# THE SCRUM PRIMER

**Leitfaden zur Theorie und Anwendung von Scrum**

Version 2.0 deutsch


**Pete Deemer**
GoodAgile
www.goodagile.com

**Gabrielle Benefield**
Evolve
www.evolvebeyond.com

**Craig Larman**
www.craiglarman.com

**Bas Vodde**
Odd-e
www.odd-e.com


Ein Hinweis für den Leser: Es sind viele präzise Scrum-Beschreibungen online verfügbar. Dieser Leitfaden versucht, die nächste Detailstufe der Praktiken zu vermitteln. Er ist nicht als letzte Stufe einer Scrum-Ausbildung gedacht. Teams, die Scrum anwenden möchten, sollten sich mit Ken Schwaber's *Agiles Projektmanagement mit Scrum*, oder Mike Cohns *Succeeding with Agile* beschäftigen, und eines der vielen Schulungs- und Coaching-Angebote zu Scrum nutzen. Sie finden dazu alle Informationen auf scrumalliance.org. Unser Dank geht an Ken Schwaber, Dr. Jeff Sutherland und Mike Cohn für ihre großzügigen Beiträge.

Die aktuellste Version des Primers auf englisch befindet sich unter: http://www.infoq.com/minibooks/Scrum_Primer. Übersetzungen befinden sich unter http://www.scrumprimer.org/.

# Jenseits der traditionellen Entwicklung

Die traditionelle Entwicklung mit ihren auf eine Funktion beschränkten Gruppen, verzögerten oder schwachen Feedback-Schleifen, der vorgelagerten voraussagenden Planung und einem sequentiellen Ablauf von der Analyse bis zum Test ist in der heutigen unberechenbaren Welt wenig erfolgreich. Dieser Ansatz verzögert das Feedback, Lernen und potenzielles ROI (engl.: Return on Investment) Erträge durch des Fehlen von wirklich, lauffähiger Software bis zu einem späten Zeitpunkt. Das bewirkt einen Mangel an Transparenz und einen Mangel an der Fähigkeit sich zu verbessern, reduzierte Flexibilität und eine Erhöhung der geschäftlichen und technischen Risiken.

Eine Alternative - funktionsübergreifende Teams mit iterativer Entwicklung - gibt es schon seit Jahrzehnten, sie war aber nicht so verbreitet wie das traditionelle Modell.

Scrum bündelt bewährte Konzepte zur Produktentwicklung in ein einfaches Framework, unter anderem: echte Teams, funktionsübergreifende Teams, sich selbst managende Teams, kurze iterative zyklische Feedback-Schleifen, und Minderung der Kosten für Änderungen. Diese Konzepte verbessern die Agilität und das Feedback, ermöglichen früheres ROI und reduzieren das Risiko.

# Überblick

Scrum ist ein Entwicklungs-Framework, in dem funktionsübergreifende Teams Produkte oder Projekte auf iterative, inkrementelle Weise entwickeln. Es strukturiert die Entwicklung in Arbeitszyklen, die **Sprints** genannt werden. Diese Iterationen dauern jeweils nicht mehr als vier Wochen (am weitesten verbreitet sind zwei Wochen) und reihen sich ohne Pause aneinander. Die Sprints sind *Time-Boxen* - sie enden an einem bestimmten Datum, ob die Arbeit abgeschlossen worden ist oder nicht, und sie werden *niemals verlängert*. Normalerweise wählen Scrum Teams eine Sprintlänge und nutzen diese für alle ihre Sprints, bis sie sich verbessern und einen kürzen Zyklus nutzen können. Am Anfang jedes Sprints wählt sich ein *funktionsübergreifendes* Team (von ca. sieben Personen) **Einträge** (Kundenanforderungen) aus einer priorisierten Liste. Das Team einigt sich auf ein gemeinsames Ziel, von dem es glaubt, dass es dieses bis zum Ende des Sprints liefern kann; etwas Konkretes, das wirklich „done“ (fertig) sein wird. Während des Sprints dürfen keine neuen Einträge aufgenommen werden; Scrum nimmt Änderungen zum *nächsten* Sprint an, aber der aktuelle Sprint soll sich auf ein kleines, klares, relativ stabiles Ziel fokussieren. Das Team versammelt sich täglich kurz, um seinen Fortschritt zu inspizieren, und die nächsten Schritte zur Fertigstellung der restlichen Arbeiten anzupassen. Am Ende des Sprints führt das Team einen Review des Sprints mit Stakeholdern durch, und demonstriert,

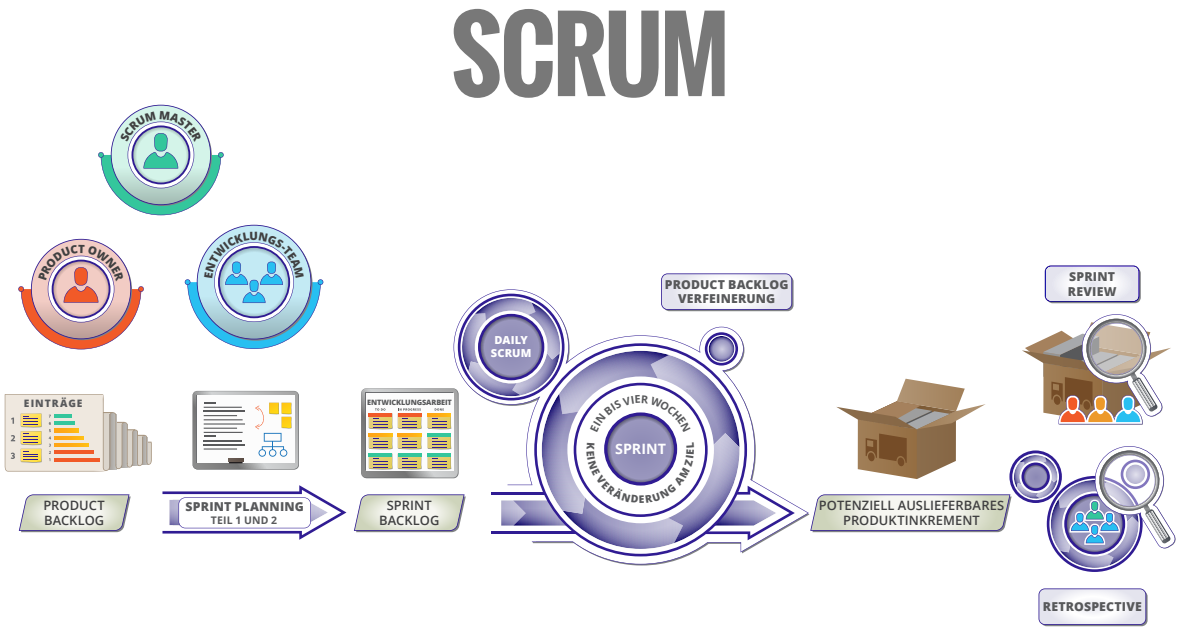


*Abbildung 1. Scrum Überblick*

was es gebaut hat. Die Mitarbeiter erhalten Feedback, das in den nächsten Sprint einfließen kann. Scrum betont ein funktionsfähiges Produkt am Ende des Sprints, das wirklich „fertig“ ist; im Fall von Software bedeutet das ein integriertes, vollständig getestet, dokumentiertes und potenziell auslieferbares System. Die Schlüsselrollen, Artefakte und Ereignisse sind in Abbildung 1 zusammengefasst.

Ein wesentliches Thema in Scrum ist „Inspektion und Adaption“ (Überprüfung und Anpassung, Anm. des. Übers.). Da die Entwicklung unvermeidlich Lernen, Innovation und Überraschungen beinhaltet, fordert Scrum in kurzen Entwicklungsschritten voranzugehen, dabei das resultierende Produkt und die Effektivität der Entwicklungspraktiken zu inspizieren und dann die Produkt-Ziele und Prozess-Praktiken anzupassen. *Und das immer wieder.*

# Scrum Rollen

In Scrum gibt es drei Rollen: Product Owner, Entwicklungs-Team und ScrumMaster. Zusammen kennt man sie als das Scrum Team.

Der **Product Owner** ist verantwortlich für die Maximierung des Return on Investment (ROI) durch die Identifikation von Produkt-Features, der Übertragung derselben in eine priorisierte Liste, der Entscheidung für die wichtigsten Items (Listeneinträge) für den kommenden Sprint, und die kontinuierliche Re-Priorisierung und Verfeinerung der Liste. Der Produkt Owner trägt die Gewinn- und Verlustverantwortung für das Produkt, wenn es sich um ein kommerzielles Produkt handelt. Im Falle einer internen Anwendung ist der Product Owner nicht im Sinne eines kommerziellen Produkts (welches Umsatz generiert) verantwortlich, aber immer noch für die Maximierung des ROI im Sinne der Auswahl der Items mit dem höchsten Wert - für jeden Sprint. In der Praxis ist „Wert“ ein unscharfer Begriff, und die Priorisierung kann dadurch beeinflusst werden, Schlüsselkunden zufrieden stellen zu wollen, strategische Ziele zu erfüllen, Risiken anzugehen, sich zu verbessern - oder von anderen Einflussfaktoren. In manchen Fällen sind Product Owner und Kunde ein und dieselbe Person; das ist häufig bei internen Anwendungen der Fall. In anderen Fällen kann der Kunde aus Millionen von Menschen mit einer Vielfalt von Bedürfnissen bestehen, wobei die Product Owner Rolle dann eher der des Produktmanagers oder Produkt-Marketing-Managers in vielen Produktorganisationen entspricht. Dennoch ist der Product Owner etwas anders als der traditionelle Produktmanager, da er aktiv und regelmäßig mit dem Entwicklungsteam interagiert, durch die Arbeit mit allen Stakeholdern Priorisierungen vornimmt, und die Ergebnisse jeden Sprint begutachtet, anstelle die Entwicklungsentscheidungen an einen Projektleiter zu delegieren. Dabei ist hervorzuheben, dass es eine und nur eine Person in Scrum gibt, die als Product Owner dient - und die letztendliche Entscheidungsgewalt hat. Diese Person ist verantwortlich für den Wert der Arbeit; auch wenn sie nicht alleine arbeiten muss.

**Das Team (auch Entwicklungs-Team genannt)** erstellt das Produkt, das der Product Owner vorgibt: zum Beispiel die Applikation oder Website. In Scrum ist das Team „funktionsübergreifend“ - es beinhaltet all die für die Lieferung des potenziell auslieferbaren Produkts erforderliche Expertise - und es ist „selbst-organisiert“ (managed sich selbst), mit einem hohen Grad an Autonomie und Eigenverantwortung. Das Team entscheidet, wie viele Items (aus der vom Product Owner angebotenen Auswahl) es in einem Sprint erstellt, und wie es dieses Ziel am besten erreicht.

