Scrum besser machen

Eine inoffizielle Sammlung von Tipps und Einsichten für eine gute Scrum-Umsetzung

Geschrieben von Certified Scrum Coach und Trainer Peter Hundermark
Deutsche Übersetzung: Andreas Schliep, Henning Wolf, Peter Beck & Sebastian Lauber

Leaders in facilitating lasting Agile change

scrumsense.com

Warum dieser Leitfaden?

Der Certified Scrum Trainer und Coach Jim York sagt: „Scrum ist einfach. Scrum machen ist schwer.℠“

Viele Menschen, denen ich in Unternehmen begegne, sagen, dass sie es schwer finden, sich über den Einstieg in Scrum zu informieren. Andere haben Teams, die zwar einigen agilen Praktiken folgen, aber weit davon entfernt sind das zu werden, was Jeff Sutherland „hyper-produktiv“ nennt.

Ich hoffe, dieses kleine Büchlein kann als Quelle der Inspiration dienen, um dir dabei zu helfen, Scrum und Agil besser einzusetzen. Wichtiger noch: Ich hoffe, dass es dich dazu ermutigt, dich selbst, dein Team und deine ganze Organisation von den alten Arbeitsweisen wegzubewegen, die einfach nicht – *“funktionieren”* – und neue Wege zu finden, die zu besserer Qualität, schnellerer Lieferung und – vor allem – mehr Spaß führen.

Lass uns wissen, was du magst und was nicht, so dass wir es verbessern können.

Und nun geh los und fang an!

Peter Hundermark
Kapstadt, November 2009
Zweite Ausgabe

Mitwirkende an der deutschen Übersetzung:
Peter Beck
Sebastian Lauber
Andreas Schliep
Henning Wolf
(2.1.1)

Was ist Scrum?

Woher kommt Scrum?

Scrum ist ein Management-Rahmenwerk, innerhalb dessen komplexe Produkte entwickelt werden können. Scrum ist abgeleitet von der Arbeit im Wissensmanagement, komplexen adaptiven Systemen und der Theorie der Steuerung von empirischen Prozessen. Es beinhaltet Einflüsse von beobachteten Software-Entwicklungsmustern und der Theory of Constraints.

Scrum und Agil

Scrum ist das beliebteste der agilen Vorgehensmodelle. Es wird häufig in Verbindung mit Extreme Programming (XP) genutzt.

Was ist das Problem?

- Auslieferungen und Produktivnahmen dauern zu lange
- Die Stabilisierung dauert zu lange
- Änderungen sind schwer einzubringen
- Die Qualität sinkt
- Todesmärsche lassen die Moral sinken

Seit Jahrzehnten haben Softwareentwickler versucht, definierte Methoden der Arbeit und des Projektmanagements einzusetzen. Definierte Methoden sind angemessen, wenn die Voraussetzungen wohl definiert sind und die Art und Weise vorhersagbar ist, wie diese in Ergebnisse überführt werden. Softwareentwicklung und andere Arten der komplexen Arbeit sind nicht für solche Vorgehensweisen geeignet. Die hohe Rate an Projektfehlschlägen und unzufriedenen Kunden illustrieren das sehr deutlich.

Wie hilft Scrum, das Problem zu lösen?

Alistair Cockburn [Cockburn 2008] beschreibt Softwareentwicklung als ‘ein kooperatives Spiel von Interaktion und Kommunikation‘.

Traditionelle Entwicklungsmethodologien verlassen sich auf Dokumente, um Wissen festzuhalten und von einem Spezialisten zum anderen weiterzugeben. Die Rückkopplungs-Zyklen sind zu lang, oder existieren gar nicht. Jahrzehnte der Projekt-Unterperformance haben diese Arbeitsmethoden als absoluten Fehlschlag aufgezeigt.

Scrum bietet den Menschen eine Plattform, effektiv zusammenzuarbeiten; und es macht unermüdlich auf jedes Problem aufmerksam, dass sich ihm in den Weg stellt.

Die Essenz von Scrum

Das ist die Essenz von Scrum:

- Das Team erhält klare Ziele vorgegeben
- Das Team organisiert sich um die Arbeit selbst
- Das Team liefert regelmäßig wertvolle Funktionalitäten
- Das Team erhält Feedback von der Außenwelt
- Das Team reflektiert seine Arbeitsweisen, um sich zu verbessern
- Die gesamte Organisation hat Einblick auf den Fortschritt des Teams
- Team und Management kommunizieren ehrlich über Fortschritt und Risiken

Diese Art der Arbeit basiert auf den Werten Selbstrespekt, Respekt gegenüber anderen, Vertrauen und Mut.

Warum schweigt Scrum über Softwareentwicklungs-Praktiken?

Scrum versucht nicht, Teams vorzuschreiben, wie sie ihre Arbeit zu machen haben. Scrum erwartet von Teams, dass sie tun, was auch immer erforderlich ist, um das gewünschte Produkt zu liefern. Es ermächtigt sie, genau das zu tun. Entwicklungspraktiken und Werkzeuge ändern und verbessern sich ständig – und gute Teams werden permanent daran arbeiten, sie zu ihrem Vorteil zu nutzen.

Anwendbarkeit

Während Scrum zuerst bei der Entwicklung von Software-Produkten eingesetzt wurde, passt es doch zu allen Formen der komplexen Arbeit. Es wird heutzutage genutzt, um Software- und Hardwareentwicklungen, Support, Werbung und Marketing, Kirchen und ganze Unternehmen zu managen.

Wie lässt sich Scrum auf herkömmliche Methoden abbilden?

Die kurze Antwort ist, dass es sich nicht auf herkömmliche Methoden abbilden lässt. Agilität und Scrum basieren auf einem anderen Paradigma. Die Begründer Jeff Sutherland und Ken Schwaber haben häufig bemerkt, dass Versuche, definierte Methoden auf empirische Methoden abzubilden, zum Scheitern verurteilt sind.

Wird Scrum in meinem Unternehmen gelingen?

Das hängt von dir ab! Die Implementierung in deinem Unternehmen kann fehlschlagen, weil die Mitarbeiter nicht resolut genug versuchen, die Probleme zu überwinden, die Scrum mit Sicherheit aufdecken wird. Und doch haben Tausende Teams auf allen Kontinenten und in jedem Wirtschaftsbereich Erfolg dabei, ihre Arbeitswelt heute besser als die von gestern zu machen.

Das Agile Manifest

Im Februar 2001 trafen sich siebzehn unabhängige ‚leichtgewichtige' Software-Methodologen, um miteinander zu reden und eine gemeinsame Basis zu finden. Unter ihnen befanden sich die Scrum Begründer Jeff Sutherland und Ken Schwaber, zusammen mit Mike Beedle, der an den initialen Scrum Patterns gearbeitet und das erste Buch über Scrum mitgeschrieben hatte. Die Gruppe nannte sich selbst die „Agile Alliance" und einigte sich auf ein Manifest für die Agile Softwareentwicklung. Des Weiteren definierten sie eine Reihe von zwölf Prinzipien hinter dem Manifest, die in voller Länge auf der nächsten Seite wiedergegeben sind.

Kommentar

Das Agile Manifest und seine zwölf Prinzipien bestehen aktuell seit nahezu einem Jahrzehnt. Sie bleiben der Lackmustest für alle agilen Methoden und Praktiker, nach dem sie ihre Arbeitsweise beurteilen können. Die am meisten geäußerte Kritik daran war die Betonung auf Softwareentwicklung, während agile Methoden wesentlich weitläufiger angewendet werden können.

