Scrum

aus Wikipedia, der freien Enzyklopädie

Scrum (engl. „Gedränge“) ist ein Vorgehensmodell der Softwaretechnik. Der Ansatz von Scrum ist empirisch, inkrementell und iterativ. Er beruht auf der Ansicht, dass die meisten modernen Entwicklungsprojekte zu komplex sind, um durchgängig planbar zu sein. Scrum versucht, die Komplexität durch drei Prinzipien zu reduzieren:[1]

1. *Transparenz:* Der Fortschritt und die Hindernisse eines Projektes werden täglich und für alle sichtbar festgehalten.
2. *Überprüfung:* In regelmäßigen Abständen werden Produktfunktionalitäten geliefert und beurteilt.
3. *Anpassung:* Die Anforderungen an das Produkt werden nicht ein und für alle Mal festgelegt, sondern nach jeder Lieferung neu bewertet und bei Bedarf angepasst.

Das Scrum Framework besteht aus drei Rollen mit jeweils verschiedenen Verantwortungen: Product Owner, Entwicklungsteam und ScrumMaster, sowie drei Zeremonien: Sprint Planning, Sprint Review und Daily Scrum, und drei Artefakten: Product Backlog, Sprint Backlog und Burndown Chart.[2] Alle Aktivitäten in Scrum sind zeitlich fest beschränkt („timeboxed“) und sollen zur unmittelbaren Vorbereitung der jeweils nächsten Aktivität dienen.

Ziel ist die schnelle, kostengünstige und qualitativ hochwertige Fertigstellung eines Produktes, das einer zu Beginn formulierten Vision entsprechen soll. Die Umsetzung der Vision in das fertige Produkt erfolgt nicht durch die Aufstellung möglichst detaillierter Anforderungslisten (vgl. Lastenheft/Pflichtenheft), sondern in zwei bis vier Wochen langen, sich wiederholenden Intervallen, so genannten *Sprints*. Meist werden User Storys eingesetzt um Lasten- Pflichtenhefte zu ersetzen. Am Ende eines jeden Sprint steht bei Scrum die Lieferung einer fertigen Software-Funktionalität (das Produkt-Inkrement). Die neu entwickelte Funktionalität sollte in einem Zustand sein, dass sie an den Kunden ausgeliefert werden kann (*potential shippable code* oder *usable software*).[3] Scrum verkörpert die Werte des Agilen Manifests.

Inhaltsverzeichnis

Geschichte und Grundlegendes

Die Anfänge von Scrum lassen sich auf das Jahr 1990 zurückverfolgen. Damals schuf *Jeff Sutherland*[4] in einem Projekt für die Guiness Peat Aviation eine neue Rolle für die damaligen Projektleiter. Diese wurden zu Teammitgliedern und ihre Rolle war eher die eines Moderators als die eines Managers. Ken Schwaber veröffentlichte auf der OOPSLA 1996 den ersten Konferenzbeitrag über Scrum. Darin schrieb Schwaber: „Scrum akzeptiert, dass der Entwicklungsprozess nicht vorherzusehen ist. Das Produkt ist die bestmögliche Software unter Berücksichtigung der Kosten, der Funktionalität, der Zeit und der Qualität.“[5]

2001 veröffentlichten Ken Schwaber und *Mike Beedle*[6] mit „Agile Software Development with Scrum“ das erste Buch über Scrum. 2003 folgte Schwabers „Agile Project Management with Scrum“. 2003 war auch das Jahr, in dem die ersten zertifizierten ScrumMasters von Schwaber ausgebildet wurden. 2007 erschien schließlich Ken Schwabers drittes Buch, „The Enterprise and Scrum“. Darin geht es nicht mehr bloß um die Einführung von Scrum in Software-Entwicklungsteams, sondern um die Ausweitung auf das gesamte Unternehmen. Schwaber betont in dem Buch, dass Scrum Probleme nicht lösen kann, sehr wohl aber diese aufzudecken und zu lokalisieren vermag. Scrum kann daher als Aufforderung zum unternehmensweiten Umdenken verstanden werden.[7]

Parallelen zu Scrum finden sich in der so genannten Schlanken Produktion (engl. *lean production*), die ihren Ursprung in japanischen Unternehmen hat und eine bessere Wertschöpfung unter anderem durch stärkeres Teamworking anstrebt. Nonaka und Takeuchi führen den Erfolg solcher Unternehmen letztlich auf ein gelungenes Wissensmanagement zurück. Im westlichen Verständnis sei die Ressource „Wissen“ im Unternehmen auf Worte und Zahlen begrenzt. Wissen kann nach diesem Verständnis erworben oder antrainiert werden. Japanische Firmen hingegen sehen in dieser Art von Wissen nur die Spitze eines Eisbergs. Für sie ist Wissen in erster Linie implizit („tacit“). Dieses implizite Wissen ist subjektiv und intuitiv, es enthält unser Bild der Realität und unsere Vision für die Zukunft. Während explizites Wissen sich leicht darstellen und verarbeiten lässt, ist dies bei implizitem Wissen deutlich schwerer. Unternehmen wie Toyota oder Canon profitieren vom impliziten Wissen ihrer Mitarbeiter, indem sie hohen Wert auf die Interaktion zwischen ihren Mitarbeitern legen.[8]

Scrum lässt sich in diesem Zusammenhang als Gegenentwurf zur Befehls-und-Kontroll-Organisation verstehen, in der Mitarbeiter möglichst genaue Arbeitsanweisungen zu erhalten haben. Statt dessen baut Scrum auf hoch qualifizierte, interdisziplinär besetzte Entwicklungsteams, die zwar eine klare Zielvorgabe bekommen, für die Umsetzung jedoch allein zuständig sind. Dadurch bekommen die Entwicklungsteams den nötigen Freiraum, um ihr Wissens- und Kreativitätspotenzial in Eigenregie zur Entfaltung zu bringen.[9]

Scrum verkörpert die Werte der agilen Software-Entwicklung, die 2001 im Agilen Manifest von Ken Schwaber, Jeff Sutherland und anderen formuliert wurden:

1. *Individuen* und *Interaktionen* gelten mehr als *Prozesse* und *Tools*.
2. *Funktionierende Programme* gelten mehr als *ausführliche Dokumentation*.
3. Die stetige *Zusammenarbeit mit dem Kunden* steht über *Verträgen*.
4. Der Mut und die Offenheit für *Änderungen* steht über dem *Befolgen* eines festgelegten *Plans*.

