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

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

"reviewed"). 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.

- 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.

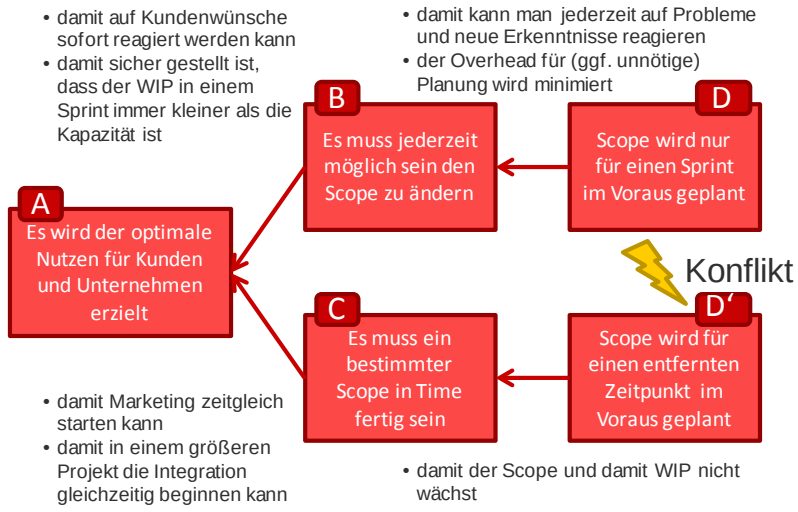

Bild 1: Die Konfliktwolke für agil (oben) vs. klassisch (unten) gilt es zu lösen.

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.

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

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.

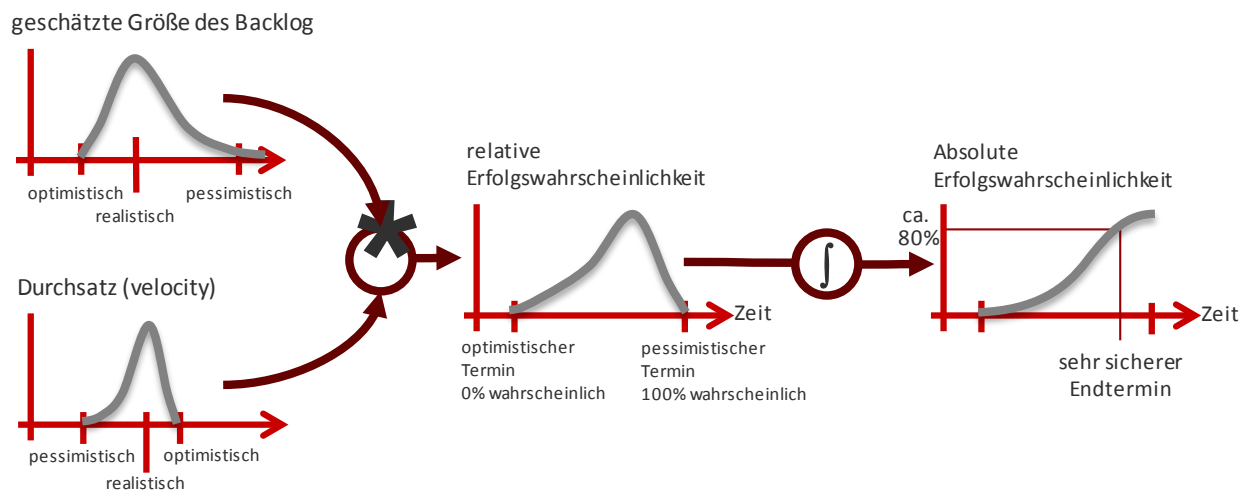

Bild 2: Berechnung der Erfolgswahrscheinlichkeit zu einem bestimmten Termin.

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

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.

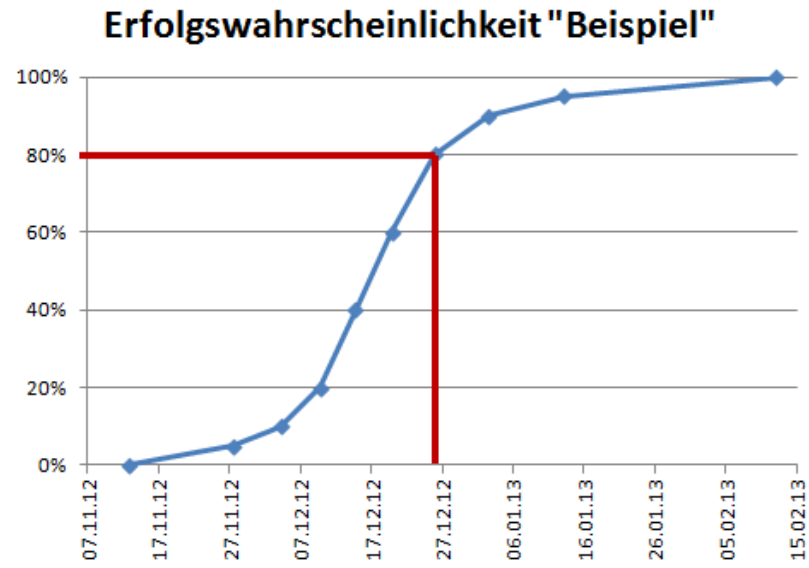

Bild 3: typischer Verlauf der Erfolgswahrscheinlichkeit über der Zeit.