Scrum ist eine Spielart der Agilität. Die Übereinstimmung mit dem Agilen Manifest und allen darin enthaltenen Prinzipien bedeutet nicht unbedingt, dass ein Team oder eine Organisation Scrum befolgt. Jedoch bedeutet eine fehlende Übereinstimmung mit IRGENDEINEM dieser Prinzipien, dass du NICHT Scrum machst – oder Agile!

Manifest für die Agile Softwareentwicklung

Wir entdecken bessere Wege, Software zu entwickeln, indem wir es tun und anderen dabei helfen, es zu tun. Durch diese Arbeit schätzen wir:

Individuen und Interaktionen höher als Prozesse und Werkzeuge
Funktionierende Software höher als vollständige Dokumentation
Zusammenarbeit mit Kunden höher als Vertragsverhandlungen
Auf Änderungen zu reagieren höher als einem Plan zu folgen

Das heißt, während wir den Wert der Dinge auf der rechten Seite sehen, schätzen wir die Dinge auf der linken Seite als wichtiger ein.

Prinzipien hinter dem Agilen Manifest

1. Unsere höchste Zielsetzung ist es, den Kunden durch die frühe und kontinuierliche Lieferung wertvoller Software zufrieden zu stellen.

2. Begrüße Anforderungsänderungen, auch spät in der Entwicklung. Agile Prozesse nutzen den Wandel für den Wettbewerbsvorteil des Kunden.

3. Liefere häufig funktionierende Software, alle paar Wochen oder Monate, eher in kürzeren Zeiträumen.

4. Geschäftsleute und Entwickler müssen täglich im Projekt zusammen arbeiten.

5. Baue Projekte um motivierte Individuen. Gib ihnen die Umgebung und Unterstützung, die sie brauchen, und traue ihnen zu, den Job zu erledigen.

6. Die effizienteste und effektivste Art der Informationsweitergabe an und in einem Entwicklungsteam ist die Konversation von Angesicht zu Angesicht.

7. Funktionierende Software ist das primäre Maß für Fortschritt

8. Agile Prozesse fördern nachhaltige Entwicklung. Die Sponsoren, Entwickler und Benutzer sollten ein gleichbleibendes Tempo ohne Unterbrechung einhalten können.

9. Die fortwährende Beachtung von technischer Exzellenz und gutem Design verbessert die Agilität.

10. Einfachheit – die Kunst der Maximierung von nicht angegangener Arbeit – ist essentiell.

11. Die besten Architekturen, Anforderungen und Designs entstehen in sich selbst organisierenden Teams.

12. In regelmässigen Intervallen reflektiert das Team darüber, wie es effektiver werden kann, und passt dann sein Verhalten entsprechend an.

Highsmith (2001)

Die Scrum Rollen

Einführung in die Rollen

Es gibt keine Projektleiter-Rolle in Scrum. Die Verantwortlichkeiten des traditionellen Projektleiters sind auf die drei Rollen im Scrum Team aufgeteilt:

- Der Product Owner managed das *Produkt* (und die Wirtschaftlichkeit)
- Der ScrumMaster managed den *Prozess*
- Das Team managed *sich selbst*.

Das ist eine Herausforderung für diejenigen, die gegenwärtig diese Rolle ausfüllen, und für Manager in Organisationen, in denen sie arbeiten. Michele Sliger und Stacia Broderich haben einen hilfreichen Leitfaden für den Übergang vom Projektleiter zum Agilen Coach geschrieben [Sliger and Broderick 2008].

Es gibt über den Product Owner und ScrumMaster hinaus keine eingesetzten Leiter des Scrum Teams; es wird keiner benötigt. Der Bedarf an Linien-Managern sinkt, wenn Teams sich größtenteils selber managen. Bei einer Organisation, die zur Agilität übergegangen ist, unterstehen häufig 50 Teammitglieder direkt einem einzelnen Vorgesetzten.

Selbstorganisation

Selbstorganisation hat überhaupt nichts mit einem laissez-faire Ansatz gemein; im Gegenteil: selbstorganisierte Teams sind hoch diszipliniert. Sie erhalten die volle Autonomie, und tragen damit eine höhere Verantwortung für die Übereinstimmung der Lieferung mit ihren eigenen Versprechen. Sie sind dazu ermutigt, absehbare Risiken einzugehen, und durch Fehlschläge und Selbstreflektion zu lernen. Ein hohes Maß an Vertrauen und eine hohe Verbindlichkeit sind automatische Resultate von wirklich selbstorganisierten Teams.

Teams, die gerade mit Scrum anfangen, werden einige Unterstützung benötigen, um ihre neuen, weiteren Grenzen auszuloten und die Verantwortung anzunehmen. Sie müssen häufig ein starkes „Muskelgedächtnis“ der schlechten Art überwinden, in der sie gesteuert wurden, in der sie teilweise seit Jahren gearbeitet hatten.

Selbstorganisation ist keine Option in Scrum; sie ist ein Kernprinzip. Ohne sie werden keine hochleistungsfähigen Teams entstehen. *Caveat emptor*!

Product Owner

Die Verantwortlichkeiten der Product Owner Rolle sind:

- Arbeit an einer gemeinsamen Vision
- Zusammenstellung von Anforderungen
- Managen und Priorisieren des Product Backlogs
- Abnahme der Software am Ende jeder Iteration
- Managen des Releaseplans
- Die Profitabilität des Projekts (ROI)

Metapher: Der Product Owner ist ein *CEO*.

ScrumMaster

Die Verantwortlichkeiten der ScrumMaster Rolle sind:

- Ermutigung und Behütung des Teams
- Beseitigung von Hindernissen
- Den Prozess in Bewegung halten
- Die Sozialisation von Scrum im Organisationsumfeld vorantreiben

Metapher: Der ScrumMaster ist ein *Moderator, Coach, Mentor* und *Bulldozer*!

Team

Die Verantwortlichkeiten des Team oder Teammitglied-Rolle sind:

- Schätzen der Größe von Backlog-Einträgen
- Lieferversprechen für Inkremente auslieferbarer Software abgeben & einhalten
- Den eigenen Fortschritt überwachen
- Sich selbst organisieren – dem Product Owner verantwortlich gegenüber zu liefern, wie versprochen

- *Teammitglieder können Entwickler, Tester, Analysten, Architekten, Autoren, Designer und sogar Benutzer sein. Das Team ist funktionsübergreifend, das heißt, dass es mit allen Mitgliedern ausreichende Fähigkeiten besitzt, die Arbeit zu erledigen. Es gibt keine vorgegebene Führungshierarchie innerhalb der Teammitglieder.*
- *Das Scrum Team enthält alle drei Rollen: einen Product Owner, einen ScrumMaster und fünf bis neun Teammitglieder.*

Die Sprint Meetings

Der Sprint ist der Herzschlag des Scrum Zyklus. Er wird markiert durch das Sprint Planning am Start und Sprint Review und –Retrospektive am Ende. Die Länge des Sprints ist fest und wird niemals erweitert. Die meisten Scrum Teams wählen zwei, drei oder vier Wochen als ihre Sprintdauer. An jedem Tag im Sprint hält das Team ein Daily Scrum Meeting ab. Jedes Meeting in Scrum ist mit einer strengen Timebox versehen. Das bedeutet, dass es eine maximale Dauer hat – nicht, dass die gesamte Zeit ausgenutzt werden muss. Für einen 30-Tage (oder vier Wochen) Sprint sind die Timeboxen für Sprint Planning 1 & 2, Review und Retrospektive auf jeweils 4 Stunden angesetzt. Die Dauer sollte für kürzere Sprints dementsprechend proportional zur Sprintlänge angepasst werden.

Einige Schlüsselmerkmale der Meetings werden in den folgenden Sektionen beschrieben. Zunächst möchte ich allerdings ein paar wertvolle Erfahrungen teilen.