Rollen

Bei Scrum gibt es insgesamt sechs Rollen, von denen drei (Entwicklungsteam, Product Owner und ScrumMaster) zum Scrum-Team (nicht zu verwechseln mit dem Entwicklungsteam) gehören.[10] Die übrigen drei (Management, Customer und User) sind externe Rollen, aber dennoch für das Gelingen von Scrum von großer Bedeutung.

Product Owner

Der Product Owner ist für die strategische Produktentwicklung zuständig. Seine Verantwortung beinhaltet die Konzeption und Mitteilung einer klaren Produktvision, die Festlegung und Priorisierung der jeweils zu entwickelnden Produkteigenschaften sowie die Entscheidung darüber, ob die vom Entwicklungsteam am Ende jedes Sprints gelieferte Funktionalität akzeptabel ist. Der Product Owner gestaltet das Produkt mit dem Ziel, den wirtschaftlichen Nutzen für das eigene Unternehmen zu maximieren.[11] Ihm allein obliegt die Entscheidung über Auslieferungszeitpunkt, Funktionalität und Kosten.

Zur Festlegung der Produkteigenschaften verwendet der Product Owner das Product Backlog. Darin trägt er in Zusammenarbeit mit dem Entwicklungsteam die so genannten User Stories ein, welche Funktionalitäten aus der Sicht des Benutzers beschreiben sollen (siehe Abschnitt Product Backlog). Zudem priorisiert er regelmäßig die eingetragenen User Stories unter Berücksichtigung ihres wirtschaftlichen Nutzen. Die Festlegungen des Product Owners sind verbindlich und dürfen nicht überstimmt werden.[12]

Als „oberster Produktentwickler“ hält der Product Owner Rücksprache mit dem Kunden und versucht, dessen Bedürfnisse und Wünsche in die Produktentwicklung einfließen zu lassen. Allerdings ist der Product Owner kein Vertreter des Kunden, da sein oberstes Ziel die Wirtschaftlichkeit des Produktes für die eigene Firma sein muss. Zudem ist der Product Owner dafür zuständig, die Interessen und Anforderungen der Interessengruppen innerhalb des eigenen Unternehmens (z.B. Marketing, Sales, IT und Service) zu bündeln.

Bei der Implementierung von Scrum passiert es häufig, dass Product Owner nicht bevollmächtigt sind, die notwendigen Entscheidungen verbindlich zu treffen. Ein weiteres Problem aus der Praxiserfahrung ist die Überlastung der Product Owner mit fremden Aufgaben.[13]

Entwicklungsteam

Das Entwicklungsteam ist für die Lieferung der Produktfunktionalitäten in der vom Product Owner gewünschten Reihenfolge verantwortlich. Zudem trägt das Entwicklungsteam die Verantwortung für die Einhaltung der zuvor vereinbarten Qualitätsstandards. Das Entwicklungsteam ist insofern autonom, als es allein entscheidet, wie viele Funktionalitäten es in einem Sprint liefern will. Hierzu geht es im Sprint

Planning 1 ein „Commitment“ ein, in dem es sich zum Erreichen der freiwillig ausgewählten Ziele verpflichtet.[14]

Ebenso sollte ein Entwicklungsteam in der Lage sein, das Ziel eines jeweiligen Sprints ohne größere Abhängigkeiten von außen zu erreichen. Deshalb ist eine interdisziplinäre Besetzung des Entwicklungsteams wichtig (z.B. ein Team mit Architekt, Entwickler, Tester, Dokumentationsexperte, Datenbankexperte, usw.). In Scrum tritt das Entwicklungsteam immer als Team auf - gute und schlechte Ergebnisse werden nie auf einzelne Teammitglieder, sondern immer auf das Entwicklungsteam als Einheit zurückgeführt. Das ideale Teammitglied ist sowohl Spezialist als auch Generalist, damit es Teamkollegen beim Erreichen des gemeinsamen Ziels helfen kann.[15]

Ein Entwicklungsteam besteht idealerweise aus 7±2 Personen, um die Bildung von sich abschottenden Kleingruppen zu vermeiden.[16]

Zu den weiteren Aufgaben eines Entwicklungsteams zählt die Schätzung des Umfangs einer jeden User Story (im Estimation Meeting) sowie das Herunterbrechen der für einen Sprint ausgewählten User Stories in technische Arbeitsschritte (so genannte Tasks), deren Bearbeitung in der Regel nicht länger als einen Tag dauert.

In der Implementierungsphase haben Team-Mitglieder bisweilen Schwierigkeiten, die interdisziplinären Anforderungen zu akzeptieren. So kann z.B. ein Entwickler nicht verstehen, warum er nun auch noch die Arbeit eines Testers leisten soll. Hinter diesen Anforderungen steht jedoch der Gedanke, dass ein starkes Entwicklungsteam den mannigfachen Unwägbarkeiten eines Projektes wesentlich besser gewachsen ist als eine Sammlung individueller Talente. Falls beispielsweise jemand mit einer Aufgabe nicht zurechtkommt, kann ein anderer aushelfen und so die Einhaltung des Sprint-Ziels gewährleisten. Und fällt jemand aus privaten Gründe für einige Zeit aus, ist ein interdisziplinär aufgestelltes Entwicklungsteam besser in der Lage, die fehlende Expertise zu kompensieren.

