeiter in den Fachbereichen und somit eine stärkere Vereinbarkeit der Projektarbeit mit dem Tagesgeschäft.

Literatur

- Cohn, Mike: Agile Estimating and Planning, Prentice Hall International, 2005 (ISBN-13: 978-0131479418)
- Cohn, Mike: User Stories Applied. For agile Software Development, Addison-Wesley Longman, Amsterdam 2004 (ISBN-13: 978-0321205681)
- Gottesdiener, Ellen: Requirements by Collaboration. Workshops for defining Needs, Addison-Wesley Longman, Amsterdam 2002 (ISBN-13: 978-0201786064)
- Leffingwell, Dean: Agile Software Requirements. Lean Requirements Practices for Teams, Programs, and the Enterprise, Addison Wesley, 2010 (ISBN-13: 978-0321635846)

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Agile Aufwandsschätzung in Scrum

Planning Poker – Techniken, Erfahrungen und Empfehlungen

Unzuverlässige Aufwandsschätzungen in der Softwareentwicklung sind eher der Normalfall als die Ausnahme. Um diesem Problem zu begegnen, sind im Lauf der letzten Jahrzehnte zahlreiche Verfahren zur Aufwandsschätzung in der Software-Entwicklung entstanden. Dabei ist in agilen Projekten das sog. Planungspoker (engl. "Planning Poker") weit verbreitet.

Dieser Beitrag stellt die Techniken und Verfahren vor, die für das Schätzen in Scrum mit Planning Poker erforderlich sind. Darüber hinaus erhalten Sie Empfehlungen, worauf Sie beim Einsatz von Planning Poker in der Praxis achten sollten.

Autor

Dr. Oliver Linssen

Diplom-Wirtschaftswissenschaftler, Geschäftsführer der Liantis GmbH, Lehrbeauftragter für Projektmanagement

Kontakt: oliver.linssen@liantis.com

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

› Zuverlässige Aufwandschätzung in agilen Teams

› Agile Softwareentwicklung mit Scrum und User Stories

sowie in den Rubriken:

› Aufwandsschätzung

› IT-Projekte

› Agiles Projektmanagement

Ziel des Planning Pokers

Der Product Owner erhält durch die Schätzung des Teams Aussagen darüber, wie "teuer" die Realisierung der einzelnen User Storys ist und verwendet diese Schätzungen, um die Einträge im Product Backlog zu priorisieren. Das Team erhält durch die Schätzung eine realistische Grundlage für die Planung der in den Sprints zu realisierenden Features.

Für beide Seiten ist es von Vorteil, dass in Scrum die Schätzung regelmäßig auf der Grundlage neuer Erkenntnisse überarbeitet wird. Dies ist eine wichtige Grundlage dafür, was der Betriebswirtschaftler auch als "rollierende Planung" bezeichnet: Eine gleitende Planung der Perioden in der Zukunft, wobei die nächste Periode konkret und die folgenden Perioden grob geplant werden.

Allgemeines zum Ablauf des Planungspokers

Aufwandsschätzungen finden in Scrum nicht nur zu Beginn des Projekts, sondern während des gesamten Projektverlaufs statt. Vor Beginn des ersten Sprints wird in einer oder mehreren Sitzungen der gesamte Inhalt des Product Backlogs geschätzt. Da sich das Product Backlog während des gesamten Projekts verändert, wird auch in jedem Sprint geschätzt. Dadurch fließen die aktuellen Erfahrungen über Produktivität des Teams, Komplexität der Technik und des Anwendungsgebiets und andere Randbedingungen in die Schätzung ein.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

An der Schätzung sind der Product Owner, das Team und der Scrum Master beteiligt. Der Product Owner erläutert die Anforderungen und beantwortet Fragen des Teams in Bezug auf die Anforderungen. Das Team schätzt den Aufwand, da es später ja auch für die Realisierung verantwortlich ist. Der Scrum Master moderiert die Veranstaltung. Die Schätzung ist eine reine Teamschätzung, d.h. der Product Owner und der Scrum Master schätzen selber nicht. Da alle Rollen bei der Schätzung beteiligt sind, ist das Verfahren sehr transparent.

! Die Schätzrunden sind ermüdend und deshalb zeitlich begrenzt: Während man durchaus die ganze Nacht pokert, sollten die einzelnen "Runden" beim Planungspoker nur etwa zwei Stunden dauern (Pichler, 2008).

Dass die Entwickler den Aufwand für die Entwicklung schätzen, ist keinesfalls eine Erfindung der agilen Softwareentwicklung. Die Anwendung dieses Prinzips erfolgt auch in Projekten, die nicht nach einem agilen Vorgehen ablaufen – denn die ausführenden Stellen geben immer noch die präzisesten Werte bei der Aufwandsschätzung ab.

User Storys

Im Product Backlog werden die funktionalen und nicht-funktionalen Anforderungen in Form von sog. User Storys erfasst (vgl. Cohn, 2004 sowie "Agile Softwareentwicklung mit Scrum und User Stories", Projekt Magazin 02/2010). Eine User Story beschreibt ein Feature, das eine zukünftige Anwendung enthalten soll. Weit verbreitet ist die Verwendung von Schablonen bei der Formulierung von User Storys. Häufig wird hierbei die Form verwendet: Als <Rolle des Anwenders> möchte ich <die gewünschte Möglichkeit>, sodass <Geschäftswert>. Die User Story wird häufig um die Beschreibung von Akzeptanztests ergänzt, mit denen die Implementierung der User Story getestet werden soll.

Story Points

Beim Planning Poker wird nicht geschätzt, wie hoch der Aufwand zur Realisierung der User Storys (in Zeiteinheiten) sein wird. Stattdessen schätzt man die relative Größe von User Storys zueinander. Die hierfür verwendete Maßeinheit wird als "Story Point" bezeichnet. Schätzt man 1 Story Point für die Realisierung einer User Story A und 3 Story Points für die Realisierung einer User Story B, bedeutet dies, dass der Zeitaufwand für die Realisierung von B vermutlich drei mal so groß sein wird wie für die Realisierung von A.

! Story Points sind eine relative Größe, die im Team durch Konsens festgelegt wird. Das bedeutet, dass unterschiedliche Teams mit unterschiedlich großen Story Points arbeiten. Über Teamgrenzen hinweg ist ein Vergleich von Story Points nicht möglich.

Fibonacci-Zahlen

Im Sprint Planning Meeting schätzt jedes Teammitglied den Aufwand so, als ob er das Feature selber realisieren würde. Zum Schätzen hat jeder Teilnehmer einen Satz Karten, auf denen eine Fibonacci-Zahl zwischen 1 und einem vorher festgelegten Höchstwert (z.B. 21) steht (1, 2, 3, 5, 8, 13, 21). Fibonacci-Zahlen werden gebildet, in dem man die Summe der beiden vorhergehenden Fibbonacci-Zahlen addiert. Sie werden deshalb gerne verwendet, weil die Abstände zwischen den einzelnen Werten immer größer werden. Damit soll den Schätzern deutlich bleiben, dass der Aufwand zur Realisierung einer Story mit dem Wert 21 im Vergleich zu einer Story mit dem Wert 13 sehr viel größer ist als zwischen zwei Storys mit den Werten 3 und 5.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Die Verwendung von Fibonacci-Zahlen ist nur eine Möglichkeit, die Komplexität der User Storys zu schätzen (Cohn, 2005). Alternativ könnten z.B. auch die Werte 1, 2, 4, und 8 verwendet werden. Auch kleinste Anforderungen sollten aber immer den Wert 1 erhalten, da die Realisierung einer Anforderung nie ohne Aufwand erfolgen kann. Schätzungen zwischen den vereinbarten Werten einer Skala werden nicht zugelassen. Für weniger konkrete User Storys am unteren Ende des Product Backlog sind auch höhere Werte (z. B. 100) zugelassen. Diese User Storys sind häufig noch vage formuliert und ihre Realisierung steht erst in späteren Sprints an. Cohn schlägt für solche Storys den Begriff "Epos" vor.

Kalibrierung

Damit die Schätzungen der Teammitglieder vergleichbar sind, muss man die Schätzung justieren. Hierfür einigt man sich auf eine (oder mehrere) User Storys, deren Realisierungsaufwand man als "mittelgroß" einschätzt. Dieser Referenzstory ordnet man dann z.B. den Wert 5 zu. Die übrigen Größen werden im Vergleich zu diesem Wert festgelegt.

Empfehlung für die Praxis

Die Maßeinheit Story Point sollte im Team nicht zu groß gewählt werden. Wenn ein Story Point beispielsweise etwa drei Personentagen entspricht, dann benötigt die Realisierung einer User Story mittlerer Komplexität (die also den Wert 5 erhalten hat) schon 15 Personentage. Eine große User Story (die den Wert 13 erhalten hat) benötigt zur Realisierung bereits 39 Personentage. Abgesehen davon, dass bei solchen Größen der Arbeitsfortschritt (Fertigstellungsgrad) im weiteren Projektverlauf nicht mehr genau genug bestimmt werden kann, lässt bei solchen Größen die Schätzgenauigkeit erheblich nach. Eine – wenn auch pauschale – Empfehlung ist, dass eine User Story mit dem Wert 1 an einen Personentag realisiert wird. Eine "mittelgroße" User Story würde also einem Aufwand entsprechen, den man in 5 Personentagen realisieren kann.

Aufwandsschätzung mit Planning Poker

Das Team geht im Sprint Planning Meeting alle zu schätzenden User Storys durch. Am Anfang muss das Team alle vorhandenen Storys schätzen, in den folgenden Sprints werden in der Regel nur neue Storys geschätzt sowie Storys, bei denen das Team der Meinung ist, dass der bisher geschätzte Aufwand nicht mehr korrekt ist. Alle Teammitglieder halten gleichzeitig die Karte mit den geschätzten Aufwänden für die Realisierung hoch (oder legen sie hin) und vergleichen ihre Ergebnisse. Nun sind drei Fälle zu unterscheiden (Schwaber, 2010):

1. Wenn alle Teammitglieder die gleiche Schätzung abgegeben haben, übernimmt man diesen Wert für die User Story.
2. Wenn die Schätzwerte der Teammitglieder nahe beieinander liegen, übernimmt man den größeren Wert als Schätzung für die User Story. Beispiel: Vier Teilnehmer schätzen 3 Story Points, drei Teilnehmer schätzen 5 Story Points. Als Ergebnis werden 5 Story Points für die Schätzung übernommen. Damit ist man auf der sicheren Seite und hat auch die Schätzung von Teammitgliedern berücksichtigt, die nach eigener Einschätzung mehr Zeit für die Realisierung benötigen würden.
3. Liegen die Schätzwerte zu weit auseinander, diskutiert man die abweichenden Ergebnisse. Nach der Diskussion pokert man den Aufwand erneut und vergleicht wieder die Ergebnisse. Beispiel: Die Bandbreite der

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Schätzungen reicht von 2 Story Points bis 8 Story Points. Über den Aufwand muss diskutiert werden, und anschließend schätzen alle Team-Mitglieder mit dem neuen Hintergrundwissen die Story erneut. In der Regel nähern sich die Ergebnisse schon bei der zweiten Schätzung deutlich an.

Eine Diskussion bei stark voneinander abweichenden Werten ist absolut notwendig, um die verschiedenen Sichtweisen der Teilnehmer einzuholen und deren Erfahrungen in die Schätzung einfließen zu lassen. Es ist nicht zu empfehlen, automatisch den höheren Wert anzunehmen. Denn dies verschiebt bei einer größeren Anzahl von User Storys den geschätzten Gesamtaufwand u.U. zu sehr nach oben. Eher ist es sinnvoll, über alle großen Abweichungen zu diskutieren.

In der Regel erzielt man im Team eine Einigung über den geschätzten Aufwand. Ist dies nicht möglich, gibt es zwei Möglichkeiten: Entweder man zerlegt die Story in kleinere Storys, die sich besser schätzen lassen oder das Team muss sich mehr Wissen über den Inhalt der Story aneignen, damit es sie präziser schätzen kann. Diese Recherche wird als "Spike" (der Begriff stammt aus dem Extreme Programming) bezeichnet. Im Rahmen eines Spikes könnte z.B. ein Prototyp entwickelt werden.

Personenspezifische Schätzung

Bei Scrum schätzt jeder den Aufwand so, als ob er die User Story selber realisieren würde. Dies ist vernünftig, weil er möglicherweise im Sprint tatsächlich die Realisierung der User Story übernimmt. Die Produktivität und die Qualifikation der Entwickler in einem Team ist aber in der Regel unterschiedlich. Wenn man im Planungspoker dazu neigt, sich an den höheren Schätzwerten zu orientieren, wird der geschätzte Aufwand tendenziell wesentlich höher ausfallen als der tatsächlich realistische – dagegen gibt es kein Patentrezept. Im besten Fall kann das Team intern die Aufgaben so verteilen, dass die einzelnen Teammitglieder möglichst selten Aufgaben erhalten, mit denen sie sich nicht gut auskennen.

Dominante Personen

Sehr erfahrene Entwickler, Meinungsführer, Experten oder dominante Persönlichkeiten, können das Ergebnis der Schätzungen im Planungspoker wesentlich beeinflussen. Natürlich kann es sinnvoll sein, sich an der Schätzung von Kollegen zu orientieren, wenn diese z.B. eine große Erfahrung mitbringen. Negative Einflüsse entstehen allerdings, wenn ein Teammitglied statt hoher Kompetenz einfach nur die Meinungshoheit hat und dadurch die Schätzung in eine bestimmte Richtung (nach oben oder unten) beeinflusst. Dem Team sollte diese Gefahr deutlich sein bzw. der Scrum Master muss hierauf hinweisen. Letztendlich muss das Team selber darauf achten, dass die Meinung aller Teammitglieder das gleiche Gewicht hat; die Schätzung des Aufwands ist eine Teamentscheidung, die auch vom gesamten Team getragen werden muss.

Magic Number

Der höchste der vereinbarten Werte (hier: 21) hat eine besondere Bedeutung. Schätzt jemand den Wert mit 21, bedeutet dies nicht, dass er den 21-fachen Aufwand von 1 vermutet. Vielmehr ist damit gemeint, dass er den Aufwand für nicht genau abschätzbar, sondern sehr groß hält. Solche Storys müssen zerlegt werden oder das Team führt einen Spike durch, wenn die Story sich nicht weiter zerlegen lässt. (s.o.)

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Sprintkapazität

Die Umrechnung der Story Points in Personentage basiert auf der Erfahrung, wie viele Story Points im Durchschnitt in einem Sprint in bisherigen Projekten (am besten vom gleichen Team) umgesetzt wurden. Dies wird als "Sprintkapazität" (auch Team-Kapazität) bezeichnet. Bei der Berechnung der Sprintkapazität muss berücksichtigt werden, wie viel Arbeitszeit tatsächlich für die Arbeit im Projekt zur Verfügung steht. Wochenenden, Feiertage, Besprechungen sind in gleicher Weise abzuziehen wie Zeiten für krankheitsbedingten Ausfall oder Reisezeiten, die als Arbeitszeiten gelten. Die Sprintkapazität ist also die Netto-Arbeitszeit, die dem Team tatsächlich zur Realisierung der User Storys im Sprint zur Verfügung steht. Hier ist die Verwendung eines realistischen Werts absolut notwendig.

Beispiel: Bei einer Sprintlänge von 30 Kalendertagen (also etwa 21 Arbeitstagen) geht man davon aus, dass jedes Teammitglied ca. 14 Arbeitstage für die Realisierung der Storys Points zur Verfügung hat. Wenn ein Team aus fünf Entwicklern besteht, erhält man somit eine Sprintkapazität von 70 Personentagen.

Mit Hilfe der Sprintkapazität lässt sich grob abschätzen, wie lange das Team für die Realisierung eines Story Points benötigt. Hat das Team z.B. bisher im Mittel 35 Story Points je Sprint realisieren können, dann benötigt es für die Realisierung eines Story Points etwa 2 Personentage. Der Wert 35 wird als "Velocity" bezeichnet und bezeichnet den geschätzten "Output" je Sprint, gemessen in Story Points (s. nachfolgender Abschnitt).

Empfehlung für die Praxis

Die Sprintkapazität eines Teams sollte nicht geschätzt werden. Hier ist der Einsatz eines Zeiterfassungssystems sinnvoll, in dem feingranular erfasst wird, wofür die Arbeitszeiten aufgewendet werden. Auf diese Weise lässt sich ermitteln, wie viel Zeit tatsächlich für Besprechungen, "Tagesgeschäft", Systemarbeiten etc. aufgewendet wird. Realistische Werte enthalten durchaus "Sprengstoff" innerhalb einer Organisation: Bei einem Projekt mit einem hohen Overhead wurde schon festgestellt, dass die Sprintkapazität des Teams weniger als 50% der gesamten Arbeitszeit betragen hat.

Velocity

Die Velocity ist eine Angabe über den Output, den ein Team in einem Sprint produziert, also wie viele Story Points in einem Sprint umgesetzt werden konnten ("umgesetzt" bezeichnet hier lauffähigen, getesteten, dokumentierten und gelieferten Code. Siehe hierzu auch den Abschnitt "Definition of Done"). Cohn diskutiert drei Möglichkeiten, die Velocity eines Teams zu bestimmen (Cohn, 2005):

1. Verwendung historischer Daten
2. Durchführung einer oder mehrerer Iterationen, um die Velocity zu bestimmen
3. Prognose einer Velocity

Die beste Möglichkeit ist, einen oder mehrere Sprints durchzuführen und dadurch die Velocity zu bestimmen. Hierbei ist zu berücksichtigen, dass ein Team zu Beginn eines Projekts eine geringere Produktivität hat. Die Velocity eines Teams sollte also nur mit Vorsicht als konstante Größe angesehen werden – in der Regel schwankt dieser Wert (Cohn, 2009).

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Gibt es keine Möglichkeit, einen oder mehrere Sprints durchzuführen, um die Velocity zu bestimmen, kann man ersatzweise einen ersten Wert aus historischen Daten bestimmen. Dieses Verfahren lässt sich wie folgt – stark vereinfacht – skizzieren:

- Zu Anforderungen, die man in der Vergangenheit in einem bestimmten Zeitfenster realisiert hat, formuliert man User Storys.
- Für diese User Storys führt man nachträglich einen Planungspoker durch.
- Schließlich dividiert man den Zeitaufwand, der für die Realisierung benötigt wurde, durch die Summe der im Planungspoker ermittelten Story Points. Wurden beispielsweise in 2959 Stunden 269 Story Points realisiert, hat man für einen Story Point 11 Stunden benötigt.