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).

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.

- 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?

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.

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.

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.

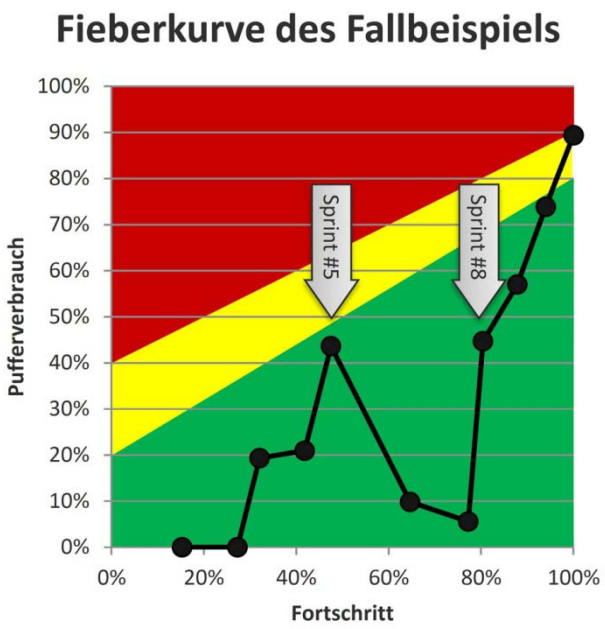

Bild 6: Reale Fieberkurve des Fallbeispiels, jeder Punkt stellt das Ende eines Sprints dar.

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.

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.

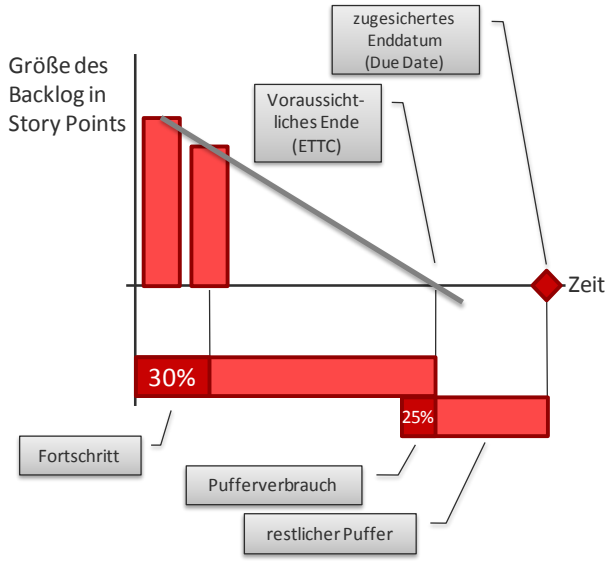

Bild 7: Berechnung des Fortschritt und des Pufferverbrauchs anhand des Product-Burndown-Chart.

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.

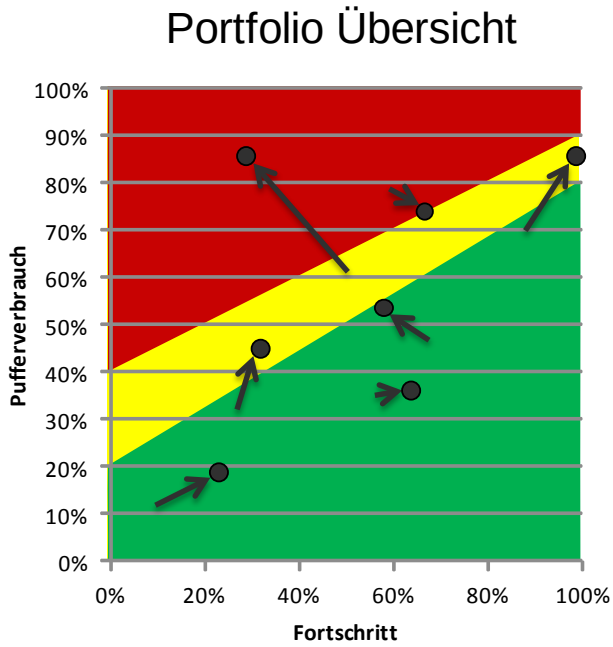

Bild 8: Beispielhafte Darstellung eines Portfolios von Scrum-Projekten.

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

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

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