Reliable Scrum

Definition Reliable Scrum

Reliable Scrum ist ein Erweiterung von Scrum mit dem Ziel Zuverlässigkeit bzgl. Zeit und Umfang herzustellen sowie Scrum kompatibel zu klassischen Projektorganisationen oder Projekten zu machen.

Reliable Scrum erweitert das klassische Scrum (gem. Scrum Guideline 2011) um die zwei Steuerungselemente (a) effektive Begrenzung der Arbeitsmenge über einen Sprint hinaus - in einem Release (b) Sicherstellung der Termintreue und des Umfangs durch Zusammenfassung der Puffer am Releaseende sowie die operative Steuerung der Ressourcen über Fortschritt zu Pufferverbrauch.

Was heißt Reliable Scrum im Kontext von Projekten?

Scrum eignet sich hervorragend bei Aufgabenstellungen in denen eine hohe Produktorientierung vorliegt, ein dediziertes Team gebildet und die Aufgabe in viele kleine unabhängige Teilaufgaben zerlegt werden kann. Solch ein Umfeld entspricht mehr einer Produktion als einem Projekt. In diesem Fall kann die Work-In-Progress Begrenzung in Form von Sprints ihre volle Wirkung entfalten und positive gruppendynamische Effekte genutzt werden. Der Vorteil von Scrum ist im Projektgeschäft aber auch ein Nachteil. Durch die hochiterative Vorgehensweise ist der Planungshorizont bei Scrum auch auf nur ein Sprint (2-3 Wochen) eingeschränkt. Scrum kann keinerlei Zusagen bzgl. Lieferumfang zu einem bestimmten Termin machen.

Um Scrum kompatibel zur Projektwelt zu machen und die Zuverlässigkeit sicher zu stellen, werden zwei Ergänzungen vorgenommen.

Es werden mehrere Sprints zu einem Release zusammengefasst. Hierdurch können Streuungen in den Schätzungen über die Sprints hinaus ausgeglichen werden. Durch diese Maßnahme wird es möglich das Backlog bzgl. des Release zu qualifizieren - welche Stories sind in / welches sind außerhalb des Releases! Des weiteren ist es notwendig alle Stories, wenigstens grob in ihrem Aufwand (Story Points) abzuschätzen. Beides bewirkt eine Auseinandersetzung mit dem Scope (und den Stakeholdern) und grobe Konzeption, so dass die Stories überschaubar und schätzbar werden. Parallel setzt sich das Team mit seiner Abarbeitungsgeschwindigkeit (Velocity sprich Kapazität) auseinander und erhält ein besseres Verständnis über sein eigene Fähigkeit. Beides - das Backlog und die Velocity - werden als Wahrscheinlichkeitsfunktionen geschätzt und betrachtet. Aus diesen Informationen lässt sich ein realistisch erreichbarer Endtermin für das Release berechnen. In diesem Termin ist ein gewisser Puffer enthalten, der so dimensioniert ist, dass der Work-In-Progress sicher auf ein sinnvolles Maß begrenzt ist.

Durch den nun expliziten Puffer wird es möglich, einen operativen Status des Releases zu ermitteln. Es wird immer der Fortschritt im Verhältnis zum Pufferverbrauch herangezogen. Ein Release ist grün, wenn der Fortschritt größer ist als der Pufferverbrauch. Durch diese zentrale Visualisierung von Fortschritt zu Pufferverbrauch kommt es zu einer Fokussierung des gesamten Teams auf den Umfang des Backlog und die Abarbeitungsgeschwindigkeit. Es wird ein operativ wirksamer und kontinuierlicher Verbesserungsprozess in Gang gesetzt. Reliable Scrum wirkt hierbei wie ein Katalysator, der alle anderen positiven Eigenschaften von Scrum verstärkt. Die Mitarbeiter erleben ihre Arbeit als planbar, konzentriert und produktiv, was sich im Folgenden positiv auf Motivation, Qualität und Innovation auswirkt. Der Product Owner erhält ein Werkzeug um jederzeit beurteilen zu können, in welchem Umfang er neue Stories dem Backlog hinzufügen kann oder entfernen muss.