Berechnung der Sprintkapazität

Sprintkapazität (SK) = Netto-Arbeitszeit x Teammitglieder

Berechnung der Velocity

Velocity (V) = SK / Dauer 1 Story Point (SP)

Beispiel: Berechnung der Dauer eines Story Points

SK = 14 Personentage x 5 = 70 Personentage

V = SK / SP → SP = SK / V

SP = 70 Tage / 35 = 2 Tage

Der dadurch ermittelte Wert gibt an, wie viele Stunden (oder Tage) für die Realisierung eines Story Points benötigt wurden. Mit Hilfe der Sprintkapazität des Teams hat man auf diese Weise einen ersten Wert für die Velocity bestimmt.

Empfehlung für die Praxis

Man benötigt mehrere Sprints, bis ein brauchbarer Wert für die Velocity des Teams zur Verfügung steht. Insbesondere bei neu gebildeten Teams, neuer Technologie und einem bisher unbekannten Fachgebiet ist aktuell kein anderer Weg erkennbar, um die Velocity zu ermitteln. Aber auch hier ist zu bedenken, dass ein neues Team, welches u.U. mit einer wenig vertrauten Technologie arbeitet und sich möglicherweise auch noch mit der Anwendungsdomäne nicht auskennt, in den frühen Sprints eine erheblich geringere Velocity aufweisen wird als in darauffolgenden. Das Team sollte am Ende jedes Sprints die Velocity berechnen und mit den Werten aus früheren Sprints vergleichen. Anschließend muss das Team entscheiden, welche Velocity für die zukünftigen Sprints zugesagt wird. Hierbei ist auch die Sprintkapazität regelmäßig zu überprüfen.

Grobe Schätzung der Projektdauer

Kennt man die Sprintkapazität des Teams und ist man in der Lage, eine Schätzung für alle User Storys im Product Backlog vorzunehmen, kann man eine grobe Schätzung der Gesamtdauer des Projekts vornehmen, indem man die Gesamtzahl der Story Points durch die Velocity des Teams dividiert.

Beispiel: Schätzt man den Gesamtaufwand aller User Storys im Product Backlog mit 200 Story Points, dann benötigt man bei einer Velocity von 35 Story Points ungefähr 6 Sprints. Bei einer Sprintlänge von vier Wochen beträgt damit die Gesamtdauer etwa ein halbes Jahr. Hierbei ist allerdings äußerste Vorsicht geboten, und dieser Wert darf nur als grobe Schätzung verstanden werden, weil

- die User Storys am unteren Ende des Product Backlogs naturgegeben noch sehr vage sein können und

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- die User Storys im Product Backlog, die nicht im aktuellen Sprint umgesetzt werden, vom Product Owner jederzeit verändert werden dürfen.

Definition of Done

Bei der Schätzung der Aufwände ist es absolut unerlässlich, dass sich das Team darauf einigt, was alles zur Realisierung einer User Story gehört. Dies wird als "Definition of Done" bezeichnet. Die Realisierung einer User Story bedeutet nicht nur, dass die geforderte Funktionalität programmiert ist. Es bedeutet außerdem (Bleek, Wolf, 2008):

- Die Funktionalität "läuft",
- sie ist getestet,
- sie ist dokumentiert,
- sie ist ausgeliefert und
- der Kunde hat die Lieferung akzeptiert.

Wird beim Planungspoker nur der Aufwand für die Programmierung geschätzt, kann man hilfsweise z.B. von folgender Verteilung der Arbeitszeit ausgehen:

- 1/3 für die Programmierung
- 1/3 für die Tests
- 1/6 für die Analyse
- 1/6 für die Dokumentation

Das bedeutet, dass der Gesamtaufwand ungefähr mit drei multipliziert werden müsste, wenn beim Planungspoker nur der Implementierungsaufwand geschätzt wurde.

Zuschläge für Dokumentation, Projektmanagement etc.

Schätzt man nur die Aufwände für die Realisierung, würde sich nach der genannten Faustformel der geschätzte Gesamtaufwand verdreifachen. Diesen Wert muss man jedoch stets kritisch hinterfragen. Bei kleinen Projekten ist der ermittelte Gesamtaufwand regelmäßig zu hoch, bei großen Projekten kann er durchaus zu niedrig sein. So wurde z.B. in einem großen Projekt festgestellt, dass der Aufwand zur reinen Implementierung nur ca. 20% des Gesamtaufwands betragen hat. Tendenziell ist hier davon auszugehen, dass der relative Anteil für die Programmierung mit zunehmender Größe des Projekts abnimmt.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Weitere Empfehlungen

Kleine und konkrete User Storys

Für die Zuverlässigkeit der Schätzung hat es sich als sinnvoll erwiesen, kleine und möglichst konkrete User Storys zu verwenden. Wir haben gute Erfahrungen damit gemacht, als Richtgröße für die User Storys einen Personentag anzustreben. Die Zuverlässigkeit der Schätzung nimmt ab, wenn User Storys zu groß oder so vage sind, dass sich die Entwickler keine konkrete Vorstellung vom notwendigen Aufwand machen können. Aus diesem Grund sollten User Storys mit dem Maximalwert (hier 21) nicht in die Schätzung für die Sprints eingehen. Dieser Wert signalisiert häufig, dass eigentlich eine Schätzung "aus dem Bauch" vorgenommen wurde. Wenn man nicht klar verstanden hat, worum es in einer User Story geht, sollte man hierfür den Aufwand nicht schätzen.

Viele User Storys

Große User Storys sollten folglich in mehrere kleine User Storys zerlegt werden. Eine gute Faustregel ist, dass die User Storys umso kleiner und konkreter sein sollten, je weiter sie oben im Product Backlog stehen.

Bei einer großen Anzahl von User Storys gleichen sich Fehleinschätzungen nach oben und nach unten gegeneinander aus (Pichler, 2008). Problematisch wird das Verfahren, wenn es nur wenige User Storys zu schätzen gibt. Hier ist die Schätzung des Gesamtaufwands mit einer hohen Unsicherheit behaftet.

Breit qualifizierte Entwickler im Team

Durch eine funktionale Arbeitsteilung entstehen hochgradig spezialisierte Entwickler. Dies kann sich nachteilig auf die Schätzung auswirken, weil die Entwickler sehr hohe Werte bei der Realisierung von User Storys angeben, die außerhalb ihrer Hauptqualifikation liegen. Ein GUI-Spezialist schätzt beispielsweise den Aufwand für die Realisierung von Datenbankoperationen hoch ein. Für den Datenbankbankspezialisten erscheint die Programmierung des Front-Ends extrem aufwändig, weil er sich mit den hierfür eingesetzten Werkzeugen nicht auskennt. Entwickler in Scrum-Teams sollten aus diesem Grund im Idealfall über eine umfassende Qualifikation verfügen (Linssen, 2009).

Trotzdem schätzt jeder im Team den Aufwand für sämtliche User Storys, da man nicht sicher ausschließen kann, dass immer das qualifizierteste/schnellste Teammitglied tatsächlich die Story im Sprint auch realisiert. Im Planning Poker spielt es eben keine Rolle, wie schnell der "schnellste" Entwickler ist, sondern wie schnell das Team ist – und zum Team gehören nun mal auch alle anderen, die weniger schnell sind. Dadurch lassen sich völlig unrealistische Schätzungen von "Assen" und "Experten" vermeiden, die dann hinterher (wenn "normale" Entwickler "am Werk" sind) überhaupt nicht eingehalten werden können.

Vergleich der Schätzungen untereinander

Story Points sind relative Größen. Unproblematisch ist, dass diese Größe vom Team und dem Projekt abhängt. Problematisch ist dagegen, wenn sich die Größe der Story Points im Zeitverlauf verändert oder die Story Points inkonsistent verwendet werden.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- Eine **Veränderung im Zeitverlauf** liegt vor, wenn eine User Story in zwei verschiedenen Schätzrunden geschätzt werden muss und in einem früheren Sprint ein anderer Wert zugeordnet wurde als in einem späteren Sprint. Diese Veränderung kann berechtigt sein, wenn man neue Erkenntnisse hinzugewonnen hat. Ist dies nicht der Fall, so ist der veränderte Schätzwert zu hinterfragen.
- Eine **inkonsistente Schätzung** liegt vor, wenn tatsächlich ähnlich komplexe User Storys unterschiedlich groß geschätzt werden oder wenn tatsächlich unterschiedlich komplexe User Storys den gleichen Wert erhalten.

Empfehlenswert ist hier der ständige Vergleich von Anforderungen untereinander, um eine in sich schlüssige Schätzung zu erhalten (Pichler, 2008).

! Wenn es möglich ist, sollten für alle beim Planungspoker verwendeten Werte mehrere Referenzbeispiele vorhanden sein.

Keine Puffer, keine Scheinpräzision

Die Aufgabe, Entwicklungsaufwände zu schätzen, ist bei Entwicklern chronisch unbeliebt, weil sie dafür häufig hinterher zur Rechenschaft gezogen werden, wenn die Schätzung so nicht eintrat. Deshalb bauen Entwickler Puffer in ihre Schätzung ein. Vermeintlich ist man dadurch in der Lage, nun präzise den Aufwand zu "berechnen". Tatsächlich wird hier eine Präzision vorgegaukelt, die hauptsächlich aus "Luft" (dem Puffer) besteht.

! Bei Scrum wird die Unsicherheit einer Schätzung toleriert. Es muss allen Beteiligten klar sein, dass keine präzisen Schätzwerte zu erwarten sind und dass Schätzungen in späteren Sprints revidiert werden können. Die Erfahrung zeigt, dass Schätzungen durch Erfahrung genauer werden. Aus diesem Grund sollten in die Schätzungen keine Puffer eingebaut werden.

Fazit

Planning Poker lässt sich schnell und ohne großen Aufwand einführen. Es schätzen die Personen den Aufwand, die hinterher auch für die Realisierung zuständig sind. Die Aufwandsschätzung ist eine Teamschätzung, das heißt, sie orientiert sich nicht an der Geschwindigkeit der schnellsten Mitarbeiter. Maßgebend ist die Geschwindigkeit, die das ganze Team glaubt, einhalten zu können, bzw. die es in der Vergangenheit hat realisieren können. Damit ist Planning Poker gegenüber anderen Verfahren, die mit idealtypischen Annahmen bzgl. Produktivität arbeiten, weit überlegen. Die im Projekt gesammelten Erfahrungen fließen in die Schätzungen ein, was sich positiv auf die Präzision der Schätzung auswirkt.

Planning Poker ist allerdings wenig geeignet, um zu Beginn eines Projekts den Gesamtaufwand "präzise" vorherzusagen, was für Manager eher unbefriedigend ist. Darüber hinaus verspricht Planning Poker keine "präzise" Schätzung. Wenn man nur wenige Storys schätzt, ist die Unsicherheit bei der Schätzung recht hoch. Sie nimmt tendenziell mit der Anzahl der zu schätzenden Storys ab. Bei der Anwendung des Verfahrens ist darauf zu achten, kleine User Storys (z.B. ein Personentag) zu verwenden. Außerdem ist eine realistische Einschätzung der Sprintkapazität wichtig, da diese tendenziell zu hoch angesetzt wird.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Literatur

- Beedle, M.; Schwaber K.: Agile Software Development with Scrum, Upper Saddle River, N.J., Prentice Hall, 2002
- Bleek, W.-G.; Wolf H.: Agile Softwareentwicklung: Werte, Konzepte und Methoden, 1. Aufl., Heidelberg, dpunkt Verlag, 2008
- Boehm, B. W.: Software Engineering Economics, Prentice Hall, 1981
- Bundschuh, M.; Dekkers C.: The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement, 1. Aufl., Berlin, Heidelberg, New York, Springer, 2008
- Cohn, M.: User Stories Applied: For Agile Software Development, Upper Saddle River, N.J., Addison-Wesley Longman, Amsterdam, 2004.
- Cohn, M.: Agile Estimating and Planning, Upper Saddle River, N.J., Prentice Hall International, 2005
- Cohn, M.: Succeeding with Agile: Software Development Using Scrum, 1. Aufl., Upper Saddle River, N.J., u.a.: Addison-Wesley Longman, 2009
- GPM Deutsche Gesellschaft für Projektmanagement, M. Gessler und SPM Swiss Project Management Association, Kompetenzbasiertes Projektmanagement (PM3): Handbuch für die Projektarbeit, Qualifizierung und Zertifizierung auf Basis der IPMA Competence Baseline Version 3.0, 2. Aufl., Nürnberg, GPM Deutsche Gesellschaft für Projektmanagement, 2009
- Koschek, H.: Geschichten vom Scrum: Von Sprints, Retrospektiven und agilen Werten, 1. Aufl., Heidelberg, dpunkt Verlag, 2009
- Linssen, O.: Agile Vorgehensweisen – Wandel im Management von IT-Projekten, in 26. Internationales Deutsches Projektmanagement Forum: Die Kunst des Projektmanagements. Inspiriert durch den Wandel: Tagungsband, 14./15. Oktober 2009, bcc Berliner Congress Center, Nürnberg, 2009, S. 330 - 339.
- McConnell, S.: Software Estimation: The Black Art Demystified, 1. Aufl., Redmond, Washington, Microsoft Press Corp., 2006
- Pichler, R.: Scrum – Agiles Projektmanagement erfolgreich einsetzen, 1. Aufl., Heidelberg, dpunkt Verlag, 2008
- Schwaber, K.: Agile Project Management with Scrum, Redmond, Washington, Microsoft Press Corp., 2004
- Schwaber, K.: Scrum im Unternehmen, 1. Aufl., Unterschleißheim, Microsoft Press Deutschland, 2008
- Schwaber, K.: Scrum-In-Depth. It Depends on Common Sense, Ver. 1.3. o.O., 2010
- Wirdemann, Ralf: Agile Softwareentwicklung mit Scrum und User Stories, Projekt Magazin 02/2010

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

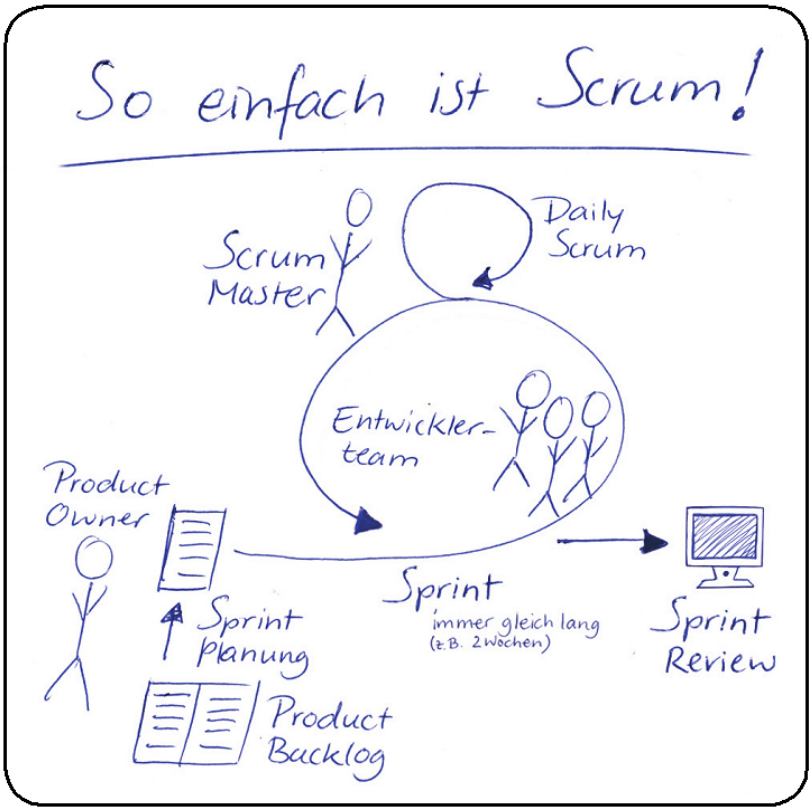


Bild 1: Die Grundlagen von Scrum sind so einfach, dass sie auf einen Bierdeckel passen.

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

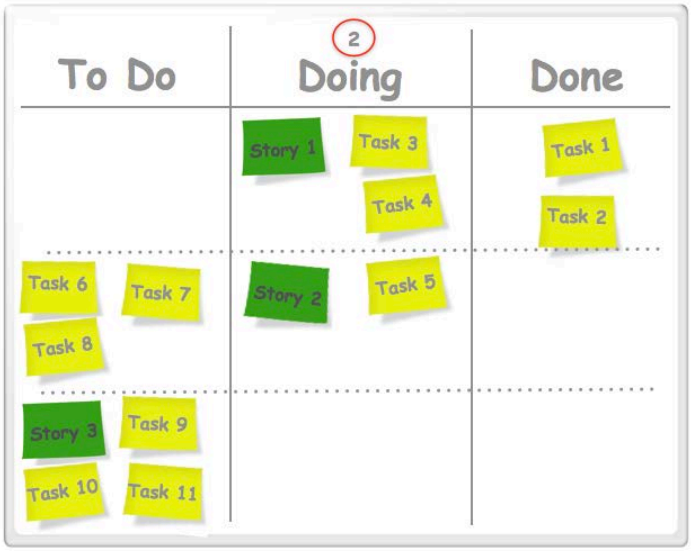


Bild 2:Scrum-Taskboard, bei dem ein Limit für User Stories eingeführt wurde, an denen gleichzeitig gearbeitet werden darf.

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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!

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

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

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Das Beste aus zwei Welten kombinieren

Scrum + Critical Chain = Reliable Scrum

Die Erfolge von agilen Methoden, insbesondere von Scrum, sind unbestritten. Die enge Zusammenarbeit im Team und die hohe Produktorientierung beflügelt die Kreativität aller Beteiligten. Die vielen Iterationen sowie die starke Einbindung des Auftraggebers stellen den wirklichen Nutzen in den Mittelpunkt. Die Abschottung der Teams während der Sprints verhindert das schlimmste Multitasking und die Retrospektiven sorgen für eine kontinuierliche Verbesserung.

Doch in der Realität fällt es vielen Organisationen immer noch schwer, mit Scrum zu arbeiten. Auch wenn Scrum einfach, eingeübt und schon weit verbreitet ist, sind die Ergebnisse oft nicht zufriedenstellend und es kommt immer wieder zu Verzögerungen. Dieser Beitrag beschreibt eine Möglichkeit, wie sich Scrum optimieren und die Planbarkeit deutlich verbessern lässt: Das "Reliable Scrum" genannte Vorgehen kombiniert Scrum (Schwaber, Sutherland, 2011) und die Critical-Chain-Methode (Goldratt, 2002; Müller, 2010) zu einer Projektmanagementmethodik, mit der die beiden Ziele "Termintreue" und "Erfüllung des Leistungsumfangs" ausgewogen gesteuert werden können.

