Scrum Whitepaper

Alles Wichtige über

Scrum,
die zentralen Begriffe,
Rollen, Events und
Herausforderungen

auf einen Blick.

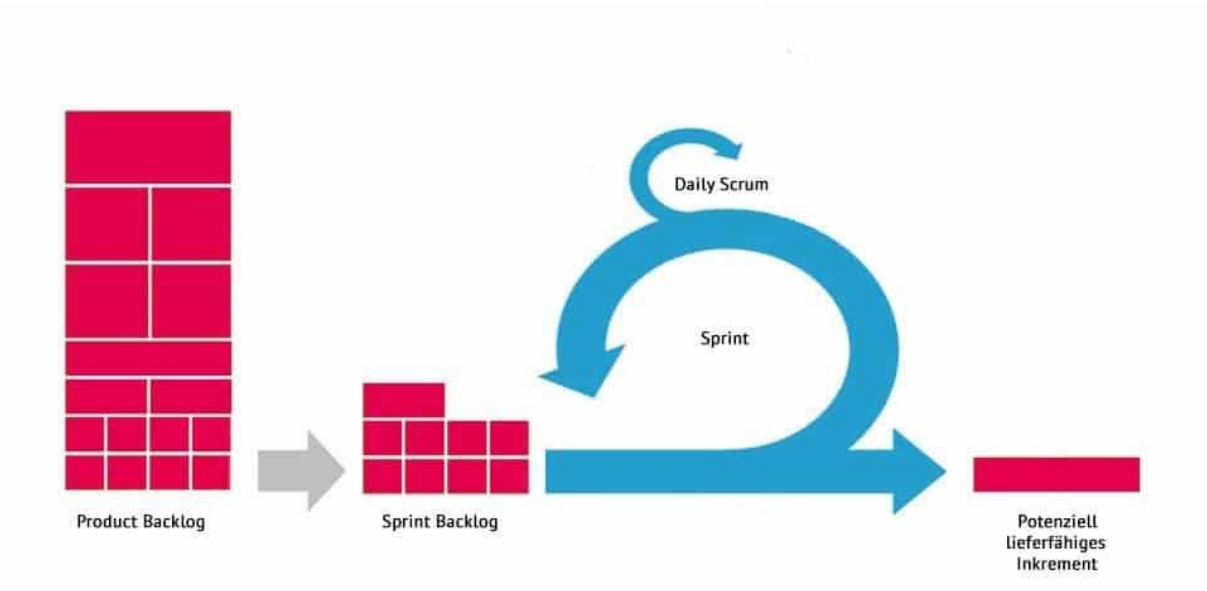
t2informatik
Softwareentwicklung und Projektmanagement aus Berlin

Scrum ist ein agiles Framework für die Entwicklung und Instandhaltung komplexer Produkte und Services. Ursprünglich für die Softwareentwicklung konzipiert, wird Scrum heutzutage in vielen Bereichen und Industrien verwendet. Wie funktioniert es, welche Events, Rollen und Elemente gibt es?

Scrum Definition

Scrum ist ein agiles Framework für die Entwicklung und Instandhaltung komplexer Produkte und Services. Ursprünglich für die Softwareentwicklung konzipiert, wird Scrum heutzutage in vielen Bereichen und Industrien verwendet. Es definiert mit Sprints, Sprint Planning, Daily Scrum, Sprint Review und Retrospektive fünf Events bzw. Aktivitäten oder Rituale. Es legt mit dem Product Owner, dem Scrum Master und dem Entwicklungsteam drei Rollen fest. Es benennt mit dem Product Backlog, dem Sprint Backlog und Product Inkrement drei Artefakte. Diese Events, Rollen und Artefakte sind der Kern von Scrum. Die Regeln zur Anwendung von Scrum sind im Scrum Guide definiert. Der Ansatz von Scrum ist empirisch, inkrementell und iterativ. Er geht davon aus, dass viele Entwicklungsprojekte zu komplex für eine exakte Planung und Beschreibung sind, da viele Anforderungen und Lösungen zu Projektbeginn unklar sind. Diese Unklarheiten werden durch ein schrittweises Vorgehen und die Entwicklung von potenziell lieferfähigen Inkrementen beseitigt.

Scrum setzt auf klare Regeln und Abläufe, und fördert die Kommunikation zwischen den Beteiligten. Da Vorhersagen und die Erstellung von realistischen Plänen bei der Entwicklung von Software und Systemen häufig schwierig sind, dreht Scrum den Ansatz um: Wissen entsteht aus Erfahrung und Erfahrung entsteht im Laufe einer Entwicklung. In Verbindung mit dem iterativen Vorgehen, der leichteren Einplanung von Änderungen und dem frühzeitigen Erkennen von Hindernissen, lässt sich durch dieses Wissen die Vorhersagbarkeit erhöhen und das Entwicklungsrisiko minimieren. Durch das Commitment des Teams, abgestimmte Anforderungen in Sprints zu realisieren und Ziele zu erreichen, erhalten Stakeholder die Möglichkeit, frühzeitig Feedback zu geben und Verbesserungsvorschläge zu äußern. Die Flexibilität beim Arbeiten mit Scrum, die konzentrierte Kommunikation zwischen den Scrum Rollen, das Feedback, die Transparenz der Arbeitsschritte und Zwischenergebnisse, die Fähigkeit, Abweichungen zu erkennen und zu korrigieren, sowie die Möglichkeit, den Prozess im laufenden Projekt zu justieren, sind die größten Vorteile von Scrum.