Jedes Mitglied des Teams ist einfach ein *Teammitglied.* Man bemerke, dass es in einer Gruppe, die nach Scrum arbeitet, keine festen Titel für Spezialisten gibt. Es gibt keinen Business Analysten, keinen DBA, keinen Architekten, keinen Teamleiter, keinen Interaktions-/UX-Designer, keinen Programmierer. Das Team arbeitet in jedem Sprint in angemessener Weise zusammen, um das selbstgesetzte Ziel zu erreichen.

Da es nur *Teammitglieder* gibt, ist das Team nicht bloß funktionsübergreifend aufgebaut, sondern es demonstriert *mehrfaches Lernen*: Jede Person hat sicherlich ihre speziellen Stärken, aber lernt auch

fortwährend andere Spezialisierungen. Jede Person hat ihre primären, sekundären und sogar tertiären Fertigkeiten, und soll sich dahin bewegen, wo gerade Arbeit anfällt; Individuen übernehmen Aufgaben in weniger vertrauten Bereichen, um zu helfen, ein Item fertig zu stellen. Zum Beispiel könnte jemand mit der primären Fertigkeit Interaktions-Design auch eine zweite Fertigkeit im Bereich des automatisierten Testens haben; jemand mit einer primären Fertigkeit in der technischen Dokumentation könnte auch bei der Analyse und Programmierung unterstützen.

Das Team in Scrum besteht aus sieben plus oder minus zwei Mitgliedern, für ein Softwareprodukt könnte es Mitglieder mit Fertigkeiten in den Bereichen Analyse, Entwicklung, Test, Interface-Design, Datenbank-Design, Architektur, Dokumentation und so weiter beinhalten. Das Team entwickelt das Produkt und gibt dem Produkt Owner Anregungen, wie er das Produkt großartig machen kann. In Scrum sind die Teams am produktivsten und effektivsten, wenn alle Mitglieder während des Sprints dediziert zu 100% an einem Produkt arbeiten. Das Team vermeidet Multitasking über mehrere Produkte oder Projekte, um die teuren Verluste der aufgeteilten Aufmerksamkeit und des Kontextwechsels zu umgehen. Stabile Teams gehen mit höherer Produktivität einher, weswegen man den Wechsel von Teammitgliedern möglichst vermeidet. Produktgruppen mit vielen Mitgliedern lassen sich in mehrere Teams aufteilen, wovon jedes auf andere Features des Produkts fokussiert ist, die ihre Arbeit eng miteinander koordinieren. Da ein einzelnes Team häufig die gesamte Arbeit (Planung, Analyse, Programmierung und Test) für ein komplettes Kunden-orientiertes Feature durchführt, werden diese Teams häufig auch *Feature Teams* genannt.

Der **ScrumMaster** hilft der Produktgruppe dabei, Scrum zu lernen und anzuwenden, um Geschäftswert zu erzielen. Der ScrumMaster setzt dabei alles in seiner Macht stehende ein, um dem Team, dem Product Owner und der Organisation zum Erfolg zu verhelfen. Der ScrumMaster ist *nicht* der Vorgesetzte der Teammitglieder; ebenso wenig, wie es einen Projektleiter, Teamleiter oder Teamsprecher gibt. Stattdessen *dient* der ScrumMaster dem Team; er hilft bei der Beseitigung von Hindernissen, schützt das Team vor Störungen von außen, und hilft dem Team bei der Einführung moderner Entwicklungspraktiken. Er unterrichtet, coached, und führt den Product Owner, das Team und den Rest der Organisation bei der geschickten Anwendung von Scrum. Der ScrumMaster ist ein *Coach* und *Lehrer*. Er stellt sicher, dass jeder (einschließlich des Product Owners und des Managements) die Prinzipien und Praktiken von Scrum versteht. Er hilft dabei, die Organisation durch die häufig schwierigen Veränderungen zu geleiten, die für die erfolgreiche agile Entwicklung notwendig sind. Da Scrum viele Hindernisse und Bedrohungen für die Effektivität des Teams und des Product Owners aufzeigt, ist es wichtig einen engagierten ScrumMaster mit Hochdruck an der Behebung dieser Probleme arbeiten zu lassen. Andernfalls könnten das Team oder der Product Owner es schwer finden, Erfolg zu haben. Es sollte einen dedizierten ScrumMaster in Vollzeit geben, auch wenn ein kleineres Team ein Teammitglied in dieser Rolle haben kann (welches dann weniger mit regulärer Arbeit ausgelastet werden kann). Großartige ScrumMaster kommen aus jeder Ausbildung oder Fachrichtung: Entwicklung, Design Test, Produktmanagement, Projektleitung oder Qualitätsmanagement.

ScrumMaster und Product Owner können nicht die gleiche Person sein, da sie sich auf sehr unterschiedliche Dinge fokussieren; eine Kombination der Rollen führt häufig zu Verwirrung und Konflikten. Ein häufig auftretendes unangenehmes Resultat der Kombination dieser Rollen ist ein mikro-managender Product Owner, also das genaue Gegenteil zu den selbst-managenden Teams, die Scrum erfordert. Anders als ein traditioneller Manager sagt der ScrumMaster den Mitarbeitern nicht, was sie tun sollen oder weist ihnen Aufgaben zu - er moderiert den Prozess, unterstützt das Team bei seiner Selbstorganisation und Selbstverwaltung. Wenn ein ScrumMaster vorher in einer Position war, bei der er das Team geleitet hat, muss er seine Einstellung und seine Art und Weise, mit dem Team zu interagieren, fundamental ändern, damit das Team mit Scrum erfolgreich sein kann.

Beachte: Es gibt in Scrum wirklich keine Projektleiter-Rolle. Sie wird nicht benötigt; die traditionellen Verantwortlichkeiten eines Projektleiters wurden auf die drei Scrum Rollen verteilt - die meisten davon eher auf das Team und den Product Owner als auf den ScrumMaster. Die Anwendung von Scrum mit einem zusätzlichen Projektleiter zeigt ein fundamentales Missverständnis von Scrum auf, und führt typischerweise zu Verantwortlichkeiten, die miteinander im Konflikt stehen, unklaren Entscheidungswegen und suboptimalen Ergebnissen. Manchmal kann ein (Ex-)Projektleiter in die Rolle des ScrumMasters wechseln, aber der Erfolg dieser Vorgehensweise hängt stark ab von dem Individuum und seinem Verständnis von der grundsätzlichen Verschiedenheit dieser Rollen, sowohl bei

den täglichen Verantwortungen als auch in der für den Erfolg wichtigen Einstellung. Ein guter Weg, die Rolle des ScrumMasters zu verstehen und die Grundfertigkeiten für den Erfolg zu entwickeln, ist der Besuch des Certified ScrumMaster Trainings der Scrum Alliance.

Zusätzlich zu diesen drei Rollen gibt es weitere Stakeholder, die zu dem Erfolg des Produkts beitragen, einschließlich der Manager, Kunden und Anwender. Einige Stakeholder, wie funktionale Manager (zum Beispiel ein Entwicklungsleiter), werden sehen, dass ihre Rolle, auch wenn sie weiterhin von Bedeutung ist, sich bei der Einführung von Scrum ändert. Zum Beispiel:

- unterstützen sie das Team, indem sie die Regeln und den Geist von Scrum respektieren
- helfen sie bei der Beseitigung von Hindernissen, die das Team und der Product Owner identifizieren
- stellen sie ihre Expertise und Erfahrung zur Verfügung

In Scrum ersetzen diese Individuen die Zeit, die sie vorher in der Rolle der „Nanny" verbracht haben (Aufgaben zuweisen, Statusreports erhalten und andere Formen des Mikromanagements) durch die Zeit als „Guru" oder „Diener" des Teams (Mentoring, Coaching, Hilfe bei der Beseitigung von Hindernissen, Hilfe bei Problemlösungen, kreative Beiträge und Anleitung bei der Ausbildung der Teammitglieder). In diesem Wandel müssen Manager vielleicht auch ihren Führungsstil ändern; zum Beispiel das Team durch sokratische Fragen bei der Entdeckung einer Problemlösung helfen, statt einfach über eine Lösung zu entscheiden und sie dem Team zu geben.

# Product Backlog

Wenn eine Gruppe den Wechsel zu Scrum plant, braucht sie vor dem Beginn des ersten Sprints ein **Product Backlog**, eine priorisierte (in der Reihenfolge 1,2,3,...) Liste von kundenorientierten Features.

Das Product Backlog besteht (und entwickelt sich) über die gesamte Lebenszeit des Produkts; es ist die Produkt-Roadmap (Abbildung 2 und Abbildung 3). Das Product Backlog ist zu jedem Zeitpunkt die einzige, definitive Darstellung von „allem, was das Team jemals liefern könnte, nach Priorität geordnet". Es gibt nur ein einziges Product Backlog für ein Produkt; also muss der Product Owner Priorisierungs-Entscheidungen über das gesamte Spektrum treffen und dabei die Interessen der Stakeholder (einschließlich des Teams) berücksichtigen.

| | | | | **Neue Schätzungen bei Sprint ...** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Priorität** | **Eintag** | **Details (wiki URL)** | **Initiale Größenschätzung** | **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | Als Käufer möchte ich ein Buch in den Warenkorb legen (siehe UI-Skizzen auf der Wiki-Seite) | ... | 5 | | | | | | |
| 2 | Als Käufer möchte ich ein Buch aus meinem Warenkorb entfernen | ... | 2 | | | | | | |
| 3 | Performance der Transaktionen verbessern (siehe Zielperformancemetriken im Wiki) | ... | 13 | | | | | | |
| 4 | Lösungen zur Beschleunigung der Kreditkartenvalidierung untersuchen (siehe Zielperformancemetriken im Wiki) | ... | 20 | | | | | | |
| 5 | Alle Server auf Apache 2.2.3 upgraden | ... | 13 | | | | | | |
| 6 | Diagnose und Fixen des Fehlers im Skript für die Bestellverarbeitung (Bugzilla ID 14823) | ... | 3 | | | | | | |
| 7 | Als Käufer möchte ich Wunschlisten erstellen und speichern können | ... | 40 | | | | | | |
| 8 | Als Käufer möchte ich Einträge zu meiner Wunschliste hinzufügen und löschen können | ... | 20 | | | | | | |

*Abbildung 2. Das Product Backlog*


*Abbildung 3. Visuelles Management: Product Backlog Items an der Wand*

Das Product Backlog beinhaltet eine Vielzahl von **Items**, hauptsächlich neue Kunden-Features („ermögliche es allen Benutzern, ein Buch in den Einkaufswagen zu legen"), aber auch größere technische Verbesserungs-Ziele (z.B. „portiere das System von C++ auf Java"), Verbesserungen (z.B. „beschleunige unsere Tests"), Forschung und Recherche („untersuche Lösungen für die schnellere Kreditkartenvalidierung"), und möglicherweise bekannte Defekte („diagnostiziere und behebe die Fehler im Skript für die Bestellverarbeitung"), wenn es davon nur eine kleine Anzahl gibt. (Ein System mit vielen Defekten hat häufig ein separates Bug-Tracking.)