Autor

Wolfram Müller

Dipl.-Ing. Mechatronik und Dipl.-Ing. Maschinenbau selbst. Berater, Geschäftsführer Speed4Projects

Kontakt:
wolfram.mueller@speed4projects.net

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in den Rubriken:

› Theory of Constraints

› IT-Projekte

› Best Practices

Service-Links

› PM-Einführung und –optimierung

Warum Reliable Scrum?

Bevor wir uns die Funktionsweise von Reliable Scrum genauer ansehen, werfen wir einen Blick in die Praxis und betrachten anhand einiger Beispiele, warum es für Reliable Scrum überhaupt einen Bedarf gibt.

Scrum kann keinen Scope garantieren – It's done when it's done

Angenommen, Sie sind für ein großes Projekt verantwortlich, in dem ein wichtiger Teil agil entwickelt wird, der zu einem definierten Zeitpunkt fertig sein muss. Mit einem Product-Burndown-Chart gibt Ihnen das Scrum-Team jederzeit volle Transparenz über den aktuellen Stand; Sie sehen nach jedem Sprint, ob der Termin gehalten wird oder nicht. Wenn Sie das Team fragen, ob alles rechtzeitig fertig wird, erhalten Sie als Antwort: "Wir liefern das Beste, was wir können", aber: "It's done when it's done!" Zuverlässigkeit bzgl. Scope und Termin ist aber bei Projekten mit vielen zu integrierenden Teilen unverzichtbar – als Verantwortlicher haben Sie jetzt ein Problem.

Oder stellen Sie sich z.B. ein börsennotiertes Unternehmen vor. Hier gilt es, große Innovationen zu schaffen. Diese werden auf Messen angekündigt, die Vertriebspipelines müssen gefüllt werden und vor allem muss rechtzeitig Marketing betrieben werden. Aber was, wenn das Produkt dann nicht verfügbar ist? Auf das fertige Produkt

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

warten und dann erst mit dem Marketing zu beginnen, verschenkt wertvollen Vorsprung. Zuverlässigkeit bzgl. Scope und Termin sind entscheidend.

Diffuses Erwartungsmanagement

In einem Scrum-Projekt repräsentiert der sog. Product Owner die Stakeholder und managt das Backlog (d.h. die Liste der zu realisierenden User Stories). Bei mehreren Stakeholdern wird es aber schnell schwierig, alle gewünschten Funktionen (formuliert in sog. User Stories, im Folgenden "Stories") aufzunehmen, denn dies kann u.U. das Projekt sprengen. Aber woher weiß der Product Owner, was zu viel ist? Und wie wird das transparent? Das Hauptproblem ist hier, dass die Stakeholder alle Wünsche äußern können, ohne dass sowohl der Preis als auch die Folgen transparent werden. Die Konsequenz ist eine unklare Erwartungshaltung.

Wunsch-Features dienen als Puffer

Selbst wenn etwas pünktlich fertig wird, ist die Enttäuschung bei den Stakeholdern häufig groß – denn was ist mit all den versprochenen Wunsch-Features? ("Should" und "Could" in der MoSCoW-Terminologie, s. Begriffsdefinition im Glossar des Projekt Magazins www.projektmagazin.de/glossar.) Ich als Stakeholder kann doch davon ausgehen, dass Wünsche erfüllt werden? Scrum arbeitet stark mit Wunsch-Funktionen, die im Notfall als Puffer verwendet werden. Das Problem dabei ist, dass die Stakeholder hören, dass viele Funktionen kommen werden und dann enttäuscht sind, wenn etwas entfällt.

Kein geeignetes Werkzeug für Planung und Steuerung

Dem Product Owner steht in der Regel kein geeignetes Werkzeug zur Verfügung, mit dem er genau erkennen kann, wie viele Stories er in ein Release noch ziehen kann oder daraus entfernen muss. Und wann kann er welche Funktionalität versprechen, wenn sich das Backlog oder die Velocity (Abarbeitungsgeschwindigkeit) laufend ändern?

Backlog und Velocity sind variabel

In vielen Fällen werden das Backlog und die Velocity als unveränderliche Größen betrachtet. Doch sowohl im Backlog (z.B. durch unvorhergesehen Ereignisse oder neue Wünsche) als auch in der Velocity (z.B. durch krankheitsbedingte Ausfälle) gibt es Streuungen, die nicht transparent sind und die mit den Stakeholdern nicht verhandelt wurden. Ein typisches Beispiel ist die Ablösung eines Altsystems. Hier tauchen im Laufe des Projekts immer wieder ungeahnte, nicht dokumentierte Probleme auf, die in Form von zusätzlichen Stories bearbeitet werden müssen. Dadurch müssen auch die Termine jedes Mal neu angepasst werden und die Stakeholder verlieren das Vertrauen in eine termingerechte Fertigstellung.

Stories enthalten Puffer

Für das Team sind wiederum die Sprints heilig – sie versprechen Ruhe vor den Stakeholdern. Für die Stakeholder sind es aber die einzigen Punkte, an denen sich der Fortschritt messen lässt – diese Transparenz ist für die Stakeholder existenziell. Vor dem Sprint einigt sich das Team daher auf eine bestimmte Menge an Stories, die es im Sprint umsetzen möchte und die es damit zusichert. Nach dem Sprint wird abgerechnet (oder neudeutsch "re-

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

viewed"). Wenn eine Story dann nicht rechtzeitig fertig wurde, wird das Team mit 0 Story-Points "bestraft". Es ist leicht nachvollziehbar, was im nächsten Sprint passiert: Das Team verpflichtet sich einfach auf weniger Stories und erzeugt damit einen impliziten bzw. verdeckten Puffer. Daher sind auch Scrum-Teams vor den Auswirkungen des Parkinsonschen Gesetzes und des Studentensyndroms nicht gefeit (s. Definitionen im Glossar des Projekt Magazins www.projektmagazin.de/glossar).

Offensichtlich gibt es eine ganze Reihe von Problemen, die dazu führen, dass Scrum nicht seine ganze Leistungsfähigkeit erreicht und deswegen mit den Stakeholdern sowie der umgebenden Organisation immer wieder in Konflikt gerät. Dafür gilt es, Lösungen zu finden, sodass das Potential von Scrum sich voll entfalten kann.

"Der Dritte Weg"

In vielen Köpfen herrscht leider eine "Ganz-oder-gar-nicht-Mentalität". Wenn man Scrum nicht "genau nach Buch" durchführt, ist es nach Ansicht von Puristen kein Scrum mehr. Mit solchen Denkmustern schränkt man sich selbst ein und verbaut sich neue Lösungswege. Zum Glück gibt es in anderen Projektmanagementansätzen Werkzeuge, die zu neuen Denkanstößen verhelfen können.

So beschreibt z.B. Eliyahu Goldratt den Ansatz, dass es keine dauerhafte Begründung für einen Konflikt geben kann, wenn ein gemeinsames Ziel existiert – es bestehen höchstens temporäre Meinungsverschiedenheiten (Goldratt 1984; Goldratt 2008). Er zeigt, wie durch die Methodik der Konfliktwolken aus jedem Konflikt eine Win-Win-Situation erzeugt werden kann, indem die Annahmen konsequent hinterfragt werden.

Jeff Sutherland, einer der Väter von Scrum, empfiehlt in einem Blog-Artikel genau diesen Ansatz (Sutherland 2006): "Eliminating bottlenecks [Anm. des Autors: es können auch Denkhindernisse sein] is the fastest way to improve productivity. See Goldratt's book 'The Goal'. All managers should read this book. Developers will find it useful also." (dt.: "Engpässe zu beseitigen ist der schnellste Weg, um Produktivität zu steigern. Vergleiche hierzu Goldratts Buch 'Das Ziel'. Alle Manager sollten dieses Buch lesen. Auch Entwickler werden es nützlich finden.")

Um nun eine Win-Win-Situation zu erzeugen, muss man die Konfliktwolke für "agil gegen klassisch" (Bild 1) oder besser gesagt "Scope ist frei gegen Scope ist fixiert" auflösen.

Gelesen wird die Konfliktwolke von links nach rechts. Begonnen wird bei A, dem gemeinsamen Ziel, denn am Schluss geht es bei beiden Ansätzen darum, den optimalen Nutzen für den Kunden und das Unternehmen zu erzielen. Um das zu erreichen, gibt es bereits an den Punkten B und C unterschiedliche Anforderungen. Um diese wiederum zu erfüllen, müssen die Handlungen D bzw. D‘ durchgeführt werden, die zueinander in Widerspruch stehen.

In Bild 1 wird der Widerspruch transparent, der letztlich den Konflikt verursacht:

- Um den optimalen Nutzen zu erzielen, muss es jederzeit möglich sein, den Scope zu ändern. Das ist auch notwendig, damit der Work-in-Progress (in einem Sprint) immer kleiner oder gleich der zur Verfügung stehenden Kapazität ist.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- Um einen bestimmten Scope zu einer bestimmten Zeit zur Verfügung zu stellen, muss der Scope unveränderlich sein und im Voraus geplant werden. Das ist notwendig, damit der Scope und damit der Work-in-Progress nicht über die zur Verfügung stehende Kapazität hinaus anwachsen.

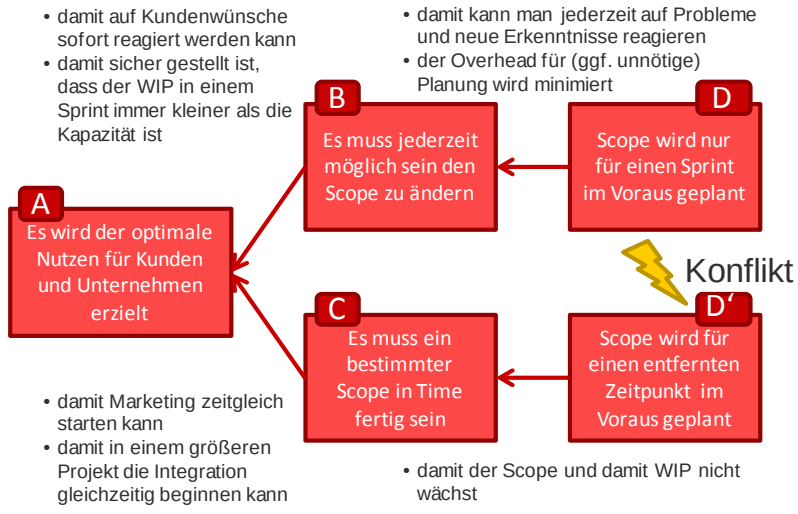


Bild 1: Die Konfliktwolke für agil (oben) vs. klassisch (unten) gilt es zu lösen.

Um diesen Konflikt zu lösen,

- muss sich der Work-In-Progress initial so planen lassen, dass er, trotz Änderungen am Scope, stets der zur Verfügung stehenden Arbeitskapazität entspricht.
- muss man jederzeit sehen und sicherstellen können, dass der Work-In-Progress unter Kontrolle ist.

Sobald Lösungen für diese beiden Punkte gefunden sind, lässt sich der Scope von Sprint zu Sprint ändern, der Work-In-Progress bleibt in einem Release unter Kontrolle und wächst nicht über die dauerhaft zur Verfügung stehende Kapazität.

Wie schafft man es nun, dass der Work-In-Progress sicher der Kapazität entspricht? Der Trick besteht darin, Scope und Velocity nicht als feste einzelne Größen zu betrachten, sondern als eine Bandbreite, d.h. als Wahrscheinlichkeitsverteilungen. Damit lässt sich die Erfolgswahrscheinlichkeit für einen bestimmten Scope zu einem bestimmten Termin berechnen. Zusammen mit dem Product Owner, ScrumMaster, dem Team und den Stakeholdern werden die Bandbreiten des möglichen Scope und der Velocity so eingestellt, dass eine ausreichende Erfolgswahrscheinlichkeit zu einem bestimmten Zeitpunkt entsteht. Dadurch bildet sich ein expliziter Zeit- und Kapazitätspuffer am Projektende, der es dem Team ermöglicht, sicher zu liefern (s.u.). Damit ist das Problem der initialen Begrenzung des Work-In-Progress' gelöst.

Wie hält man aber jetzt den Work-In-Progress dauerhaft unter Kontrolle? Mit den aktuellen Werten von Zeitpuffer und Projektfortschritt lässt sich jederzeit ein objektiver Projektstatus ermitteln. Darauf aufbauend ist es möglich, ein visuelles Steuermittel anzufertigen, dass den Fortschritt gegenüber dem Pufferverbrauch visualisiert (s. unten). Das Projekt ist "grün", wenn der Fortschritt größer ist als der Pufferverbrauch. Das Projekt ist "rot", wenn der Fortschritt kleiner ist als der Pufferverbrauch. Dazwischen gibt es ggf. einen kleinen "gelben" Übergangsbereich, den es anzustreben gilt. Der Product Owner (und die Stakeholder) erhalten dadurch einen operativen, wirksamen Indikator, ob der Work-In-Progress noch zum vereinbarten Termin mit gegebener Kapazität erreichbar ist. Der Product Owner kann mit diesem Werkzeug sein Product Backlog so managen, dass der Work-In-Progress unter Kontrolle ist und die Erfolgswahrscheinlichkeit des Teams erhalten bleibt.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Beide Maßnahmen, d.h. die Planung nach ausreichender Erfolgswahrscheinlichkeit und die Steuerung nach Pufferverbrauch, stammen fast identisch aus dem Critical Chain Projektmanagement und sind dort seit knapp 15 Jahren bekannt und bewährt. Auf diese Weise lässt sich ein Scope für einen bestimmten Zeitpunkt in der Zukunft sicher einhalten und dabei flexibel von Sprint zu Sprint variieren.

Im nachfolgenden Fallbeispiel werden die Werkzeuge im Detail beschrieben. Die Microsoft-Excel-Dateien mit den vorgestellten Berechnungsverfahren finden Sie im Anhang dieses Artikels.

Fallbeispiel "Infrastrukturprojekt"

Wie funktioniert das Ganze nun in der Praxis? Im folgenden Fallbeispiel geht es um ein großes Infrastruktur-Projekt. Dabei galt es, eine bestehende Altanwendung abzulösen. Es handelte sich um eine hochintegrierte Self-Service-Anwendung, die viele Produkte für sehr viele Kunden unter einem Dach vereinte. Mit der Zeit wuchs die Anwendung zu einem monströsen "Monolit", bei dem alles mit allem zusammenhing – jede Änderung erzeugte große Risiken sowie einen hohen Aufwand.

Also musste eine neue Plattform her, in der alle Produkte wie "Apps" funktionieren – eine "Plug&Play-Plattform". Keiner wusste genau, wie eine solche Plattform konkret aussehen sollte und welche Überraschungen in der Altanwendung schlummerten. Nur eines schien klar: Ein Vorgehen nach dem Wasserfallmodell mit vollständig definiertem Lastenheft war undenkbar – daher konnte es nur agil gehen!

Der Backlog-Sumpf

Als ich nach gut einem halben Jahr in das Projekt kam, befand sich das Team in der dritten Architekturdiskussion. Der ScrumMaster konnte nicht nur ein Backlog vorweisen, sondern gleich vier – für jeden Stakeholder eines. Die Euphorie im Team war längst verflogen, erste Ergebnisse konnte man nicht zeigen. Man rettete sich von Sprint zu Sprint. Die Stakeholder waren nervös und hatten eine viel zu frühe Deadline gesetzt.

Die Frage nach der Velocity wurde von jedem anders beantwortet. Was wann geliefert werden sollte, war völlig unklar. Offensichtlich war hingegen, dass das Backlog viel zu groß und im Hinblick auf den gewünschten Termin nicht realisierbar war – aber wie sollte man es den Stakeholdern sagen? Die Aussage: "Unser Gefühl sagt, dass es zu viel ist, wir brauchen mehr Zeit!" hätte hier nicht ausgereicht!

Klar, hätte man Scrum von Beginn an richtig angewandt und hätte man die "Besten-der-Besten-Senior-Entwickler" und ebensolche Product Owner gehabt, wäre alles kein Problem gewesen. In der Realität stehen allerdings ganz normale Mitarbeiter mit ein paar Schwächen und vielen Stärken zur Verfügung, aber keine Superhelden.

Um aus dieser Situation herauszukommen, gingen wir nun Schritt für Schritt vor.

Transparenz als Schlüssel zum Erfolg

Der Schlüssel ist, die Erfolgswahrscheinlichkeit des Teams transparent zu machen und damit den Scope mit der vorgegebenen Teamgröße und der Zeit auszutarieren. Die zentralen Fragen sind: "Hat das Team überhaupt eine

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Chance, einen bestimmten Scope zu einem verhandelten Termin in Topqualität zu liefern?" und "Wie berechne ich die Erfolgswahrscheinlichkeit?"

Ein Teil der Lösung ist, dass das Backlog und die Velocity nicht mehr als unveränderliche Werte beschrieben werden, sondern als Wahrscheinlichkeitsverteilungen. Dies gelingt mit einer 3-Punkt-Schätzung in Form eines optimistischen, realistischen und pessimistischen Werts. Im Fallbeispiel zeige ich, wie man zu dieser Schätzung kommt. An dieser Stelle genügt die Tatsache, dass diese drei Punkte eine relative Wahrscheinlichkeitskurve aufspannen. Der optimistische und pessimistische Schätzwert haben hierbei die Wahrscheinlichkeit 0%, der realistische die "größte Wahrscheinlichkeit" – wie hoch diese auch immer sein mag (Bild 2).

Mit diesen Wahrscheinlichkeiten kann man nun rechnen. Wenn man die pessimistische Schätzung des Backlogs (in Story Points) durch die pessimistische Schätzung der Velocity (Story Points/Zeit) dividiert, erhält man den Worst-Case – also die längste denkbare Projektdauer. Der sich daraus ergebende späteste Endtermin ist zu 100% wahrscheinlich. Nimmt man die optimistischen Schätzungen, so erhält man den Best-Case, d.h. die kürzest mögliche Projektdauer und den frühesten Zeitpunkt für den Projektabschluss. Dieser hat eine Wahrscheinlichkeit von nahezu 0%, da es äußerst unwahrscheinlich ist, dass wirklich alles glatt läuft. Den Verlauf zwischen den Extremwerten erhält man, indem man alle Kombinationen aus den Wahrscheinlichkeitsverteilungen von Scope und Velocity berechnet – was mathematisch einer sog. Faltung mit anschließender Integration entspricht. Diese Berechnungen sind in den bereitgestellten Excel-Tabellen programmiert.