ScrumMaster

Der ScrumMaster ist dafür verantwortlich, dass Scrum gelingt. Dazu arbeitet er mit dem Entwicklungsteam zusammen, gehört aber selber nicht zu ihm. Er führt die Scrum-Regeln ein und überprüft deren Einhaltung, moderiert die Meetings und kümmert sich um jede Störung des Scrum-Prozesses. Arbeitsmittel des ScrumMasters ist das Impediment Backlog. Darin festgehalten sind alle Hindernisse („Impediments“), die das Entwicklungsteam an effektiver Arbeit hindern. Der ScrumMaster ist verantwortlich für die Beseitigung dieser Hindernisse. Dazu gehören Probleme im Entwicklungsteam (z.B. mangelnde Kommunikation und Zusammenarbeit, persönliche Konflikte) und im Scrum-Team (z.B. in der Zusammenarbeit zwischen Product Owner und Entwicklungsteam) sowie Störungen von außen (z.B. Aufforderungen der Fachabteilung zur Bearbeitung zusätzlicher Aufgaben während eines Sprints).

Ein ScrumMaster ist gegenüber dem Entwicklungsteam eine Führungskraft, aber kein Vorgesetzter. Weder kann er einzelnen Team-Mitgliedern konkrete Arbeitsanweisungen geben, noch kann er diese beurteilen oder disziplinarisch belangen. Seine Rolle ist die eines *Servant Leaders* - der ScrumMaster sichert sich Anerkennung und Gefolgschaft, indem er sich der Bedürfnisse der Team-Mitglieder annimmt.[17]

Im Idealfall wird der ScrumMaster vom Entwicklungsteam gewählt. Zu Beginn einer Scrum-Implementierung ist der ScrumMaster ein Vollzeitjob, da die Umstellung der Abläufe, das Zusammenwachsen des Teams und das Einlernen der Rollen meist sehr aufwändig sind. Ist Scrum erst einmal wohl etabliert, kann der ScrumMaster seine Rolle als Change-Manager wahrnehmen. Er hat dann die Zeit und auch die nötige Erfahrung, um Scrum im Unternehmen bekannt zu machen und die Akzeptanz dafür zu steigern.[18]

Customer

Mit Customer (Kunde) ist der Auftraggeber gemeint, dem das Produkt nach Fertigstellung zur Verfügung gestellt wird. Customer kann je nach Situation sowohl die interne Fachabteilung als auch ein externer Kunde sein. Es ist Aufgabe des Product Owners, seinen Customer durch Lieferung des Wunschproduktes zu begeistern. Deshalb sollten Product Owner und Customer für die Dauer des Projektes im engen Austausch stehen.[19] Vor Beginn der Entwicklung sollte der Product Owner ein möglichst genaues Verständnis von der Wunschvorstellung seines Customers gewinnen. Und der Customer sollte schon nach den ersten Sprints Gelegenheit haben, sich die neuen Funktionalitäten anzuschauen und hierzu Feedback zu geben. Schließlich ist Scrum explizit darauf ausgerichtet, dass sich die Liste der gewünschten Produktfunktionalitäten während der Entwicklungsphase ändert.

User

Die Rolle des Users (Anwender) umfasst diejenigen Personen, die das Produkt benutzen. Der User kann, muss aber nicht zugleich der Kunde sein. Die Rolle des Users ist für das Team von besonderer Bedeutung, denn nur der User kann das Produkt aus der Perspektive des Benutzers beurteilen, der die eigentliche Zielgruppe darstellt. Der User sollte beim Sprint Planning 1 sowie beim Sprint Review hinzugezogen werden, um ihn das Produkt ausprobieren zu lassen und Feedback darüber einzuholen.[20]

Management

Das Management gehört ebenso wenig zum Scrum-Team wie der User und der Customer. Doch trägt es Verantwortung dafür, dass die Rahmenbedingungen für gelingendes Scrum stimmen. Dazu gehören die Bereitstellung von materiellen Ressourcen (Räumlichkeiten, Arbeitsmittel), aber auch genereller die Unterstützung für den eingeschlagenen Kurs. Das Management ist dafür verantwortlich, das Scrum-Team vor externen Arbeitsanforderungen zu schützen, adäquate personelle Besetzungen zu finden sowie den ScrumMaster dabei zu unterstützen, Hindernisse auszuräumen.[21]

Zusammenspiel

Bei der Rollenaufteilung ist zu berücksichtigen, dass das Team sich selbst organisiert. Weder der Product Owner noch der ScrumMaster geben vor, welches Teammitglied wann was macht und wer mit wem zusammenarbeitet. Der ScrumMaster hat die Pflicht, darauf zu achten, dass keiner in diesen Selbstorganisationsprozess eingreift und das Team stört oder Verantwortlichkeiten an sich nimmt, die ihm nicht zustehen.

Meetings

Sprint Planning Meeting 1

In diesem Treffen stellt der Product Owner seinem Entwicklungsteam die im Product Backlog festgehaltenen User Stories vor (in der zuvor priorisierten Reihenfolge). Damit sich das Entwicklungsteam ein klares Bild der einzelnen User Stories verschaffen kann, müssen die jeweiligen Anforderungen mit dem Product Owner geklärt werden und vom Team schriftlich festgehalten werden. Neben dem Product Owner kann hier auch der User wertvolle Informationen liefern, indem er z.B. dem Entwicklungsteam erklärt, wie er sich eine Funktionalität im alltäglichen Gebrauch wünscht. Außerdem einigt sich der Product Owner mit dem Entwicklungsteam hinsichtlich der Kriterien, die am Ende des Sprints darüber entscheiden, ob die neue Funktionalität fertig ist oder nicht.

Ziel muss immer die Lieferung von „usable software“ sein: Einer Funktionalität also, die hinreichend getestet und integriert ist, um für den Benutzer freigegeben werden zu können. Bei der Festlegung der Abnahmebedingungen spielen Akzeptanzkriterien (z.B. Einsatz oder Usability), Aspekte (z.B. Performance)

und Testfälle (z.B. User Acceptance Test) eine wichtige Rolle. Anhand solcher Festlegungen kann am Ende des Sprints klar beurteilt werden, ob die gelieferte Funktionalität tatsächlich fertig ist. Das im Sprint Planning 1 festgelegte Ziel spezifiziert somit die Abnahmebedingungen für das Review.