Product Backlog Einträge sind auf beliebige Weise, die klar und nachhaltig verwendbar ist, formuliert. Entgegen eines beliebten Missverständnisses, enthält das Product Backlog *nicht* „User Stories", sondern einfach Einträge *(Items)*. Diese Items können als User Stories, Use Cases oder gemäß jedes anderen Anforderungs-Ansatzes, den die Gruppe nützlich findet, formuliert sein. Ungeachtet des Ansatzes sollten die meisten Items aber darauf fokussiert sein, den Kunden Mehrwert zu liefern.

Ein gutes Product Backlog ist DEEP…

**Detailed appropriately (Angemessen detailliert)**. Die Items mit höchster Priorität sind feingranularer und detaillierter als die mit niedrigerer Priorität, weil man an den ersteren eher als an den letzteren arbeiten wird. Zum Beispiel bestehen die oberen 10% des Backlogs aus sehr kleinen, gut analysierten Items, und die restlichen 90% eher nicht.

**Estimated (Geschätzt)**. Die Items im aktuellen Release müssen Schätzungen haben, und sollten darüber hinaus nach jedem Sprint auch für eine Neu-Schätzung in Betracht gezogen werden, wenn alle dazu lernen und neue Informationen aufkommen. Das Team gibt dem Product Owner *Aufwands*-Schätzungen für jedes Item im Product Backlog, vielleicht auch Einschätzungen für das *technische Risiko*. Der Product Owner und andere Business-Stakeholder liefern die Einschätzung des Werts der Produktanforderungen, was die Umsatzsteigerung, Kostenreduzierung, geschäftlichen Risiken, und Bedeutung für verschiedene Stakeholder einschließen kann.

**Emergent (Aus sich selbst heraus entstehend)**. Als Antwort auf das Lernen und die Variabilität wird das Product Backlog regelmäßig verfeinert. Jeden Sprint können Items hinzugefügt, herausgenommen, angepasst, aufgeteilt und umpriorisiert werden. Daher wird das Product Backlog kontinuierlich vom Product Owner angepasst, um Änderungen der Kundenbedürfnisse, neue Ideen oder Einsichten, Aktionen der Mitwerber, technische Hürden und so weiter zu reflektieren.

**Prioritized (Priorisiert)**. Die Einträge an der Spitze des Product Backlogs sind priorisiert oder *angeordnet* in einer 1-N Reihenfolge. Generell sollten die Items mit der höchsten Priorität den meisten

Geschäftsnutzen *bang for your buck* liefern: eine hohe Auswirkung (*bang)* für niedrige Kosten *(buck)*. Eine weitere Motivation, die Priorität eines Items zu erhöhen ist es, *hohe Risiken früh anzugehen, bevor die Risiken zuschlagen*.

Die traditionelle Entwicklung betont üblicherweise nicht die Lieferung des höchsten *bang for your buck,* aber das ist ein Grundthema von Scrum. Daher muss der Product Owner lernen, die Auswirkung des „Geschäftswerts" zu beurteilen. Das ist etwas, wobei ihm der ScrumMaster helfen kann. Was bedeutet „Geschäftswert"? Einige Produktgruppen verwenden eine einfache relative Punkte-Bewertung für jedes Product Backlog Item, das ein „guesstimate" - eine Baucheinschätzung - von Faktoren wie Umsatzsteigerung, Kostenminderung, Stakeholder-Vorlieben, Marktabgrenzung und so weiter verbindet. Andere finanzieren ein bestimmtes Item durch ein oder zwei Kunden, die für die Entwicklung zahlen, und nutzen so den exakten (kurzfristigen) Umsatz für dieses Item als Annäherung für den Wert. Wieder anderen ist diese Item-spezifische Schätzung zu unfokussiert oder granular, sie wenden einen breiteren auf Geschäftsergebnisse basierten Ansatz an („erhöhe die Abonnements um 10% bis zum 1. September"), in dem der Wert nur geliefert wird, wenn mehrere Items, die sich auf das Ergebnis auswirken, zusammen ausgeliefert werden. In diesem Fall muss der Product Owner das nächste Inkrement dieses Minimum Viable Product (kleinsten verwendbaren Produkts) definieren.

Eine häufige Technik für Aufwandsschätzungen ist die Schätzung in relativen Größen (unter Berücksichtigung von Aufwand, Komplexität und Unsicherheit) unter Verwendung einer Einheit von „Story Points" oder einfach „Points".

Das sind bloß Vorschläge; Scrum definiert nicht die Technik für die Formulierung oder Priorisierung von Items im Product Backlog, ebenso wenig, wie es die Schätz-Technik oder -Einheiten vorgibt.

Eine weitere häufige Technik, die in Scrum verwendet wird, ist die Nachverfolgung der fertiggestellten Arbeit pro Sprint; zum Beispiel 26 Punkte pro Sprint geschafft. Mit dieser Information lässt sich ein Release-Datum für die Fertigstellung aller Features vorhersagen, oder die Menge der Features zu einem bestimmten Datum, wenn der Durchschnitt weiterhin gültig ist und sich nichts ändert. Dieser Durchschnittswert wird „Velocity" (Geschwindigkeit) genannt. Die Velocity wird in der gleichen Einheit wie die Größenschätzungen der Product Backlog Items ausgedrückt.

Die Items im Product Backlog können signifikant in ihrer Größe oder ihrem Aufwand voneinander abweichen. Größere Einträge werden in kleinere während des Product Backlog Verfeinerungs-Workshops heruntergebrochen, kleinere können zusammengefasst werden. Die Product Backlog Items für die nächsten paar Sprints sollten klein und feingranular genug sein, dass sie vom Team verstanden werden - was die Prognosen im Sprint Planning Meeting sinnvoll macht; das nennt man auch eine „handhabbare" Größe.

Größere technische Verbesserungen, die viel Zeit und Geld erfordern, sollten im Product Backlog stehen, da sie eine optionale Geschäftsinvestition darstellen, die letztendlich der geschäftsorientierte Product Owner tätigen muss. Bemerke, dass das Team in Scrum die unabhängige Kompetenz hat, wie viele Items aus dem Product Backlog es in einen Sprint mit aufnimmt; damit kann es kleinere technische Verbesserungen eigenständig einplanen, welche man als Teil der normalen Kosten ansehen kann, die für einen Entwickler erforderlich sind, um seine Arbeit richtig zu erledigen. Ungeachtet dessen sollte der *Hauptteil* der Zeit des Teams in jedem Sprint für die Ziele des Product Owners und nicht für interne Entwicklungsaufgaben verwendet werden.

Einer der Mythen von Scrum ist, dass es einen vom Schreiben detaillierter Spezifikationen befreit. In der Realität liegt es am Product Owner und Team, über die erforderliche Detailtiefe zu entscheiden. Das kann von einem Backlog Item zum nächsten variieren, abhängig von dem Verständnis des Teams und anderen Faktoren. Beschreiben Sie, was wichtig ist, im kleinstmöglichen Umfang. Mit anderen Worten, beschreiben Sie nicht jedes mögliche Detail eines Items, sondern machen Sie klar, was für das Verständnis erforderlich ist, und bestärken Sie dieses durch den kontinuierlichen Dialog zwischen Team, Product Owner und Stakeholdern. Product Backlog Items mit niedriger Priorität, an denen für eine Weile nicht gearbeitet werden wird, sind normalerweise „grob-granular" (groß, mit weniger detaillierten Anforderungen). Feiner granulare Product Backlog Items mit hoher Priorität, die bald implementiert werden, haben tendenziell zu mehr Details.

# Definition of Done

Das Ergebnis jeden Sprints wird offiziell als Potenziell Auslieferbares Produktinkrement bezeichnet. Bevor der erste Sprint beginnt, müssen der Product Owner, das Team und der ScrumMaster begutachten, was alles erfüllt sein muss, damit ein Product Backlog Item potenziell auslieferbar ist. Alle Aktivitäten, die notwendig sind, um das Produkt ausliefern zu können, sollten in der Definition von Potenziell Auslieferbar enthalten sein und daher im Sprint ausgeführt werden.

Leider können Teams, wenn sie mit Scrum beginnen, das Ziel, ein Potenziell Auslieferbares Inkrement in jedem Sprint zu erzeugen, oft nicht erreichen. Das liegt oft daran, dass es dem Team an Automatisierung fehlt, oder dass es nicht cross-funktional genug ist (z.B. noch kein technischer Autor im cross-funktionalen Team vorhanden ist). Im Laufe der Zeit muss sich das Team verbessern, so dass es ein Potenziell Auslieferbares Inkrement in jedem Sprint liefern kann. Um dies zu können, muss es eine Bestandsaufnahme seiner existierenden Fähigkeiten durchführen. Diese wird in der Definition of Done festgehalten.

Vor dem ersten Sprint müssen sich der Product Owner und das Team auf eine Definition of Done einigen, die einen Teil der Aktivitäten abdeckt, die für die Erzeugung eines Potenziell Auslieferbaren Produktinkrements nötig sind (für ein gutes Team ist es das gleiche). Das Team plant seine Arbeit im Sprint im Einklang mit der Definition of Done.

Ein guter Product Owner will erreichen, dass die Definition of Done so weit wie möglich mit Potenziell Auslieferbar übereinstimmt, da dies die Transparenz in der Entwicklung erhöht und *Verzögerungen und Risiken* verringert. Wenn die Definition of Done nicht mit Potenziell Auslieferbar übereinstimmt, wird Arbeit bis kurz vor das Release verschoben, was *Risiko und Verzögerungen* verursacht. Diese verschobene Arbeit wird manchmal *unerledigte (undone) Arbeit* genannt.

Ein Scrum Team sollte sich laufend verbessern, was sich in der Ausweitung der Definition of Done zeigt.

# Sprint Planning

**Zusammenfassung**: Ein Meeting zur Vorbereitung des Sprints, typischer Weise in zwei Teile geteilt (Teil 1 ist "was" und Teil 2 ist "wie").
**Teilnehmer**: Teil 1: Product Owner, Team, ScrumMaster. Teil 2: Team, ScrumMaster, Product Owner (optional, aber er sollte für Fragen erreichbar sein)
**Dauer**: Jeder Teil ist auf eine Stunde pro Sprint-Woche begrenzt.

Am Anfang jeden Sprints findet das **Sprint Planning Meeting** statt. Es ist in zwei getrennte Unter-Meetings geteilt, wovon das erste **Sprint Planning Teil 1** genannt wird.