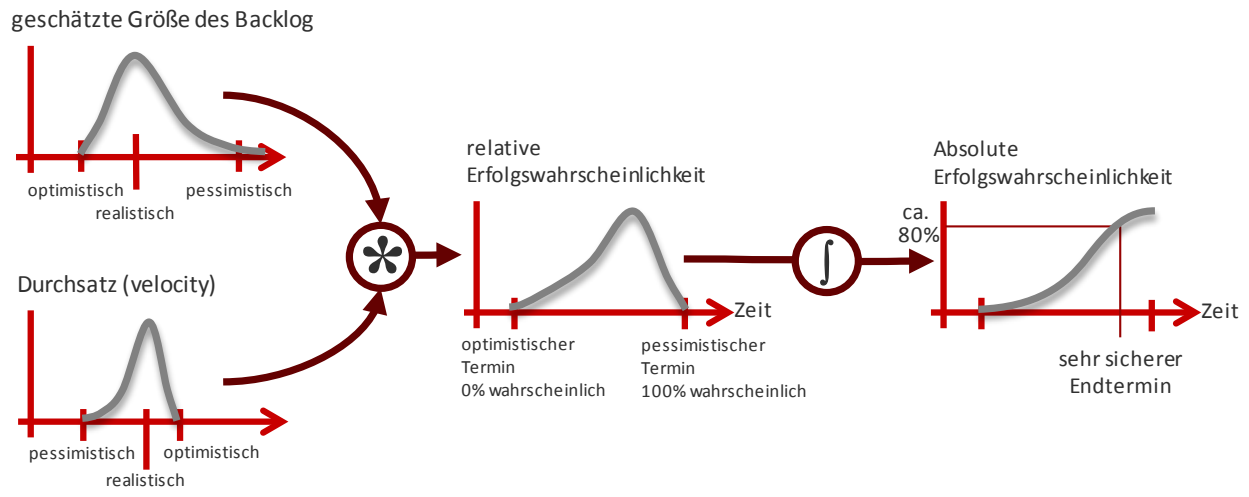


Bild 2: Berechnung der Erfolgswahrscheinlichkeit zu einem bestimmten Termin.

Mit dieser Berechnung erhält man die Verteilung der Erfolgswahrscheinlichkeit über der Zeit. Daraus lässt sich ablesen, mit welcher Wahrscheinlichkeit der Scope mit den vorgegebenen Ressourcen zu einem bestimmten Termin eingehalten wird. Auf diese Weise lassen sich die Größe des Backlogs, der Termin und/oder die Velocity so ausbalancieren, dass das Team eine ausreichend hohe Erfolgswahrscheinlichkeit erhält. Als langjähriger Erfahrungswert ist eine Erfolgswahrscheinlichkeit von ca. 80% anzustreben. 100%ige Sicherheit zu erreichen, wäre

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

unverhältnismäßig teuer (Ressourcen und Zeit) und birgt die Gefahr, dass Parkinsonsches Gesetz und Studentensyndrom das Projekt gefährden.

Beispiel: Das Team hat aktuell den Umfang des Backlogs auf 500 Story Points (SP) geschätzt und rechnet realistisch mit 100 SP, die wahrscheinlich dazu kommen. Als Worst Case schätzten sie, dass das Backlog auf maximal 750 SP anwachsen könnte. Bei der Velocity gehen sie von 40 SP/Woche und einer Streuung von +/- 20% aus. Das Release wurde am 01.09.2012 gestartet.

Für die maximale Dauer ergeben sich mit diesen Zahlen 750 SP / 32 SP/Woche = 23,4 Wochen. Ausgehend vom Starttermin 01.09.2012 ergibt das einen 100%ig sicheren Worst-Case-Endtermin am 12.02.2013. Die gleiche Rechnung für den Best-Case ergibt 500 SP / 48 SP/Woche = 10,4 Wochen und damit den 12.11.2012 als frühest möglichen Fertigstellungstermin.

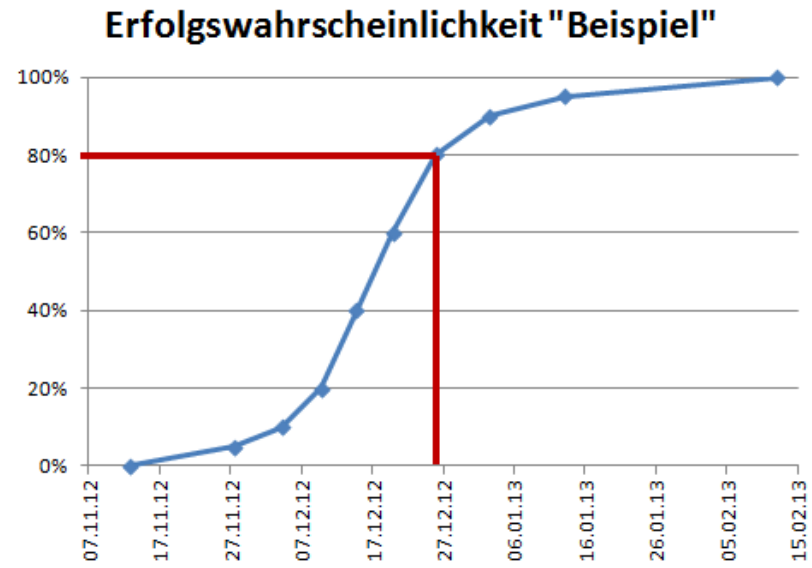


Bild 3: typischer Verlauf der Erfolgswahrscheinlichkeit über der Zeit.

Die Wahrscheinlichkeitswerte für eine Fertigstellung zwischen diesen beiden Extremwerten ergeben sich über die Faltung und Integration. Sie lassen sich deshalb nur mit Hilfe der beigefügten Excelkalkulation (s.u.) berechnen. Das Ergebnis dieser Berechnung ist folgende typische S-Kurve für die Erfolgswahrscheinlichkeit über die Zeit (Bild 3):

Der gewünschte Termin für eine 80% Sicherheit ist im Rechenbeispiel der 25.12.2012 (s. Bild 3).

Wie gingen wir nun im Fallbeispiel konkret vor?

Folgende Schritte führten wir durch, um schließlich zum Ziel zu kommen:

1. Der Product Owner erstellte einen Releaseplan, in dem jeweils mehrere (6-10) Sprints zu einem Release zusammengefasst wurden.
2. Der Product Owner verteilte die vorhandenen Stories auf diese Releases.
3. Fehlende Stories wurden vom Team ergänzt.
4. Nicht geschätzte Stories schätzte das Team mit Hilfe einer "Affinity Estimation" (relative Aufwandsschätzmethode, bei der die Stories nach geschätztem Arbeitsumfang sortiert werden).

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

5. Große Stories wurden vom Team aufgegliedert, bis der Anteil der Stories mit einer Größe von über oder gleich 42 Story-Points kleiner war als 10% des Gesamtumfangs.
6. Die Piorisierung (nach MoSCoW) wurde entfernt und ersetzt durch "in Release XYZ". Hierzu mussten einige Stories aufgeteilt werden, sodass das Team den zwingenden Teil der Story bereits in Release XYZ realisierte und die weiteren Teile für die nachfolgenden Releases geplant wurden.
7. Im Folgenden schätzte das Team zusammen mit dem Product Owner ab, wie viele Stories bei realistischer und wie viele bei pessimistischer Betrachtung noch zum entsprechenden Release hinzu kommen konnten. Zusammen mit der aktuellen Größe des Backlogs (optimistischer Wert) bildeten diese Schätzungen die Grundlage für die Wahrscheinlichkeitsverteilung des Backlogs.

Damit war das Backlog initial erstellt und jede Story hinsichtlich Release und Größe ausreichend definiert.

Tabelle 1 zeigt für das Beispiel die Stories, ihre Zuordnung zu den Releases, ihren Aufwand und den voraussichtlichen Realisierungstermin, bzw. ob die betreffende Story bereits abgeschlossen ist. Damit lässt sich zu jedem Zeitpunkt die aktuelle Größe des Backlogs ermitteln, z.B. 8 von 18 Story Points für Release 1 und 5 Story Points für Release 2 (s. Tab. 1).

Story	Release	Story-Points	Fertig
Persona X will Funktion A	R1	10	01.08.2012
Persona Y will Funktion B	R1	8	offen
Persona Z will Funktion C	R2	5	offen
...			

Tabelle 1: Auszug aus der Übersicht über die ersten beiden Releases.

Jetzt fehlte noch die Wahrscheinlichkeitsfunktion der Velocity:

8. Das Team schätzte zusammen mit dem ScrumMaster die mittlere Velocity bis zum geplanten Release-Ende und deren Spannweite (min/max), also die optimistische und die pessimistische Velocity. Als Grundlage wurden hierzu die Velocity-Werte der letzten Sprints herangezogen.

Damit ist auch die Wahrscheinlichkeitsverteilung der Velocity definiert.

Bild 4 zeigt die Daten aus dem Fallbeispiel für die oben genannten Schritte in der beigefügten Excel-Tabelle "Reliable Scrum WIP and Execution Control" und dort im Tabellenblatt "Part I - WIP Control". Die folgende Beschreibung bezieht sich auf diese Excel-Tabelle.

- Nachdem die fehlenden Stories ergänzt und alle Stories geschätzt wurden, gibt man unter "expected Load" die Größe des Backlogs an. Eingaben erfolgen nur in die gelb hinterlegten Felder (s. Bild 4), alle anderen Felder werden berechnet. Der obere, optimistische Wert für die expected Load ist die Größe des aktuellen Backlogs in Story Points. Der realistische sowie der pessimistische Wert ergeben sich aus einer Schätzung des Teams zusammen mit dem Product Owner.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- Die Velocity wird unter "expected Velocity" beschrieben. Hier ist der realistische Wert typischerweise der Mittelwert der letzten Sprints. Weiter gibt man eine Abweichung nach oben und nach unten an, die auf Schätzungen des Teams gemeinsam mit dem ScrumMaster beruhen.
- Für die Berechnung der Termine ist noch das Startdatum für das Release erforderlich, d.h. der Zeitpunkt, an dem mit diesem Release begonnen wurde.
- Mit Hilfe der Wahrscheinlichkeitsfunktion des Backlog und der Velocity wird nun die Erfolgswahrscheinlichkeit zu einem bestimmten Termin ("sucess probability based on wished Due-Date") oder der Termin bei einer Erfolgswahrscheinlichkeit von 80% ("Due-Date based on a given probability of success") errechnet. Die 80% sind hierbei ein Erfahrungswert, der sich in unterschiedlichen Umgebungen bewährt hat.

Beispiel

Reliable Scrum

Part I - WIP Control

The goal of Part I - WIP-Control is to make sure, that the probability to fulfill the expectations of the stakeholder is in a realistic range. There are two factors that define the probability of success a) the amount of story points in the backlog that has to be burned down and b) the mean velocity during until the due-date. If you take both as probabilities you'll get a good estimate on the overall success-rate at a specific time. A calculated success rate of greater than 80% will lead in practice to a success of the project/release.

expected Load - BACKLOG (regarding probability)	total		
optimistic Load (Story Points)	618		baseline, the current known backlog items
realistic Load (Story Points)	742	20%	estimate on how many backlog items usually occur
pessimistic Load (Story Points)	834	35%	estimate on how many backlog items in worts case occur
expected VELOCITY (regarding probability)	mean		
pessimistic velocity (Story Points per Week)	16	-20%	estimate on how much the velocity can drop in worst case
realistic velocity (Story Points per Week)	20		
optimistic velocity (Story Points per Week)	24	20%	estimate on how much the velocity can increase in best case
start of burn down	04.11.2011		date when the burn down of the backlog really started
Results			
sucess probability based on wished Due-Date			
due date	26.07.2012		
absolute probability of success to deliver at due date	63%		above 80% will typically lead to a successful project
Due-Date based on a given probability of success			
reasonable absolute probability	80%		
reasonable due date	09.08.2012		

Bild 4: Beispiel für die Berechnung der Erfolgswahrscheinlichkeit in einem Reliable-Scrum-Projekt.

Im Fallbeispiel waren diese Schritte nach ca. drei Wochen erledigt und das erste Ergebnis für die Erfolgswahrscheinlichkeit lag vor: Mit den geplanten Stories im Release sowie der aktuellen Velocity kamen wir für den Liefertermin auf lediglich 3% Erfolgswahrscheinlichkeit! Was nun? Auf die Stakeholder zuzugehen, frei nach dem Motto: "Houston, we have a problem!", erschien uns zu passiv. Folglich stellten wir uns die Fragen: Brauchen wir tatsächlich alle Stories für das Release? Was ist wirklich wichtig? Gibt es Stories, die zu groß geschätzt wurden?

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Wie sieht es mit unserer Velocity aus – ist sie wirklich so gering? Kann man Stories weiter teilen in "Must" für das aktuelle und "Should" für das nächste Release?

Backlog neu erstellen ...

Im ersten Ansatz verkleinerten wir das Backlog so weit, dass zum gewünschten Termin die Erfolgswahrscheinlichkeit auf 80% stieg. Die Folge war zunächst allgemeines Entsetzen darüber, wie wenig wir im ersten Release umzusetzen konnten. Wir präsentierten diese Ergebnisse den Stakeholdern – und plötzlich war die Diskussion einfach. "Nein, ohne Story X und Y ist das nicht verkaufbar, die muss noch rein." Aber bei jeder Story, die hinzugenommen würde, war klar, dass die Erfolgswahrscheinlichkeit sinkt und der Termin wackelt.

... Termin anpassen

Nach dieser Diskussion mit den Stakeholdern kamen ein paar Features wieder hinzu und die Erfolgswahrscheinlichkeit unseres ersten Releases lag damit bei 63%. Die Stakeholder hatten klar gesagt, was im Release unbedingt umgesetzt werden sollte, waren aber nicht bereit, die Ressourcen weiter aufzustocken. Um die angestrebten 80% zu erreichen, mussten wir deshalb zusätzlich den Termin verschieben. Wir vereinbarten mit den Stakeholdern eine Verlängerung der Projektdauer um 12 Arbeitstage.

Als Ergebnis dieses Vorgehens hatten wir erreicht: Das Backlog war so eingestellt, dass die Erfolgswahrscheinlichkeit nun ausreichend hoch war, das Release bis zum vereinbarten Termin umzusetzen. Als Nebeneffekt wurde der Projektauftrag deutlich schneller und besser geklärt als in den Monaten zuvor. Es entstand eine Verhandlung mit den Stakeholdern, wie viel "Agilität" und Risikopuffer sie bezahlen wollen. Die Erwartungshaltung war beiderseitig eindeutig geklärt.

Dead or Alive – die Alternative zu "Melonengrün"

Die bekannteste Ampelfarbe in herkömmlichen Projektreports ist "Melonengrün" – außen grün und je tiefer man bohrt, desto "roter" wird es. Im agilen Kontext ist das Pendant zu Melonengrün "Dead or Alive". Im Product-Burndown-Chart sieht man nur, ob der Fortschritt besser ist als der lineare Forecast, also "Alive" – oder eben nicht "Dead". Die Möglichkeit, bei einer schlechten Entwicklung einzugreifen und gegenzusteuern, ist hier ohne Puffer nur über das Prinzip Hoffnung gegeben.

Damit das Team aktiv steuern kann, benötigt es noch ein objektives Fortschritts-Monitoring, das den bestehenden Handlungsspielraum aufzeigt.

Scrum mit Critical Chain kombinieren

Aus der Critical Chain-Methodik ist ein Konzept des Monitorings bekannt: "Fortschritt gegenüber Pufferverbrauch". Wenn der Fortschritt größer ist als der Pufferverbrauch, ist das Projekt "on track", also "grün". Andernfalls sind Maßnahmen zu ergreifen, um den Puffer "zurückzuholen". Dazwischen gibt es einen kleinen gelben Bereich, der dem Team dazu dient, früh die Initiative selbst zu ergreifen.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Durch das Aushandeln eines realistischen Backlogs sowie des passenden Termins, der eine Erfolgswahrscheinlichkeit von 80% ermöglicht, ist automatisch ein gewisser Zeit- und Kapazitäts-Puffer entstanden. Zum aktuellen Zeitpunkt umfasst das Backlog ja nur den aktuell bekannten, optimalen Umfang und die Velocity entspricht dem Mittelwert der letzten Sprints. Daher dürfte der aktuelle Forecast für das Release-Ende (Estimated Time To Complete) etwas vor dem zugesicherten Termin (Due Date) liegen. Dieser Puffer (s. Bild 7) kann nun dazu verwendet werden, um die Fieberkurve gemäß Critical Chain zu zeichnen (Bild 6).

Auch hierzu gibt es in der Excel-Datei ein Tabellenblatt, das die Berechnung der Fieberkurve übernimmt (Bild 5).

Reliable Scrum

Part II - Execution Control

The goal of Part II - Execution Control is to make sure, that the progress is in balance with the buffer consumption! In addition to the product burn down chart there is a new "fever curve" that shows the buffer consumption over progress. If the curve get's into the yellow or red area the team is questioned to generate ideas how to get back into the green – "buffer regain". This is valid because in Part I the due date was adjusted in a way that the team has a realistic probability of success.

start of project/release	04.11.2011
start of the buffer	15.05.2012
commited due date	07.08.2012

30% percentage of buffer to the whole duration of the release project

# sprint	end date of sprint	elapsed time	story point in backlog [story points]	calculated velocity [story points/day]	set velocity [story points/day]	estimed date of finishing	progress	buffer consumption
0	04.11.2011	-	618	-		-	-	-
1	15.11.2011	11	557	-		-	-	-
2	29.11.2011	25	506	3,6	3,6	15.04.2012	15%	0%
3	13.12.2011	39	439	4,2	4,2	26.03.2012	27%	0%
4	10.01.2012	67	400	2,8	2,8	31.05.2012	32%	19%
5	31.01.2012	88	380	2,3	3,1	01.06.2012	42%	21%
6	21.02.2012	109	293	2,4	2,4	20.06.2012	47%	44%
7	13.03.2012	130	171	2,8	2,4	23.05.2012	65%	10%

Bild 5: Tabellenblatt „Part II - Execution Control“ zur Berechnung der Fieberkurve nach Critical Chain.