- *Ich finde zweiwöchige Sprints gut für den Anfang. Nach drei Sprints kann das Team die Sprintlänge neu beurteilen.*
- *Teams brauchen drei Sprints, um die neuen Konzepte zu verstehen, alte Gewohnheiten abzulegen, und einen Zusammenhalt als Team zu erlangen.*
- *Halte kein Sprint Planning am Montagmorgen ab. Das Team ist noch nicht auf der Höhe, und Montag ist der übliche Tag für Ferien und Krankmeldungen. Vermeide Reviews und Retrospektiven am Freitag Nachmittag. Das Team ist müde und aufs Wochenende eingestellt. Daher solltest du Sprintübergänge von Dienstag bis Donnerstag legen.*
- *Teams in zweiwöchigen Sprints neigen dazu, alle Sprintübergangsmeetings an einem Tag abzuhalten. Anders ausgedrückt, sie starten den Tag mit dem Review, gefolgt von der Retrospektive. Nach dem Mittagessen kommen dann Sprint Planning 1 und 2. Die Überlegung dazu ist, alle Meetings aus dem Weg zu kriegen und 9 volle Tage zum Arbeiten zu haben. Nach meiner Erfahrung gibt es zwei Probleme mit diesem Ansatz:*
  - *Das Team versteht nicht, dass diese Meetings Teil ihrer Arbeit sind - genau genommen der wichtigste Teil, der wohl gelingen muss!*
  - *Am letzten Teil des Tages, dem Sprint Planning 2 - ist das Team praktisch hirntot.*

  Und doch, wenn das Team es wünscht, sollen sie es so ausprobieren - wie immer.

Sprint Planning - Teil 1

Der erste Teil des Sprint Planning (SP1) ist tatsächlich ein detaillierter *Anforderungs-Workshop*. Der Product Owner stellt eine Reihe von angedachten Features vor, und

das Team stellt Fragen, um die Anforderungen in ausreichendem Detail zu verstehen, damit sie das Versprechen abgeben können, das jeweilige Feature im kommenden Sprint zu liefern. Das Team alleine entscheidet, was es in dem Sprint liefern kann, in Abhängigkeit von der Sprintdauer, der Größe und aktuellen Fertigkeiten der Mitglieder, seiner Definition von *DONE*, bekannten Abwesenheitszeiten und Maßnahmen, die sie in der zuletzt abgehaltenen Retrospektive beschlossen hatten.

Der Product Owner muss während dieses Meetings anwesend sein, um das Team in die richtige Richtung zu führen und Rückfragen zu beantworten – von denen es viele geben wird. Der ScrumMaster muss sicherstellen, dass beliebige andere vom Team zum Verständnis der Anforderungen benötigte Stakeholder anwesend oder rufbereit sind.

Eventuelle neue Backlog-Einträge, die für den laufenden Sprint aufgenommen und noch nicht geschätzt wurden, werden in diesem Meeting sofort bemessen. Das soll aber keine Ausrede für das Versäumnis sein, das Backlog zu pflegen – siehe unten!

Am Ende des SP1 verspricht das Team dem Product Owner die Lieferung einer selbsteingeschätzten Menge an lauffähigen und getesteten Funktionen. Ein erfahrendes Team kann die frühere Velocity als Referenz nehmen („Wetter von gestern"). Das nennt man auch die Velocity-basierte Planung. Meine Empfehlung an die meisten Teams ist die Commitment-basierte Planung. Die Backlogeinträge, die das Team zur Lieferung zugesagt hat, nennt man das *Selected Product Backlog.*

Sprint Planning - Teil 2

Wenn Teil 1 ein Anforderungs-Workshop ist, so ist Teil 2 des Sprint Planning (SP2) ein Design-Workshop. In dieser Sitzung arbeitet das Team gemeinsam daran, einen Grobentwurf der versprochenen Funktionalitäten zu erarbeiten. Ein Ergebnis dieser Sitzung ist das Sprint Backlog, oder die Liste der Aufgaben, die das Team gemeinsam erledigen muss, um aus den Einträgen im Selected Product Backlog lauffähige getestete Funktionalität zu machen. Diese Aufteilung von Aufgaben nennt man das Sprint Backlog, das meistens durch ein physisches Task Board dargestellt wird.

Während des SP2 kann das Team weitergehende Fragen zu den Anforderungen haben. Der ScrumMaster muss sicherstellen, dass der Product Owner oder andere benötigte Stakeholder für die Beantwortung bereit stehen.

Das Design ist emergent, wie alles andere in der Agilität auch. Dazu ist das Meeting auch noch zeitlich beschränkt. Daher ist es normal, dass das Team einen Entwurf nicht vollständig in dieser Sitzung fertig stellen kann. Das ist kein Zeichen dafür, dass etwas falsch läuft. Das Team nimmt sich einfach eine Haftnotiz und einen Stift – und notiert weitere Aufgaben, wann immer sie im Sprint aufkommen.

> ✺ *Du merkst, dass das SP2 funktioniert, wenn das Team lautstark diskutierend um das Whiteboard versammelt ist, und vielleicht sogar noch über die „beste" oder „richtige" Implementierung einer Funktionalität streitet.*

Daily Scrum Meeting

Das tägliche Scrum Meeting ist einer der drei primären *Inspektions- und Anpassungspunkte* in Scrum. Das Team trifft sich, um seine Arbeit zu kommunizieren und synchronisieren. Für die Zusammenarbeit des Teams ist dieses Meeting essentiell, um kontinuierlichen Fortschritt zu gewährleisten und Blockaden zu vermeiden. Zudem beurteilt das Team fortlaufend seinen eigenen Fortschritt zur Erreichung seines Sprintziels.

Das Daily Scrum Meeting dient nicht dazu, den Fortschritt an den ScrumMaster, Product Owner oder irgendjemanden sonst zu berichten. Der Product Owner darf teilnehmen, wenn er sich dementsprechend benimmt – also nur spricht, wenn er angesprochen worden ist. Der ScrumMaster stellt sicher – mit all seinen Fähigkeiten – dass sich jedes Teammitglied etwas Arbeit für die nächsten 24 Stunden vornimmt, dass die Arbeit ausschliesslich zur Lieferung des nächsten Backlog-Eintrags beiträgt, und dass irgendwelche Hindernisse so schnell wie möglich aus dem Weg geräumt werden. Der ScrumMaster sorgt auch dafür, dass dieses Meeting 15 Minuten nicht überschreitet – was überraschenderweise eine ausreichende Zeitspanne ist.

Jason Yip [Yip 2006] bietet einen nützlichen Leitfaden, um ScrumMastern bei der guten Durchführung dieses Meetings zu unterstützen.

Sprint Review

Der Sprint Review wird manchmal fälschlicherweise als Sprint Demo bezeichnet. Während er eine Demonstration der neuen Features enthält, die das Team im Sprint fertiggestellt hat, ist sein wesentlicher Sinn die Inspektion der Lieferung des Teams und die Sammlung von Feedback der Teilnehmer, um den Plan für den folgenden Sprint zu adaptieren. Daher ist er wesentlich mehr als eine Vorführung.

Der Fokus des Sprint Reviews ist das Produkt, welches das Team erstellt.

Wenn ich gefragt werde, wer zum Sprint Review eingeladen werden sollte, entgegne ich: „die ganze Welt“. Meine Intention ist hierbei, dem ScrumMaster und der gesamten Organisation zu vermitteln, dass die direkte Aufmerksamkeit und das Feedback eines breiten Teils der Organisation von entscheidender Bedeutung für die Maximierung des Werts ist, den das Team in zukünftigen Sprints liefern wird. Sprint Reviews haben viele mögliche Ergebnisse, inklusive der Beendigung des Projekts. Meistens wird das Team dazu ermächtigt, für einen weiteren Sprint mit der Arbeit fortzufahren, und eine Zielsetzung für diesen nächsten Sprint wird vereinbart.