Im Sprint Planning Teil 1 begutachten der Product Owner und das Team die hochprioren Einträge im Product Backog, die der Product Owner im Sprint implementiert haben möchte. Normaler Weise wurden diese Einträge sehr gut in einem vorangegangenen Sprint analysiert (während der Product Backlog Verfeinerung), so dass in diesem Meeting nur kleinere last-minute Fragen geklärt werden. In diesem Meeting diskutieren der Product Owner und das Team die Ziele und den Kontext für diese hochprioren Items im Product Backlog, was dem Team Einblick in die Gedanken des Product Owners erlaubt. Teil 1 konzentriert sich darauf, *was* der Product Owner möchte und *warum* es gebraucht wird. Am Ende von Teil 1 kann der Product Owner gehen, aber er *muss* während Teil 2 des Meetings verfügbar sein (z.B. per Telefon).

Im Teil 1 können das Team und der Product Owner auch das **Sprint-Ziel** formulieren. Das ist eine zusammenfassende Aussage über die im Sprint anvisierten Ziele, die idealer Weise ein zusammenhängendes Thema haben. Das Sprint-Ziel gibt dem Team auch Flexibilität bezüglich des Umfangs, den es tatsächlich ausliefert, denn auch wenn es etwas auslassen muss (weil der Sprint zeitlich

begrenzt ist), sollte es sich dennoch verpflichten, etwas Greifbares und Fertiges zu liefern, das im Sinne des Sprint-Ziels ist.

Wie groß sollten die Backlog-Einträge sein, die in einen Sprint aufgenommen werden? Jeder Backlog-Eintrag sollte klein genug heruntergebrochen sein, so dass es deutlich kleiner als der gesamte Sprint geschätzt wird. Eine gebräuchliche Richtlinie ist, dass der Backlog-Eintrag als klein genug geschätzt wird, in einem Viertel oder weniger des Sprints vom ganzen Team fertig gestellt werden zu können.

**Sprint Planning Teil 2** konzentriert sich darauf, *wie* die Backlog-Eintrage, die das Team entschieden hat aufzunehmen, implementiert werden sollen. Das Team prognostiziert die Menge der Backlog-Eintrage, die es bis zum Ende des Sprints fertig stellen kann, indem es am oberen Ende des Product Backlogs beginnt (mit anderen Worten, mit den Einträgen, die für den Product Owner die höchste Priorität haben) und die Liste in geordneter Reihenfolge abarbeitet. *Das ist eine Schlüsselpraktik in Scrum. Das Team entscheidet, wie viel Arbeit es fertig stellen wird, anstatt sie vom Product Owner zugewiesen zu bekommen.* Dies macht die Prognose verlässlicher, weil das Team sie auf Basis seiner eigenen Analyse und Planung abgibt. Auch wenn der Product Owner keine Kontrolle darüber hat, wie viel das Team annimmt, weiß er oder sie, dass die Einträge vom oberen Ende des Product Backlogs genommen werden – mit anderen Worten, die Einträge, die er oder sie als die wichtigsten eingestuft hat. Das Team hat die Möglichkeit, sich für Einträge einzusetzen, die weiter unten in der Liste stehen; das geschieht normaler Weise, wenn das Team und der Product Owner erkennen, dass etwas von niedrigerer Priorität leicht und angemessen mit Einträgen höherer Priorität zusammenpasst.

Das Sprint Planning Meeting dauert oft mehrere Stunden, aber nicht länger als vier Stunden für einen zweiwöchigen Sprint – das Team liefert eine ernstzunehmende Prognose über die fertigzustellende Arbeit, und das erfordert sorgfältige Überlegung, um erfolgreich zu sein. Teil 1 und Teil 2 sind von gleicher begrenzter Länge; für einen zweiwöchigen Sprint dauert jeder Teil maximal zwei Stunden.

Scrum legt nicht genau fest wie das Sprint Planning Teil 2 durchzuführen ist. Einige Teams benutzen ihre Velocity der vorangegangenen Sprints, um sie anzuleiten, was sie anstreben können. Andere Teams benutzen einen feingranulareren Prozess, indem sie zunächst ihre Kapazität errechnen.

Wenn der Kapazitäts-Ansatz genutzt wird, beginnt das Team das Sprint Planning Teil 2 damit abzuschätzen, wie viel Zeit jedes Teammitglied für Sprint-bezogene Arbeit hat. Für die meisten Menschen läuft es auf 4-6 Stunden pro Tag hinaus, die sie für Sprint-bezogene Arbeit zur Verfügung haben. Das ist die Kapazität des Teams für den nächsten Sprint; die restliche Zeit verbringen die Teammitglieder mit Meetings, E-Mails, Mittagessen und Kaffeetrinken. Wenn die Kapazität bestimmt ist, muss das Team herausfinden, wie viele Product Backlog Einträge es in dieser Zeit fertig stellen können, und wie es die Aufgabe angehen will. Das fängt oft mit einer Design-Diskussion an einem Whiteboard an. Wenn das übergreifende Design verstanden ist, bricht das Team die Product Backlog Einträge in feingranulare Arbeitsschritte herunter. Bevor die Product Backlog Einträge angenommen werden, kann sich das Team darauf konzentrieren, Aufgaben für Verbesserungsziele zu generieren, die in der Retrospective des vorangegangenen Sprints erzeugt wurden. Dann wählt das Team den ersten Eintrag des Product Backlogs – den mit der höchsten Priorität des Product Owners – und arbeitet die Einträge von oben nach unten ab, bis es „voll“ ist. Für jeden Eintrag erzeugt es eine Arbeitsliste, die entweder aus Arbeitsaufgaben besteht, die sich aus der Aufteilung des Product Backlogs ergeben, oder einfach aus dem Product Backlog Eintrag selbst, wenn der Product Backlog Eintrag so klein ist, dass er nur ein paar Stunden zur Implementierung benötigt. Diese Liste von zu erledigender Arbeit wird **Sprint Backlog** genannt (Abb. 4 und 5).

Am Ende des Sprint Planning Meetings, setzt das Team ein realistisches Ziel für das, was sie glauben bis zum Ende des Sprints liefern zu können. Traditionell wurde das Sprint Commitment genannt – das Team verpflichtet sich, ihr Bestes zu tun, um ihr Ziel zu erreichen. Leider wurde das häufig als ein „mit Blut unterschriebenes“ Versprechen fehlinterpretiert, anstatt dass das Team sich ernsthaft darum bemüht. Um diese Verwirrung zu vermeiden, wird der Vorgang jetzt „Prognose“ genannt und so zum Product Owner kommuniziert.

| | | | | Neue Aufwandsschätzung Übrig am Ende des Tages | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Product Backlog Eintrag** | **Sprint-Aufgabe** | **Freiwilliger** | **Anfängliche r Schätzwert** | **1** | **2** | **3** | **4** | **5** | **6** |
| Als Käufer möchte ich ein Buch in den Warenkorb legen | Datenbank verändern | | 5 | | | | | | |
| | Webseite erzeugen (UI) | | 8 | | | | | | |
| | Webseite erzeugen (Javascript Logik) | | 13 | | | | | | |
| | Automatischen Akzeptanztest erzeugen | | 13 | | | | | | |
| | Hilfe-Seite für Nutzer aktualisieren | | 3 | | | | | | |
| | . . . | | | | | | | | |
| Performance der Transaktionen verbessern | DCP Code mergen und layer-level Tests vervollständigen | | 5 | | | | | | |
| | Maschinenbefehl für pRank vervollständigen | | 8 | | | | | | |
| | DCP und reader ändern, so dass sie pRank http API benutzen | | 13 | | | | | | |

Abbildung 4. Ein Beispiel für einen Weg, ein Sprint Backlog zu erstellen

Scrum ermuntert zu Arbeitskräften mit mehreren Fähigkeiten, anstatt nur nach seiner Arbeitsbeschreibung zu arbeiten, wie „Tester“, die nur testen. Mit anderen Worten gehen die Teammitglieder dahin „wo die Arbeit ist“ und helfen sich gegenseitig wo möglich. Wenn es viele Test-Aufgaben gibt, können *alle* Team-Mitglieder helfen. Das heißt nicht, dass jeder ein Generalist ist; kein Zweifel, dass einige Leute besonders befähigt beim Testen sind (und so weiter), aber Team- Mitglieder arbeiten zusammen und lernen neue Fähigkeiten voneinander. Folglich ist es weder notwendig noch angemessen, dass Personen sich während der Erzeugung und Schätzung von Aufgaben im Sprint Planning freiwillig für alle Aufgaben melden, die „sie am besten tun können“. Statt dessen ist es besser, sich nur für eine Aufgabe gleichzeitig zu melden, wenn es an der Zeit ist, eine neue Aufgabe zu wählen, und in Betracht zu ziehen, bewusst Aufgaben zu wählen, die Lernen mit einschließen (vielleicht indem man mit einem Spezialisten im Paar zusammenarbeitet). Das ist der Grund, warum Aufgaben im Sprint Planning nicht vorab zugeteilt werden; dies sollte statt dessen während des Sprint „wenn notwendig“ gemacht werden.

Es gibt *seltene* Fälle, dass *John* eine bestimmte Aufgabe erledigen sollte, wenn es für andere zu lange dauern würde oder unmöglich wäre, sie zu erlernen – vielleicht ist John die einzige Person mit artistischen Fähigkeiten, Bilder zu malen. Andere Team-Mitglieder können nicht einmal ein Strichmännchen malen wenn ihr Leben davon abhinge. In diesem seltenen Fall – und wenn es nicht so selten vorkommt und nicht seltener wird, obwohl das Team lernt, stimmt etwas nicht – könnte es notwendig sein zu fragen, ob die gesamte geplante Zeichenaufgabe, die von John erledigt werden muss, in einem kurzen Sprint machbar ist.

Viele Teams haben einen Sprint Backlog in Form eines wandfüllenden Task Boards (oft **Scrum Board** genannt), wo Aufgaben (auf Post-It Zetteln) während des Sprints durch Spalten wandern, die als „zu erledigen“ „in Arbeit“ und „erledigt“ bezeichnet sind. Siehe Abb. 5

Einer der Pfeiler von Scrum ist, dass alle Zusätze oder Änderungen auf den nächsten Sprint verschoben werden müssen, sobald sich das Team sein Ziel für den Sprint gesetzt hat. Das bedeutet, dass wenn der Product Owner entscheidet, dass es einen neuen Backlog-Eintrag gibt, von dem er oder sie möchte, dass das Team ihn bearbeitet, kann er diese Änderung nicht vor dem Beginn des nächsten Sprints einbringen. Wenn äußere Umstände auftreten, die die Prioritäten erheblich ändern, und bedeuten, dass das Team seine Arbeit verschwenden würde, wenn es weiterarbeitet, kann der Product Owner oder das Team den Sprint abbrechen. Das Team hört auf zu arbeiten, und ein neues Sprint Planning Meeting initiiert einen neuen Sprint. Dies verursacht eine erhebliche Störung, was eine Abschreckung für den Product Owner und das Team sein sollte, diese dramatische Entscheidung zu treffen.