Zuerst werden die Start- und Endtermine des Releases eingetragen. Hier wird auch der Beginn des Puffers durch den ScrumMaster und das Team definiert. Da es um Schätzungen und Wahrscheinlichkeiten geht, gibt es keine eindeutige Angabe, wie groß dieser Puffer sein muss. Puffergrößen von 15% bis 30% der geplanten Gesamtlaufzeit haben sich allerdings in der Praxis bewährt. Ist der Puffer zu klein, besteht die Gefahr, dass das Team wieder Puffer in die Stories einplant und die Fieberkurve zu schnell reagiert. Ist der Puffer zu groß, setzen Phänomene wie Parkinsons Gesetz und Studentensyndrom ein.

Die Fieberkurve in Bild 6 zeigt für das Fallbeispiel den Fortschritt gegenüber dem Pufferverbrauch. Jeder Punkt der Kurve ist das Ende eines Sprints. Reliable Scrum wurde dabei im fünften Sprint des Releases "aktiviert". Da zu diesem Zeitpunkt das Projekt bereits ein Stück weit vorangeschritten war, wählten wir den Puffer so, dass der relative Pufferverbrauch ungefähr gleich dem prozentualen Fortschritt (d.h. Fertigstellungsgrad) war. Der Punkt Nummer 5 kam damit fast im gelben Bereich zum liegen.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Im Folgenden wurde nach jedem Sprint die aktuelle Größe des Backlogs in Story Points angegeben und das Team schätzte gemeinsam mit dem ScrumMaster die voraussichtliche Velocity bis zum Ende des Releases. Das Ergebnis ist ein voraussichtlicher Endtermin für das Release. Daraus lassen sich der Fortschritt und der Pufferverbrauch berechnen und in das Diagramm übertragen.

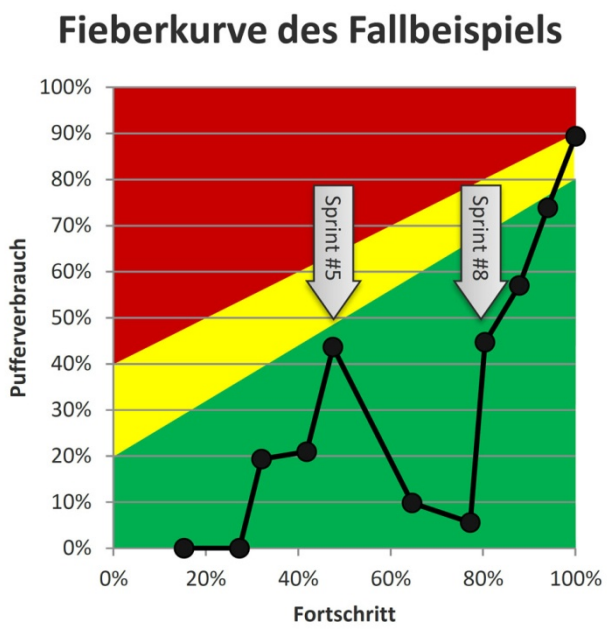


Bild 6: Reale Fieberkurve des Fallbeispiels, jeder Punkt stellt das Ende eines Sprints dar.

Mehr Sicherheit für das Team

Deutlich sichtbar ist der Velocity-Sprung nach Sprint 5. Durch den Puffer am Ende des Releases bekam das Team mehr Sicherheit und konnte befreiter agieren. Die Schätzungen wurden aggressiver und der Burndown nahm deutlich zu. Verstärkt wurde dieser Effekt dadurch, dass es jetzt durch die gute Auftragsklärung ein klares Ziel gab, auf das motiviert hingearbeitet werden konnte.

Der nächte interessante Punkt ist nach Sprint Nummer 8 zu sehen. Die Velocity war zu diesem Zeitpunkt so hoch, dass das Team den Puffer offensichtlich nicht mehr brauchen würde. Die Projektlaufzeit wurde deshalb um 13% verkürzt und der Endtermin entsprechend nach vorne verlegt. Zusätzlich wurden weitere Stories in das Release einbezogen. Daher lässt sich ab diesem Punkt ein Sprung in Richtung "gelb" beobachten, wobei der Product Owner genau so viele Stories in das Release neu aufnahm, dass die Velocity gut genutzt werden konnte und die Sicherheit des Releases zu keiner Zeit gefährdet war. Diese Maßnahme stärkte erheblich das Vertrauen bei den Stakeholdern. Dennoch konnte sich das Team darauf verlassen, dass ihre Erfolgswahrscheinlichkeit erhalten blieb.

Was ist der Nutzen dieser Darstellung?

An dem Diagramm können die Stakeholder jederzeit genau nachvollziehen, wo das Release gerade steht und wie es sich entwickelt. Der Product Owner hat ein Werkzeug, mit dem er sein Backlog managen kann und das Team erhält die notwendige Ruhe, um konzentriert zu arbeiten. Da der Fokus nicht mehr auf dem Sprint liegt, sondern auf dem Release, versteckt das Team den Puffer nicht mehr in den Schätzungen. Ängste nehmen ab, Studentensyndrom und Parkinsons Gesetz werden vermieden: Die Velocity steigt.

Nach jedem Sprint

- werden neu hinzu gekommene Stories geschätzt und einem Release zugeordnet,
- freigegebene Stories als fertig registriert und
- halbfertige bleiben offen.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Somit erhält man nach jedem Sprint den aktuellen Stand der Story Points im Backlog. Im Sprint Review schätzt das Team erneut (unter Berücksichtigung aller neuen Erkenntnisse) die mittlere Velocity, die es glaubt, bis Ende des Releases halten zu können.

Dividiert man die aktuell verbleibenden Story Points durch die geschätzte mittlere Velocity, so erhält man eine Prognose für die restliche Projektlaufzeit und daraus für das voraussichtliche Projektende. Um den Fortschritt zu einen bestimmten Betrachtungszeitpunkt (Fertigstellungsgrad) zu berechnen, bildet man das Verhältnis von der Projektdauer bis zu diesem Zeitpunkt zur prognostizierten gesamten Projektdauer vom Projektstart bis zum vorrausichtlichem Ende. Der verbleibende Restpuffer ist die Zeitspanne zwischen prognostiertem Projektende und dem zugesicherten Enddatum. Die Differenz von Restpuffer zur gesamten Pufferzeit ist der verbrauchte Puffer. Dividiert man den verbrauchten Puffer durch die gesamte Pufferzeit, erhält man den prozentualen Pufferverbrauch (vgl. Bild 7). Auf diese Weise kann man nach jedem Sprint die Fieberkurve aktualisieren.

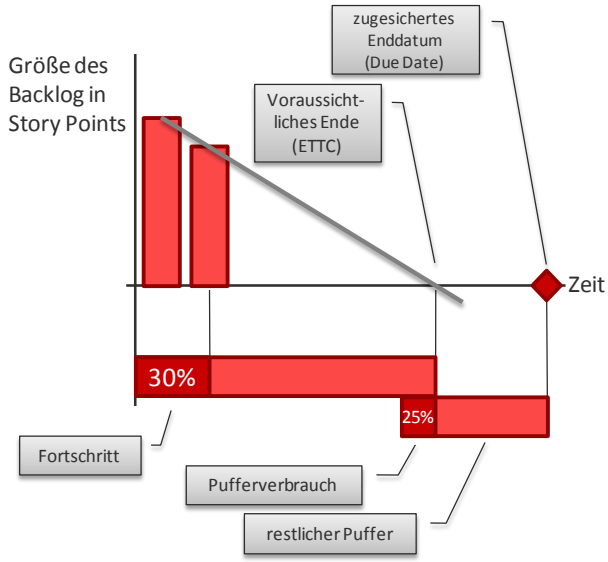


Bild 7: Berechnung des Fortschritt und des Pufferverbrauchs anhand des Product-Burndown-Chart.

Fieberkurve für Projektportfolio

Auch für das Portfoliomanagement mit mehreren parallel durchgeführten Scrum-Projekten lässt sich das Diagramm einsetzen. Doch wie behält man die Übersicht? Wenn man die Kurven aller Scrum-Streams in ein Diagramm zeichnen würde, hätte man keinen Überblick mehr. Für die Stakeholder sind ohnehin nur der letzte Stand und die Entwicklung wichtig. Die Lösung ist daher, nur die aktuellen Endpunkte und die jeweiligen Entwicklungen im letzten Sprint als Pfeile in ein Diagramm einzuzeichnen.

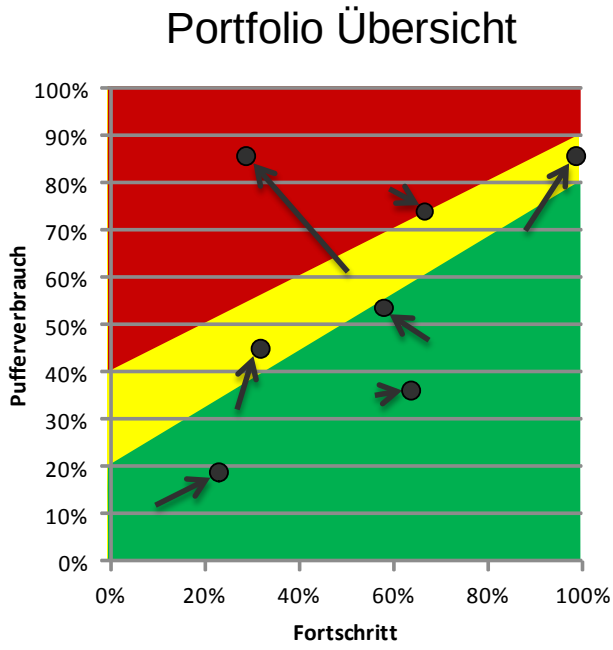


Bild 8: Beispielhafte Darstellung eines Portfolios von Scrum-Projekten.

In Bild 8 ist dies beispielhaft für sieben unterschiedliche Projekte dargestellt. Auffällig ist das Projekt links oben, das sich aus dem gelben in den roten Bereich entwickelt hat. Hier ist die Aufmerksamkeit der Stakeholder gefordert. Die anderen Projekte bewegen sich tendenziell nach rechts oben und befinden sich im oder in der Nähe des gelben Bereichs. Dies ist in Ordnung und es bedarf keiner weiteren Intervention. Falls sich mehr als 10% der Projekte im roten Bereich bewegen, ist das

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Portfolio nicht mehr unter Kontrolle. In diesem Falle sollte die Anzahl der Projekte reduziert und die Ressourcen besser gebündelt oder Endtermine sowie Scopes der unwichtigeren Projekte anpasst werden.

Für die Erstellung dieses Diagramms gibt es ebenfalls eine Microsoft Excel-Tabelle "Reliable Scrum Portfolio", die Sie mit diesem Artikel herunterladen können.

Die Grenzen von Reliable Scrum

In zwei Fällen lässt sich Reliable Scrum nicht sinnvoll einsetzen. Zum einen, wenn ein über eine längere Zeit bereits stabiles Produkt schrittweise um neue Features ergänzt werden soll und dabei nach jedem Sprint unverzüglich ausgerollt wird. Hier ist der Einsatz von Reliable Scrum nicht sinnvoll, da keine Notwendigkeit besteht, einen Termin abzusichern – Scrum in der ursprünglichen Form kann in diesem Fall seine Vorteile ausspielen. Außerdem ist es bei dieser Konstellation nicht möglich, mehrere Sprints zusammenzufassen.

Der andere Fall liegt vor, wenn in einem Projekt mehrere Scrum-Teilprojekte existieren, die zusammenspielen müssen. Die Komplexität der Abhängigkeiten kann hier die Vorteile von Scrum im Allgemeinen aufzehren. In diesem Fall ist ein Übergang zu Critical Chain, ggf. mit Teilprojekten nach Reliable Scrum vorzuziehen. Critical Chain stellt hierfür Werkzeuge wie Zulieferpuffer, Projektpuffer, Engpassidentifikation und Management von Abhängigkeiten zur Verfügung, die in einer solchen Konstellation erfolgsentscheidend sind.

Wirkungen und Ergebnisse

Reliable Scrum fokussiert sich im ersten Schritt auf die Konkretisierung des Backlogs sowie auf die Verhandlung einer erfolgversprechenden "Backlog-Termin-Kombination", die es ermöglicht, den Termin sicher zu erreichen. Damit führt Reliable Scrum umgehend zu folgenden positiven Ergebnissen:

- Das Projektziel wird schneller geklärt.
- Die Stakeholder stimmen explizit Time, Scope und Budget miteinander ab – und zwar so, dass der Scope und der Termin vom Team auch realistisch eingehalten werden können.
- Es entsteht ein expliziter Zeitpuffer am Release-Ende.

Der zweite Schritt besteht darin, den operativen Status des Releases als Fieberkurve in Form des Fortschritts gegenüber dem Pufferverbrauch darzustellen. Dies hat folgende Auswirkungen:

- Das Team und der Product Owner erhalten ein empfindliches und in der täglichen Arbeit sofort anwendbares Monitoring bzgl. des aktuellen Stands des Backlogs. Da das Team mit den Stakeholdern den Umfang des Backlogs auf Basis seiner Schätzungen so aushandelt, dass eine ausreichend hohe Erfolgswahrscheinlichkeit besteht, fühlt es sich verpflichtet, seine Zusagen auch einzuhalten. Sobald das Release in den Gelb-Roten-Bereich kommt, setzt sofort der Mechanismus des "Buffer-Regain" ein, das Team versucht also, Puffer zurückzuholen und wieder in den grünen Bereich zu gelangen.
- Die Stakeholder erhalten eine einfache und verständliche Form des Reportings. Die Fieberkurve zeigt entstehende Probleme sehr früh an und stellt die Wirkung der vom Team umgesetzten Maßnahmen transparent

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

dar. Damit entsteht im Projekt bereits zu einem frühen Zeitpunkt Vertrauen zwischen dem Team und den anderen Stakeholdern, wodurch das Team ungestört arbeiten kann.

- Durch den Puffer am Projektende wird der Druck auf den einzelnen Sprint gemildert, ohne die Zuverlässigkeit der Projektplanung zu gefährden. Hierdurch kann das Team "befreit aufspielen". Es wird in der Folge optimistischer schätzen und aggressiver planen, wodurch letztendlich die Velocity steigt.

Im dritten Schritt können auch ganze Portfolios von Scrum-(Teil-)Projekten mit Reliable Scrum gemanagt werden. Folgender Nutzen stellt sich hierdurch ein:

- Die Stakeholder und Portfolioverantwortlichen erhalten eine sehr einfache Übersicht über das Portfolio. Sie bekommen ein klares Signal, welche Scrum-Projekte ihre Aufmerksamkeit erfordern und welche nicht.
- Wenn der Großteil der Scrum-Projekte im grünen Bereich ist, werden deren Sprintergebnisse sicher geliefert. Damit können auch mehre voneinander abhängige Scrum-(Teil-)Projekte mit überschaubarem Aufwand sicher ins Ziel gebracht werden.

Ganz unabhängig von den Ergebnissen wird mit Reliable Scrum das Beste der agilen Welt mit dem Besten der traditionellen Projektwelt vereint: Scrum + CCPM = Reliable Scrum. Agil wird kompatibel zu herkömmlichen Projekten und Projektumgebungen. Der Konflikt "Klassisch gegen Agil" ist aufgelöst – es heißt jetzt "Klassisch und Agil". Somit entsteht der größtmögliche Nutzen für den Kunden und das Unternehmen – die Basis für ein "agiles Enterprise".

Literatur

- Schwaber, K. und Sutherland, J.: The Scrum Guide – the official rulebook, 2011, unter: www.scrum.org/Scrum-Guides, zuletzt eingesehen am 23.08.2012
- Goldratt, E.: Die kritische Kette : Ein Roman über das neue Konzept im Projektmanagement, Campus: Frankfurt/M., 2002
- Müller, W.: Critical Chain Project Management - der Joker des Lean Projektmanagement, Verlag SIGS-Datacom, OBJEKTspektrum, Ausgabe 02/2010 (zu beziehen über den Autor: mail@speed4projects.net)
- Goldratt, E: The Goal, Campus-Verlag, 1984
- Goldratt, E.: The Choice, North River Press: Great Barrington (MA), 2008
- Sutherland, Jeff: Blog-Artikel: scrum.jeffsutherland.com/2006/06/why-three-questions-in-daily-scrum.html, zuletzt eingesehen am 23.08.2012

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Fachbeitrag

Agiles Projektmanagement

Schneller geht's nicht – Ultimate Scrum

Agile Methoden haben in den letzten Jahren einen Siegeszug bei der Software-Entwicklung angetreten und finden auch bei anderen Projektarten immer größere Verbreitung. Mit den bekanntesten Methoden Scrum und (Software-)Kanban ziehen auf diese Weise Ideen, die eigentlich aus der Produktionssteuerung kommen (s.u.), auch in die Projektwelt ein.

Aber auch diese Ansätze haben noch Schwächen. Zwar begrenzen sie den Work-In-Progress (aktuell in Bearbeitung befindliche Aufgaben), allerdings tun sie dies nur lokal und nicht über die gesamte Laufzeit und den gesamten Umfang eines Projekts hinweg. Darüber hinaus können weder nach Scrum noch nach Kanban arbeitende Teams verbindliche Zusagen für einen definierten Leistungsumfang zu einem bestimmten Zeitpunkt geben. Dies wird erst durch den von mir beschriebenen Ansatz "Reliable Scrum" (Müller, Projekt Magazin 18/2012) erreicht, auf den dieser Artikel aufbaut.

Reliable Scrum sorgt zwar für eine höhere Planungssicherheit, kann aber nicht verhindern, dass es bei Scrum und Kanban einen verhältnismäßig großen Bestand an offenen Aufgaben gibt. Die Folge davon ist, dass die Durchlaufzeiten länger als notwendig sind. Wie ich im Folgenden zeige, ist es für eine Steigerung der Produktivität aber wichtig, die Anzahl der offenen Aufgaben zu minimieren und die Durchlaufzeiten bis zum Optimum zu verkürzen. Durch die starke Fokussierung auf Reduktion des Bestands und die Verkürzung der Durchlaufzeit werden alle störenden Hindernisse ausgeräumt. Die Produktivität und damit der Durchsatz steigen nachhaltig. In Folge entsteht der größte Nutzen für den Kunden, das Unternehmen und genauso für die Mitarbeiter.

Alle agilen Ansätze setzen sehr stark auf Selbstorganisation. Das ist zwar für das Funktionieren der agilen Methoden sehr gut und wichtig, aber die Selbstorganisation gewährleistet weder, dass schnell ein Leistungsoptimum erreicht wird, noch dass es überhaupt erreicht wird.

Im Folgenden zeige ich, wo das Optimum der Leistungsfähigkeit eines agilen Teams liegt, woran man erkennt, wie weit man noch davon entfernt ist, und wie man es durch ein kombiniertes Steuerungsverfahren erreichen kann. Diesem Steuerungsverfahren habe ich, da es Scrum als Basis hat, den Namen "Ultimate Scrum" gegeben.