Nachdem die Klärungen erfolgt sind, fragt der ScrumMaster das Entwicklungsteam, wie viele User Stories es im bevorstehenden Sprint erledigen will. Dabei fängt er mit der ersten User Story an, nimmt dann die zweite hinzu und fragt das Entwicklungsteam so lange, bis ein oder mehrere Mitglieder des Entwicklungsteam Zweifel an der Erreichbarkeit des Ziels äußern. Wichtig ist hierbei, dass das Entwicklungsteam ohne äußeren Druck entscheiden darf, für wie viele User Stories es sich verpflichten will.[22]

Dauer: 60 Minuten pro Sprint-Woche.

Sprint Planning Meeting 2

Während im Sprint Planning 1 das „Was“ im Vordergrund stand, spielt hier das „Wie“ die zentrale Rolle. Das Entwicklungsteam weiß bereits, was es zu erledigen hat und macht sich nun daran, die technische Umsetzung der mit an Bord genommenen User Stories zu klären (ohne jedoch mit der eigentlichen Umsetzung zu beginnen). Dieses Treffen organisiert das Entwicklungsteam eigenverantwortlich. Oftmals bilden sich während des Meetings Kleingruppen, in denen jeweils verschiedene Aspekte (z.B. Architektur, Datenelemente, Lücken) geklärt werden. Ergebnis dieses Meetings sind Aufgaben oder „Tasks“, deren Erledigung nicht länger als einen Tag dauern sollte. Die Aufgaben werden zusammen mit den zuvor angenommenen User Stories an eine Pinnwand oder ein Whiteboard gehängt, das so genannte Taskboard. Darauf lässt sich jederzeit erkennen, welche Aufgaben zu bearbeiten sind, welche in Bearbeitung sind, und welche bereits bearbeitet wurden.[23]

Dauer: 60 Minuten pro Sprint-Woche.

Sprint Planning 1 und Sprint Planning 2 sollten am selben Tag durchgeführt werden.

Daily Scrum

Zu Beginn eines jeden Arbeitstages trifft sich das Scrum-Team zu einem max. 15-minütigen Daily Scrum. Zweck des Daily Scrums ist der Informationsaustausch. Im Daily Scrum werden keine Probleme gelöst - vielmehr geht es darum, sich einen Überblick über den aktuellen Stand der Arbeit zu verschaffen. Dazu hat sich bewährt, dass jedes Teammitglied mit Hilfe des Taskboards sagt, was es seit dem letzten Daily Scrum erreicht hat, was es bis zum nächsten Daily Scrum erreichen möchte, und was dabei im Weg steht.

Beim Daily Scrum kann offensichtlich werden, dass die Erledigung eines Task länger als geplant, also länger als einen Tag dauert. Dann ist es sinnvoll, den Task in kleinere Aufgaben herunterzubrechen, die dann auch von anderen Mitgliedern des Entwicklungsteam übernommen werden können.

Treten beim Daily Scrum inhaltliche Fragen auf, die sich nicht innerhalb der strikten 15-Minuten-Vorgabe beantworten lassen, notiert der ScrumMaster diese und geht sie entweder selber an (z.B. bei der Beseitigung von Hindernissen) oder vertagt deren Beantwortung auf ein späteres Meeting.[24]

Sprint Review

Das Sprint Review steht am Ende des Sprints. Das Entwicklungsteam präsentiert seine Ergebnisse und es wird überprüft, ob das zu Beginn gesteckte Ziel erreicht wurde. Es ist Aufgabe des Product Owners, die entwickelten Funktionalitäten zu begutachten und anhand der im Sprint Planning 1 festgelegten Bedingungen zu entscheiden, ob sie abgenommen werden können oder nicht. Dabei sollten keine Kompromisse eingegangen werden: Ein Team hat auch dann sein Ziel verfehlt, wenn es eine „fast fertige“

(aber z.B. noch nicht getestete) Funktionalität liefert. In diesem Fall kehren die nicht fertiggestellten User Stories in das Product Backlog zurück und werden vom Product Owner neu priorisiert.

Im Sprint Review ist die Beteiligung des Users besonders wichtig, da dieser nun zum ersten Mal eine fertige Funktionalität benutzen kann. Hieraus kann sich wichtiges Feedback für die weitere Produktgestaltung ergeben. Es kann sogar passieren, dass die Funktionalität technisch einwandfrei umgesetzt wurde (sie erfüllt alle Abnahmekriterien) und dennoch aus der Perspektive des Benutzers unbrauchbar ist (z.B. weil ein Button an einer kaum auffindbaren Stelle platziert wurde). Solches Feedback ist dann als Aufforderung an den Product Owner zu verstehen, seine User Stories besser an den Bedürfnissen des Benutzers auszurichten.

Häufig entsteht während des Reviews ein lebhafter Dialog, in dem den Anwesenden neue Funktionalitäten einfallen. Diese werden vom ScrumMaster notiert und an den Product Owner als mögliche neue Einträge für das Product Backlog weiter gereicht.[25]

Das Sprint Review dauert nicht länger als 90 Minuten.

Retrospektive

Die Retrospektive steht zwischen Review und dem neuen Sprint Planning Meeting. Sie soll dem Team Gelegenheit und Zeit geben, über die gemachten Erfahrungen zu reflektieren und Verbesserungsmöglichkeiten zu identifizieren. Sinn der Retrospektive ist nicht, Kritik an Personen zu üben und Schuldige zu suchen, sondern aus den Ereignissen zu lernen. Die Retrospektive sollte in einem geschützten Raum ablaufen, um ein Gefühl der Sicherheit zu vermitteln.