Ein machtvoller, positiver Einfluss geht davon aus, dass das Team geschützt ist vor Zielen, die sich während des Sprints ändern. Erstens geht das Team an die Arbeit mit der absoluten Sicherheit, dass

*Abbildung 5:* Visuelles Management - Sprint Backlog Aufgaben an der Wand

sein Ziel sich nicht ändern wird, was seinen Fokus darauf verstärkt, die Vervollständigung sicherzustellen. Zweitens wird der Product Owner diszipliniert, die Einträge, die er priorisiert und dem Team für den Sprint anbietet, wirklich zu durchdenken.

Der Product Owner gewinnt zwei Dinge, indem er diese Regeln befolgt. Erstens hat er das Vertrauen, dass das Team sich verpflichtet, sein Bestes zu tun, einen realistischen und klaren Satz von Arbeitsaufgaben fertigzustellen, den es ausgewählt hat. Mit der Zeit kann das Team sehr bewandert darin werden, eine realistische Prognose zu wählen und auszuliefern. Zweitens macht der Product Owner alle Änderungen, die er haben möchte, vor dem Start des *nächsten* Sprints. Bis zu diesem Punkt ist es möglich und akzeptabel, Dinge hinzuzufügen, zu löschen, zu ändern und umzupriorisieren. Während der Product Owner nicht berechtigt ist, während des laufenden Sprints Änderungen an den ausgewählten in Entwicklung befindlichen Product Backlog Einträgen vorzunehmen, ist er nur eine Sprintlänge oder weniger entfernt, um alle Änderungen machen zu können, die er sich wünscht. Das Stigma um Veränderung – Veränderung der Richtung, Veränderung der Anforderungen, oder einfach Veränderung der Meinung – ist Vergangenheit, und es mag aus diesem Grund sein, dass Product Owner normaler Weise so enthusiastisch in Bezug auf Scrum sind wie jeder andere.

# Daily Scrum

**Zusammenfassung**: Aktualisierung und Koordination unter den Teammitgliedern
**Teilnehmer**: Das Team ist notwendig, der Product Owner ist optional; der ScrumMaster ist normaler Weise anwesend, stellt aber zumindest sicher, dass das Team einen Daily Scrum durchführt.
**Dauer**: maximale Länge von 15 Minuten

Wenn der Sprint begonnen hat, beteiligt sich das Team an einer weiteren der Schlüsselpraktiken in Scrum: dem **Daily Scrum**. Das ist ein kurzes (15 Minuten oder weniger) Meeting, das jeden Tag zu einer geeigneten Zeit stattfindet. Jeder aus dem Team nimmt teil. Um es kurz zu halten, ist es

empfehlenswert, dass jeder stehenbleibt. Das ist die Gelegenheit für das Team, seine Arbeit zu synchronisieren und sich gegenseitig über Hindernisse zu informieren. Im Daily Scrum berichtet ein Teammitglied nach dem anderen *den anderen Teammitgliedern* drei Dinge. (1) Was wurde erreicht seit dem letzten Meeting?; (2) Was wird getan bis zum nächsten Meeting?; (3) Welche Hindernisse sind im Weg?. Zu beachten: das Daily Scrum ist kein Statusmeeting, um einem Manager zu berichten, sondern es ist eine Zeit für ein selbstorganisierendes Team, miteinander zu teilen was gerade vor sich geht, um ihm zu helfen, sich zu koordinieren. Jemand macht Notizen von den Blockaden, und der ScrumMaster ist verantwortlich dafür, dem Team zu helfen, sie zu beseitigen. Tiefgehende Diskussionen finden im Daily Scrum wenig oder gar nicht statt, das Thema ist Anworten auf die drei Fragen zu *berichten*; wenn Diskussion notwendig ist, findet diese direkt nach dem Daily Scrum statt in einem oder mehreren Folgemeetings, auch wenn in Scrum niemand verpflichtet ist, diesen beizuwohnen. Ein Folgemeeting ist ein übliches Ereignis, in dem einige oder alle Teammitglieder sich an die Informationen anpassen, die sie im Daily Scrum erhalten haben: mit anderen Worten, ein weiterer „inspect and adapt“ (inspiziere und passe an) Zyklus. Für Teams, die gerade mit Scrum beginnen, ist es empfehlenswert, dass *kein* Manager oder eine andere als Autorität wahrgenommene Person am Daily Scrum teilnimmt. Dies bringt das Risiko mit sich, dass das Team sich „kontrolliert“ fühlt – unter dem Druck jeden Tag einen größeren Fortschritt zu erzielen (eine unrealistische Erwartung), und daran gehindert, über Probleme zu berichten – es neigt dazu, das Selbst-Management des Teams zu unterwandern und lädt zu Mikromanagement ein. Es wäre sinnvoller für einen Stakeholder, sich nach dem Meeting an das Team zu wenden und Hilfe bei allen Blockaden anzubieten, die den Fortschritt des Teams verlangsamt.

# Fortschrittsverfolgung während des Sprints

| | | | | **Neue Aufwandschätzung Übrig am Ende des Tages** | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Product Backlog Eintrag** | **Sprint-Aufgabe** | **Freiwilliger** | **Anfänglicher Schätzwert** | **1** | **2** | **3** | **4** | **5** | **6** |
| Als Käufer möchte ich ein Buch in den Warenkorb legen | Datenbank verändern | Sanjay | 5 | 4 | 3 | 0 | 0 | 0 | |
| | Webseite erzeugen (UI) | Jing | 3 | 3 | 3 | 2 | 0 | 0 | |
| | Webseite erzeugen (Javascript Logik) | Tracy & Sam | 2 | 2 | 2 | 2 | 1 | 0 | |
| | Automatischen Akzeptanztest erzeugen | Sarah | 5 | 5 | 5 | 5 | 5 | 0 | |
| | Hilfe-Seite für Nutzer aktualisieren | Sanjay & Jing | 3 | 3 | 3 | 3 | 3 | 0 | |
| | . . . | | | | | | | | |
| Performance der Transaktionen verbessern | DCP code mergen und layer-level Tests vervollständigen | | 5 | 5 | 5 | 5 | 5 | 5 | |
| | Maschinenbefehl für pRank vervollständigen | | 3 | 3 | 8 | 8 | 8 | 8 | |
| | DCP und reader ändern, so dass sie pRank http API benutzen | | 5 | 5 | 5 | 5 | 5 | 5 | |
| | . . . | . . . | . . . | | | | | | |
| | | **Total** | 50 | 49 | 48 | 44 | 43 | 34 | |

*Abbildung 6:* Tägliche Aktualisierung des Restaufwands im Sprint Backlog

Das Team organisiert sich selbst, und um das erfolgreich machen zu können, muss es wissen wo es steht. Jeden Tag aktualisieren die Teammitglieder ihre Schätzung für den Restaufwand zur Fertigstellung ihrer Arbeit im **Sprint Backlog** (Abb. 6). Es ist auch üblich, dass jemand die Restaufwände aufsummiert und im **Sprint Burndown Chart** aufträgt (Abb. 7 und 8). Diese Kurve zeigt jeden Tag einen neuen Schätzwert, wie viel Arbeit noch zu tun ist, bis das Team fertig ist. Idealer Weise ist es eine *abwärts* gerichtete Kurve, deren Verlängerung „null Restaufwand“ am letzten Tag des Sprints erreicht. Daher wird sie *Burndown* Chart genannt. Auch wenn sie manchmal gut aussieht, ist es oft nicht der Fall; das ist die Realität bei der Produktentwicklung. Das Wichtige daran ist, dass sie dem Team seinen Fortschritt in Bezug auf das Ziel zeigt, nicht durch die Zeit, die bereits *verbraucht* wurde (eine für den Fortschritt irrelevante Information), sondern durch die Zeit, die für die *Zukunft* noch zu tun *bleibt* – die das Team von seinem Ziel trennt. Wenn die Burndown-Kurve nicht abwärts zeigt in Richtung Fertigstellung gegen Ende des Sprints, muss das Team sich anpassen, indem zum Beispiel der

Umfang der Arbeit reduziert wird oder effizienter gearbeitet wird unter Beibehaltung eines nachhaltigen Schritttempos.