Autor

Wolfram Müller

Dipl.-Ing. Mechatronik und Dipl.-Ing. Maschinenbau selbst. Berater, Geschäftsführer Speed4Projects

Kontakt:
wolfram.mueller@speed4projects.net

Mehr Informationen unter:
› projektmagazin.de/autoren

ähnliche Artikel

in den Rubriken:

› Agiles Projektmanagement

› Projektabwicklung

› IT-Projekte

› Informations- und Kommunikationstechnologie

› Best Practices

Service-Links

Dienstleister

› PM-Einführung und -Optimierung

Software

› Agiles Projektmanagement

Termin

› IT-Projektmanagement

Buch

› Agiles Projektmanagement

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Optimale Geschwindigkeit – der "Heilige Gral des Projektmanagements"

Die Suche nach der optimalen Geschwindigkeit eines Projektteams hört sich so an wie die "Suche nach dem Heiligen Gral". Tatsächlich haben sich schon Generationen von Projektmanagern mit dieser Herausforderung befasst. Z.B. wurde die traditionelle Netzplantechnik in erster Linie dazu erfunden, um Projekte schneller durchführen zu können. Auch heute geht es in Projekten immer noch darum, Ergebnisse möglichst schnell zu liefern: Wir befinden uns in einer leistungsorientierten Gesellschaft, in der genau derjenige einen Wettbewerbsvorteil hat, der ein Stückchen näher am Optimum arbeitet als die anderen.

Gleich vorweg: Es geht dabei nicht um die Optimierung der Leistung zu Gunsten einzelner Nutznießer (z.B. der Shareholder des Unternehmens) und auf Kosten vieler Leistungserbringer (der Teammitglieder). Bei meinen Überlegungen steht der Mensch im Mittelpunkt. Für mich bedeutet "Optimum" stets, dass der angestrebte Zustand auch aus Sicht der Mitarbeiter optimal ist. Denn was gibt es Schöneres und Motivierenderes als jeden Abend sagen zu können: "Ich habe heute das Bestmögliche erreicht!"

Um das Optimum zu erreichen, muss man jedoch viele Aspekte berücksichtigen: Man muss die richtigen Produkte und Projekte auswählen und man braucht sehr gute Führungskräfte wie auch Mitarbeiter. Weiterhin gilt es, Wissen aufzubauen und dieses in Können umzusetzen. Last-but-not-least muss die Arbeitsorganisation alles perfekt unterstützen.

Für die absolute, optimale Leistungsfähigkeit müssten auch alle diese Aspekte optimal sein – dies ist ein hehres Ziel. Jedoch steht dem Erreichen dieses Ziels leider entgegen, dass Ressourcen und Zeit begrenzt sind. Man muss sich daher fokussieren. Doch welchen der genannten Aspekte adressiert man zuerst? Die Arbeitsorganisation hat hier eine interessante Sonderstellung: alle Verbesserungen dieses Aspekts sind – wie der Name schon sagt – organisatorischer Natur. D.h. Arbeitsorganisation besteht "lediglich" aus Regeln und Mustern in unseren Köpfen. Diese lassen sich, wenn man es richtig macht, unverzüglich und kostenlos ändern. Als Dank erhält man mehr Kapazität, die für Verbesserungen der anderen Aspekte genutzt werden kann (Goldratt, 1984 und 1997; Techt, 2006).

Mein Lösungsweg, um zur optimalen Geschwindigkeit von agilen Projekten zu gelangen, besteht daher darin, als erstes die Arbeitsorganisation zu optimieren. In meiner Praxis als Leiter bzw. Coach von agilen Teams konnte ich die im Folgenden dargestellten Steuerungsverfahren empirisch Schritt für Schritt entwickeln und ihren Erfolg verifizieren.

Durchlaufzeit minimieren, Performance steigern, Stress beseitigen

Das Schönste an Ultimate Scrum ist für mich, dass sich die Teamkultur und die Außenwahrnehmung des Teams vollständig gewandelt haben. Wo zuvor alle Arbeitsschritte in einem Mikromanagement überwacht wurden und Leistungsdruck aufgebaut wurde, herrscht jetzt Vertrauen in die Leistungsfähigkeit des Teams. Wenn man in das Büro dieses Teams geht, fällt als erstes auf, dass es leise ist und keinerlei Anzeichen von Hektik zu spüren sind. Nur unterbrochen von ein paar Witzen und Lachen sitzen die Entwickler an ihren Arbeitsplätzen und haben auch noch Zeit, miteinander eine Tasse Tee zu trinken. Das Team wird mittlerweile als Vorzeigeteam wahrgenommen – andere Teams gehen aktiv auf uns zu und wollen wissen, was wir hier anders machen.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Ultimate Scrum, wie ich es im Folgenden vorstelle, ist mittlerweile in mehreren Teams und Unternehmen im Einsatz und bewährt sich jeden Tag in der Praxis. Neben der veränderten Teamkultur lassen sich die positiven Auswirkungen an handfesten Messgrößen erkennen:

- Die täglichen Stand-Up-Meetings verkürzen sich auf ca. 3-5 Minuten – auch in standortübergreifenden Teams und in Teams von bis zu 9 Mitgliedern.
- Der Bestand an offenen Tasks sinkt auf ein extrem niedriges Niveau. Idealerweise gibt es weniger offene Tasks als Entwickler.
- Die Durchlaufzeiten von Tasks sind minimiert – oft liegen sie unter einem Tag.
- Hindernisse werden noch deutlicher, schneller sichtbar und können meist sofort aufgelöst werden.
- Bei den Retrospektiven geht es nicht mehr um Prozessverbesserungen, sondern darum, wie nicht direkt wertschöpfende Tätigkeiten eliminiert werden können.
- Die Performance ("Velocity") des Teams steigt deutlich – ca. 30% Steigerung wurden bisher erreicht und ein Ende ist noch nicht in Sicht.

Die Performance-Steigerung ist hierbei ein indirekter Effekt. Wenn Sie das Ziel verfolgen, den Bestand an offenen Tasks auf ein Minimum zu reduzieren z.B. "weniger offene Tasks als Entwickler", dann entsteht schnell die Situation, dass ein Entwickler aufgrund eines Hindernisses nicht mehr an der begonnenen Task weiter arbeiten kann. Typische Hindernisse sind hierbei: fehlende oder unvollständige Spezifikationen und Dokumentation, fehlende Freigaben, fehlende automatisierte Tests, fehlende automatische Integration oder Nichtverfügbarkeit von Ressourcen.

In der Vergangenheit wurden diese Hindernisse aber selten ausgeräumt, sondern stattdessen eine neue Task begonnen. Genau dies ist bei Ultimate Scrum aber nicht gestattet. Der Entwickler muss seine nun freie Kapazität dazu einsetzen, das Hindernis zu beseitigen. Die Idee, dass die Entwickler die Hindernisse selbst auflösen, befördert in ungeahnter Weise die Selbstorganisation in Teams. In kürzester Zeit übernimmt das Team die Verantwortung für die Optimierung der Arbeitsorganisation. Die Entwickler warten nicht mehr auf Anstoß von außen (z.B. Retrospektiven), sondern lösen die Hindernisse unverzüglich selbst auf.

Mit jedem Hindernis, das nachhaltig gelöst wird, reduziert sich der Aufwand für die Erstellung von zukünftigen Tasks. Der Durchsatz und damit die Performance steigen.

Als weitere Optimierung bleibt jetzt nur noch, das Wissen und Können der Entwickler selbst zu verbessern. Damit steht nicht mehr die Organisation, sondern der Mensch im Mittelpunkt. Die gewonnene Kapazität durch die Leistungssteigerung und Overhead-Vermeidung investieren wir in Weiterbildung. Wir führen Sessions zu unterschiedlichen Themen durch, die alle dazu dienen, das vorhandene Können möglichst gut zu verteilen und neues Können von außen in das Team zu holen.

Natürlich gibt es auch hier Phasen höchster Produktivität und Phasen, in denen nicht alles rund läuft. Es ist physiologisch unmöglich, die ganze Zeit zu "sprinten". Es gibt also auch hier Schwankungen – aber auf einem viel höheren Niveau.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Ich hoffe, dass ich jetzt Ihre Neugier geweckt habe und Sie wissen wollen, wie Ultimate Scrum funktioniert.

Selbststeuerung braucht ein Ziel und ein Feedback, wo man steht

Die agilen Projektsteuerungen setzen im Kern auf Selbstorganisation und den sog. Deming-Zyklus ("Plan-Do-Check-Act") für die kontinuierliche Verbesserung. Beides sind notwendige Voraussetzungen, aber leider nicht hinreichend für eine optimale Leistungsfähigkeit.

In der Praxis habe ich beobachtet, dass in vielen Teams dieser Prozess der Selbststeuerung sehr lange dauert, nicht sicher zum Optimum führt und sehr stark von der Erfahrung der Führung oder des Coaches abhängt. Die Folge sind nicht oder erst spät genutzte Performance-Potenziale, die den Erfolg des Teams und des Unternehmens schmälern. Den Grund dafür sehe ich darin, dass den Teams eine klare Zielvorgabe für die Arbeitsorganisation fehlt und sie kein Feedback erhalten, wie weit sie vom Ziel entfernt sind und wie schnell (oder langsam) sie sich auf das Ziel hin bewegen.

Erst wenn dem Team klar ist, wo das Optimum liegt, kann es sich selbstorganisiert im Rahmen von Deming-Zyklen dort hinentwickeln. Dann wird das Team auch das dafür notwendige, systemische Wissen aufbauen. Diesen Prozess kann der Teamleiter oder ein Coach beschleunigen, indem er dem Team Tipps für die wirksamsten nächsten Verbesserungsmaßnahmen gibt.

Agile Vorgehensweisen sind verwandt mit Produktionssteuerungen

Die agilen Methoden entsprechen, wie ich im Folgenden zeige, in ihrem Kern Strategien aus der Produktionssteuerung. Dies erscheint überraschend, da Produktion und Projektwelt sich unter anderem dadurch voneinander unterscheiden, dass in Projekten neue Produkte entwickelt werden, während in der Produktion viele identische Produkte genau nach Vorgabe erstellt werden. Ein Produktionsablauf zeichnet sich daher u.a. durch folgende Eigenschaften aus:

- Es gibt sehr viele und kleine Aufträge, die weitgehend unabhängig voneinander zu verarbeiten sind.
- Die Reihenfolge, in der die Aufträge bearbeitet werden, kann leicht geändert werden.
- Die Durchführungszeit (Netto-Bearbeitungszeit, s.u.) ist sehr viel geringer als die sog. Overall-Lead-Time (Gesamtdurchlaufzeit, s.u.).
- Es gibt wenige bis gar keine externen Abhängigkeiten.

Wenn man die agilen Methoden betrachtet, so sieht man, dass

- es ein großes Backlog mit vielen weitgehend unabhängigen Stories gibt,
- die Reihenfolge der Stories ohne größeren Aufwand verändert werden kann,
- die Bearbeitungszeit einer Story im Verhältnis zur Liegezeit im Backlog extrem kurz ist
- und das Team möglichst autark organisiert ist, so dass alle notwendigen Arbeiten im Team erledigt werden können und wenig externe Abhängigkeiten existieren.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Die agilen Methoden weisen also eine große Ähnlichkeit mit der Produktion auf. Daher liegt es auf der Hand, sich Produktionssteuerungen und ihre systemischen Zusammenhänge genauer anzuschauen.

Gerade für die Gestaltung von Teamarbeit stammen viele Ideen aus dem Produktionsumfeld. Produktion ist im Übrigen nicht zwingend mit Taylorismus verbunden, d.h. Produktion bedeutet nicht automatisch eine bis ins kleinste Detail vorgegebene und kontrollierte Folge von Arbeitsabläufen.

Die Optimierung der Produktion ist der Optimierung der Projektarbeit um Jahre voraus. Produktionssteuerungen sind erforscht und die systemischen Zusammenhänge sind bekannt. Man kennt die Optimierungen sowie das Optimum selbst. Diesen Vorteil gilt es zu nutzen.

Produktion hat noch einen weiteren Vorteil: Durch die wesentlich einfacheren Randbedingungen im Vergleich zur Projektwelt sind Produktionssteuerungen erheblich einfacher aufgebaut und benötigen deutlich weniger Planungs-Overhead. Weniger Overhead bedeutet stets auch mehr Kapazität, um Kundennutzen zu erzielen. Mehr Kundennutzen ist zugleich ein starker Motivator für das Team. Deshalb sollten nach Möglichkeit bevorzugt Produktionssteuerungen anstatt individuell geplanter Projektabläufe eingesetzt werden.

Aus der Produktionssteuerung weiß man, was das Optimum ist

Zum Verständnis der folgenden Ausführungen erläutere ich zunächst einige Fachbegriffe der Produktion. Ich folge dabei der Darstellung von Nyhuis und Wiendahl (Nyhuis, 1999).

Arbeitssystem, Auftrag, Zufluss, Abfluss, Kapazität, Bestand

Die Produktionssteuerung betrachtet "Arbeitssysteme" als kleinste zu steuernde Einheit. Arbeitssysteme führen "Aufträge" mit einer bestimmten "Kapazität" (Einheit: Aufträge pro Zeit) durch. Erhält ein Arbeitssystem einen neuen Auftrag, so wird dies als "Zufluss" bezeichnet, schließt das Arbeitssystem einen Auftrag ab und übergibt es das Ergebnis an die Arbeitsumgebung, so stellt dies einen "Abfluss" dar. Die Anzahl der gerade im Arbeitssystem befindlichen Aufträge nennt man "Bestand". Oft wird der Bestand als relative Größe im Verhältnis zur Kapazität angegeben. Wenn der Bestand z.B. den Wert "2" hat, sind doppelt so viele offene Aufträge im System als Kapazität für den betrachteten Zeitraum.

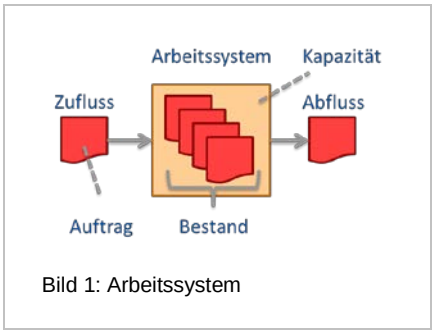


Bild 1: Arbeitssystem

Auslastung, Engpass

Normalerweise besteht eine Produktion nicht aus einem, sondern aus mehreren miteinander verknüpften Arbeitssystemen. Jedes dieser Arbeitssysteme hat eine eigene Kapazität. Durch die Vernetzung der Arbeitssysteme mit ihren jeweiligen Zu- und Abflüssen werden sie unterschiedlich ausgelastet. Die "Auslastung" eines Systems gibt an, zu welchem Anteil seine Kapazität für einen bestimmten Zeitraum produktiv verwendet wird. Das am stärksten ausgelastete System wird "Engpass" (Goldratt, 1984; Nyhuis, 1999) genannt.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Durchlaufzeit (engl. Cycle Time), Durchführungszeit (engl. Touch Time)

Die Durchlaufzeit ist die Zeitspanne, die ein Auftrag vom Eingang in das Arbeitssystem (Zufluss) bis zum Ausgang benötigt (Abfluss). Wenn ein Produktionsablauf aus mehreren Aufträgen besteht, dann dauert die Durchlaufzeit vom Zufluss des ersten Auftrags bis zum Abfluss des letzten. Die Durchlaufzeit beinhaltet alle Warte-, Rüst-, Transport- und Durchführungszeiten.

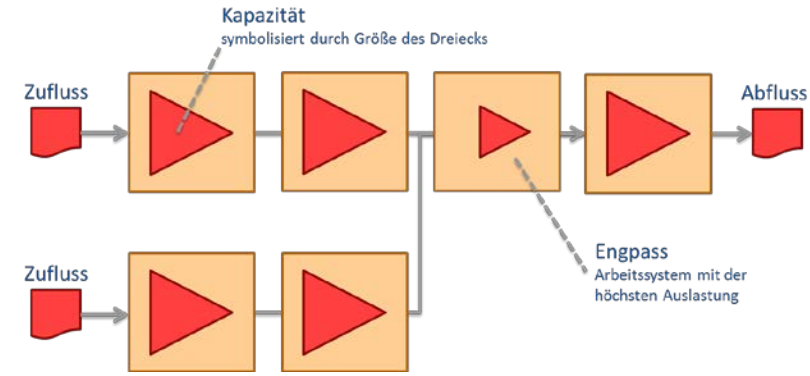


Bild 2: Vernetzte Arbeitssysteme mit Engpass.

Die Durchführungszeit (auch Netto-Bearbeitungszeit) ist die Summe aller Zeitabschnitte, in denen das Arbeitssystem (bzw. die Arbeitssysteme) direkt den Auftrag bearbeitet.

Durchsatz & Engpass

Aus der Systemtheorie sind diverse Ansätze bekannt, die das Thema Durchsatz aufgreifen: Trichtermodell von P. Nyhuis und H.-P. Wiendahl (Nyhuis, 1999), Warteschlangentheorie u.a. von A. K. Erlang (Erlang, 1909) oder die "Theory of Constraints" (TOC) von E. Goldratt (Goldratt, 1984 und 1997).

Allen gemeinsam ist die Kernaussage, dass in jedem gekoppelten System aus mehreren Teilsystemen genau ein Teilsystem existiert, dessen Kapazität die Leistungsfähigkeit (Durchsatz) des Gesamtsystems maßgeblich beeinflusst. Dieses Teilsystem wird oft als "Flaschenhals", "Engpass", "Constraint" oder "bestimmendes System" bezeichnet. Zweck aller Maßnahmen ist es, durch den Fokus auf ein bestimmendes Element die Komplexität der Steuerung zu verringern und so mit möglichst geringem Aufwand die Leistung zu optimieren.

Die TOC geht unter allen Ansätzen am weitesten und deckt das größte Anwendungsspektrum ab. Der Kern der TOC sind die "fünf Fokusschritte" (Techt, Projekt Magazin 7/2007):

1. Finde den Engpass, bzw. definiere einen sinnvollen Engpass.
2. Nutze den Engpass optimal aus, bzw. entlaste den Engpass von unnötiger Arbeit.
3. Ordne alles dem Engpass unter, bzw. richte alle Regeln im Unternehmen am Engpass aus, insbesondere auch die Prozesse des Top-Managements.
4. Erweitere den Engpass. Dies darf erst nach den anderen Schritten erfolgen, da sonst störende Regeln zementiert werden.
5. Beginne wieder mit Schritt 1 und identifiziere den neuen Engpass. Achte darauf, dass dein eigenes Denken nicht zum Engpass wird.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Am besten lässt sich dieses Vorgehen an einem stark vereinfachten Beispiel erklären: Ein produzierendes System besteht aus den drei nacheinander angeordneten Teilsystemen A, B und C (z.B. Konzeption, Umsetzung, Test). Alle Projekte benötigen jeweils genau 10 Arbeitseinheiten von A, B und C. Teilsystem A hat eine Kapazität von 120, B von 50 und C von 110 Einheiten pro Zeit (s. Bild 3).