Durch die Einführung von Reliable Scrum kann Scrum zuverlässig Scope und Termin einhalten und wird damit kompatibel zu Projektorganisationen.

Hintergrund zu Reliable Scrum

Im Folgenden werden die Element von Reliable Scrum und ihre Wirkung im Einzelnen beschrieben. Reliable Scrum setzt voraus, dass das Team etwas Erfahrung mit Scrum aufweist und seine Velocity abschätzen kann. Dies ist normalerweise nach 2-3 Sprints gegeben. Wie beim klassischen Projekt ist diese Zeit notwendig, um sich Klarheit über den Auftrag und die Kapazitäten zu verschaffen.

Jedes Scrum Team beginnt also mit ganz normalem Scrum. Erst nach 2-4 Sprint entsteht der fließende Übergang zu Reliable Scrum.

Begrenzung des "Work-In-Progress" (Einlastung)

Eine der Stärken von Scrum ist die harte Begrenzung des Work-In-Progress in einem Sprint. Gleichzeitig gibt es keine Begrenzung über den Sprint hinaus, was es unmöglich macht eine Termin oder Scopeaussage zu treffen. Mit folgendem Vorgehen kann man einen sicheren Endtermin und damit eine sichere Begrenzung des Work-In-Progress erreichen:

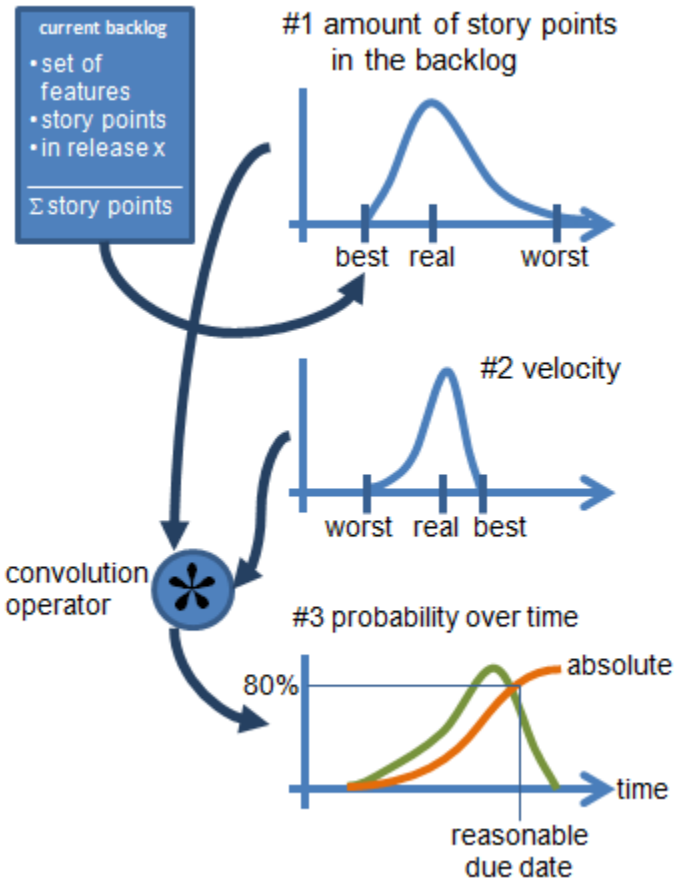
1. Als erstes muss das Backlog qualifiziert werden. Jede Story muss einem Release zugeordnet werden. Ein Release sollte min. 6 Sprints umfassen, da ansonsten die ausgleichende Wirkung nicht genutzt werden kann.
2. Für jede Story muss eine Abschätzung des Aufwandes erfolgen. Dies geschieht in als Story Points z.B. in Form von Schätz-Poker-Runden. Wichtig ist, dass alle Stories geschätzt sind und dass der Anteil der "großen" Stories <3-5% beträgt. Als groß gilt eine Story, wenn sie 42 oder mehr Story Points aufweist. In diesem Fall muss z.B. durch eine "Research Story" im nächsten Sprint konzeptionelle Arbeit geleistet werden, so dass die Story zerlegt werden kann und unter die einzelnen Stories weniger als 42 Story Points aufweisen.
3. Der Arbeitsaufwand des Backlog wird als Wahrscheinlichkeitsfunktion beschrieben. Der aktuelle Inhalt des Backlog in Story Points für