Sprint Retrospektive

Die Sprint Retrospektive ist das abschliessende Meeting eines Sprints. Sie folgt direkt im Anschluss an den Sprint Review und wird niemals ausgelassen.

Während der Sprint Review auf das Produkt fokussiert ist, ist die Retrospektive auf den Prozess fokussiert - die Weise, in der das Scrum Team zusammen arbeitet,

einschliesslich der technischen Fertigkeiten und der verwendeten Entwicklungspraktiken und -Werkzeuge.

Und während der Sprint Review der ganzen Welt offen steht, ist die Sprint Retrospektive auf die Mitglieder des Scrum Teams beschränkt - den Product Owner, die Entwicklungsteam-Mitglieder und den ScrumMaster. Außenstehende, einschließlich Manager aus jeder Ebene der Organisation, sind streng ausgeschlossen, wenn sie nicht explizit vom Team eingeladen wurden.

Diese Regel muss in dem Kontext des Ziels der Retrospektive gesehen werden, auf tiefer Ebene zu verstehen, wie das Team zusammenarbeitet und performt, und Maßnahmen für die Verbesserung anzugehen. Das erfordert häufig intensive Selbstbetrachtung und Austausch, die wiederum eine sichere Umgebung erfordern. Norman Kerth's Oberste Direktive für Retrospektiven unterstreicht das: *„Was auch immer wir entdecken, wir verständigen uns darauf, und sind fest davon überzeugt, dass alle den besten Job gemacht haben, den sie konnten, mit ihren Kenntnissen zu dem Zeitpunkt, ihren Fertigkeiten und Fähigkeiten, den verfügbaren Mitteln und in der konkreten Situation.“* [Kerth 2001]

Boris Gloger [Gloger 2008] zeigt ein einfaches Muster, genannt *Heartbeat Retrospectives*, für neue Teams auf, um die Durchführung wertvoller Retrospektiven zu lernen. Esther Derby und Diana Larsen [Derby und Larsen 2006] stellen hilfreiche Aktivitäten für Moderatoren von Scrum Retrospektiven vor.

Estimation Meeting

Dieses Meeting wird in einem Teil der Scrum Literatur nicht erwähnt [oder taucht unter dem Namen *Release Planning* oder *Backlog Grooming* auf, Anm. d. Übs.]. Es ist jedoch essentiell, wenn du einen kontinuierlichen Strom der wichtigsten fertigen Features von deinen Teams erreichen möchtest.

In jedem Sprint setzt der Product Owner ein oder zwei Meetings an, in denen sich das Scrum Team und – wenn erforderlich – andere Stakeholder treffen, um neue Backlogeinträge einzuschätzen, oder große Backlogeinträge neu zu beurteilen, die für die Bearbeitung in den kommenden Sprints in kleinere aufgeteilt werden müssen.

> *Teams sollten 5-10% ihrer Zeit im Sprint der Vorbereitung der folgenden ein bis zwei Sprints widmen. Das Estimation Meeting wie oben beschrieben, ist ein Beispiel dafür. Andere Beispiele sind Story Writing und Release Planning-Workshops. Das ist wichtig, um einen Stop-Start-Effekt beim Übergang zwischen den Sprints zu vermeiden. Die andere Implikation ist natürlich, das Teams 90-95% ihrer Zeit für die Arbeit am laufenden Sprint verwenden sollten.*

Scrum Artefakte

Scrum fordert nur vier Artefakte [Prozessdokumente, Anm. d. Übs.] zwingend:

- Product Backlog
- Sprint Backlog
- Burndown Chart
- Impediment Backlog

Scrum schweigt sich absichtlich über alle weitere Dokumentation und Artefakte aus. Das führt manchmal zu dem Missverständnis, dass agile Teams keine Dokumentation erstellen müssten. Ich coache Teams dahin, nur die Artefakte zu produzieren, die in der Zukunft wirklichen Wert für sie selbst und andere darstellen. Das grenzt wertlose Arbeit und die unnötige Rodung von Wäldern aus!

Product Backlog

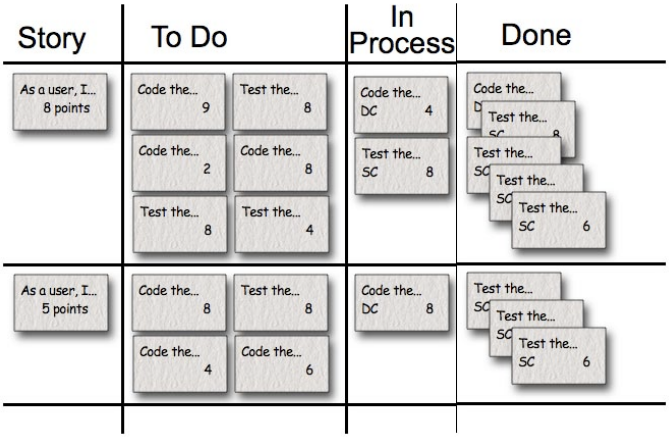
Das Product Backlog ist einfach eine Liste von Arbeitsgegenständen [oder besser Anforderungen an das Produkt, Anm. d. Übs.], die über die Zeit erledigt werden müssen. Einträge können dem Backlog von jedem hinzugefügt werden, aber nur der Product Owner hat die Berechtigung zu entscheiden, in welcher Reihenfolge sie vom Team bearbeitet werden sollen. Natürlich wird ein guter Product Owner darüber mit Stakeholdern und dem Team verhandeln.

Anforderungen sind emergent, das bedeutet, dass wir weder jedes Detail wissen, noch wissen können, welches wir in einem Produkt haben wollen. Daher ist das Product Backlog ein lebendes Dokument, welches konstante Pflege benötigt, um es aktuell und nützlich zu halten. Viele neue Einträge werden im Lauf der Zeit hinzugefügt; einige Einträge können entfernt werden, wenn sich herausgestellt hat, dass ein gewünschtes Feature nicht mehr benötig ist. Darüber hinaus müssen die Einträge bemessen werden, um das wahrscheinliche Verhältnis zwischen Wert, Zeit und Kosten zu bestimmen. Und nicht zuletzt wird sich die Reihenfolge der Einträge im Backlog ändern, wenn der relative Wert zwischen ihnen heute anders gesehen wird als gestern.

In fast allen Fällen ist es ausreichend, und generell vorzuziehen, das Backlog als eine Sammlung von Stories auf physischen 150 x 100 mm (6“ x 4“) Karten zu erstellen und verwalten. Ron Jeffries [Jeffries 2005] prägte das alliterative Tripel *Card, Confirmation, Conversation* (die 3Cs), um die Arbeit mit Stories zu beschreiben. Mike Cohns Buch über User Stories [Cohn 2004] wird dir alles erzählen, was du darüber wissen musst.

Sprint Backlog

Die meisten Teams kennen das Sprint Backlog als ein Task Board, welches die physische Repräsentation der Liste der Arbeit ist, die sie sich für den laufenden Sprint vorgenommen haben. Das Task Board ist ein Anwendungsbeispiel eines Kanban – ein japanischer Ausdruck, der Zeichen und sichtbares Signal bedeutet. Es sagt dem gesamten Team und allen anderen, welche Arbeit für den Sprint geplant wurde, und in welchem Status die Arbeit ist.