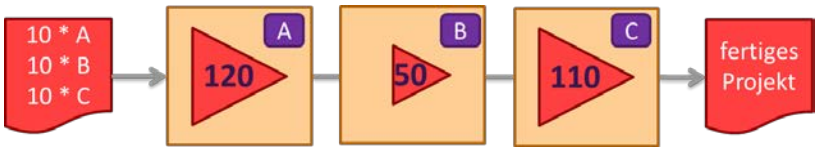


Bild 3: Einfaches Produktionssystem aus drei Arbeitssystemen

Bei optimaler Auslastung des Engpasses erhält man die in Bild 4 dargestellten Auslastungen: Es ist offensichtlich, dass B (im Beispiel die Entwicklung) den Engpass darstellt. Die Kapazität der gesamten Produktionskette wird ausschließlich vom Engpass B bestimmt, der pro Zeiteinheit 50 Aufträge durchführt. Da jedes Projekt aus zehn Aufträgen besteht, können aktuell nur fünf Projekte pro Zeiteinheit fertig gestellt werden.

Alle Optimierungen müssen sich daher auf das Arbeitssystem B konzentrieren. Wenn man die Kapazität in A oder C erhöhen würde, hätte dies keine Auswirkung auf den Durchsatz.

Wenn man aber den mittleren Aufwand eines Projekts in B halbiert oder die Kapazität in B verdoppelt – verdoppelt sich der Durchsatz an Projekten, da sowohl A als auch C über ausreichend freie Kapazität verfügen.

System	Kapazität	Auslastung
B	50	100%
C	110	45%
A	120	42%

Bild 4: Auslastungen bei Arbeitssystemen mit unterschiedlicher Kapazität.

! Wenn man dagegen pauschal die Kapazität aller drei Teilsysteme verdoppelt, wird sich der Durchsatz zwar ebenfalls verdoppeln, aber auch die Kosten steigen um 100%. Dieses Phänomen ist landläufig als "magisches Dreieck des Projektmanagements" bekannt: Der Aufwand steigt proportional zum Leistungsumfang. Dies gilt, wenn keine strukturellen Verbesserungen durchgeführt werden. Sobald man sich aber auf den Engpass fokussiert, kann man auch das magische Dreieck gewissermaßen "sprengen" und mit nur wenig höherem Aufwand ein deutlich größeres Ergebnis erzielen.

Das Management muss sich daher darauf fokussieren, den Engpass eines Produktionsprozesses optimal zu nutzen und sich zugleich in allen Entscheidungen dem Engpass unterordnen.

Durchlaufzeit & Bestand

Ergänzend zur TOC benötigen wir noch Little's Law (Little 1961), dessen Gültigkeit John D.C. Little für die Länge von Warteschlangen im Einzelhandel bewies. Es beschreibt allgemein den Zusammenhang zwischen Bestand und mittlerer Durchlaufzeit:

$$L = \lambda \cdot W$$

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Little's Law besagt, dass die durchschnittliche Anzahl von Kunden (L) in einem Wartesystem (dies entspricht dem Bestand in einem Arbeitssystem) gleich der durchschnittlichen Ankunftsrate (λ) multipliziert mit der durchschnittlichen Verweildauer (W) ist.

In unserem Zusammenhang ist vor allem die Auflösung von Little's Law nach der durchschnittlichen Verweildauer interessant:

$$W = \frac{L}{\lambda}$$

Wenn in einem eingeschwungenen System der Zufluss (Ankunftsrate λ) stabil ist, dann ist die Durchlaufzeit (Verweildauer W) proportional zum Bestand (Anzahl von Kunden bzw. offenen Aufträgen L).

Nyhuis und Wiendahl (Nyhuis, 1999) haben Little's Gesetz in eine grafische Form gebracht und um die Auslastung ergänzt. Damit wird deutlich, wo das Ziel der Optimierung sein sollte.

Die grüne Kurve gibt die Auslastung an, d.h. welcher Anteil der Kapazität für die Produktion genutzt wird. Wenn der Bestand unter den Wert 1 sinkt, verringert sich die Auslastung proportional zum Bestand. Entgegen der irrigen Annahme von vielen Managern steigt die Auslastung aber nur noch unwesentlich an, wenn man mehr Arbeit einplant als Kapazität vorhanden ist. In der täglichen Arbeit kommt es auch immer wieder zu kleinen Wartezeiten die verhindern, dass unverzüglich mit einem Arbeitspaket begonnen werden kann. Daher ist typischerweise die Auslastung bei Bestand 1 etwas geringer als 100% und es kommt zu einem sanften Übergang.

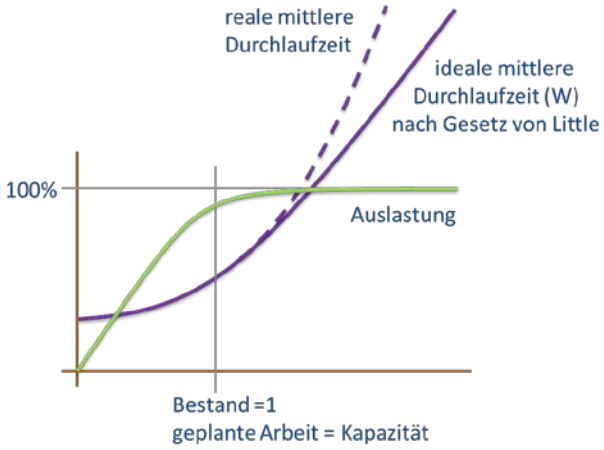


Bild 5: Auslastung und Durchlaufzeit eines Arbeitssystems als Funktion des Bestands.

Die violette Kurve zeigt die mittlere Durchlaufzeit an. So lange der Bestand einen Wert kleiner als 1 hat, können die Arbeitspakete mit wenig oder gar keiner Wartezeit bearbeitet werden. Je geringer der Bestand ist, desto näher liegt die Durchlaufzeit an der reinen Durchführungszeit. Sobald der Bestand den Wert 1 überschreitet, d.h. die eingeplante Arbeit die vorhandene Kapazität übersteigt, wächst die mittlere Durchlaufzeit proportional zum Bestand an. In diesem Fall kommt Little's Law zum Tragen. Da gerade bei Wissensarbeitern die Bestände physikalisch nicht sichtbar sind, kommen hier schnell Werte für den Bestand von 5 und mehr vor – dementsprechend lange sind die Durchlaufzeiten für die Arbeitspakete. Dieser Zustand wird oft auch als "negatives Multitasking" bezeichnet.

In der Realität (gestrichelt) sind die Durchlaufzeiten sogar überproportional länger, weil zusätzlich geistige Umschaltzeiten bei der parallelen Bearbeitung mehrerer Aufgaben hinzukommen und Desynchronisation auftritt, d.h. wechselseitige Behinderungen der Aufgaben. Aufgrund dieser Effekte wirkt sich eine Verringerung der Durchlaufzeit überproportional positiv auf die Produktivität auf den Durchsatz aus.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Optimierungsziele

Aus den aufgeführten Zusammenhängen ergeben sich die beiden theoretischen Optimierungsziele:

1. Der Bestand an offener Arbeit sollte so gering wie möglich sein: Dadurch wird die Durchlaufzeit optimal und der Durchsatz verbessert.
2. Der Bestand muss gerade so hoch sein, dass der Engpass zu 100% ausgelastet ist. Damit wird der Durchsatz optimal.

Hieraus ergibt sich auch die Formulierung des Optimierungsziels in der Praxis: "Weniger offene Arbeitspakete als Entwickler im Team!".

Doch welche Steuerung erreicht nun sicher beide Ziele?

Die Entwicklung in der Produktionssteuerung

Die Produktionssteuerungen haben im Gegensatz zu den Projektsteuerungen einen gewissen Erfahrungsvorsprung. Die ersten systematischen Überlegungen bzgl. Produktion sind mit dem Namen Henry Ford verknüpft. Er gilt als Erfinder der ersten Generation von Fertigungssteuerungen. Um den Durchsatz zu steigern und die Durchlaufzeit zu verkürzen, hat er eine rigide Begrenzung des Work-In-Progress, d.h. des Bestands, eingeführt (Goldratt, 2008).

Ausgehend von einem zentralen, steuernden Element "der Hochzeit", bei der die Motor-, Getriebe- und Fahrwerkseinheiten zusammengefügt werden, ist eine vielstufige Fertigungsstraße aufgebaut. Die Menge an Zulieferteilen (Work-In-Progress) wurde durch markierte Flächen vor den einzelnen Stufen begrenzt. Die vorgelagerten Stufen durften nur genau so viel produzieren, wie auf dieser Fläche abgelegt werden konnte. Henry Ford hat mit diesem Verfahren eine nie wieder erreichte kurze Durchlaufzeit erzielt – allerdings um den Preis der Inflexibilität: Vielzitiert ist sein Ausspruch, dass der Kunde sein Auto in jeder beliebigen Farbe haben könne – solange sie schwarz sei.

Der nächste große Fortschritt bei den Produktionssteuerungen war die Erfindung und Einführung von Kanban in den 1950er Jahren bei Toyota durch Taiichi Ohno (Ohno, 1988; Goldratt, 2008). Dies gilt als 2. Generation der Produktionssteuerung. Hier wurde die strenge Begrenzung des Work-In-Progress über die Fläche auf dem Fertigungsboden durch die Steuerung über die Bestände in Gitterboxen ersetzt. Immer dann, wenn vor einem Fertigungsschritt eine Gitterbox entnommen wurde, wird über einen Begleitschein (Kanban = Signalkarte) dem vorgelagerten Produktionsschritt signalisiert, dass dieses Teil nachproduziert werden muss. Kanban ist wesentlich flexibler und kann damit auch variantenreiche Produkte beherrschen. Der Nachteil sind die hohen Bestände (Puffer vor jedem Fertigungsschritt) und das lange Einregeln der Kanban-Limits.

Die 3. Generation der Produktionssteuerung wurde von Goldratt (Goldratt, 1987) unter der Bezeichnung "Drum-Buffer-Rope" eingeführt. Hier tritt an Stelle der Steuerung über den Bestand vor jeder Fertigungsstufe die Steuerung über den Puffer vor dem Engpass und über die Zeit bis zum Puffer. Diese Steuerung ist noch wesentlich flexibler, sehr einfach, robust und weist die kleinsten möglichen Bestände auf. Damit ist auch die Durchlaufzeit auf ein Optimum verkürzt.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Drum-Buffer-Rope ist eine Fertigungssteuerung, deren Bezeichnung aus ihren drei Hauptelementen abgeleitet ist. Goldratt stellt in seinem Roman "The Goal" das Prinzip von Drum-Buffer-Rope als Anekdote von "Herbie bei der Wanderung der Pfadfinder" dar (vgl. Goldratt, 1987). Die Herausforderung der in Bild 6 dargestellten Situation besteht darin, möglichst schnell den Zielpunkt der Wanderung zu erreichen und zwar als geschlossene Gruppe.

Die "Drum" (Trommel) ist hierbei die begrenzende Ressource in einer Wertschöpfungskette. Wenn man einen möglichst guten Durchsatz erreichen will, muss man alles tun, um diesen Flaschenhals (Herbie als der kleinste und schwächste Wanderer) optimal zu nutzen. Er wird zum Taktgeber für den ganzen Prozess, was durch die Trommel symbolisiert wird.

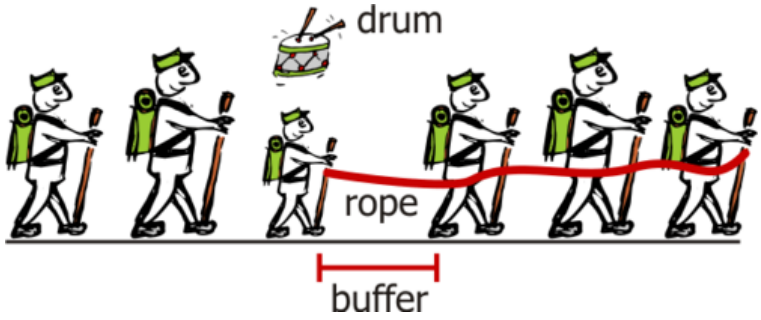


Bild 6: Visualisierung von Drum-Buffer-Rope durch eine Wandergruppe.
Lizenzbedingung: http://creativecommons.org/licenses/by-sa/3.0/de/legalcode

Damit der Flaschenhals nie leer läuft, sollte vor diesem immer eine gewisse Menge an Arbeit bereit liegen (ein kleiner freier Bereich zum Laufen). Das ist der "Buffer" (Puffer).

Damit nicht zu viele Aufträge gestartet werden, muss die Freigabe von neuen Aufträgen anhand des Füllstands des Puffers vor dem Flaschenhals erfolgen. Die Signalleitung vom Puffer zur Auftragsfreigabe wird als "Rope" (Seil zum vordersten Wanderer) symbolisiert.

Im Beispiel der wandernden Pfadfinder ist damit gewährleistet, dass die Gruppe zusammenbleibt und gemeinsam mit der maximalen Geschwindigkeit des langsamsten Wanderers vorankommt.

In der Produktion (Bild 7) ist die Reihenfolge der Elemente zwar umgekehrt – die Bedeutungen sind aber die gleichen. Die "Drum" ist die Station mit der geringsten Kapazität. Der "Buffer" ist direkt vor dieser Station und besteht aus Aufträgen, die bereit zur Bearbeitung sind. Die "Rope" ist ein Signal, durch das ein neuer Auftrag gestartet wird, sobald ein gewisser Füllstand im Puffer unterschritten wird.

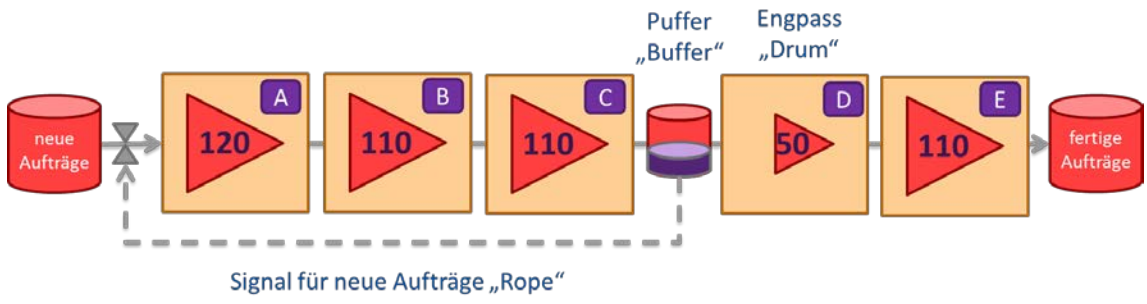


Bild 7: Drum-Buffer-Rope für eine Produktionskette mit Engpass.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Die Höhe des Puffers wird so eingestellt, dass der Puffer nur extrem selten leer läuft. Auf dem Weg zum Puffer werden die Aufträge so priorisiert, dass der Auftrag bevorzugt wird, der noch am wenigsten Zeit zum Puffer hat.

Die Entwicklung der agilen Steuerungen im Vergleich

Bei den agilen Methoden gab es eine ähnliche Entwicklung. Ausgehend vom agilen Manifest (Kent, 2001) sind unterschiedliche agile Methoden entstanden. Der Durchbruch gelang mit Scrum (Schwaber, 2011). Wenn man die Steuerung von Scrum betrachtet (und dabei etwas blinzelt), sieht sie so aus wie die Steuerung von Henry Ford und entspricht damit einer Steuerung der 1. Generation. Die Begrenzung des Work-In-Progress findet natürlich nicht über die Fläche vor der Fertigungsstufe statt, sondern durch das Time-Boxing in Form von Sprints. In einen Sprint passt nur eine gewisse Menge an Stories, die der Scrum Master aus seiner Erfahrung heraus bemisst. Damit ist der Work-In-Progress effektiv begrenzt. Um einen hohen Durchsatz zu erreichen, sollten die Stories möglichst vergleichbar und ähnlich klein sein. Bei Scrum sind die Sprints allerdings mit einer Dauer von bis zu vier Wochen relativ lang. Das ist vergleichbar mit großen Batch-Sizes oder Beständen in der Produktion. Große Batch-Sizes führen gemäß Little's Law zu langen Durchlaufzeiten. Die einzelnen Stories haben bei Scrum die Durchlaufzeit von mindestens einer Sprintdauer. Falls eine Story nicht innerhalb eines Sprints umgesetzt wird, verlängert sich ihre Durchlaufzeit jeweils um eine weitere Sprintdauer.

Die zweite Generation der agilen Methoden ist bereits wie die zugrunde liegende Produktionssteuerung benannt: "Kanban". In Kanban entfallen die Sprints und damit die großen Batch-Sizes. Es ist wie in der Produktionssteuerung der Übergang von großen Batches auf One-Piece-Flow. Man erhält hier wesentlich mehr Flexibilität – es kann auch z.B. eine dringliche Story vorgezogen werden. Auf der anderen Seite erkauft man sich die Steuerung mit einem mehrstufigen Prozess und Beständen (Kanban-Limits) vor jeder Stufe. Der Work-In-Progress ist durch die Kanban-Limits in den einzelnen Stufen zwar wirksam begrenzt, insgesamt weist Kanban aber in Summe einen hohen Bestand an offenen Arbeiten auf, was zu einer deutlichen Verlängerung der Durchlaufzeit führt. Weiterhin dauert es relativ lang, bis man die richtigen Kanban-Limits bestimmt hat. In dieser Zeit ist der Work-In-Progress höher als erforderlich, was die Durchlaufzeit verschlechtert. Wenn sich im Betrieb die Größe oder die Art der Stories ändert, müssen die Limits erneut langwierig gefunden und überflüssige Bestände abgebaut werden.

Für eine optimale Abarbeitungsgeschwindigkeit benötigen wir also eine agile Steuerung, die ohne Sprints, ohne viele Stufen und ohne große Bestände vor den Stufen auskommt und sich noch dazu schnell einregelt.