Die Retrospektive steht nur dem Entwicklungsteam und dem ScrumMaster offen, sowie dem Product Owner, alle anderen Rollen dürfen nur auf Einladung dazukommen. Als Vorgehensweise bietet es sich an, zunächst alle relevanten Ereignisse der vergangenen Sprints auf einem Zeitstrahl festzuhalten (ohne Bewertungen vorzunehmen), um anschließend auf zwei separaten Flipcharts aufzuschreiben, was gut lief bzw. was verbessert werden könnte. Hinsichtlich der Verbesserungsmöglichkeiten lassen sich die identifizierten Punkte in teamspezifische Belange (z.B. Testumgebung, klare Formulierung der Stories, bessere Absprache mit dem Kunden) und in organisationsweite Belange (z.B. zu kleiner Raum, mangelnde Hardware, Störungen von außen) trennen. Anschließend empfiehlt es sich, die Liste der Hindernisse nach Dringlichkeit zu priorisieren.

Hindernisse, die das Team allein betreffen, werden von diesem selbst gelöst. Die anderen Hindernisse werden vom ScrumMaster in seinem Impediment Backlog aufgenommen. Die allein das Team betreffenden Impediments werden im darauf folgenden Sprint Planning 1 dem Product Owner zusammen mit dem für ihre Lösung geschätzten Aufwand vorgestellt. Der Product Owner entscheidet dort, welche Hindernisse vom Team im kommenden Sprint angegangen werden dürfen.[26]

Die Dauer einer Retrospektive beträgt 90 Minuten. Wird für die Retrospektive und deren Vorbereitung nicht genug Zeit eingeräumt, bleiben die Erkenntnisse und Ergebnisse oberflächlich und die Beobachtungen nach jedem Sprint ähneln sich. Dann läuft man Gefahr, dass die Retrospektiven an Stellenwert verlieren oder ganz gestrichen werden, weil die Retrospektiven keinen Mehrwert ergeben.

Sprint

Im Sprint wird das Produkt entwickelt. Die zwei oben erwähnten Sprint Plannings stehen am Anfang eines jeden Sprints. An deren Ende steht die Lieferung fertiger Produktfunktionalitäten, wie sie im Sprint Planning festgelegt wurden. Die Arbeit des Entwicklungsteams orientiert sich am Taskboard. User Stories mit der höchsten Priorisierung werden zuerst bearbeitet. Alle arbeiten gemeinsam an jeweils einer User Story. Dadurch weiß jeder, welche Funktionalität gerade entwickelt wird, und man ist in der Lage, sich gegenseitig auszuhelfen: Der Software-Entwickler kann mit dem Tester die ersten Testläufe starten, der

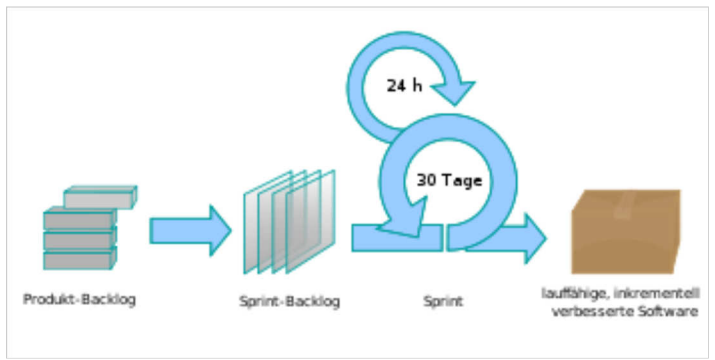
Business Analyst kann dem User bereits einige Teile der Funktionalität zeigen. Das Entwicklungsteam macht sich erst dann an die Bearbeitung der nächsten Funktionalität, wenn alle zuvor vereinbarten Akzeptanz- und Testkriterien erfüllt und die Funktionalität damit tatsächlich fertig ist. Dadurch wird verhindert, dass am Ende des Sprints viele Funktionalitäten bearbeitet aber keine davon fertiggestellt sind.

Während des Sprints trägt das Entwicklungsteam allein die Verantwortung, das vereinbarte Ziel zu erreichen. Um diese Verantwortung auch wahrnehmen zu können, muss das Entwicklungsteam unbedingt vor Störungen geschützt werden. Von außen herangetragene, zusätzliche Aufgaben gehören in den Verantwortungsbereich des ScrumMasters. Das gilt auch dann, wenn diese Aufgaben nur ein Mitglied des Entwicklungsteams betreffen. Der ScrumMaster entscheidet, ob und was dem Entwicklungsteam zugemutet werden kann. Jegliche Hindernisse, die das Team am Erreichen ihres Zieles hindern, sind vom ScrumMaster zu beseitigen. Dazu gehören auch Probleme innerhalb des Entwicklungsteams. Werden Regeln nicht eingehalten oder Tasks nicht fertiggestellt, muss der ScrumMaster eingreifen. Dabei darf er nicht dem Entwicklungsteam sagen, wie es etwas zu erledigen hat - dazu ist nur das Team selbst befugt. Der ScrumMaster erteilt somit keine Anweisungen, er erinnert vielmehr an die getroffenen Abmachungen und deren Konsequenzen.