das Release entspricht der optimistischen Schätzung einer 3-Punkt-Schätzung. Das Team gibt nun zwei weitere Schätzungen ab (a) die realistische - was denkt das Team, welcher Umfang an Stories im Laufe des Release noch realistisch auftauchen wird (typ. 10-20%) und (b) was denkt das Team, welcher Umfang an Stories im "Worst Case" noch auftauchen wird (typ. 30-50%). Diese drei Punkte spannen die Wahrscheinlichkeitsfunktion des Backlog (Arbeitsumfanges) auf.

Durch diese drei Aufgaben kommt typischerweise ein sehr guter Prozess in Gang. Das Team und der Product Owner setzen sich intensiv mit ihren Stakeholdern und dem Backlog auseinander. Es kommt zur Klärung der Erwartungshaltung was im Folgenden dem Release deutlich zu gute kommt. Parallel hierzu wird die Abarbeitungsgeschwindigkeit ermittelt.

1. Es wird aus den vorangegangenen Sprints die Velocity herangezogen bereinigt und gemittelt. Es entsteht ein mittlere Velocity pro Woche. Diese bildet wiederum den realistischen Wert einer Wahrscheinlichkeitsfunktion.
2. Das Team gibt nun zwei weitere Schätzungen ab (a) die pessimistische - was denkt das Team welche Velocity auch im "Worst Case" noch erreicht werden kann. Hier sind Urlaub, Krankheit, Wegfall von Teammitgliedern oder auch gruppendynamische Prozesse zu berücksichtigen. (b) die optimistische - was denkt das Team welche Velocity es erreichen kann wenn es wirklich läuft. Typisch sind hier Schätzungen um die +/- 20%

Durch die Frage nach der Velocity kommt typischerweise eine Auseinandersetzung mit den eigenen Fähigkeiten des Teams in Gang. In Folge entsteht ein starkes Verantwortungsgefühl bzgl. der eigenen Velocity. Um nun einen realistischen Endtermin des Releases zu ermitteln, muss man nur noch beide Wahrscheinlichkeiten falten und integrieren. Man erhält die absolute Erfolgswahrscheinlichkeit über der Zeit. Aus der Praxis hat es sich bewährt, eine Erfolgswahrscheinlichkeit um die 80% anzustreben. Unter 80% steigt das Versagensrisiko schnell und deutlich an. Sicherheiten über 80% sind sehr teuer und verführen zu Studentensyndrom und Parkinson. Die Erfahrung zeigt, dass 80% ausreichend ist und etwaige Probleme vom Team in den Griff zu bekommen sind.



Das Bild zeigt die drei Wahrscheinlichkeitsfunktionen (a) Backlog (b) Velocity und (c) Erfolgswahrscheinlichkeit über der Zeit (grün = relative und orange = absolut). Ein einfaches Excel-Sheet zur Berechnung findet sich auf der Website zu Reliable Scrum. Das Ergebnis ist ein realistisch erreichbarer Endtermin des Release und ein Puffer bzgl. Backlog und Velocity.

Puffer ans Projektende

Wie in Critical Chain wird der Puffer aggriegt am Releaseende. Doch wie groß ist der Puffer eigentlich?

Aus Psychologie der Schätzungen und Erfahrung weiß man, dass in klassischen Projektorganisationen ca. 50% Puffer in den Aufwänden versteckt sind. In Organisationen die Erfahrung mit Scrum haben, ist dies weniger stark ausgeprägt. Hinzu kommt, dass durch die Ermittlung des Endtermins des Releases in Form der Wahrscheinlichkeiten gar keine Abschätzung von Puffern in Arbeitspaketen mehr notwendig ist. Der Puffer ist durch den Endtermin implizit festgelegt.