Sprint Backlog oder Task Board

Übernommen von http://epf.eclipse.org

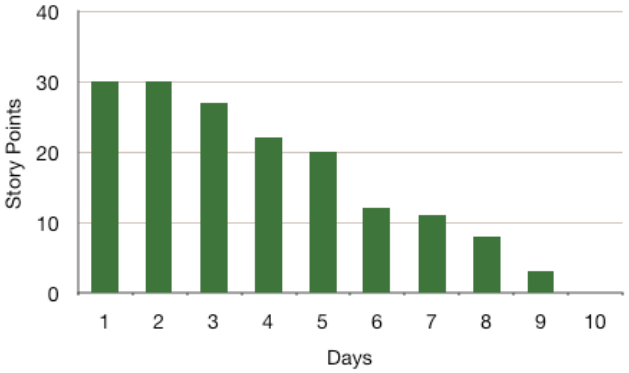
Sprint Burndown Chart

Das Sprint Burndown Chart wurde konzipiert, um dem Team bei der Überwachung seines Fortschritts zu helfen; und bei der Frage, ob es sein Lieferversprechen am Ende des Sprints einhalten kann. Das klassische Format verlangt einem Team ab, die Dauer jeder Arbeit in Stunden zu schätzen, und auf täglicher Basis die gesamte restliche Arbeitszeit für alle unvollendeten Arbeiten in das Diagramm einzutragen.

Ich coache Teams dahin, ihren Sprint in Story Points zu verfolgen. Die Idee dahinter ist:

- Die Schätzung neuer Arbeiten und die Neuschätzung von laufenden Arbeiten erfordert Aufwand.
- Die Schätzung von Arbeiten ist nicht akkurat
- Schätzungen in Zeiteinheiten wirft uns die alten Arbeitsweisen zurück
- Die Fertigstellung von Arbeiten liefert keinen Wert, nur abgeschlossene Stories (Features) liefern Wert.

Also ist in meiner Coaching-Welt der Sprint Burndown genau so aufgebaut, wie das Product oder Release Burndown, nur dass die Skala der x-Achse Tage statt Sprints umfasst.



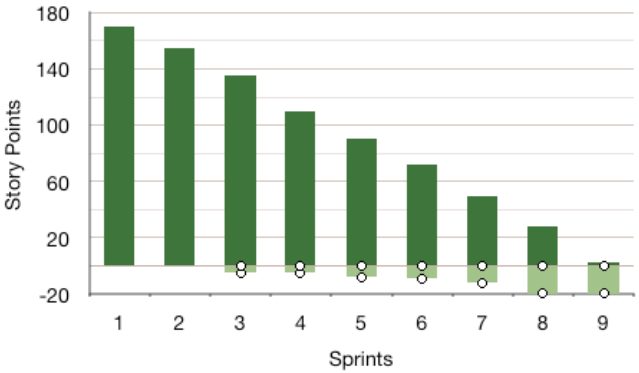
Sprint Burndown Chart

Product oder Release Burndown chart

Das Product Burndown Chart – manchmal bekannt als Release Burndown Chart – misst die Rate der Lieferung eines Stroms von lauffähigen, getesteten Features über die Zeit. Diese Rate ist als die Velocity des Teams bekannt. Weil Features in ihrer Komplexität – und damit Aufwand und Laufzeit – variieren, nutzen wir eine Skala, um ihre Größe zu vergleichen. Die am weitesten verbreitete Skala ist als Story Points bekannt. Sobald ein Team eine Reihe von Sprints zusammen gearbeitet hat, hat sich in der Regel seine Velocity in einem definierbaren Rahmen eingependelt. Product Owner nutzen diese Velocity zur Vorhersage der zukünftigen Lieferrate des Teams, was zu glaubwürdigen Releaseplänen führt.

Ich bringe Teams bei, eine alternative Form des Product Burndown Charts zu verwenden, die es dem Product Owner gleichzeitig ermöglicht, Änderungen am Product Backlog nachzuverfolgen. Das ist bei der dynamischen Natur dieser Liste essentiell.

Mit diesem Diagramm können Product Owner den Fortschritt berichten, Releasedaten ermitteln und den Release-Umfang vorhersagen.



Product Burndown Chart

Impediment Backlog

Das Impediment Backlog ist einfach die aktuelle Liste der Dinge, die das Team davon abhalten, Fortschritte zu erzielen oder sich zu verbessern. Dabei handelt es sich um Dinge, die der ScrumMaster aus dem Weg räumen muss, in ihrer niemals endenden Mission, dem Team so gut wie möglich zu helfen. Impediments [Hindernisse] können die Reparatur der Kaffeemaschine umfassen, aber auch die Neubesetzung des Geschäftsführerpostens! Ein guter ScrumMaster versucht, Impediments innerhalb von 24 Stunden nach ihrem Auftauchen zu beseitigen. (OK, vielleicht nicht den Geschäftsführer.)

Scrum starten

Ken Schwaber [Schwaber 2007] sagt, dass vor dem Start von Scrum keine besonderen Maßnahmen vonnöten seien. Ich interpretiere das so, dass keine Modifikation des Prozesses erforderlich ist, um starten zu können. Dennoch gibt es wenig Hinweise in der Literatur, wie man in Gang kommen kann – und wir alle bemühten uns, unser erstes Scrum Team ohne Hilfe von außen zum Laufen zu bringen.

Das Beste, was du tun kannst, ist einen erfahrenen Coach zu beauftragen. Falls das nicht gehen sollte, kannst du ein Muster ausprobieren, dass für mich mit Dutzenden von Teams funktioniert hat.

Offensichtlich (hoffe ich) brauchst du ein Scrum Team. Das heißt, einen Product Owner, einen ScrumMaster und fünf bis neun Teammitglieder. Folge dann dieser Reihenfolge von Schritten, die auf den nächsten Seiten erläutert werden.

1. Trainiere das Scrum Team in den Grundlagen von Scrum
2. Erarbeite die Vision
3. Schreibe User Stories, um das Product Backlog aufzubauen
4. Ordne das Backlog nach Geschäftswert
5. Bemesse die Backlogeinträge
6. Sortiere das Backlog wenn nötig nach weiteren Faktoren um
7. Erstelle einen initialen Releaseplan
8. Plane den ersten Sprint
9. Starte den Sprint!

- *Ich beginne bei neuen Teams mit einem ganztägigen Training über die Grundlagen von Agile und Scrum. Das reicht aus, um ein Scrum Team ins Laufen zu bringen, das einen erfahrenen ScrumMaster/Coach als Unterstützung in den ersten Sprints zur Verfügung hat.*
- *Die Schritte 2 bis 7 können bequem innerhalb eines zweitägigen Produkt-Workshops mit dem gesamten Scrum Team durchgeführt werden. Die besten Workshops werden durch Stakeholder unterstützt, wie Enterprise-Architekten, Manager und Fachbereichs-Vertreter.*

Training

Ich verwende eine Reihe von Gruppenübungen und Spielen in meinem Team Training, um die Prinzipien zu illustrieren. Ich behandle alle oder die meisten der folgenden Themen:

- Die Macht der Selbstorganisation
- Empirische versus definierte Prozesse
- Der Wert des gemeinsamen Arbeitsplatzes
- Die Bedeutung von Vertrauen
- Agile Prinzipien (Agiles Manifest)
- Der Scrum Flow (Zyklus der Meetings)
- Rollen und Verantwortungen (drei Scrum Rollen und mehr)
- Die Verwendung von User Stories für Anforderungen
- Agiles Schätzen mit Planning Poker
- Konzepte von Größe und Velocity
- Done!
- Verwenden eines Taskboards
- Überwachung des Fortschritts (Burndown Charts)
- Simulation eines ganzen Sprints

Erarbeitung der Vision

Katzenberg und Smith [2002] haben bestätigt, dass das Vorhandensein von klaren Ziele essentiell bei der Schaffung von hochleistungsfähigen Teams sind.