Im Jahr 2001 formulierten 14 Autoren, u.a. Ken Schwaber und Jeff Sutherland, das agile Manifest. Es definiert Werte und agile Prinzipien, die in Scrum Anwendung finden:

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
- Funktionierende Software ist wichtiger als eine umfassende Dokumentation
- Die Zusammenarbeit mit Kunden ist wichtiger als die Vertragsverhandlung
- Das Reagieren auf Veränderung ist wichtiger als das Festhalten an Plänen

Diese Werte beinhalten immer zwei Seiten. Die linke Seite ist dabei wertvoller als die rechte. Natürlich darf die rechte Seite nicht zu weit in den Hintergrund rücken; bspw. lassen sich nicht alle Funktionen einer Software ohne Dokumentation nachvollziehen. Bei der Entwicklung von Lösungen ist also immer auch auf eine Ausgewogenheit der Werte zu achten.

Scrum Rollen

Scrum definiert mit dem Product Owner, dem Scrum Master und dem Entwicklungsteam drei aktive Rollen. In den Mittelpunkt der Entwicklung stellt es die Stakeholder, die nicht nur als eine primäre Quelle für die Erhebung von Anforderung dienen, sondern auch an einzelnen Scrum Events teilnehmen können.

Product Owner

Der Product Owner repräsentiert die Vorstellungen der Stakeholder, erstellt und pflegt das Product Backlog, und arbeitet eng mit dem Entwicklungsteam zusammen. Er legt das Projektziel fest und ist für den Erfolg der Entwicklung verantwortlich. Er steuert die Entwicklung über die Formulierung und Priorisierung von Epics, Features, Anforderungen und User Storys. Er verantwortet die Releases und Releasepläne, pflegt einen Kazapitätsplan und betreibt Risikomanagement. Er vereint somit Produkt- und Projektmanagementaufgaben. Trotz seiner Verantwortung und seinen zahlreichen Aufgaben ist er aber nicht der Vorgesetzte des Entwicklungsteams. Er sollte sich weder in technische Lösungsdiskussionen oder die Schätzung von Aufwänden einmischen. Damit der Product Owner seine Rolle gut ausführen kann, muss er vom Management bevollmächtigt sein, um bspw. Entscheidungen bzgl. Anforderungen und Funktionsumfang treffen zu können. Darüber hinaus ist es wichtig, dass er sowohl fachlich als auch kommunikativ qualifiziert und stets verfügbar ist.

Scrum Master

Der Scrum Master trägt die Verantwortung für die Einhaltung des Scrum-Prozesses und dessen Implementierung im Unternehmen. Er moderiert die Scrum-Meetings, sorgt für die Einhaltung der vereinbarten Timebox, erfasst und beseitigt Hindernisse (die sogenannten Impediments) und schützt das Team vor unberechtigten Eingriffen während der Entwicklung. Er fördert die Kommunikation zwischen Entwicklungsteam und Product Owner und behält dabei verschiedene Artefakte wie das Product und Sprint Backlog, Burndown Charts, Taskboards oder das User Story Mapping im Auge. Im Rahmen der Retro-

spektive versucht er die Zusammenarbeit im und mit dem Team, sowie die gelebten Prozesse zu verbessern. Dort hat er auch Gelegenheit, Feedback zu seiner eigenen Tätigkeit zu erfragen. In seiner Rolle ist ein Scrum Master gleichzeitig Coach, Moderator und Vermittler. Seine Rolle sollte aber nicht mit einem Projektleiter verwechselt werden, denn das Entwicklungsteam organisiert sich selbst und kommuniziert auch direkt mit dem Product Owner. Eine Doppelfunktion als Product Owner oder Teammitglied sollte auf jeden Fall vermieden werden.

Entwicklungsteam

Ein Entwicklungsteam in Scrum besteht aus 3 bis 9 Personen. Idealerweise werden Entwicklungsteams funktionsübergreifend zusammengestellt, so dass sie gemeinsam alle erforderlichen Fähigkeiten besitzen, um ein Inkrement am Ende des Sprints zu erstellen. Dabei muss das Team vor allem in der Lage sein, Architekturen zu entwerfen oder zu optimieren, sowie Software zu entwickeln und zu testen. Unabhängig von den Fähigkeiten der einzelnen Mitglieder definiert Scrum keine zusätzlichen Rollen oder Titel für Teammitglieder. Damit betont Scrum die gemeinsame Verantwortung des Teams, ein potenziell lieferbares Inkrement zu liefern. Grundsätzlich gilt, dass Entwicklungsteams selbstorganisierend sind. Weder der Product Owner noch der Scrum Master sind dem Team vorgesetzt. Damit wird die Kommunikation im Zuge der verschiedenen Scrum Events um so wichtiger.

Stakeholder

In Scrum haben Stakeholder keine explizite Rolle, nehmen aber an verschiedenen Stellen maßgeblich Einfluss auf die Entwicklung. Mit Ihren Zielen liefern sie oft eine Grundlage für Anforderungen. Der Product Owner stimmt sich bspw. mit wesentlichen Stakeholdern ab und vertritt ihre Meinungen bspw. im Sprint Planning. Wer die Aufgaben der Stakeholderidentifikation und Stakeholderanalyse übernimmt, entscheiden Organisationen individuell. Am Sprint Planning 1 dürfen sie – sofern es Sinn macht und sie Interesse am Gelingen des Vorhabens haben – als Ansprechpartner für die Entwickler teilnehmen. Beim Sprint Review dürfen sie aktiv mitwirken und Feedback zu den umgesetzten User Storys sowie Verbesserungsvorschläge äußern. Bei weiteren Scrum Events wie bspw. dem Daily Scrum dürfen sie auf Einladung als stille Beobachter teilnehmen.