Es gibt zwei einfache Möglichkeiten die Puffergröße festzulegen.

1. Die Pufferberechnung findet typischerweise nach dem 3-4 Sprint statt. Man kann nun den Puffer so einstellen, dass der Fortschritt des Projektes ungefähr dem Pufferverbrauch entspricht. Das ist ein fairer Start für das Team.
2. Man kann den Puffer einfach je nach Situation wählen. Die ideale Puffergröße ist ca. 25-30% der Releasedauer. Kleine Puffer führen dazu, dass die Projektampel schneller rot wird und Probleme sehr sehr schnell eskalieren (das kann gewünscht sein). Puffer unter 15% werden nicht mehr als Puffer wahrgenommen und verlieren ihren Effekt. Puffer über 35% verführen zum Studentensyndrom und Parkinson und reagieren sehr langsam

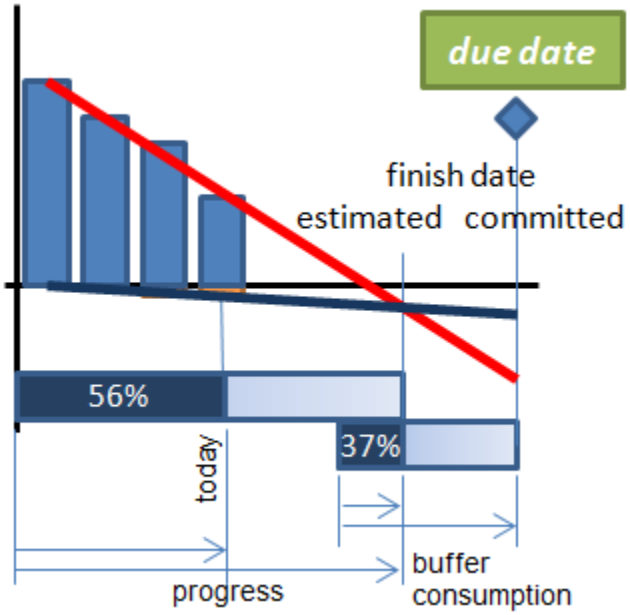
Idealerweise legt man den Puffer mit 33% fest. Dieser Wert hat sich bewährt und er zeigt, dass es sich hier um eine "willkürliche" Festlegung handelt. Je nach Situation kann diese aber variiert werden.

Execution Control oder operative Projektsteuerung

Wenn man einen Projektpuffer hat, kann man endlich operativ aussagekräftige Projektampeln definieren. O.k. es ist keine Ampel mehr sondern eine Fieberkurve - aber die Farben sind die gleichen.

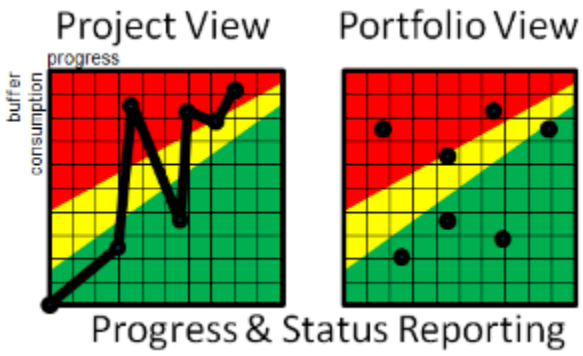
Das Prinzip ist einfach:

- Wenn der Fortschritt auf der kritischen Kette größer ist als der Pufferverbrauch > grün
- Wenn der Fortschritt auf der kritischen Kette kleiner ist als der Pufferverbrauch > rot
- Der Übergang von grün auf rot ist gelb (ist zwar komisch aber so ist es halt bei Ampeln)
- Wenn der Puffer verbraucht ist > schwarz = tot



Der Fortschritt wird berechnet, in dem das Verhältnis der Dauern vom jetzigen Zeitpunkt und vom aktuell geschätzten Ende zum Releasestart gebildet wird. Der Pufferverbrauch ermittelt sich aus dem Verhältnis von jetzigem Zeitpunkt zum Pufferstart zur Pufferdauer. Ein einfaches Excel-Sheet zur Berechnung findet sich auf der Website zu Reliable Scrum.