Der Product Owner teilt üblicherweise seine oder ihre Vision für das Produkt mit. Eine von mir genutzte Methode ist, jedes Teammitglied dazu zu bringen, seine oder ihre eigene Version der Vision aufzuschreiben. Dann arbeiten die Mitglieder in Paaren daran, eine einzige gemeinsame Vision aus ihren ursprünglichen Ergebnissen zu formulieren. Der Prozess der gemeinsamen Bearbeitung setzt sich fort, bis das gesamte Team daran arbeitet, eine einzige Aussage zu formulieren.

Diese Übung nutzt die Macht der Gruppe und resultiert in einer größeren Identifikation für die entstandene Vision. Das Team präsentiert die Vision deutlich in ihrem Arbeitsbereich. Die Visioning-Übung kann eine bis drei Stunden dauern.

Erstellung des Product Backlogs

Die nächste Stufe ist die Abhaltung eines Story-Workshops. Es ist von Vorteil, an dieser Stelle Stakeholder aus den Geschäftsbereichen mit einzubeziehen. Sicherlich ist das gesamte Scrum Team involviert. Natürlich moderiert der ScrumMaster (oder Coach).

Das Verfassen guter User Stories ist einfach eine Frage der Übung. Das Pareto-Prinzip (80/20) findet hier – wie immer – Anwendung.

Mike Cohn [Cohn 2004] hat einen exzellenten Leitfaden zur Erstellung guter User Stories geschrieben. Ich empfehle jedem Product Owner, sein persönliches Exemplar stets griffbereit zu haben.

> Template: As a **role / persona**, I want **function** so that **reason**.
> Example: As a **programmer**, I want **coffee** so that I can **stay awake**.

Bill Wake schlug das hilfreiche INVEST-Akronym für die Eigenschaften einer guten User Story vor [Wake 2003].

> **Independent**—ideally can be implemented in any order
> **Negotiable**—and negotiated
> **Valuable**—to the customer
> **Estimatable**—enough to rank and schedule it
> **Small**—and with short descriptions
> **Testable**—I could write a test for it

Das Backlog muss ausreichend viele Einträge enthalten, damit das Team den ersten Sprint planen kann. Für gewöhnlich enthält das Backlog alle Einträge, die das nächste geplante (oder erhoffte) Produkt Release ausmachen.

Als Leitlinie, sollte das Backlog am Ende des ersten Tages zu 80% vollständig (aber nicht sortiert oder geschätzt) sein. Im ersten Teil des zweiten Tages kann man die restlichen 20% ergänzen und offene Fragen klären. Die Pause über Nacht ist eine sehr nützliche „Hiatus“ (Kunstpause), die frische Gedanken über die Arbeit hervorbringen kann.

Reihenfolge des Product Backlogs

Das Backlog ist nun nach dem Geschäftswert sortiert. Das erscheint leichter gesagt als gut getan. Mike Cohn [Cohn 2006] beschreibt zwei Methoden, um Begehrlichkeiten zu priorisieren: das Kano-Modell der Kundenzufriedenheit und Wiegers Ansatz der relativen Gewichtung. Was mir an beiden gefällt ist, dass sie nicht nur den Nutzen des Features betrachten, sondern auch den Schaden, wenn das Feature nicht vorhanden ist. Das ist besonders hilfreich, wenn das Backlog technische Einträge enthält, deren Geschäftswert sich nicht unmittelbar erschließt.

Als Mindestanforderung ist die subjektive Einschätzung des Product Owners über den Wert eines Features im Vergleich zum anderen gut genug. Besser ist eine eher quantitative Beurteilung mit einer dem Planning Poker vergleichbaren Technik.

Wie es auch immer angestellt wird, es ist eine Kernverantwortung des Product Owners, das Backlog in eine Reihenfolge zu bringen.

Bemessung des Backlogs

Die Projektplanung hatte immer die Schätzung als kritischen Faktor. Projektleiter wie ich haben Wochen damit verbracht, komplexe Modelle mit historischen Daten zu kalibrieren.

Die nackte Realität ist, dass die Besten dieser Techniken keine besseren Ergebnisse liefern als wesentlich einfachere und schnellere Techniken wie Planning Poker und Affinity Estimating. Planning Poker funktioniert zum Teil, weil es ein solides theoretisches Fundament hat, aber hauptsächlich, weil die Leute, die schätzen, auch diejenigen sind, welche die Arbeit erledigen werden. Wer hätte das gedacht?

Planning Poker ist schnell. Ein geübtes Team kann in einer durchschnittlichen Rate von 3 Minuten pro Backlogeintrag schätzen. Planning Poker ist akkurat. Schätzungen auf der Basis von Planning Poker sind so gut wie die besten traditionellen Methoden. Und Planning Poker macht Spaß. Es nimmt einem den Schmerz, der normalerweise mit diesem Thema verbunden ist. Kommerzielle Planning Poker-Karten sind bei diversen Quellen zum Preis von ca. $10 (inklusive Versand) für einen Satz von 4 Paketen erhältlich. Du kannst deine eigenen Karten auf Blankokarton für fast umsonst drucken.

Affinity Estimating geht noch schneller als Planning Poker. Es ist großartig, um in die Gänge zu kommen, wenn wir ein gesamtes Backlog schätzen müssen – und die Zeit wichtiger als die Korrektheit und der Informationsaustausch sind.

Trotzdem müssen wir immer noch ein paar grundlegende Konzepte über das agile Schätzen verstehen:

- Wir bemessen Dinge relativ zueinander. Warum? Weil wir das als Menschen natürlicher finden, und die Ergebnisse zuverlässiger sind. So ist es leicht, sich darauf zu einigen, dass „diese Story die doppelte Komplexität wie jene“ hat – auch wenn wir nicht wissen, wie lange es dauern wird, jede zu implementieren.
- Wir bemessen Dinge in Komplexitäts-Einheiten statt in Zeit. Warum? Weil es uns ermöglicht, die Rate in der ein Team arbeitet von der Größe oder Komplexität der Arbeit zu trennen. Das bewahrt uns davor, unsere Schätzungen anhand dessen zu ändern, der die Arbeit macht, oder wenn die Fertigkeiten und Kapazität des Teams sich mit der Zeit ändert. Wir verwenden Story Points als Einheit.
- Wir benutzen eine nicht-lineare Bemessungsskala, weil die Differenz zwischen einer ‚1’ und einer ‚2’ offensichtlich mehr aussagt – relativ betrachtet – als die zwischen einer ‚20’ und ‚21’. Meine Vorliebe sind die Pseudo-Fibonacci-Zahlen: 1, 2, 3, 5, 8, 13, 20, 40 und 100. Und ich definiere 1-8 oder vielleicht 13 als den Größenbereich von Features, die ein Team innerhalb eines Sprints liefern kann. Die höheren Zahlen sind für große Stories (Epics) reserviert, die in kleinere Stories heruntergebrochen werden müssen, bevor sie in einen Sprint aufgenommen werden.
- Wir stellen den Bezug zwischen unseren Größenbestimmungen zur Dauer durch die Verwendung der Velocity her, die Rate, in der ein Team lauffähige, getestete Features an den Product Owner liefern kann. Wir sagen, ein Team hat eine Velocity von 25, wenn es am Ende jedes Sprints „fertige Stories“

abliefern kann, deren aufsummierte Größen im Durchschnitt 25 Punkte betragen.