Neben Stakeholder wird oft auch der Begriff Shareholder benutzt. Der Shareholder-Ansatz zielt auf die wirtschaftlichen Interessen und Erwartungen der Anteilseigner des Unternehmens ab. Vorrangige Ziele sind also Umsatz- und Gewinnmaximierung. Dies ist der sogenannte Shareholder-Value. In der Theorie werden Interessen der Stakeholder lediglich dann berücksichtigt, sofern eine positive Wirkung auf den Unternehmenserfolg zu erwarten ist. Da Unternehmen jedoch kaum in der Lage sind, sich ausschließlich auf die Interessen der Shareholder oder die Bedürfnisse der Stakeholder zu konzentrieren, werden beide Ansätze in der Praxis gleichzeitig verfolgt.

Scrum Events

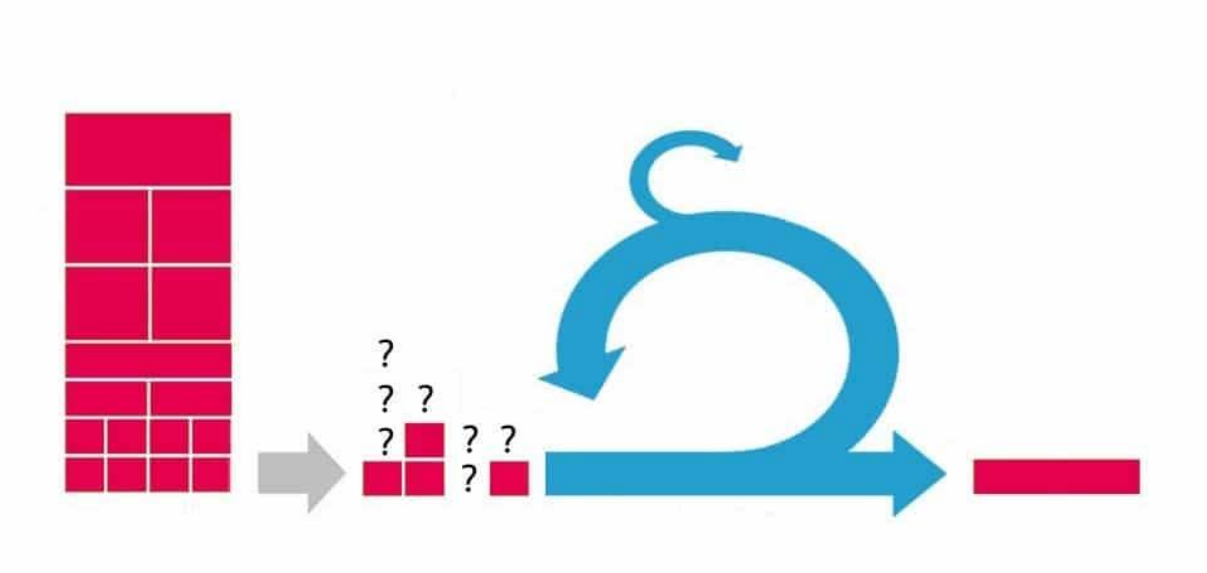
Ein Leitmotiv von Scrum ist die Entwicklung von Produkten in kurzen Zyklen bzw. Inkrementen. Diese kurzen Zyklen werden Sprint genannt. Sie sind das Herz von Scrum. Ursprünglich wurden Sprints auf 30 Tage terminiert, in der Zwischenzeit sind Sprints mit einer Dauer zwischen 1 und 4 Wochen üblich.

Wichtig ist, dass Sprints im gesamten Entwicklungsablauf eine konsistente Dauer haben. Beim Arbeiten mit parallelen Teams sollten Organisationen darauf achten, dass Sprints synchron verlaufen, so dass es nicht zu unnötigen Wartezeiten zwischen den Teams kommt. Während eines Sprints gilt es Änderungen zu vermeiden, um das vom Team verabschiedete Sprintziel gemeinsam zu erreichen und ein potenziell lieferfähiges Inkrement zu entwickeln, also eine Lösung, die funktioniert und Kunden zumindest theoretisch zur Verfügung gestellt werden könnte. Werden einzelne User Storys nicht umgesetzt, so wird der Sprint nicht verlängert, sondern die entsprechenden User Storys gegebenenfalls für einen der nächsten Sprints eingeplant. Die Anpassung von Qualitätszielen und Akzeptanzkriterien ist nicht vorgesehen, sollte jedoch das Sprintziel obsolet werden, kann der Product Owner einen Sprint auch abbrechen.

Scrum definiert verschiedene Events, die regelmäßig und kontinuierlich stattfinden: Spint Planning, Daily Scrum, Sprint Review und Sprint Retrospektive. Das Backlog Refinement gilt nicht als Scrum Event, auch wenn es ebenfalls regelmäßig durchgeführt werden sollte.