Die Dauer von Sprints sollte immer gleich lang sein und darf nicht verlängert werden. Ein Sprint sollte nicht kürzer als eine Woche und nicht länger als vier Wochen dauern. Ist das Ziel eines Sprints offensichtlich nicht mehr zu erreichen, beispielsweise weil teamintern ein heftiger Streit entbrannt ist oder weil Anforderungen missverstanden wurden, dann kann der Sprint vom Entwicklungsteam oder vom Product Owner abgebrochen werden. Im Fall eines Sprintabbruchs folgt auf den Sprint kein Review, sondern eine Retrospektive und das Sprint Planning für den nächsten Sprint.[27]

Artefakte

Product Backlog



Der Scrum-Prozess

Das Product Backlog ist eine priorisierte Liste, die alles enthält, was im zu entwickelnden Produkt enthalten sein sollte. Es ist zudem der einzige Ort, in dem Änderungsanforderungen aufgenommen werden. Der Product Owner ist für alle Eintragungen ins Product Backlog verantwortlich. Er verantwortet auch deren Priorisierung. Das Product Backlog ist nie vollständig und erhebt auch nicht diesen Anspruch. Zu Beginn eines Projektes enthält es die bekannten und am besten verstandenen Anforderungen. Die Priorisierung der Eintragungen erfolgt unter Gesichtspunkten wie wirtschaftlicher Nutzen, Risiko und Notwendigkeit. Eintragungen mit der höchsten Priorisierung werden als erste im Sprint umgesetzt. Das Product Backlog verändert sich zusammen mit dem Produkt und der Arbeitsumgebung.[28]

Die im Product Backlog eingetragenen Anforderungen sind nicht technik-, sondern benutzerorientiert. Sie beschreiben die Anforderungen aus der Sicht des Endanwenders und heißen deshalb „User Stories“.[29] Eine gute User Story sollte die Antwort auf mindestens drei Fragen liefern und folgendes Format haben:

Als User (Wer?) möchte ich diese Funktionalität (Was?), damit ich folgenden Nutzen habe (Wozu?).[30]

Bei einem Projekt zur Entwicklung eines städtetauglichen Elektrofahrrads könnte eine User Story demnach folgendermaßen lauten:

Als 30-jährige Geschäftsfrau möchte ich auf dem Weg zur Arbeit wenig in die Pedale treten müssen, damit

ich nicht verschwitzt in der Firma ankomme.

Damit ist eine Eigenschaft des Produktes aus Sicht eines potenziellen Benutzers geklärt. Es ist Aufgabe des Product Owners und des Teams, im Sprint Planning 1 zu klären, was genau damit gemeint ist, und welches die Akzeptanzkriterien sein sollen. So könnte zum Beispiel vereinbart werden, dass bis zu einer Steigung von maximal 30% der elektrische Antrieb so stark sein muss, dass der Fahrer nicht mehr als 100 Watt durch sein eigenes Treten beisteuern muss. Zudem muss das Entwicklungsteam mit dem Product Owner klären, ob sich diese User Story überhaupt in einem Sprint erledigen lässt oder ob sie in kleinere Stories heruntergebrochen werden muss. Sobald eine User Story umgeschrieben oder um weitere Information ergänzt wird, werden auch diese Änderungen im Product Backlog festgehalten.[31]

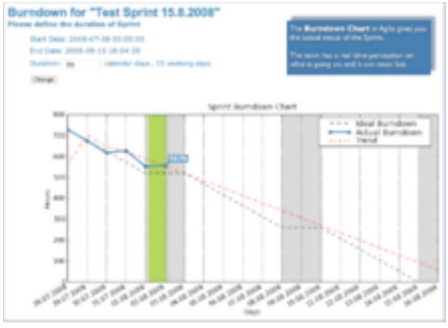
Fragen nach dem „Wie“, also nach der technischen Umsetzung dieser User Story, gehören ins Sprint Planning 2 und werden nicht im Product Backlog, sondern im Taskboard mit Hilfe der einzelnen Tasks festgehalten.

Sprint Backlog

Das Sprint Backlog dient zur Übersicht der für einen Sprint zu erledigenden Aufgaben. Zu diesem Zweck kann ein Taskboard eingesetzt werden. Es besteht aus vier Spalten. In der ersten Spalte („Stories“) werden die User Stories aufgehängt, für die sich das Entwicklungsteam zu diesem Sprint verpflichtet hat (in der vom Product Owner priorisierten Reihenfolge). Die drei weiteren Spalten enthalten die Aufgaben oder Tasks, die sich aus den einzelnen User Stories ergeben (und die im Sprint Planning 2 festgelegt worden sind). Je nach Bearbeitungsstand sind die Tasks entweder offen („Tasks to Do“), in Bearbeitung („Work in Progress“) oder erledigt („Done“). Im Daily Scrum erklärt jedes Mitglied des Entwicklungsteams anhand des Taskboards, an welcher Aufgabe es am Vortag gearbeitet hat, und ob diese erledigt wurde. Tasks, die an einem Tag nicht beendet werden konnten, werden mit einem roten Punkt markiert. So können Hindernisse schnell identifiziert werden.[32]

Burndown-Charts

Burndown-Charts dienen der Visualisierung bereits geleisteter und noch verbleibender Arbeit.



Beispiel eines Burndown Charts

Ein Sprint-Burndown-Chart erfasst auf der x-Achse den Zeitverlauf (in Tagen) und auf der y-Achse die Anzahl der nicht erledigten Tasks. Alternativ können statt der Anzahl der Tasks die geschätzten Aufwände für jeden einzelnen Task (in Stunden) eingetragen werden. All diese Charts sollen dazu dienen, die Erreichung des Sprint-Ziels besser abschätzen zu können. Sie sagen jedoch nur etwas über die Abarbeitung der einzelnen Aufgaben aus. Die im Product Backlog festgehaltenen User Stories bleiben hier unberücksichtigt.