Umsortierung des Backlogs

Nach der Bemessung der Backlogeinträge sollten wir eventuell einige davon neu sortieren. Faktoren, die hierbei neben dem Geschäftswert eine Rolle spielen, sind unter anderem:

- Größe: Wir werden natürlich eine einfache (kleine) Story vor einer komplexen (großen) Story umsetzen, wenn beide einen ähnlichen Geschäftswert haben.
- Lernen: Wir können eine Story zur Implementierung vorziehen, wenn das dem Team dabei etwas über die Geschäftsdomäne oder eine neue Technologie lernen kann.
- Risiko: Wir können eine Story zu einem frühen Zeitpunkt implementieren, um ein identifiziertes Risiko abzumildern. Offensichtliche Beispiele sind Einträge, die Performance- und Skalierungskriterien implementieren helfen.

Dabei dürfen wir nicht vergessen, dass der Product Owner das letzte Wort bezüglich der Reihenfolge der Einträge im Backlog hat.

Release Planning

Wenn wir einmal das Backlog sortiert und bemessen haben, ist der nächste Schritt die Erstellung des initialen Releaseplans. Dafür benötigen wir eine Einschätzung der Velocity unseres Teams. Nun haben wir noch nicht mit der Arbeit begonnen, weswegen wir eine einfache Technik verwenden, die als *Commitment-basierte Planung* bekannt ist.

Zuerst müssen wir natürlich die Teamgröße im Sprint wissen – ob irgendein Mitglied im Urlaub, auf Schulung ist, etc. Und wir müssen die Sprintlänge bestimmen – ich empfehle für gewöhnlich zwei Wochen für ein neues Team. Und wir müssen eine Definition of DONE für das Team aufstellen – was heißt es, wenn das Team sagt, dass es eine Story abgeschlossen hat.

Der ScrumMaster wählt nun den ersten Eintrag von der Spitze des Backlogs und fragt das Team „könnt ihr diesen Eintrag im Sprint fertig stellen?“. Er fährt auf diese Weise fort bis das Team sich nicht mehr sicher fühlt, weitere Einträge dazu zu nehmen. Durch die Aufsummierung der Storypoint-Werte aller versprochenen Einträge ermittelt das Team seine initiale Velocity-Schätzung.

Dieser Velocity-Wert wird verwendet, um die übrigen Backlogeinträge (zumindest die geschätzten) auf die folgenden Sprints aufzuteilen. Diese Liste von Sprints zugeordneten Einträgen ist unser initialer Releaseplan. Ist er akkurat? Vielleicht nicht, aber er ist wahrscheinlich glaubwürdiger als alles, was ein Projektleiter bisher zu einem solchen frühen Zeitpunkt aufstellen konnte. Und mit dem Beginn der Arbeit und dem Abschluss von mehr und mehr Sprints, werden wir anfangen, unsere tatsächliche Velocity aufzuzeichnen und sie als Vorhersage für die zukünftigen Ergebnisse zu

verwenden. Daher ist der Releaseplan ein lebendes Dokument, das immer mehr an Glaubwürdigkeit gewinnt, je weiter wir voranschreiten.

> ✺ *Lass den Produkt-Workshop nicht zu lange dauern. Es ist innerhalb von zwei Tagen zumindest möglich. mit jedem Team und jedem Produkt genügend Backlog-Einträge für die ersten paar Sprints vorzubereiten.*

Kollokation und Teamräume

Alistair Cockburn [Cockburn 2007] definiert Softwareentwicklung als ein *kooperatives Spiel von Interaktion und Kommunikation*. Das Agile Manifest sagt aus, dass Entwickler und Geschäftsvertreter auf täglicher Basis zusammen arbeiten müssen und dass die Kommunikation von Angesicht zu Angesicht die beste Art ist, Informationen weiterzugeben.

Um effektiv zusammen arbeiten zu können, gibt es keinen einzelnen Faktor, der mehr Gewinn bringt, als Kollokation. Eine Studie über kolozierte Softwareentwicklungsteams [Teasley, Covi, Krishnan & Olson, 2000] hat aufgezeigt, dass sie doppelt so produktiv wie nicht-kolozierte Teams sind.

Die Definition, die ich für Kollokation verwende ist, dass Teammitglieder nicht weiter als 6 m (20') von ihrem am weitesten entfernten Teammitglied sitzen. Daraus folgt, dass die Anzahl der Menschen begrenzt ist, die in einen solchen Raum passen. Praktisch ist diese Anzahl von 8 bis 10, was glücklicherweise mit der Maximalgröße eines Scrum Teams übereinstimmt.

Des Weiteren muss jedes Teammitglied in der Lage sein, alle anderen Teammitglieder ohne größeren Aufwand als durch eine Schulterdrehung oder Stuhldrehung sehen zu können. Das bedeutet: keine Trennwände zwischen Schreibtischen, lebe wohl, Dilbertville!

Außer der Gewohnheit gibt es meist keinen erkennbaren Grund, warum Organisationen (beachte: nicht Teams) sich weigern, die Arbeitsumgebung zu verändern, um eine effektive Kollokation zu ermöglichen.

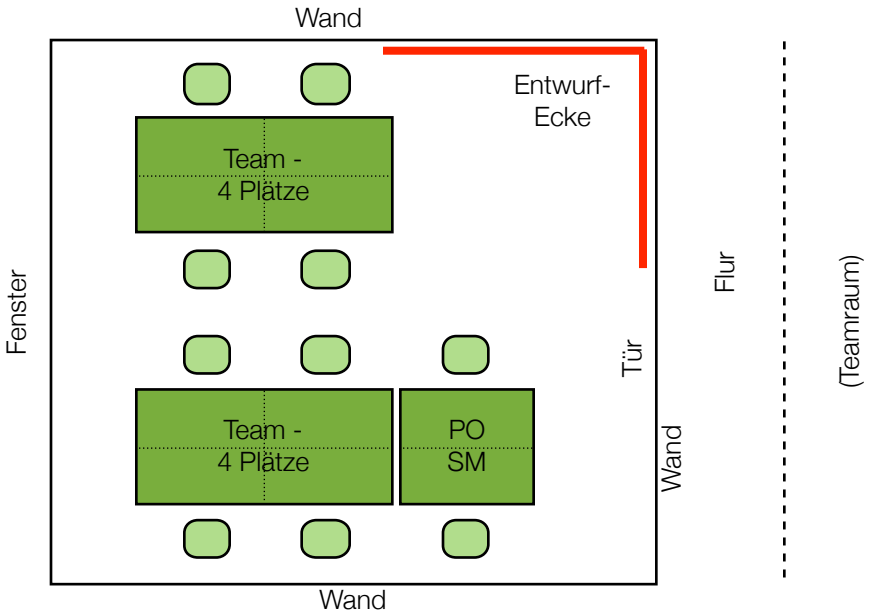
Und das im Widerspruch zu der Tatsache, dass kollozierte Teamräume im Durchschnitt nicht mehr Grundfläche benötigen, als hergebrachte Anordnungen.

Das Management wird dich wegen des Aufbaus von Trockenbauwänden herausfordern, da sie aus ihrer Sicht Flexibilität behindern. Das ist einfach nicht wahr – Flexibilität ist eine Anforderung der Organisationsstruktur, nicht der Teamräume!

Meiner Erfahrung nach haben 90% aller Scrum Teams in Organisationen, die mit agilen Vorgehensweisen gerade begonnen haben, ein Jahr lang Schwierigkeiten mit ihrer miserablen Arbeitsumgebung, bis das Management davon überzeugt ist, die notwendigen Änderungen durchzuführen. Sobald die Änderungen erfolgt sind, stimmen alle zu, dass die Verbesserung sofort eingetroffen ist – und deutlich spürbar ist. Warum ist das so?

In der Hoffnung, dass dieser absurde Widerstand sich auflösen wird, stelle ich eine simple Anordnung eines kolozierten Teamraums auf der nächsten Seite vor. Es würde den Rahmen dieses kleinen Leitfadens sprengen, alle Überlegungen bezüglich dieses Designs zu erörtern – dazu müsstest du mich schon beauftragen!