Sprint Planning

Zu Beginn eines Sprints findet das Sprint Planning (auch als Sprint Planning Meeting oder Sprint-Planungssitzung bezeichnet) als Kick-off statt. Es ist ein Treffen zur Planung der Anforderungen und Arbeitspakete, die im aktuellen Sprint umgesetzt werden sollen. Voraussetzung für das Meeting ist ein gepflegtes und priorisiertes Product Backlog. Es dient als Quelle zur Erstellung eines Selected Backlogs mit den wichtigsten und am höchsten priorisierten Anforderungen. Das Selected Backlog ist quasi eine Wunschliste des Product Owners. Wie viele und welche Backlog Items ins Sprint Backlog aufgenommen und in der Folge tatsächlich umgesetzt werden, entscheidet das Entwicklungsteam.



Beim Sprint Planning nimmt das Entwicklungsteam – also alle Entwickler, der Scrum Master und der Product Owner – teil. Organisiert wird das Meeting vom Product Owner, moderiert vom Scrum Master, der für die Einhaltung des vereinbarten Ablaufs und die Kommunikation zwischen den Beteiligten verantwortlich ist. Stakeholder wie bspw. Anwender, Partner, Vertreter aus Vertrieb oder Marketing können

zum Sprint Planning eingeladen werden, sie sollten sich aber passiv verhalten. Idealerweise hat der Product Owner bereits vor dem Meeting mit den Stakeholdern über die Inhalte und die Bedeutung der Anforderungen gesprochen.

Das Sprint Planning teilt sich in zwei Phasen bzw. Teile auf: Sprint Planning 1 und 2. Das Sprint Planning 1 kümmert sich um „What" – also darum, „was" getan werden soll. Es setzt sich aus folgenden Schritten zusammen:

- Vision des Sprints – vorgetragen durch den Product Owner. Durch die Vision erhält das Entwicklungsteam eine übergeordnete Sicht und ein besseres Verständnis, worum es im nächsten Sprint gehen wird. Auch wenn der Scrum Guide diesen Schritt nicht explizit fordert, so zeigt die Praxis, dass die Orientierung an der Vision dem Team bei der Erreichung des Sprintziels hilft. Je länger ein Sprint dauert, desto wichtiger wird diese Orientierung. Aus der Vision entsteht das Sprint Goal.
- Nennung der Items im Selected Backlog durch den Product Owner und anschließende Diskussion mit dem Entwicklungsteam. Hier geht es darum, fachliche und technische Hintergründe und Zusammenhänge zu erkennen und eine Vorstellung zur Machbarkeit der Anforderungen zu gewinnen.
- Verfeinerung der Akzeptanzkritieren der User Storys, gegebenenfalls bereits Formulierung von konkreten Akzeptanztests.
- Aufwandschätzung zur Realisierung der User Storys durch das Entwicklungsteam in Personentagen, Story Points oder T-Shirt Sizes.
- Übernahme der besprochenen Anforderungen in das Sprint Backlog unter Beachtung der maximal zu realisierenden Menge an Aufgaben. Ist das Sprint Backlog „voll" kann die Schätzung von weiteren Anforderungen auf das nächste Sprint Planning verschoben werden.
- Commitment des Entwicklungsteams, d.h. eine Verpflichtung des Teams die besprochenen Anforderungen im Laufe des Sprints zu realisieren.

Mit dem Commitment des Teams endet Phase 1 des Sprint Plannings. In Teil 2 des Meetings diskutieren die Entwickler ohne den Product Owner, der aber für Rückfragen erreichbar sein sollte, und ohne die möglicherweise anwesenden Stakeholder, über die technischen Details der zu realisierenden User Storys. Das Sprint Planning 2 beschäftigt sich mit dem „How", also der Frage, „wie" die vereinbarten Anforderungen umgesetzt werden sollen. Folgende Schritte werden durchgeführt:

- Gliederung der User Storys in Arbeitspakete – sprich Tasks. Idealerweise werden die Anforderungen nach ihrer Priorität in einzelne Aufgaben, die nicht länger als einen Arbeitstag dauern sollten, zerlegt. Zeitliche und technische Abhängigkeiten sollten möglichst vermieden werden, um die parallele Bearbeitung von Tasks zu ermöglichen. In der Praxis kann es nützlich sein, manche Anforderungen erst im Laufe des Sprints iterativ zu zerlegen, um so neue Erkenntnisse nutzen zu können. Eine namentliche Zuordnung der Tasks auf einzelne Entwickler wird nicht vorgenommen.
- Visualisierung der Tasks mit einem Taskboard. Hier unterscheiden sich die Ansätze: entweder wird mit einem physikalischen Taskboard zur Umsetzung der Storys und Tasks gearbeitet oder eine Software kommt zum Einsatz. Die Haptik der Story- und Task-Karten spricht für die physikalische Lösung, die Nachvollziehbarkeit und Revisionssicherheit für den Einsatz einer Software.

- Entwicklung einer Realisierungsstrategie, bspw. das Festlegen von Klassen, Interfaces, Testcases und Unit-Tests.

Am Ende von Phase 2 sollte das Team in der Lage sein, dem Product Owner und dem Scrum Master zu erklären, wie er als selbstorganisierendes Team arbeiten will, um das Sprint Goal zu erreichen und das erwartete Inkrement zu erstellen.