Um den Stand der ausgewählten User Stories zu beobachten, bietet sich der Story-Burndown-Chart (auch Sprint-Product-Burndown-Chart genannt) an. Darin werden statt der Tasks die noch nicht erledigten User Stories eines Sprints angezeigt. Da nicht jede User Story gleich groß ist, werden diese mit unterschiedlichen großen Story Points versehen. Die Summe der noch verbleibenden Story Points wird auf der y-Achse angezeigt, während auf der x-Achse die Dauer eines Sprints verzeichnet ist. Der Kurvenverlauf ist hier treppenförmig. Eine Absenkung ergibt sich immer dann, wenn eine User Story erledigt wurde (bei der Erledigung einer 8 Punkte großen User Story ergäbe sich somit eine Absenkung um 8 Punkten). Um den Verlauf des Gesamtprojektes anzuzeigen, eignet sich der Release-Burndown-Chart. Hier werden auf der y-Achse alle im Product Backlog festgehaltenen und noch nicht erledigten User Stories (statt nur die eines Sprints) in Form von Story Points zusammengezählt. Auf diese Weise wird angezeigt, wie viel noch bis zum Ende des Projektes (nach derzeitigem Stand der Dinge) geliefert werden muss.[33]

Impediment Backlog

Der ScrumMaster führt ein Impediment Backlog, in dem alle aktuell identifizierten Hindernisse eingetragen werden. Neben einer kurzen Beschreibung des Problems sollte das Impediment Backlog das Datum enthalten, an dem das Problem zum ersten Mal aufgetreten ist. Zudem trägt der ScrumMaster am Ende eines Daily Scrum das Datum ein, an dem das Hindernis beseitigt wurde.[34]

Definition of Done

Die Definition of Done (DoD) ist eine Checkliste von Aktivitäten, die zur Implementierung einer User Story gehören und die Qualität der Software beeinflussen. Dazu gehört beispielsweise das Schreiben von Kommentaren, Unit Tests und Design-Dokumenten. Die DoD wird von den Beteiligten zu Beginn eines Projektes festgelegt, kann aber im Laufe der Entwicklung angepasst werden. Die DoD hilft zu Beginn eines Sprints, die Anzahl und den Umfang der Tasks festzulegen. Es müssen aber nicht alle Aktivitäten der DoD auf jede User Story zutreffen. Am Ende des Sprints dient die DoD neben den detaillierten Anforderungen für eine spezifische User Story dazu, zu entscheiden, ob eine User Story als *done* akzeptiert wird. [35]

Die Grenzen von Scrum

Keine Erfolgsgarantie

Scrum kann genau so wenig wie andere Prozesse und Frameworks eine Erfolgsgarantie bieten. Die Anwendung von Scrum erzeugt Transparenz sowie regelmäßige Produktlieferungen. Hindernisse werden sichtbar und das Produkt kann Schritt für Schritt entwickelt und bei Bedarf geändert werden.

Verwertung gewonnener Erkenntnisse

Bei der Verwendung von Scrum muss man sich darauf einstellen, dass die ursprünglichen Einschätzungen permanent über- oder untertroffen werden. Scrum zeigt vom ersten Tag an Abweichungen vom Soll-Zustand an. Ob Scrum dazu führt, dass Produkte schnell, gut, günstig oder qualitativ hochwertig entwickelt werden, hängt davon ab, was das Scrum-Team mit den gewonnenen Erkenntnissen macht. Nach *Schwaber* kann auch ein Team von Idioten nach Scrum arbeiten[36]. Das Team liefert am Ende jedes Sprints zuverlässig Produktinkremente, hält alle Meetings ab, und verteilt die Rollen nach Scrum. Wenn aber das Scrum-Team die Ergebnisse nicht nutzt, um anders zu arbeiten und Anpassungen vorzunehmen, wird auch das Produkt nicht besser oder früher fertig sein.

Hinderliche Einflüsse bei der Teamzusammensetzung

Die Implementierung von Scrum kann durch verschiedene Faktoren behindert werden. Die Selbstorganisation im Entwicklungsteam beinhaltet den Grundsatz, dass es im Team keine permanente Hierarchie geben darf. Mitglieder, die nicht bereit sind, ihre bisherige Position innerhalb des Entwicklungsteams aufzugeben, können daher zu Konflikten führen. Zudem sollte ein Entwicklungsteam ausreichend interdisziplinär ausgebildet sein, um möglichst viele Aufgaben ohne externe Hilfe zu bearbeiten. Auch bei umfangreichen und vielschichtigen Projekten wird von Scrum gefordert, dass prinzipiell alle Teammitglieder alle Aufgaben eines Sprints bearbeiten können. Somit passt jemand, der sich exklusiv als Tester, Programmierer oder Architekt sieht, nicht in ein Entwicklungsteam nach Scrum. Ebenso verhindernd wirken Einzelkämpfer und Gurus, da sie den von Scrum geforderten Informationsfluss zwischen allen Teammitgliedern behindern.

Aufwand für Tests

Häufige Änderungen erfordern Refactoring, was wiederum eine ausreichende Testabdeckung durch Unittests nötig macht. Häufige Lieferungen erfordern eine ausreichende fachliche Testabdeckung durch Regressionstests. Gerade in Altsystemen ist eine ausreichende Testabdeckung durch Testautomatisierung mit vertretbarem Aufwand oft nicht zu erreichen. Manuelle Tests generieren nach jedem Sprint erneut Testaufwand und verhindern somit das effektive Arbeiten nach Scrum.

Juristische Erwägungen

Auch vor einem juristischen Hintergrund (z.B. Produkthaftungsgesetz) kann die Anwendung von Scrum begrenzt sein. Im Verlauf des Entwicklungsprozesses und insbesondere bei der Beauftragung existieren keine verbindlichen Vorgaben, da sich diese erst während des Projektes formieren. Insofern besteht eine deutliche Unschärfe über die zu erbringende Leistung und deren Abnahmekriterien aus juristischer Sicht. Bei Streitigkeiten im Rahmen von Werkverträgen kann dies dazu führen, dass keine eindeutige Aussage zur Vertragserfüllung getroffen werden kann und auch Verantwortlichkeiten unklar bleiben.

Tools

Es gibt diverse Tools wie Agilo, Pangoscrum, AgileZen, Tinypm, ThoughtWorks Studios, Greenhopper, VersionOne, ScrumWorks Pro, Banana Scrum und ScrumTable, die darauf ausgelegt sind, die Einführung von Scrum und die darin anfallenden Prozesse zu erleichtern.