(nächste Teamraum)



(nächste Teamraum)

- *Einige Anordnungsmerkmale für den Teamraum*
  - *Schreibtische müssen rechteckig sein, um Pairing zu unterstützen.*
  - *1,5–1,8 m x 0,8 m ist eine gute Schreibtischgröße*
  - *1,2 m Platz zwischen Schreibtisch und Wand*
  - *Design-Ecke 3 x 3 m*
  - *Typische Raumgröße: 7 x 8 m = 50 bis 60 m²*
  - *Der ScrumMaster kann das ganze Team sehen*
  - *Der Product Owner hat einen - nicht permanenten - flexiblen Arbeitsplatz im Raum*
  - *Trockenbauwände mit internen Fenstern bilden Schallbarrieren und genug Platz für Informations-Radiatoren, ohne dabei das Tageslicht einzuschränken*

- *Teamräume vermindern den Bedarf an Meetingräumen erheblich. Teams können Sprint Plannings, Daily Meetings und Reviews in ihrem Raum abhalten.*
- *Sei vorgewarnt, dass dein Architekt oder Bürodesigner dir keine große Hilfe sein wird. Sie sind nicht im Design von Teamräumen ausgebildet.*
- *Die Kosten der Anpassungen (Trockenbauwände und Möbel) amortisieren sich in einer Woche bis zu einem Monat. Klingt unglaublich, oder?*

Metriken

Es macht Sinn, von jedem Manager zu erwarten, dass er – innerhalb eines vernünftigen Zeitraums – in der Lage sein will, die Ergebnisse des Wechsels zu agilen Vorgehensweisen eines Teams oder einer Organisation messen zu können.

Zum Glück gibt es Metriken, die einfach und schnell zu implementieren sind. Auf der Basis meiner eigenen Forschung, und der anderer agiler Praktiker, habe ich eine Reihe von Metriken zusammengestellt, die euer Team einführen kann und sollte [Hundermark 2009]. Sobald das Team die Grundlagen des Scrum Frameworks verstanden, und eine Vorstellung über die eigene Velocity erlangt hat – was normalerweise am Ende des dritten Sprints der Fall sein sollte – macht es möglicherweise Sinn, Messungen aufzunehmen. Allerdings lassen sich Kunden- und Teamuntersuchungen bereits vor dem Start von Scrum durchführen!

Ohne die Erläuterung hier zu vertiefen, liste ich hier die Metriken, die ich für eine initiale Implementierung empfehle::

- Kunden- und Teamumfragen
- Velocity Diagramm
- Burnup oder Burndown Diagramm
- Ausführung automatisierter Tests
- Technische Schulden
- In Bearbeitung befindliche Aufgaben
- Story-Durchlaufzeit

Und sobald du in der Lage bist, finanzielle Metriken für den Geschäftswert mit einzubeziehen, führe die folgenden Metriken mit ein:

- Kosten pro Sprint und Story Point
- Echte Wertschöpfung
- ROI oder NPV

Coaching

Was macht ein Coach?

> *'Scrum Coaching bedeutet ein Engagement mit einer oder mehreren Organisationen / Teams, währenddessen der Coach für die betroffenen Teams als Mentor oder Moderator zur Verbesserung des Verständnisses und der Anwendung von Scrum handelt, damit die Teams ihre aufgestellten Ziele erreichen können. Das Engagement beinhaltet die eins-zu-eins und Team Betreuung, die Einführung des Prozesses, Organisationsentwicklung, Beratung über die Einstellung, und die Interaktion mit allen Führungsebenenen innerhalb einer Organisation. Es kann auch die Mentorenschaft in mit der Effektivität von Scrum verknüpften Methoden, Prinzipien aus dem Agilen Manifest, Lean-Prinzipien oder Praktiken des Extreme Programming beinhalten.'*
>
> *Certified Scrum Coach Application, Scrum Alliance*

Der Coach bietet in den Kundenorganisationen Anleitung auf die folgenden Arten:

- Beratung zur Verbesserung und Beschleunigung des Selbst-Entdeckungs-Prozesses
- Moderation der Annahme, Implementierung und Erlernung von Scrum beim Kunden
- Agile Führung, auf der Basis eines Servant Leadership Stils
- Organisationsentwicklung, die die vorhandenen Fertigkeiten, Ressourcen und Kreativität des Klienten verbessert

Warum braucht meine Organisation Coaching?

Softwareentwicklung und andere von Wissensarbeitern durchgeführte Aktivitäten, basieren auf implizitem Wissen – im Gegensatz zu formellem oder expliziten Wissen. Das unterschwellige Wissen lässt sich schwierig von einer Person zur anderen übertragen, es wird hauptsächlich durch persönliche Erfahrung erworben. Ein Beispiel ist das Fahrradfahren.

Die ScrumMaster Rolle muss dem Rest des Scrum Teams und der Organisation die Prozess-Führung bieten. Schulungen, wie das Certified ScrumMaster Training, sind für den Erwerb des erforderlichen unterschwelligen Wissens nicht ausreichend, um agile Praktiken und das Scrum Framework zu meistern. Das Team und die Organisation brauchen die Fähigkeiten eines erfahrenen Praktikers, um sie durch den Irrgarten der Herausforderungen zu geleiten, die unvermeidbar auf ihrem Wandel zur Agilität auf sie einstürmen werden.

Fazit

Auf die Frage, was sie anders machen würden, wenn Sie ihren Übergang zur Agilität und Scrum noch einmal angehen würden, antworten die meisten Manager aus Unternehmen auf der ganzen Welt: mehr Coaching!'.

Referenzen

1. Cockburn, Alistair (2007). *Agile Software Development: The Cooperative Game (2nd edition)*. Addison Wesley
2. Cohn, Mike (2004). *User Stories Applied*. Addison Wesley.
3. Cohn, Mike (2006). *Agile Estimating and Planning*. Prentice Hall.
4. Gloger, Boris (2008). *Heartbeat Retrospectives*. http://borisgloger.com/2008/04/24/heart-beat-retrospectives-1-introduction/.
5. Highsmith, Jim, et al (2001). *Manifesto for Agile Software Development*. http://www.agilemanifesto.org/.
6. Hundermark, Peter (2009). *Measuring for Results*. http://www.scrumsense.com/coaching/measuring-for-results.
7. Jeffries, Ron (2001). *Card, Conversation, Confirmation*. http://www.xprogramming.com/xpmag/expCardConversationConfirmation.
8. Katzenberg, Jon R. And Smith, Douglas K. (2002). *The Wisdom of Teams*. Collins.
9. Nonaka, Ikujiro and Takeuchi, Hirotaka (1986). *The New, New Product Development Game*. Harvard Business Review, Jan-Feb 1986.
10. Sliger, Michele and Broderick, Stacia (2008). *The Software Project Manager's Bridge to Agility*. Addison Wesley.
11. Schwaber, Ken (2007). *The Enterprise and Scrum*. Microsoft Press.
12. Schwaber, Ken (2009). *Scrum Guide*. http://www.scrumalliance.com/resources].
13. Teasley, Covi, Krishnan, & Olson (2000). *How Does Radical Collocation Help a Team Succeed?* Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (pp. 339 - 346). New York: ACM.
14. Wake, William (2003). *INVEST in Good Stories, and SMART Tasks*. http://xp123.com/xplor/xp0308/index.shtml
15. Yip, Jason (2006). *It's Not Just Standing Up: Patterns of Daily Stand-up Meetings*. http://martinfowler.com/articles/itsNotJustStandingUp.html.