Es gibt eine Reihe von Erfahrungen, die Teams im Laufe einer Entwicklung sammeln. So werden sie besser beim Schätzen von Aufwänden oder können Störungen leichter vorhersehen. Hier finden Sie einige Tipps für das Sprint Planning:

- Vereinbaren Sie einen Prozentsatz der Kapazität für Fehlerbehebungen und Support-Arbeiten.
- Planen Sie mit effektiven Arbeitstagen, notieren Sie sich die freien Tage des Teams, Feiertage oder andere Ereignisse, die sich auf die Sprint-Lieferung auswirken könnten.
- Manche Organisationen nutzen Fälligkeitsdaten, denn die Verwendung von Fälligkeitsterminen kann ein guter Mechanismus sein, um den Fortschritt voranzutreiben und zu verfolgen. Zusätzlich kann er nützlich für Tests und die Stakeholderkommunikation sein.
- Stellen Sie sicher, dass Sie eine Definition of Done nutzen, da dies am Ende jedes Sprints zu besserer Software führt.

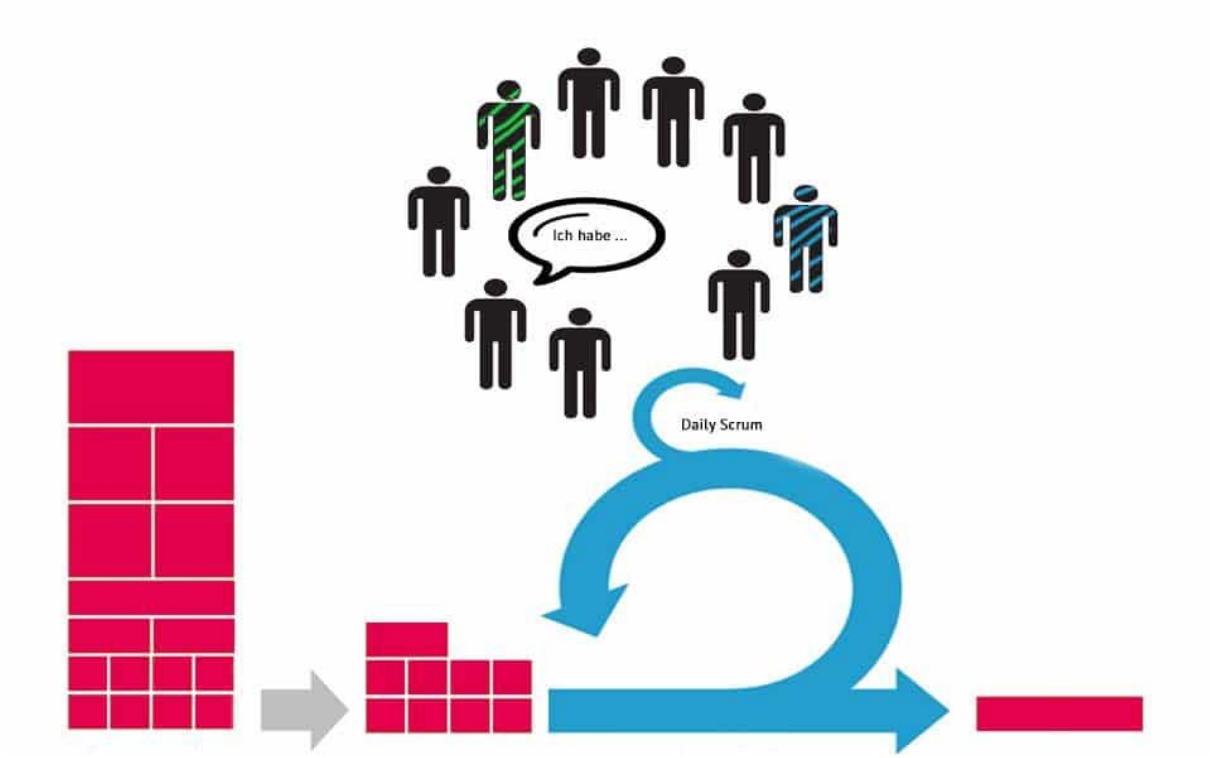
Es ist schwierig, die Zukunft vorherzusagen, trotz aller Erfahrungen und Techniken. Die Realität ist, dass selbst die Planung eines einfachen Softwareentwicklungsprojekts eine Herausforderung ist. Es gibt viele verschiedene Variablen, die berücksichtigt werden müssen, und sehr leicht kann es passieren, dass der Aufwand zur Realisierung einer Anforderung falsch eingeschätzt wird. In der Folge schaffen es Unternehmen oft nicht, Zusagen zu halten. Teams unterschätzen die Komplexität bei der Entwicklung von Lösungen, Zusammenhänge bei Anforderungen werden übersehen oder technische Schwierigkeiten lassen sich schlicht nicht einfach lösen. Die typische Projektplanung verwendet Puffer, um das Unerwartete zu berücksichtigen. Die zugrundeliegende Hoffnung ist, dass der Puffer groß genug ist und dass das Unerwartete nicht eintrifft. Wie können Sie diesen Fehler vermeiden? Scrum definiert die Erstellung von potenziell lieferbaren Software-Inkrementen am Ende eines Sprints. Damit minimieren Sie das Risiko, denn wenn etwas wirklich Unerwartetes passiert, besteht so noch die Möglichkeit, einen Nutzen aus dem zu ziehen, was bereits entwickelt wurde.