Literatur

- Ken Schwaber: *Agiles Projektmanagement mit Scrum*. Microsoft Press Deutschland, 4. Oktober 2007 (Originaltitel: *Agile Project Management with Scrum*, übersetzt von Thomas Irlbeck), ISBN 978-3866456310.
- Holger Koschek: *Geschichten vom Scrum: Von Sprints, Retrospektiven und agilen Werten*. dpunkt.verlag, 2009, ISBN 978-3-89864-640-6
- Ken Schwaber: *Scrum im Unternehmen*. Microsoft Press Deutschland, 14. April 2008 (Originaltitel: *The Enterprise and Scrum*), ISBN 978-3866456433.
- Boris Gloger: *Scrum-Produkte zuverlässig und schnell entwickeln*. 3. Auflage. Hanser Verlag, 2011, ISBN 978-3446425248
- Boris Gloger: *Scrum: Der Paradigmenwechsel im Projekt- und Produktmanagement. Eine Einführung.* In: Informatik Spektrum, Vol. 33, No. 2. 2010.
- Ken Schwaber: *Scrum Development Process, Advanced Development Methods*, 131 Middlesex Turnpike Burlington, MA01803

Siehe auch

- Feature Driven Development
- Extreme Programming (XP)
- Testgetriebene Entwicklung
- Kanban in der IT
- Story-Cards
- Agile Softwareentwicklung
- Agiles Datawarehousing

Weblinks

- Die offizielle Website der Scrum Alliance (http://www.scrumalliance.org/) (englisch)

- Die offizielle Webseite der Scrum.org (http://www.scrum.org/) (english)
- Artikel über Scrum (Original in IX 6/09 veröffentlicht) (http://www.agile42.com/cms/articles/2009/6/16/Scrum-risks/)
- Scrum auf einen Blick (http://media.agile42.com/content/Scrum_Cheat_Sheet_0909.pdf) (englisch; PDF-Datei; 665 kB)
- Artikel: Agile Testing, Scrum im Test. (http://www.pentasys.de/uploads/blickpunkte/attachments/pentasys_blickpunkte_1106_agile_testing.pdf)
- Jeff Sutherland's Scrum Handbook (http://jeffsutherland.com/scrumhandbook.pdf) (englisch; PDF-Datei; 7,2 MB)
- Das neue Scrum-Plakat (http://www.scrum-plakat.de/scrum-plakat-neu/) (eine grafische Darstellung des Scrum-Prozesses auch zum Download)

Einzelnachweise

1. Jeff Sutherland, Ken Schwaber: *The Scrum Guide* (http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf) . Abgerufen am 10. August 2011. S. 4.
2. Roman Pichler 2009: Scrum - Agiles Projektmanagement erfolgreich einsetzen. dpunkt Verlag, Heidelberg. S. 1.
3. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 12.
4. *Jeff Sutherland*. (http://scrum.jeffsutherland.com/) Abgerufen am 21. Oktober 2011.
5. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 19.
6. *Mike Beedle*. (http://www.scrumalliance.org/profiles/3278-mike-beedle) Abgerufen am 21. Oktober 2011.
7. Ken Schwaber 2008: Scrum im Unternehmen. Microsoft Press Deutschland. S. XI-XII.
8. Ikujiro Nonaka u. Hirotaka Takeuchi 2008: A Theory of the Firm's Knowledge-Creation Dynamics. In: Alfred Chandler et al. (Hrsg.): The Dynamic Firm. The Role of Technology, Strategy, Organization, and Regions. Oxford University Press. S. 215-216.
9. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 27-30.
10. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 14-15
11. Roman Pichler 2009: Scrum. Agiles Projektmanagement erfolgreich einsetzen. dpunkt.verlag, Heidelberg. S. 9-12.
12. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 78-87
13. Roman Pichler 2009: Scrum - Agiles Projektmanagement erfolgreich einsetzen. d.punkt Verlag, Heidelberg. S. 10-13
14. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 67-77.
15. Roman Pichler 2009: Scrum - Agiles Projektmanagement erfolgreich einsetzen. d.punkt Verlag, Heidelberg. S. 15
16. Ken Schwaber, Mike Beedle: *Agile Software Development with Scrum*. Prentice Hall, Upper Saddle River 21. Oktober 2001, ISBN 978-0130676344, S. 36.
17. Roman Pichler 2009: Scrum - Agiles Projektmanagement erfolgreich einsetzen. d.punkt Verlag, Heidelberg. S. 20-23
18. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 88-101
19. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag,

München. S. 101-103
20. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 103-104
21. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 104-107
22. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 164-166
23. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 166-169
24. Roman Pichler 2009: Scrum. Agiles Projektmanagement erfolgreich einsetzen. dpunkt.verlag, Heidelberg. S. 104-107.
25. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 103-104
26. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 180-192
27. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 192-205.
28. Jeff Sutherland, Ken Schwaber: *The Scrum Guide* (http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf) . Abgerufen am 10. August 2011. S. 12.
29. Roman Pichler 2009: Scrum - Agiles Projektmanagement erfolgreich einsetzen. d.punkt Verlag, Heidelberg. S. 46-47
30. Boris Gloger: *Scrum Essentials: Die sieben Fragen der User Story* (http://borisgloger.com/2011/06/20/scrum-essentials-die-sieben-fragen-der-user-story/) . Abgerufen am 11. August 2011. S. 12.
31. Roman Pichler 2009: Scrum - Agiles Projektmanagement erfolgreich einsetzen. d.punkt Verlag, Heidelberg. S. 46-47.
32. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 167-169.
33. Boris Gloger 2011: Scrum. Produkte zuverlässig und schnell entwickeln. 3. Auflage. Hanser Verlag, München. S. 209-213.
34. Roman Pichler 2009: Scrum - Agiles Projektmanagement erfolgreich einsetzen. d.punkt Verlag, Heidelberg. S. 119.
35. Dhaval Panchal: What is Definition of Done (DoD)? http://www.scrumalliance.org/articles/105-what-is-definition-of-done-dod
36. *Ken Schwaber: Scrum et al. (Minute 14)* (http://www.youtube.com/watch?v=IyNPeTn8fpo) Abgerufen am 12. August 2011.