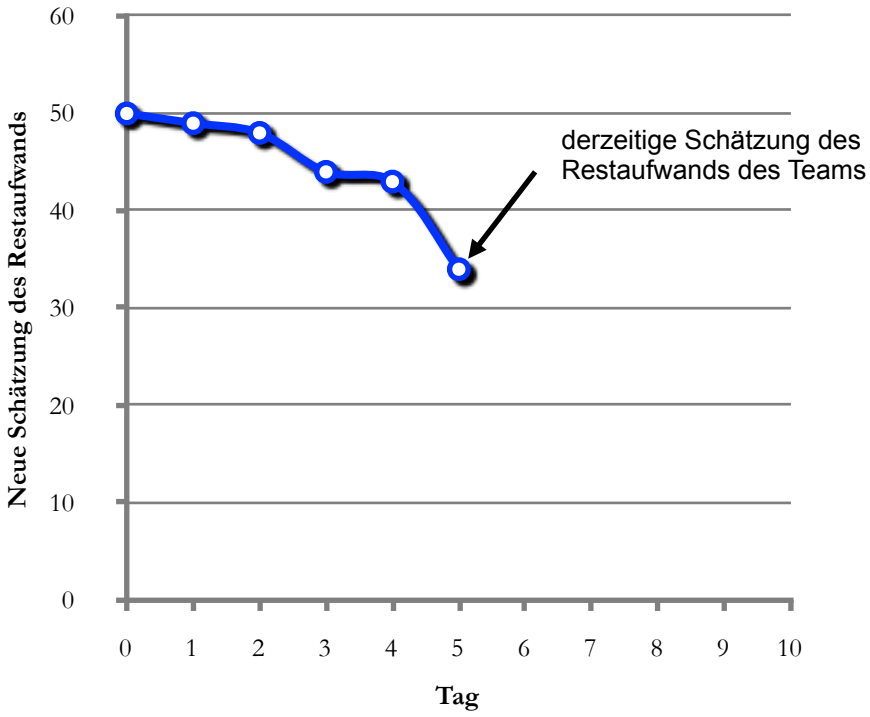


*Abbildung 7: Sprint Burndown Chart*

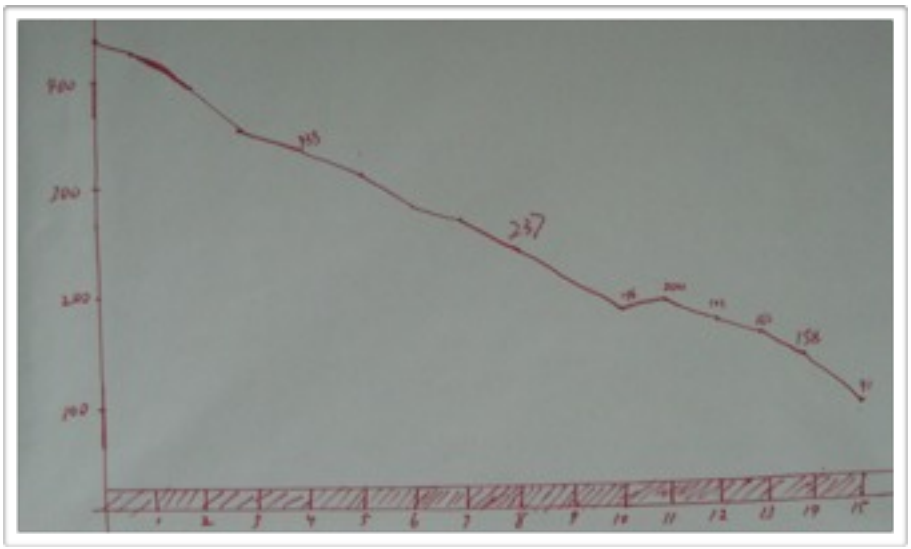

*Abbildung 8. Visuelles Management: Handgezeichnetes Sprint Burndown Chart*

Auch wenn die Sprint Burndown Kurve als Kalkulationstabelle erzeugt und dargestellt werden kann, finden viele Teams es effektiver, sie auf Papier an einer Wand in ihrem Arbeitsraum zu zeigen, mit handgeschriebenen Änderungen. Diese „low tech/high touch“ Lösung ist schnell, einfach, und oft sichtbarer als eine Computer-Graphik.

# Product Backlog Verfeinerung

**Zusammenfassung**: Große Backlog-Einträge aufteilen, Einträge analysieren, den aktuellen Schätzwert für die verbleibende Arbeit des Teams aktualisieren und re-priorisieren für *zukünftitge* Sprints..
**Teilnehmer**: Team; Product Owner nimmt während der ganzen Aktivität teil, wenn er der Experte ist, der bei der detaillierten Verfeinerung helfen kann. Andernfalls nimmt er nur stellenweise teil, um die Richtung anzugeben oder zu re-priorisieren; andere, die die Anforderungen verstehen, können dem Team helfen; der ScrumMaster nimmt an anfänglichen Sitzungen teil, um die Gruppe zu coachen, effektiv zu werden; sonst muss er nicht teilnehmen.
**Dauer**: Normaler Weise nicht mehr als 10% der Teamkapazität des Sprints, obwohl es auch länger dauern kann für "Analyse-gewichtige" Backlog-Einträge. Zum Beispiel wird bei einem zweiwöchigen Sprint ein Tag mit Verfeinerung verbracht.

Einer der weniger bekannten, aber wertvollen Richtlinien in Scrum ist, dass ein Prozentsatz jedes Sprints vom ganzen Team der Verfeinerung (oder "grooming") des Product Backlogs gewidmet werden sollte, um zukünftige Sprints vorzubereiten. Das beinhaltet detaillierte Anforderungsanalyse, das Aufteilen von großen Einträgen in kleinere, Schätzung von neuen Einträgen, und erneute Schätzung von existierenden Einträgen. Scrum schweigt sich darüber aus, wie diese Arbeit zu tun ist, aber eine häufig angewandte Technik ist ein fokussierter Workshop in der Mittte oder gegen Ende des Sprints, so dass das Team und der Product Owner and andere Stakeholder sich dieser Arbeit ohne Unterbrechung widmen können.

Die Verfeinerung bezieht sich nicht auf Einträge, die für den aktuellen Sprint ausgewählt wurden; sie ist für zukünftige Einträge, die wahrscheinlich für die nächsten ein oder zwei Sprints anstehen. Mit dieser Praktik wird das Sprint Planning ziemlich einfach, weil der Product Owner und das Scrum Team die Planung mit einem klaren, gut analysierten und sorgfältig geschätzten Satz von Backlog-Einträgen beginnen. Ein Zeichen dafür, dass der Verfeinerungs-Workshop nicht stattgefunden hat (oder nicht gut stattgefunden hat), ist, dass im Sprint Planning bedeutende Fragen, Entdeckungen oder Verwirrungen auftreten, und dass es sich unvollständig anfühlt. Die Planungsarbeit zieht sich dann oft in den Sprint hinein, was typischer Weise nicht wünschenswert ist.

# Sprint Review

**Zusammenfassung**: Überprüfung und Anpassung in Bezug auf die Funktionalitäten des Produktinkrements
**Teilnehmer**: Team, Product Owner, ScrumMaster. falls angemessen andere Stakeholder eingeladen durch den Product Owner.
**Dauer**: Timeboxed auf eine Stunde pro Sprintwoche.

Nachdem der Sprint beendet ist, findet das **Sprint Review** statt, bei dem der Sprint gereviewed wird. An diesem Meeting nehmen der Product Owner, die Teammitglieder und der ScrumMaster teil. Zusätzlich können Kunden, User, Stakeholder, Experten, Führungskräfte und jeder, der sonst ein Interesse an dem Produkt hat, teilnehmen. Für einen zweiwöchigen Sprint beträgt die maximale Dauer des Reviews zwei Stunden. Jeder Teilnehmer hat das Recht, Fragen zu stellen und Feedback zu geben.

Das Review wird oft fälschlicherweise als "Demo" bezeichnet, aber dieser Begriff spiegelt die Absicht des Meetings nicht wider. Eine Kernidee hinter Scrum ist *inspect and adapt*, also zu überprüfen und zu lernen, was gerade passiert und dann basierend auf Feedback sich weiterzuentwickeln, in sich wiederholenden Zyklen. Das Sprint Review ist eine Inspect-and-adapt-Aktivität für das *Produkt*. Für den Product Owner ist es der Zeitpunkt zu lernen, was gerade mit dem Produkt und dem Team passiert (das bedeutet, den Sprint zu reviewen); und für das Team ist es der Zeitpunkt darüber zu lernen, was gerade mit dem Product Owner und dem Markt geschieht. Folglich ist ein kritisches Element des Reviews ein tiefgehendes *Gespräch* zwischen dem Team und dem Product Owner, um die Situation zu verstehen, Ratschläge zu geben, und so weiter. Das Review beinhaltet definitiv, die aktuelle Software, die das Team während des Sprint entwickelt hat, live zu verwenden. Jedoch wenn der Fokus

des Reviews ausschließlich darauf liegt, das Produkt zu begutachten, anstatt einen verbalten Austausch zu haben, dann existiert ein Ungleichgewicht.

Der "Live-Software" Teil des Sprint Reviews dreht sich nicht darum, dass das Team eine "Präsentation" gibt - im Sprint Review werden keine Folien gezeigt. Das Meeting ist dazu da, eine Inspektion der echten Software im Live-Betrieb zu sein, zum Beispiel in einer Entwicklungsumgebung. Im Meetingraum sind ein oder mehrere Computer vorhanden, an denen die Teilnehmer die Software im Live-Betrieb begutachten und benutzen können. Eine aktive Sitzung, in der echte Benutzer und der Product Owner eine hands-on Interaktion mit der Software vornehmen, sollte einer passiven Demo-Sitzung des Teams vorgezogen werden.

Das Ziel sollte sein, nicht mehr als 30 Minuten für die Vorbereitung des Sprint Reviews aufbringen zu müssen. Andernfalls deutet es darauf hin, dass etwas falsch läuft.

# Sprint Retrospective

**Zusammenfassung**: Überprüfung und Anpassung hinsichtlich des Prozesses und der Arbeitsumgebung.
**Teilnehmer**: Team, ScrumMaster, Product Owner (optional). Andere Stakeholder können vom Team eingeladen werden, aber dürfen ansonsten nicht teilnehmen.
**Dauer**: Timeboxed auf 45 Minuten pro Sprintwoche.

Das Sprint Review dreht sich darum, das *Produkt* zu inspizieren und anzupassen. Die **Sprint Retrospective**, die sich direkt an das Sprint Review anschliesst, dreht sich darum, *den Prozess und die Arbeitsumgebung* anzupassen. Sie ist eine Gelegenheit für das Team, all das, was funktioniert und was nicht funktioniert, zu diskutieren, und Änderungen zu beschließen, die es ausprobieren will. Manchmal kann der ScrumMaster als ein effektiver Moderator für die Retrospektive fungieren, aber es kann auch besser sein, eine neutrale Person außerhalb des Teams für die Moderation des Meetings zu finden; ein guter Ansatz für mehrere ScrumMaster ist, die Retrospektive des jeweiligen anderen zu moderieren. Dadurch entsteht ein Austausch über Teamgrenzen hinweg.

Für Sprint Retrospektiven existieren viele Techniken. Das Buch *Agile Retrospectives* (Derby, Larsen 2006) bietet einen nützlichen Katalog von Techniken.

Viele Teams halten ihre Retrospektiven fokussiert auf *Probleme* ab, und das ist nicht gut. Es kann dazu führen, dass Beteiligte die Retrospektiven als erdrückende oder negative Ereignisse ansehen. Stattdessen sollte jede Retrospektive auch auf Positives und Stärken fokussieren; für mehr detailliertere Tipps existieren viele Bücher zum Thema *appreciative inquiry*.

Retrospektiven können schnell langweilig werden, wenn immer die gleiche Analysetechnik verwendet wird; deshalb sollten im Laufe der Zeit verschiedene Techniken zum Einsatz kommen.

# Beginn des nächsten Sprints

Nach dem Sprint Review kann der Product Owner das Product Backlog mit neuen Einsichten aktualisieren - neue Einträge hinzufügen, veraltete entfernen oder bereits existierende überarbeiten. Der Product Owner ist dafür verantwortlich, dass sich diese Änderungen im Product Backlog widerspiegeln. Abb. 9 zeigt ein Beispiel für ein angepasstes Product Backlog.

Zwischen zwei Sprints gibt es keine Ruhezeit - normalerweise gehen Teams von der Sprint Retrospective an einem Nachmittag direkt zum Sprint Planning am folgenden Morgen über (oder nach dem Wochenende).