Daily Scrum

Das Daily Scrum ist ein täglich stattfindendes Meeting, zu dem sich das Entwicklungsteam trifft, um sich gegenseitig über den Fortschritt, die anstehenden Tätigkeiten und mögliche Probleme auszutauschen. Da sich die Teilnehmer im Kreis zum Dialog hinstellen, wird es auch als Standup Meeting oder als *daily stand-up* bezeichnet. Die Teilnahme an dem Meeting ist für das Entwicklungsteam verpflichtend, zusätzlich sollte der Scrum Master und könnte der Product Owner teilnehmen. Auch Vertreter aus anderen Bereiche wie bspw. Marketing, Vertrieb oder Dokumentation können an dem Meeting teilnehmen und sich so ein Bild vom Status quo der Entwicklung machen. Gäste haben allerdings kein Rederecht. Die Uhrzeit, die Dauer und der Ort des Daily Scrums sind fixiert und müssen daher nicht von Meeting zu Meeting neu vereinbart werden.

Beim Daily Scrum soll jedes Teammitglied drei Fragen beantworten:

- Was habe ich seit gestern getan?
- Was mache ich bis morgen?
- Was behindert mich bei meiner Arbeit?



Wichtig ist bei der Beantwortung der Fragen die individuelle Orientierung am *Sprint Goal*. Das Sprint Goal wird durch das Team gesetzt und das Daily Scrum dient dem Abgleich der Tätigkeiten in Richtung des vereinbarten Ziels. *„In Bezug auf das Sprint Goal habe ich seit gestern …"* – wäre eine nützliche Satzstruktur, die den Fokus der Antwort auf die Erreichung des Ziels legt. So kann das Team gut erkennen, falls Anpassungen zur gemeinsamen Erreichung des gesteckten Ziels notwendig werden.

Impediments sind Hindernisse jeglicher Art, die einzelne Teammitglieder oder das gesamte Team betreffen. Sie werden durch die Frage „Was behindert mich bei meiner Arbeit" offenbart. Hier finden Sie eine Auswahl mit „typischen" Impediments:

- „Mein Scanner ist kaputt und ich benötige heute einen neuen."
- „Ich kann den Support bei unserem Lieferanten nicht erreichen, um …"
- „Unser Lieferant kann nicht beginnen, solange wir den Auftrag nicht schicken."
- „Die Klimaanlage spinnt, wie soll ich bei der Hitze arbeiten?"
- „Ich habe Probleme beim Debuggen von ABC."
- „Der Bereichsleiter kam rein und hat mir eine ganz wichtige Aufgabe gegeben."
- „Wir sind 8 Teammitglieder, haben aber nur 5 Lizenzen von X."

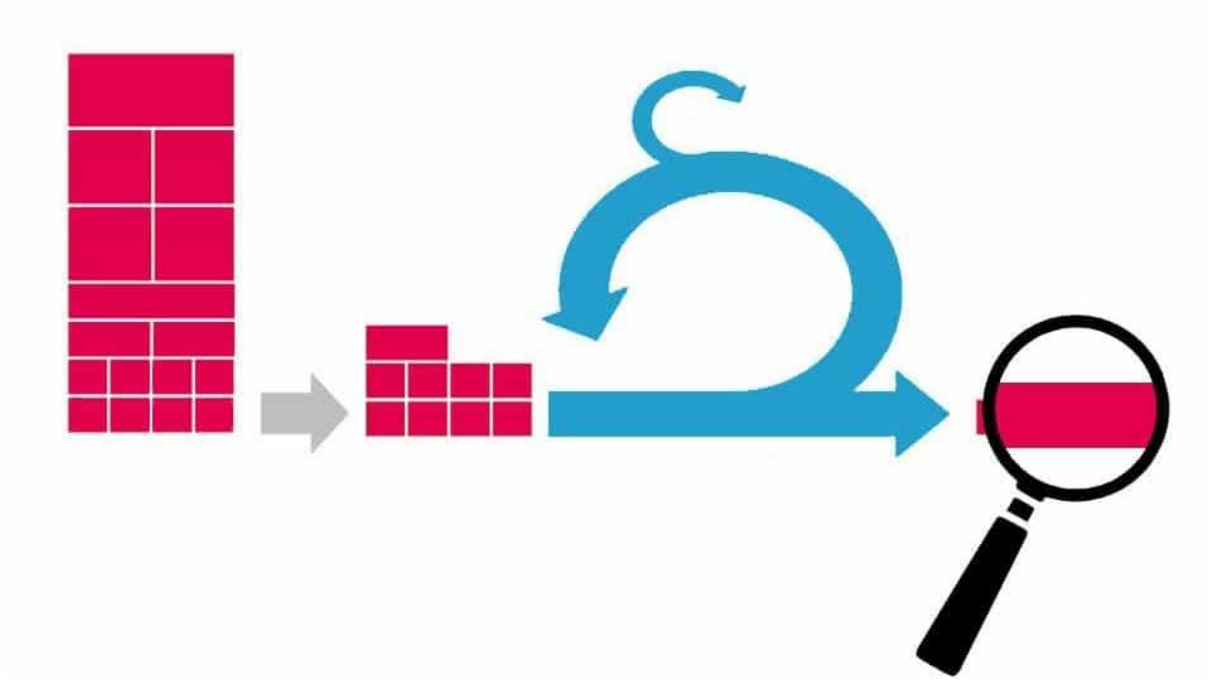
Es ist die Aufgabe des Scrum Masters die Impediments zu dokumentieren und zu beseitigen. Diese Beseitigung und auch die Diskussion darüber erfolgt nicht im Daily Scrum, denn sonst wäre die vereinbarte Timebox nicht zu halten. Die Dokumentation erfolgt in einem Impediment Backlog. Idealerweise werden die Hindernisse so schnell wie möglich gelöst, da dies aber nicht immer gelingt, ist die Kommunikation vom Scrum Master in Richtung Team und gegebenenfalls dem Management sehr wichtig.