In der Praxis wird das in eine Fieberkurve eingetragen. Jedes Projekt beginnt links unten bei Fortschritt 0% und Pufferverbrauch 0% und wandert über die Zeit nach rechts oben 100% Fortschritt und lässt wenn möglich ein bisschen Puffer übrig.



Die Projektsicht dient vor allem dazu das Projektteam auf Kurs zu halten. Die Wirkungen sind typischerweise:

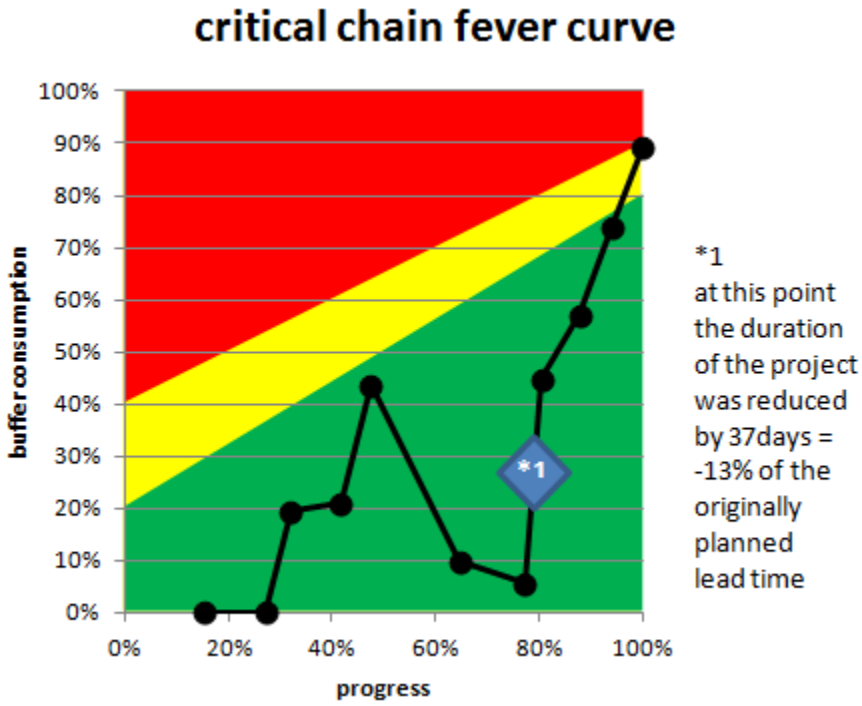
- das Team erhält schnell Feedback über den Fortschritt und echten Status
- wenn das Projekt in die rote Zone gerät, fokussiert sich das Team auf die Rückgewinnung des Puffers (selbstständig)
- die Projektprobleme werden schneller sichtbar
- die Verantwortlichkeit des Team für den Termin steigt - das Team wird gestärkt
- der Product Owner kann anhand des Diagramm einfach ermitteln, wie viele Stories er hinzunehmen kann (grün) oder weglassen muss (rot)

Die Portfoliosicht dient vor allem den Stakeholdern den Führungsebenen. Die Wirkungen sind typischerweise:

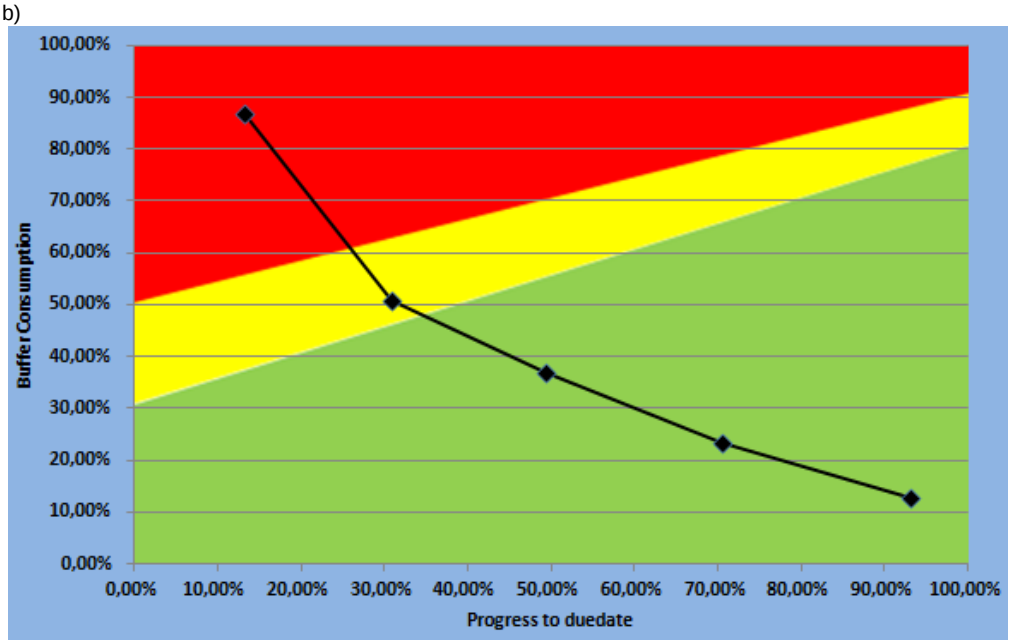
- transparente und visuelle, schnell zu erfassende Information über den Zustand des kompletten Portfolios
- objektiver Status täglich
- Vertrauen in die Projekte und Teams steigt und damit weniger Reporting und µManagement
- Information über den Belastungszustand des Unternehmens - wenn mehr als 10% der Projekte rot sind ist das System nicht mehr selbststeuernd - dann muss die Last reduziert werden

Hier ein paar Beispiele aus der Praxis:

a)



b)



In Projekt a) konnte nach ein paar Sprints sogar der Zieltermin vorverlegt werden. In Projekt b) wurde mit eine sehr kleine Projektpuffer gestartet und das Team hat alles daran gesetzt, diesen Puffer zu vergrößern ("buffer regain") - was deutlich gelungen ist. Jetzt haben sie eine stabilen Puffer und werden aller Voraussicht nach auch schneller als geplant fertig.

Und das gibt es noch viel mehr was plötzlich passiert

- Wenn ein Release rot wird, kann man zusammen mit der Velocity schnell erkennen wo das Problem liegt. Wenn die Velocity gleich bleibt dann muss es beim Product Owner liegen (zu viele Stories hinzugenommen) oder eben im Team - Velocity nicht gehalten.
- Probleme werden schnell sichtbar - auch das im Scrum gefürchtete "sich im Kreis drehen" wird früher erkannt und es kann gegen gesteuert werden.
- Durch die zuverlässige Absicherung des Termins kann man auch in zeitkritischen Projekten Scrum einsetzen.
- Man kann durch die gleiche Darstellung des Status in einem Portfolio Scrum und Critical Chain mischen.
- Man kann sogar in einem Projekt Teilprojekt oder Phasenweise zwischen Scrum und Critical Chain wechseln.

Hintergrund zu Reliable Scrum

Reliable Scrum wurde 2011 von Wolfram Müller von Speed4Projects beschrieben.

Und nun?

- Wo finde ich Erfahrungsberichte? Auf der Website zu Reliable Scrum unter "Beginnings" oder "In Practice"
- Wo finde ich Literatur: E. Goldratt "die Kritische Kette" - hier sind die Grundlagen für Reliable Scrum gelegt
- Wo gibt's Seminare zu Reliable Scrum? Speed4Projects oder Projekthaus-Stuttgart
- Wo gibt es die Werkzeuge zu Reliable Scrum? Auf der Website zu Reliable Scrum unter "Downloads"
- Gibt es eine Übersicht über Reliable Scrum? Ja - Reliable Scrum on an Page (o.k. auch von mir - und die Bilder sind die gleichen)

Beiträge auf openPM zu Reliable Scrum

erscheinen hoffentlich bald viele ...