Eines der Prinzipien agiler Entwicklung ist "sustainable pace" (nachhaltiges Arbeitstempo), und nur solange Teams reguläre Arbeitszeiten auf einem vertretbaren Level einhalten, können sie diesen Zyklus unendlich lange wiederholen. Die Produktivität wächst über die Zeit, dadurch dass das Team seine Praktiken weiterentwickelt und Hürden für die Teamproduktivität entfernt werden, jedoch nicht dadurch, dass Überstunden gemacht werden oder zu Lasten der Qualität.

| | | | | Neue Schätzung in Sprint ... | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| **Priorität** | **Eintrag** | **Details (Wiki-URL)** | **ursprüngliche Schätzung** | **1** | **2** | **3** | **4** | **5** |
| 1 | Als Käufer möchte ich ein Buch in meinen Einkaufwagen legen. (siehe UI-Skizzen im Wiki) | ... | 5 | 0 | 0 | 0 | | |
| 2 | Als Käufer mächte ich ein Buch aus meinem Einkaufswagen entfernen können. | ... | 2 | 0 | 0 | 0 | | |
| 3 | Verbesserung der Performance bei der Transaktionsverarbeitung (see Zielperformancemetriken im Wiki) | ... | 13 | 13 | 0 | 0 | | |
| 4 | Untersuche Lösungen zur Beschleunigung der Kreditkartenvalidierung (siehe Zielperformancemetriken im Wiki) | ... | 20 | 20 | 20 | 0 | | |
| 5 | Alle Sever auf Apache 2.2.3 upgraden | ... | 13 | 13 | 13 | 13 | | |
| 6 | Diagnose und Fixen des Fehlers im Skript für die Bestellverarbeitung (Bugzilla ID 14823) | ... | 3 | 3 | 3 | 3 | | |
| 7 | Als Käufer möchte ich eine Wunschliste erstellen und speichern können. | ... | 40 | 40 | 40 | 40 | | |
| 8 | Als Käufer möchte ich Einträge auf meiner Wunschliste hinzufügen und löschen können. | ... | 20 | 20 | 20 | 20 | | |
| | ... | | ... | ... | | | | |
| | | | **537** | **580** | **570** | **500** | | |

*Abbildung 9. Überarbeitetes Product Backlog*

Sprints werden so lange durchgeführt, bis der Product Owner entscheidet, dass das Produkt fertig zum Release ist. Die perfekte Vision von Scrum ist, dass das Produkt am Ende jedes Sprints potenziell auslieferbar ist. Das impliziert, dass keine Nacharbeiten notwendig sind, wie zum Beispiel Testen und Dokumentation. Die Implikation ist, dass *alles* komplett *fertig* gestellt wird in jedem Sprint; so dass die Software tatsächlich direkt nach dem Sprint Review ausgeliefert oder deployed werden kann. Allerdings besitzen viele Organisationen schlechte Entwicklungspraktiken, Tools und Infrastruktur und können damit diese perfekte Vision nicht erreichen. In diesen Firmen gibt es deshalb die Notwendigkeit eines "Release Sprints", um die verbleibende Arbeit zu erledigen. Falls ein "Release Sprint" notwendig ist, ist das ein notwendiges Übel, und die Aufgabe der Organisation liegt darin, ihre Praktiken zu verbessern, damit "Release Sprints" nicht länger benötigt werden.

# Releases managen

Manchmal wird die Frage gestellt, wie in einem iterativen Modell Langzeitplanung geschehen kann. Hierbei gibt es zwei Fälle zu betrachten: (1) das erste Release eines neuen Produkts und (2) ein weiteres Release eines bereits bestehenden Produkts.

Im Fall des neuen Produkts oder eines existierenden Produkts, das gerade erst beginnt, mit Scrum weiterentwickelt zu werden, muss das initiale Product Backlog vor dem ersten Sprint durch den Product Owner und das Team verfeinert werden, damit ein Product Backlog im Scrum-Sinn entsteht. Dies kann ein paar Tage oder eine Woche dauern und umfasst einen Workshop (manchmal Initial Product Backlog Creation oder Release Planning genannt), ein wenig detaillierte Anforderungsanalyse und Abschätzung aller für das erste Release identifizierten Einträge.

Überraschender Weise sollte es im Fall eines etablierten Produkts mit einem bestehenden Product Backlog in Scrum keine Notwendigkeit geben, ein spezielles oder umfangreiches Release Planning für das kommende Release stattfinden zu lassen. Warum? Weil der Product Owner und das Team in jedem Sprint an der Verfeinerung des Product Backlogs arbeiten (fünf bis zehn Prozent der Zeit in jedem Sprint), und damit kontinuierlich für die Zukunft vorarbeiten sollten. Dieser Modus der *kontinuierlichen Produktentwicklung* umgeht die Notwendigkeit für die dramatisch punktuellen Phasen des Vorbereitens, Ausführens und Abschließens, die in einem traditionellen sequentiellen Entwicklungszyklus zu sehen sind.

Während eines initialen Product Backlog Verfeinerungsworkshops und während der kontinuierlichen Verfeinerung in jedem Sprint werden das Team und der Product Owner das Release Planning, die

Verfeinerung der Schätzungen, der Prioritäten und des Inhalts vornehmen, um ihren Wissensstand widerzuspiegeln.

Einige Releases sind Datums-getrieben; zum Beispiel: "Wir werden Version 2.0 unseres Projekts bei der Messe am 10. November veröffentlichen." In dieser Situation wird das Team in der zur Verfügung stehenden Zeit so viele Sprints wie möglich fertig stellen (und so viele Features wie möglich einbauen). Andere Produkte erfordern, bevor sie fertig genannt werden können, dass bestimmte Features eingebaut wurden, und das Produkt wird nicht veröffentlicht, bis diese Anforderungen erfüllt sind, wie lange auch immer das dauern mag. Da Scrum hervorhebt, in jedem Sprint potenziell auslieferbaren Code zu produzieren, kann der Product Owner entscheiden, Zwischenreleases zu erstellen, um dem Kunden früher zu ermöglichen, an den Vorteilen der fertig gestellten Arbeit teilzuhaben.

Da man nicht alles von Beginn an wissen kann, liegt der Fokus darauf, einen Plan zu erstellen und zu verfeinern, um dem Release eine grobe Richtung zu geben, und klarzustellen, wie Trade-Off Entscheidungen gefällt werden sollen (beispielsweise Inhalt vs. Termin). Das kann man sich als eine Roadmap zum endgültigen Ziel vorstellen; welche Route man genau nimmt und welche Entscheidungen man auf dem Weg trifft, kann während der Reise festgestellt werden.

*Das Ziel ist wichtiger als der Weg.*

Die meisten Product Owner entscheiden sich für einen Release-Ansatz. Zum Beispiel entscheiden sie sich für einen Release-Termin und arbeiten dann mit dem Team zusammen, um die Product Backlog Einträge abzuschätzen, die bis zu diesem Datum fertig gestellt werden können. Die Einträge, die voraussichtlich im aktuellen Release enthalten sein werden, werden manchmal *Release-Einträge* genannt. In Situationen, in denen eine Zusage zu einem "festen Preis, festen Datum und festen Umfang" notwendig ist - zum Beispiel bei Auftragsentwicklung - muss bei einem oder mehreren dieser Parameter ein eingebauter Puffer eingeplant werden, um Unsicherheit und Änderung zuzulassen; in dieser Hinsicht ist Scrum nicht anders als andere Ansätze.

# Anwendungs- oder Produktfokus

Für Anwendungen oder Produkte - entweder für den allgemeinen Markt oder auch für interne Verwendung innerhalb einer Firma - bewegt Scrum Gruppen weg von dem alten *Projekt*-zentrierten Modell hin zu einem Modell der *kontinuierlichen Anwendungs-/Produktentwicklung.* Es gibt nicht länger ein Projekt mit einem Anfang, einer Mitte und einem Ende. Und deshalb gibt es auch keinen traditionellen Projektmanager. Stattdessen gibt es einfach einen stabilen Product Owner and ein langlebiges selbstmanagendes Team, das in "unendlichen" Serien von Sprints fester Länge zusammenarbeitet, bis das Produkt oder die Anwendung abgekündigt wird. Alle notwendigen Arbeiten, um das "Projekt" zu managen, werden vom Team und dem Product Owner, der ein interner Geschäftskunde oder Produktmanager ist, übernommen. Das Team wird nicht von einem IT Manager oder jemand aus dem Project Management Office gemanaged.

Scrum kann auch für echte *Projekte* verwendet werden, die einmalige Initiativen sind (im Gegensatz zur Entwicklung einer langlebigen Applikation); auch in diesem Fall erledigen das Team und der Product Owner das Project Management.

Was passiert, wenn es nicht genügend Arbeit von einer oder mehreren existierenden Anwendungen gibt, um ein dediziertes langlebiges Team für jede Anwendung zu ermöglichen? In diesem Fall kann ein stabiles langlebiges Team Arbeit von einer Anwendung in einem Sprint und dann Arbeit einer anderen Anwendung im nächsten Sprint übernehmen; in diesem Fall sind Sprints häufig recht kurz, beispielsweise eine Woche.

Manchmal gibt es selbst für dieses Vorgehen nicht genügend neue Arbeiten. Dann kann das Team Arbeiten von mehreren Anwendungen im selben Sprint übernehmen; diesen Ansatz sollte man aber mit Vorsicht genießen, da es in unproduktives Multitasking über mehrere Anwendungen degenerieren kann. Ein Grundsatz für Produktivität in Scrum besteht darin, dass das Team fokussiert an einem Produkt oder Anwendung in einem Sprint arbeitet.

# Häufige Herausforderungen

Scrum ist nicht nur ein konkreter Satz an Praktiken - mehr noch, und noch wichtiger, Scrum ist ein Framework, das Transparenz herstellt und einen Mechanismus, der "Inspect and Adapt" zulässt. Scrum funktioniert, indem es Dysfunktionen und Hindernisse sichtbar macht, die die Effektivität des Product Owners und des Teams beeinträchtigen, so dass diese adressiert werden können. Zum Beispiel kennt der Product Owner vielleicht nicht den Markt, die Features oder wie deren relativer Geschäftswert bestimmt wird. Oder das Team ist unfähig zur Aufwandsabschätzung oder zur Entwicklungsarbeit.

Das Scrum Framework wird schnell diese Schwächen offenbaren. Scrum löst diese Probleme in der Entwicklung nicht; Scrum macht diese nur auf schmerzhafte Weise sichtbar und bietet ein Framework für alle Beteiligten, um Lösungswege für diese Probleme in kurzen Zyklen und mit kleinen Verbesserungsexperimenten zu erforschen.

Nehmen wir an, das Team schafft es aufgrund von schlechter Aufgabenanalyse und Schätzfähigkeiten nicht, im ersten Sprint das zu liefern, was es vorhergesagt hat. Für das Team ist das ein Fehlschlag. In Wirklichkeit aber ist diese Erfahrung der notwendige erste Schritt, um realistischer und bedachter im Umgang mit seinen Vorhersagen zu werden. Dieses Muster von Scrum - Dysfunktionen sichtbar zu machen und dem Team die Möglichkeit zu geben, etwas dagegen zu tun - ist der Grundmechanismus, der die signifikantesten Vorteile bewirkt, die Teams beim Einsatz von Scrum erfahren.