Da sich die agilen Methoden in ähnlicher Weise entwickelten wie die Produktionssteuerungen und da bei beiden die gleichen systemischen Gesetze gelten, liegt es nahe, das Drum-Buffer-Rope-Prinzip (Produktionssteuerung der 3. Generation) auf die agilen Methoden zu übertragen.

Drum-Buffer-Rope in agilem Projektmanagement

Seit rund zehn Jahren arbeite ich unter anderem mit agilen Methoden und versuche, diese zu optimieren. Die Kombination von Scrum mit einer Drum-Buffer-Rope-Steuerung begann ich vor ca. einem Jahr, was mich zu "Reliable Scrum" führte, das ich im Artikel "Scrum + Critical Chain = Reliable Scrum" (Müller, Projekt Magazin 18/2012) beschrieb. Reliable Scrum basiert auf Scrum und fasst mehrere Sprints zu einem Release/Projekt zusammen. Damit ist es möglich, durch den Einsatz von etwas Mathematik den Work-In-Progress (Backlog) von Beginn an auf die

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

richtige Menge zu begrenzen. Für die operative Steuerung dient ein Puffer am Projektende, der dafür Sorge trägt, dass der Work-In-Progress während des Projekts unter Kontrolle bleibt. Reliable Scrum verwendet zwar noch Sprints, diese sind aber eigentlich nicht mehr notwendig, um den Work-In-Progress zu begrenzen. Dies wird bereits durch die Fortschritt-zu-Pufferverbrauch-Steuerung (Müller, Projekt Magazin 18/2012) gewährleistet.

Aus Sicht der Produktionssteuerung könnte man ohne weitere Anpassungen auf eine One-Piece-Flow-Steuerung auf Story-Ebene umschalten. Allerdings steuert das Team seine Arbeit nicht auf Story-Ebene. Für die tägliche Arbeitsorganisation sind ausschließlich die aus den Stories erstellten Tasks relevant. Für einen funktionierenden Produktionsablauf müssen daher die Ebenen "Stories" und "Tasks" getrennt gesteuert werden.

Die Weiterentwicklung von Reliable Scrum zu Ultimate Scrum besteht nun darin, auf die Sprints zu verzichten und stattdessen die Stories und Tasks mit Varianten der Drum-Buffer-Rope-Steuerung zu managen.

Der Ansatz von Ultimate Scrum ist noch radikaler als Kanban. Kanban braucht viele Stufen und viele Puffer - diese entfallen bei Ultimate Scrum komplett. Damit kann man den Bestand noch weiter senken und noch schnellere Durchlaufzeiten erreichen (s. auch Müller: "Drum-Buffer-Rope als bessere Alternative zu Scrum und Kanban", 2012).

Durch die minimalen Bestände werden darüber hinaus Störungen sofort deutlich und können schneller beseitigt werden. Die Selbststeuerung wird befeuert und das Optimum schneller erreicht.

Ultimate Scrum besteht aus zwei Steuerkreisen (Bild 8):

1. Stories in Tasks zerlegen (Bild 8, linke Seite), gesteuert durch eine "Just in Time"-Steuerung – einer extrem vereinfachten Form der Drum-Buffer-Rope
2. Tasks abarbeiten (Bild 8, rechte Seite), gesteuert durch eine einstufige Drum-Buffer-Rope

Die Steuerung der linken Seite in Bild 8 ist extrem einfach. Das Aufteilen einer Story in Tasks erfordert für eine Person nur wenige Stunden Aufwand – ungefähr so viel wie für eine Task selbst. Daher wird das Schneiden von Stories ausschließlich durch die Menge der Tasks im Taskpuffer gesteuert. Wenn nur noch zwei Tasks übrig sind wird einer der Entwickler die nächste Story aus dem Backlog nehmen und sie in Tasks aufbrechen. Im einfachsten Fall startet man das Schneiden einer neuen Story genau dann, wenn kein Task mehr im Taskpuffer vorhanden ist. Der erste Entwickler, der keine Task mehr hat,

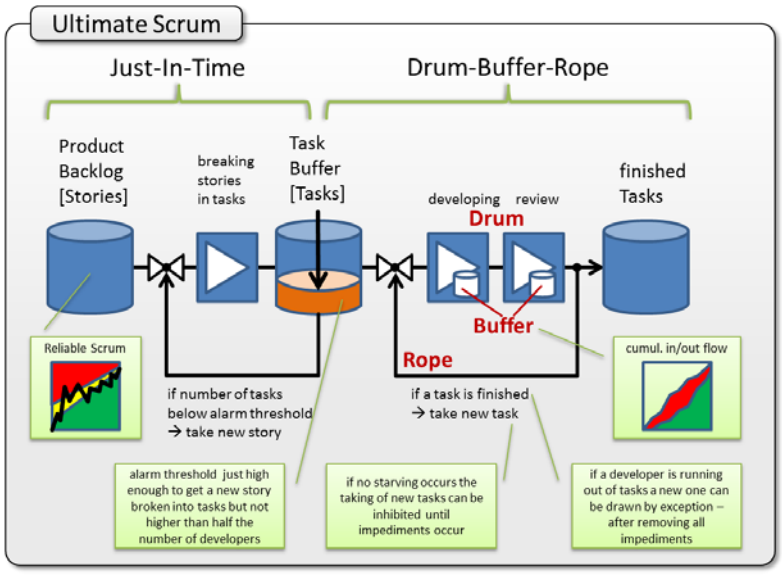


Bild 8: Steuerungselemente von Ultimate Scrum.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

übernimmt die Aufgabe des Schneidens. Normalerweise ist nach wenigen Minuten klar, was die erste Task in dieser Story ist und damit ist der Taskpuffer schon wieder ausreichend gefüllt.

Es kann sein, dass die Alarmgrenze von null, ein oder zwei Tasks zu riskant ist und ein Pufferloch auftritt – also im Taskpuffer keine Aufgabe mehr übrig ist. Dies kann passieren, wenn es während dem Schneiden der Story zu Problemen kommt. Wenn sich z.B. die Architektur geändert hat, wenn Restunklarheiten beseitigt werden müssen oder eben das Ziel der Story an die aktuellen Gegebenheiten angepasst werden muss. In diesem Fall dauert es länger, die Story zu scheiden und die Gefahr besteht, dass im Taskpuffer zu wenig Tasks verfügbar sind.

Wenn Pufferlöcher auftreten, muss man einfach die Alarmschwelle solange langsam und schrittweise erhöhen, bis keine Pufferlöcher mehr auftreten. Die Alarmschwelle sollte dabei nicht die Hälfte der Anzahl der beteiligten Entwickler überschreiten, ansonsten ist dies ein deutlicher Hinweis auf Probleme. Es würden zu früh neue Stories geöffnet. Es besteht die Gefahr, dass die Entwickler sich um mehr als eine Story kümmern und damit die Durchlaufzeit beider Stories verlängern und Probleme bei der Taskbearbeitung kaschieren.

Das Monitoring über die Abarbeitung der Stories geschieht über die aus "Reliable Scrum" (Müller, Projekt Magazin 8/2012) bekannten Werkzeuge – also das klassische "Product Burndown Chart" erweitert um die neue "Fieberkurve" (Pufferverbrauch im Vergleich zum Projektfortschritt). Diese Diagramme werden immer aktualisiert, sobald eine Story fertig gestellt wurde. Durch die vielen Messpunkte ist eine sehr genaue "Echtzeit-Überwachung" möglich, die wesentlich detaillierter ist als bei einer Taktung durch Sprints. Durch die extrem geringe Zahl offener Stories und offener Tasks kann man auch den Fertigstellungsgrad von angefangen Stories mit einbeziehen und hierdurch den Fortschritt noch detaillierter und genauer berechnen.

Die Steuerung des eigentlichen Entwicklungsprozesses auf der rechten Seite in Bild 8 ist eine einstufige Drum-Buffer-Rope! Um dies zu erkennen, muss man etwas genauer hinschauen. Normalerweise hat eine Drum-Buffer-Rope viele Prozessschritte (wie in Kanban). Hier haben wir aber einen kontinuierlichen Prozess nur mit den zwei Schritten "Entwicklung" und "Review/Test". Das Besondere ist jedoch, dass es nur eine Ressourcenart gibt: Entwickler. Diese erhalten zwar Unterstützung durch die Qualitätssicherung, deren Mitarbeiter die Tests schreiben, aber am Ende sind auch für die Reviews die Entwickler der begrenzende Faktor. Daher hat es keinen Sinn, zwischen den Prozessschritten zu unterscheiden. Beide werden durch die Verfügbarkeit der Entwickler begrenzt. Von außen sieht es aus wie eine Stufe (Entwicklung + Review).

Die Drum-Buffer-Rope Steuerung des Entwicklungsprozesses besteht nun aus folgenden drei Teilen:

1. Die Trommel ist die begrenzende Ressource - in diesem Fall sind dies die Entwickler. Die Trommel ist wie der Herzschlag der Produktionskette. Sie gibt den Takt vor – nach ihr müssen sich alle richten. Alle Prozesse vor- und nach der begrenzenden Ressource sind so auszulegen, dass es zu keinem Engpasses kommt. Die Integration und End-2-End-Tests sowie der Roll-Out sind so zu automatisieren und mit Ressourcen zu bestücken, dass fertig gestellte Funktionen unverzüglich dem Kunden zu Verfügung gestellt werden können.
2. Dann benötigen wir einen Puffer, der sich in der Regel direkt vor der Trommel befindet. Bei Ultimate Scrum gibt es jedoch nur eine begrenzende Ressource. Dadurch ist der komplette Bestand (Anzahl der offenen Tasks) selbst der Puffer. Es spielt dabei für die Steuerung keine Rolle, ob ein angefangener Task in der Be-

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

arbeitung ist oder in einem Puffer liegt. Letztendlich ist es sogar ideal, wenn es keine dedizierten Puffer gibt, da hierdurch der Bestand offener Tasks weiter reduziert wird.

3. Das "Seil" (Rope) ist die Signalleitung vom Puffer zur Freigabe neuer Tasks aus dem Taskpuffer, um neue Aufgaben zu starten. Bei Ultimate Scrum besteht die Signalleitung aus einer ganz einfachen Regel: Wenn eine Aufgabe abgeschlossen ist, dann darf eine neue Aufgabe gestartet werden, d.h. aus dem Taskpuffer entnommen werden.

Das Monitoring für die rechte Seite erfolgt über das sog. "Task-Flow-Diagramm", bestehend aus einem "aggregierten Input-Output-Diagramm" wie es Bild 9 zeigt. Aufgetragen über die Zeit sind die aufsummierten Zu- und Abflüsse der Aufgaben, die Differenz der beiden Linien ergibt den Bestand.

Das Ziel ist, den Bestand (Differenz zwischen Ein- und Ausgangslinie) so niedrig wie möglich zu halten. Dies kann auf sehr einfache Weise erreicht werden:

Initial werden keine neuen Tasks gestartet, auch wenn Hindernisse auftauchen und Entwickler an bereits begonnenen Aufgaben nicht mehr weiterarbeiten können. Hier gilt es der Versuchung zu widerstehen, eine neue Task zu beginnen. Wenn ein Entwickler eine Task nicht unverzüglich fertig stellen kann, ist diese Situation voll von Informationen über Hindernisse oder verfahrenstechnische Probleme. Es gilt vorrangig, die Hindernisse zu beseitigen und hierfür die Entwickler/das Team einzusetzen.

Wenn alle Hindernisse soweit als möglich beseitigt wurden und wenn trotz dieser Bemühungen ein Entwickler an einer begonnenen Aufgabe nicht weiterarbeiten kann, erst dann wird ein zusätzlicher Task gestartet und damit der Bestand an offenen Task um eins erhöht, um den Durchsatz zu sichern. Dies sollte aber eine Ausnahme sein und es müssen die Ursachen hierfür untersucht und dauerhaft beseitigt werden.

Wenn für eine lange Zeit keine Pufferlöcher, d.h. Arbeitsunterbrechungen trotz offener Tasks, aufgetreten sind, kann man den Bestand an in Bearbeitung befindlichen Aufgaben Schritt für Schritt wieder reduzieren, z.B. durch eine einfach temporäre Regel wie "Nach jeweils fünf 5 Tasks wird eine Task weniger gestartet".

Im Task-Flow-Diagramm eines realen Projekts (Bild 9) kann man schön sehen, wie wenig Bestand an offenen Tasks im Umlauf ist. Das Team des dargestellten Projekts war durch Reliable Scrum (bis Tag 6) bereits sehr effektiv – durch den hohen Fokus und die Umstellung der Steuerung auf Ultimate Scrum wurde die Anzahl offener Tasks noch weiter reduziert (s. Tag 7 und 8) – aktuell sind es sogar weniger offene Tasks als Entwickler und die Durchlaufzeiten sind mittlerweile unter einem Tag.

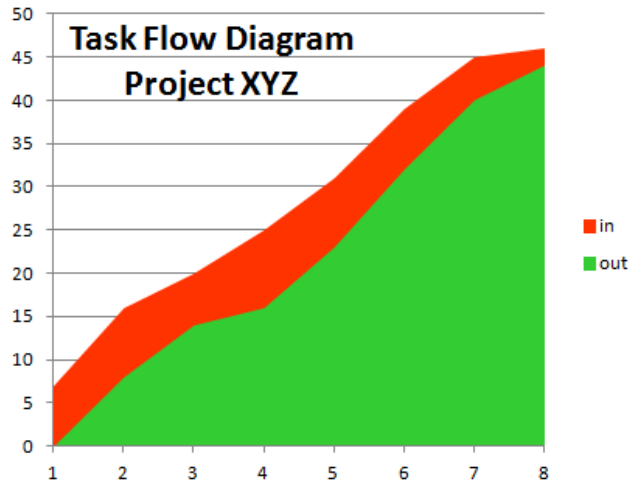


Bild 9: Task-Flow-Diagramm mit aggregierten Zu- und Abflüssen der Aufgaben.

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

Wie führt man Ultimate Scrum ein?

Voraussetzung für die Umsetzung von Ultimate Scrum ist ein etabliertes Scrum- oder Kanban-Team, das schon nach Reliable Scrum (Müller, Projekt Magazin 8/2012) arbeitet. Reliable Scrum bietet den notwendigen Schutz gegenüber zu viel Work-In-Progress, den beim traditionellen Scrum die Sprints gewährleisten. Daher ist Reliable Scrum eine zwingende Voraussetzung für die oben beschriebene Steuerung nach Ultimate Scrum. Nur so kann sich das Team voll auf die neue Verpflichtung "so wenig offene Tasks wie möglich" einlassen. Auf der Reliable-Scrum-Website (www.reliable-scrum.info) findet sich ein detailliertes Tagebuch, wie die Einführung von Reliable und Ultimate Scrum von statten gehen kann.

Da Reliable Scrum genauso wie Scrum auf der Selbstorganisation des Teams und dem Deming-Zyklus beruht, kann man Ultimate Scrum nicht durch eine einfache Anordnung des Scrum-Masters oder Product Owners erzwingen. Dies würde die Selbstbestimmtheit verletzen und den Erfolg gefährden.

Die Entscheidung für Ultimate Scrum kann nur im Team gemeinsam erfolgen. Stellen Sie die hier hergeleiteten Überlegungen vor und diskutieren Sie die daraus entwickelten Steuerungsmechanismen gemeinsam durch. Letztlich ist es eine Entscheidung des Teams, ob es bereit ist, Ultimate Scrum auszuprobieren. Die Leitfrage der Diskussion sollte sein: "Arbeiten wir mit unserer optimalen Leistungsfähigkeit? Was fehlt uns, um diese zu erreichen?" Dabei gilt es zwei Größen zu betrachten: Den Durchsatz, d.h. Tasks pro Zeiteinheit und die Durchlaufzeit, d.h. die durchschnittliche Gesamtbearbeitungsdauer einer Task.

Sowohl Durchlaufzeit als auch Durchsatz werden durch den Bestand an offenen Task getrieben. Die einfachste Frage an das Team ist daher: "Wie ist bei euch das Verhältnis von offenen Tasks zu Entwicklern?". Dies ist auch eine Frage, die jederzeit von den Stakeholdern gestellt werden kann.

Zusammenfassung

Die Suche nach dem Leistungsoptimum ist nicht nur für das Top-Management existenziell. Es gibt für ein Team nichts Motivierenderes, als wenn es von sich behaupten kann, dass es so gut arbeitet, dass der größte mögliche Nutzen für den Kunden und das Unternehmen erreicht wird.

Ultimate Scrum verschlankt die Arbeitsorganisation im Sinne von Theory of Constraints und Lean Management. Es ist darauf fokussiert, den Engpass (Kapazität der Entwickler) zu schützen und den Work-In-Progress zu reduzieren. Das Ergebnis sind optimaler Durchsatz und minimale Durchlaufzeiten. Dies setzt schnell Kapazitäten frei, die für die Optimierungen des Overheads und anschließend in die Entwicklung der Mitarbeiter investiert werden können.

Literatur

- Techt, Uwe: Theory of Constraints (ToC). Produktivitätssteigerung: Den Engpass in fünf Schritten überwinden, Projekt Magazin, 7/2007
- Techt, Uwe: Goldratt und die Theory of Constraints (ToC) – der Quantensprung im Management, Editions La Colombe, Moers, 2006

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion

- Müller, Wolfram: Das Beste aus zwei Welten kombinieren. Scrum + Critical Chain = Reliable Scrum , Projekt Magazin 18/2012
- Müller, Wolfram: Drum-Buffer-Rope als bessere Alternative zu Scrum und Kanban, 2012,
- Little, John D. C.: A Proof for the Queuing Formula: L = λW, Operations Research May/June 1961, vol. 9 no. 3 383-387
- Nyhuis, Peter und Wiendahl Hans-Peter: Logistische Kennlinien – Grundlagen, Werkzeuge und Anwendungen, Springer Verlag, 1999
- Erlang, Agner Krarup: The Theory of Probabilities and Telephone Conversations, 1909
- Goldratt, Eliyahu M.: Standing on the Shoulders of Giants - Production concepts versus production applications. The Hitachi Tool Engineering example, 2008
- Goldratt, Eliyahu M.: Critical Chain, 1997
- Goldratt, Eliyahu M.: The Goal: A Process of Ongoing Improvement, 1984
- Ohno, Taiichi: Toyota Production System: Beyond Large-Scale Production, 1988, Productivity Press
- Beck, Kent u.a.: Manifesto for Agile Software Development, 2001,
- Schwaber, Ken u. Sutherland, Jeff: The Scrum Guide, 2011,

© 2013 Projekt Magazin. Vervielfältigung, auch auszugsweise, nur mit schriftl. Genehmigung der Redaktion