Beim Daily Scrum geht es um die Planung des Tags durch das Entwicklungsteam. Es geht um erledigte und offene Aufgaben, es geht um Hindernisse und es geht um Zusammenarbeit. Idealerweise ist der Product Owner anwesend und in der Lage, Auskünfte zu geben. Der Scrum Owner sorgt für einen effektiven Ablauf und kümmert sich um die Beseitigung der Impediments. Durch den Austausch untereinander synchronisiert sich das Team und Redundanzen werden vermieden. Wichtig ist, dass das Daily Scrum nicht zu einer Pflichtveranstaltung im Rahmen von Scrum verkommt, weil es keinen ernsthaften Austausch zwischen den Teilnehmern gibt. Denken einzelne Mitarbeiter, dass sie in der Zeit des Meetings sinnvollere Dinge tun und bspw. einzelne User Storys umsetzen könnten, läuft das Daily Scrum alles andere als ideal. Hier ist der Scrum Master gefordert und hier hilft die Orientierung an den gemeinsamen Sprintzielen. Agile Softwareentwicklung heißt nicht, dass jeder einzelne Entwickler seine Aufgaben erledigt, es bedeutet, dass das gesamte Team in der Verantwortung steht. Dieser Verantwortung sollte sich jeder Entwickler bewusst sein und so das Daily Scrum als Basis für eine erfolgreiche Zusammenarbeit verstehen. Dabei hilft die konsequente Orientierung am Sprint Goal.

Sprint Review

Das Sprint Review ist ein Treffen am Ende eines Sprints zur Beurteilung und Abnahme der erledigten Arbeit in Bezug auf das gesteckte Sprintziel. Der Scrum Guide definiert „A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed. During the Sprint Review, the Scrum Team and stakeholders collaborate about what was done in the Sprint.“ Idealerweise hat das Team jedes Sprint Backlog Item fertiggestellt, wichtiger ist aber die Erreichung des Sprintziels, das im Sprint Planning gemeinsam vereinbart wurde. Präsentiert werden beim Review lediglich die Items, die tatsächlich auch fertig – entsprechend der Definition of Done – gestellt wurden.

Am Sprint Review nehmen das gesamte Entwicklungsteam – also die Entwickler, der Scrum Master und der Product Owner – und idealerweise ausgewählte Stakeholder – also bspw. Anwender, Manager, Vertreter aus Bereichen wie Marketing, Vertrieb oder IT – teil. Natürlich adressiert das Sprint Review nicht alle Stakeholder; Wettbewerber sind auch von den Ergebnissen betroffen, sie werden aber nicht dazu beitragen wollen, das Produkt eines Konkurrenten zu verbessern. Im Fokus des Meetings steht vor allem der Austausch der Entwickler mit dem Product Owner. Kann dieser nicht alle fachlichen Aspekte abdecken, sollte der Auftraggeber am Sprint Review teilnehmen oder einen Stellvertreter schicken. Ist dies nicht möglich, verschiebt sich die Abnahme entsprechender User Storys. Bei komplexen Storys sollte der Auftraggeber schon vor dem eigentlichen Review eingebunden werden, denn sonst könnte die Abnahme zu umfangreich werden und scheitern.



Wie sieht ein möglicher Ablauf des Sprint Reviews aus? Nach einer kurzen Begrüßung – ein Unfreezing ist an sich unnötig und geht gegen die vereinbarte Timebox – durch den Product Owner und einer kurzen Vorstellung der Teilnehmer, sofern diese sich nicht kennen, folgt die Erläuterung der Regeln. Eine goldene Regel wie bei der Sprint Retrospektive ist besonders nützlich. Bevor das Team mit der Präsentation der erledigten User Storys beginnt, sollte der Product Owner eine Liste präsentierten und gegebenenfalls verteilen, aus der ersichtlich wird, welche Items umgesetzt wurden und demonstriert werden und welche nicht. Diese Liste sollte im Vorfeld zwischen dem Product Owner und dem Entwicklungsteam abgestimmt werden, denn aus ihr ergibt sich auch die spätere Präsentationsreihenfolge der umgesetzten Items. In manchen Organisationen wird diese Liste bereits bei der Einladung zum Sprint Review verschickt, denn so fällt es Besuchern aus Marketing oder Vertrieb leichter zu entscheiden, ob eine Teilnahme für sie sinnvoll ist. Das Herzstück des Sprint Reviews ist die Präsentation der neuen Funktionalität, idealerweise durch die Entwickler selbst. Hier wird das Feedback der Teilnehmer abgefragt und die User Storys werden bestenfalls abgenommen, gegebenenfalls aber eben auch nicht abgenommen. Bevor es zu einer Diskussion über die demnächst anstehenden Backlog Items kommt, kann der Scrum Master noch die größten Herausforderungen des Sprints erläutern. Meist beziehen sich diese Punkte eher auf den Prozess als auf die Umsetzung der Anforderungen. Das Sprint Review sollte mit einem Dank an die Entwickler und Präsentatoren und dem Termin für das nächste Sprint Review enden.

Es gibt eine Reihe von Tipps und Tricks, die zum Gelingen eines Sprint Reviews beitragen können:

- Die Anzahl der Teilnehmer kann sehr groß werden, daher sind eine gute Moderation, klare Regeln und Abläufe, sowie die konsequente Ausrichtung an der vereinbarten Timebox sehr wichtig.
- Meist ist der Product Owner auch der Moderator des Scrum Events. Er hat also zwei Hüte auf, die er auf jeden Fall trennen muss. Er darf natürlich Feedback geben, aber er sollte sich bspw. bei der direkten Einplanung von neuen User Storys aufgrund des gelieferten Feedbacks zurückhalten.