Wenn eine Scrum Praktik schwierig erscheint, ist ein häufiger Fehler, Scrum selbst anzupassen. Zum Beispiel könnten Teams, die Probleme damit haben, am Ende des Sprints auszuliefern, die Sprintlänge anpassbar machen, damit ihre Zeit niemals abläuft - und in diesem Ablauf stellen sie sicher, dass sie niemals lernen müssen, wie sie besser schätzen und ihre Zeit einteilen können. Auf diese Art können Firmen ohne Coaching und ohne die Unterstützung eines erfahrenen ScrumMasters Scrum in ein Spiegelbild ihrer eigenen Schwächen und Dysfunktionen verwandeln, und damit den echten Vorteil von Scrum unterwandern: das Gute und das Schlechte sichtbar machen, und die Organisation vor die Wahl stellen, sich selbst auf das nächste Level zu bringen.

Ein weiterer häufiger Fehler ist es anzunehmen, dass eine Praktik nicht erwünscht oder verboten ist, nur weil Scrum sie nicht direkt verlangt. Zum Beispiel verlangt Scrum nicht, dass der Product Owner eine Langzeitstrategie für sein Produkt vorlegt; ebenfalls verlangt Scrum nicht, dass Entwickler die Ratschläge von erfahreneren Entwicklern zu komplexen technischen Problemen einholen. Scrum überlässt es den Beteiligten, die jeweils richtigen Entscheidungen zu treffen; und in den meisten Fällen sind beide dieser Praktiken sehr empfehlenswert (zusammen mit so vielen anderen).

Ebenfalls sollte man mit Managern vorsichtig sein, die ihren Teams Scrum aufzwängen; Scrum dreht sich darum, Teams den Freiraum und die Hilfsmittel zu geben, um sich selber zu managen, und das von oben herab zu befehlen, ist kein Rezept zum Erfolg. Ein besserer Ansatz könnte sein, mit einem Team zu beginnen, Scrum von einem Kollegen oder Manager kennenzulernen, ein ausgiebiges professionelles Training zu erhalten und dann als Team eine Entscheidung zu treffen, ob die Praktiken für einen begrenzten Zeitraum vertrauensvoll ausprobiert werden können; am Ende dieses Zeitraums wertet das Team seine Erfahrungen aus und entscheidet darüber, ob mit Scrum weitergemacht werden soll.

Die gute Nachricht ist, dass, obwohl der erste Sprint normalerweise sehr herausfordernd für das Team ist, die Vorteile von Scrum am Ende des Sprints sichtbar sind. Das führt dazu, dass viele neue Scrum Teams einhellig sagen: "Scrum ist schwer, aber es ist auf jeden Fall wesentlich besser als das, was wir davor getan haben!"

# Anhang A: Literaturhinweise

Es gibt eine Menge Material zu Scrum. In dieser Literatursektion würden wir gerne auf weiteres Onlinematerial und ein paar Bücher verweisen.

**Online Material:**

- The Lean Primer - An introduction to Lean Thinking, an important influence to Scrum. http://www.leanprimer.com
- The Distributed Scrum Primer - Additional tips for teams who aren't co-located. http://www.goodagile.com/distributedscrumprimer/
- The ScrumMaster Checklist - A list of question that good ScrumMasters use. http://www.scrummasterchecklist.org/
- Feature Team Primer - Scaling Scrum with Feature Teams, http://www.featureteams.org
- The Agile Atlas - Core Scrum. ScrumAlliance description of Scrum. http://agileatlas.org/atlas/scrum
- Scrum Guide - Scrum.org description of Scrum. http://www.scrum.org/Scrum-Guides
- Agile Contracts Primer - How to make Scrum-friendly contracts. http://www.agilecontracts.org/

**Bücher:**

- Leading Teams - Richard Hackman
- Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum - Craig Larman, Bas Vodde
- Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum - Craig Larman, Bas Vodde
- Agile Project Management with Scrum - Ken Schwaber
- Succeeding with Agile: Software Development using Scrum - Mike Cohn

# Anhang B: Glossar

**Burndown**
Der Trend von verbleibender Arbeit in einem Sprint, einem Release oder einem Produkt über die Zeit. Die Quelle für die Rohdaten sind das Sprint Backlog oder das Product Backlog mit der verbleibenden Arbeit über die vertikale Achse und dem Zeitraum (Tage im Sprint oder Sprints) auf der horizontalen Achse.

**Daily Scrum**

Ein kurzes Meeting, das jedes Team täglich abhält, und während dessen die Teammitglieder ihre Arbeit betrachten, ihre Arbeit und den Fortschritt synchronisieren und Hindernisse an den Scrum Master geben, um diese zu beseitigen. Um anfallende Arbeiten zur Optimierung des Sprints anzupassen können Folgemeetings nach dem Daily Scrum Meeting stattfinden.

**Entwicklungsteam**

Ein anderer Name für die Rolle des Teams.

**Done**

Vollständig wie von allen Beteiligten vereinbart und übereinstimmend mit den Standards, Konventionen und Guidelines der Organisation. Wenn etwas als "done" beim Sprint Review Meeting vorgestellt wird, muss es dieser Definition genügen.

**Geschätzer verbleibender Aufwand (Sprint Backlog Einträge)**

Die Anzahl an Stunden, die ein Teammitglied schätzt, noch an einer Task arbeiten zu müssen. Diese Schätzung wird am Ende des Tages aktualisiert, wenn an einer Sprint Backlog Task gearbeitet wurde. Die Schätzung spiegelt den vollständig verbleibenden geschätzten Restaufwand wider, unabhängig von der Anzahl an Leuten, die die Arbeit durchführen.

**Inkrement**
Produktfunktionalität, die während jedes Sprints vom Team entwickelt wird, die potenziell auslieferbar ist oder für die Stakeholder des Product Owners nützlich ist.

**Potenziell auslieferbares Produktinkrement**
Ein vollständiger Schnitt durch das ganze Produkt oder System, der vom Product Owner oder den Stakeholdern eingesetzt werden kann, wenn sie es denn möchten.

**Sprint**

Eine Iteration, oder ein sich wiederholdender Zyklus an vergleichbarer Arbeit, der ein Inkrement eines Produkts oder Systems produziert. Nicht länger als ein Monat und meistens mehr als eine Woche. Die Länge ist fest über die Gesamtarbeit und alle Teams, die am gleichen System oder Produkt arbeiten, verwenden die gleiche Sprintlänge.

**Product Backlog**
Eine priorisierte Liste von Anforderungen mit geschätzten Zeiten, um diese in vollständig Produktfunktionalitäten zu verwandeln. Schätzungen sind um so präziser, je höher ein Eintrag im

Produkt Backlog priorisiert ist. Die Liste ist im ständigen Fluss, angepasst an die Geschäftsbedingungen oder die technologischen Veränderungen.

**Product Backlog Eintrag**
Funktionale Anforderungen, nicht-funktionale Anforderungen und Probleme, priorisiert nach der Wichtigkeit für das Business und nach Abhängigkeiten, außerdem geschätzt. Der Detailgrad der Schätzung hängt von der Priorität und der Granularität des Product Backlog Items ab, wobei die am höchsten priorisierten Items für den nächsten Sprint ausgewählt werden können, da sie sehr granular und detailliert sind.

**Product Owner**
Die Person, die für das Managen des Product Backlogs verantwortlich ist, um den Geschäftswert des Produkts zu maximieren. Der Product Owner ist dafür verantwortlich, die Interessen jedes Stakeholders innerhalb des Projekts und des daraus resultierenden Produkts zu repräsentieren.

**Scrum**
Kein Akronym, aber ein Mechanismus im Rugby, um einen Ball außerhalb des Spiels wieder ins Spiel zurückzuholen.

**ScrumMaster**
Die Person, die für den Scrum Prozess, dessen korrekte Implementation und die Maximierung seiner Vorteile verantwortlich ist.

**Sprint Backlog**
Eine Liste der Arbeit des Teams innerhalb eines Sprints. Diese ist häufig zergliedert in einen Satz von detaillierteren Aufgaben. Die Liste entsteht während des Sprint Plannings und kann vom Team während des Sprints dadurch angepasst werden, dass Einträge entfernt werden oder neue Aufgaben hinzugefügt werden, wenn dieses notwendig wird. Jede Aufgabe im Sprint Backlog wird während des Sprints verfolgt und macht den verbleibenden Aufwand transparent.

**Sprint Backlog Aufgabe**
Eine der Aufgaben, die das Team oder ein Teammitglieder als notwendig erklärt hat, um ein zugesagtes Product Backlog Item in eine Systemfunktionlität zu verwandeln.

**Sprint Planning Meeting**
Ein auf vier Stunden (bei zwei-wöchigen Sprints) zeitlich begrenztes Meeting. Das Meeting wird unterteilt in zwei zweistündige Segmente, die ebenfalls zeitlich begrenzt sind. Während des ersten Teils präsentiert der Product Owner die Product Backlog Items mit der höchsten Priorität dem Team. Das Team und der Product Owner arbeiten dann zusammen, um dem Team bestimmen zu helfen, wie viele Product Backlog Items es im kommenden Sprint in Funktionalität verwandeln kann. Im zweiten Teil plant das Team, indem es die Arbeit designed und zerteilt, um zu verstehen, wie es das Sprintziel erreichen wird.

**Sprint Retrospective Meeting**
Ein vom ScrumMaster moderiertes Meeting, bei dem das gesamte Team den gerade beendeten Sprint diskutiert und entscheidet, was geändert werden soll, damit der nächste Sprint angenehmer oder produktiver werden kann.

**Sprint Review Meeting**
Ein zeitlich begrenztes zwei Stunden Meeting (bei einer Sprintlänge von zwei Wochen) am Ende jedes Sprints, in dem das Team mit dem Product Owner und den Stakeholdern zusammenkommt und den Output des Sprints inspiziert. Dieses beginnt normalerweise mit einem Review der vervollständigten Product Backlog Items, einer Diskussion von Möglichkeiten, Randbedingungen und Risiken und einer Diskussion darüber, welche Dinge am besten als nächstes angegangen werden sollten (diese können potenziell zu weiteren Änderungen am Product Backlog führen). Nur vollständige Funktionalität kann hier gezeigt werden.

**Stakeholder**
Jemand mit einem Interesse am Ergebnis des Projekt, entweder weil er es bezahlt, es einsetzen will oder davon beeinträchtigt ist.

**Team**
Eine funktional übergreifende Gruppe von Leuten, die dafür verantwortlich ist, sich selbst zu managen, um ein Produktinkrement in jedem Sprint zu entwickeln.

**Timebox**
Ein Zeitrahmen, der nicht verlängert werden kann, und innerhalb dessen ein Event oder Meeting stattfindet. Zum Beispiel hat, das Daily Scrum Meeting eine Timebox von fünfzehn Minuten und wird am Ende der fünfzehn Minuten beendet. Für Meetings kann diese Timebox verlängert werden. Für Sprints ist sie fix und kann nicht verlängert werden.