- Die Teilnahme von wichtigen Stakeholdern ist wesentlich, denn durch ihr Feedback wird der tatsächliche Wert der Realisierung eindeutig verifiziert.
- Geringe Präsentationsfähigkeiten der Entwickler können bei Stakeholdern zu mangelnder Akzeptanz führen. Hier sollte eine bewusste Entscheidung getroffen werden, ob denn eine interne Regel – das Feature wird von dem Entwickler präsentiert, der es umgesetzt hat – beibehalten oder im besonderen Fall alternativ gehandhabt wird.
- Die Freigabe von realisierten User Storys ist wichtig, aber das Feedback ist viel wichtiger. Aus dem Feedback ergeben sich die Freigaben. Verkommt das Sprint Review zu einem reinen Freigabe-Event wird der eigentliche Sinn verfehlt.

Die Durchführung eines Sprint Reviews am Ende eines Sprints bietet verschiedene Vorteile:

- Beim Sprint Review wird sichtbar, was bereits erarbeitet wurde. Da nur realisierte User Storys und deren Umsetzung live im Produkt präsentiert werden dürfen, ist der Fortschritt der Entwicklung für alle Beteiligten direkt ersichtlich.
- Zwischen dem Entwicklungsteam und den Stakeholdern gibt es direktes Feedback und somit Kritik, Lob oder Verbesserungsvorschläge. Diese können gegebenenfalls vom Product Owner für den nächsten Sprint eingeplant werden.
- Die Stakeholder erhalten lebendige Information über die Entwicklung anstelle anonymer Berichte. Somit ist die Auseinandersetzung mit den Ergebnissen deutlich leichter.
- Der Vorbereitungsaufwand ist für Entwickler gering, denn nur fertige Lösungen werden präsentiert. PowerPoint-Präsentationen oder ähnliche Medien sind im Review nicht zulässig.
- Die Identifikation des Teams mit eigenen Arbeitsergebnissen steigt.
- Ideen für weitere Funktionalitäten lassen sich gemeinsam entwickeln.

Es gibt gute Gründe, warum Features einer Software live präsentiert werden, bevor die Software vollständig entwickelt ist. Auch wenn es Manager gibt, die für solche Präsentationen wenig Zeit investieren wollen, so ist es auf Dauer die einzige Möglichkeit, einen eigenen Blick auf den Fortschritt einer Entwicklung zu werfen. Fortschrittsmeldungen per Mail oder Informationen zwischen Tür und Angel können funktionieren, es ist aber deutlich wahrscheinlicher, dass am Ende Ergebnisse entstehen, die für den einen oder anderen Verantwortlichen überraschend sind. Durch die regelmäßige Präsentation der Ergebnisse erhalten das Management und auch wichtige andere Stakeholder die Möglichkeit, eigene Eindrücke zu gewinnen, Fehlentwicklungen zu erkennen, Feedback zu geben oder Wünsche und Ideen zu äußern. Das Sprint Review bietet eine hervorragende Möglichkeit der Steuerung, es ist eine Art Entwicklungs- und Projektcontrolling in Echtzeit. Aussagen wie „Wir sind im Plan“ können nun selbst überprüft werden – alles was dazu benötigt wird, ist einmal im Monat ein Investment von 4 Stunden oder weniger. Darüber hinaus transportiert auch die Teilnahme der Stakeholder eine Aussage in Richtung Entwicklungsteam: „Wir interessieren uns für eure Arbeit und eure Ergebnisse“. Die Alternative könnte als Ignoranz wahrgenommen werden und sogar zu Demotivation führen.

Sprint Retrospektive

Eine Retrospektive ist ein Rückblick. Als Begriff in der Kunst beschreibt er die Auseinandersetzung mit Werken eines Künstlers und/oder einer Epoche. In Scrum ist eine Retrospektive ein regelmäßiges Event, zu dem sich das Entwicklungsteam trifft, um die jüngere Vergangenheit – also den zurückliegenden Sprint – zu beleuchten und dadurch die zukünftige Zusammenarbeit im Team zu verbessern. Es ist ein Meeting, bei dem Prozesse, Werkzeuge, Fähigkeiten, Beziehungen, Herausforderungen und Erfahrungen reflektiert werden. Das Feedback bietet dabei Chancen sowohl für das Team als Ganzes als auch für jeden einzelnen Teilnehmer.

An der Retrospektive nehmen das Entwicklungsteam und der Scrum Master teil. Über die Teilnahme des Product Owners gibt es unterschiedliche Auffassungen. Der Scrum Guide definiert „*The Sprint Retrospective is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.“*. Daraus ergibt sich die Möglichkeit der Teilnahme des Product Owners, denn er gehört neben den Entwicklern und dem Scrum Master zum Scrum Team. Alternativ könnte es auch reine Team Retrospektiven (also ohne Scrum Master und Product Owner) oder Overall Retrospektiven mit einem Vertreter des Teams, dem Scrum Master, dem Product Owner und einem Manager geben.

Die Durchführung der Retrospektive lässt sich in 5 Phasen einteilen:

1. Intro mit Begrüßung, Klärung der Ziele, Rückblick auf das letzte Treffen.
2. Sammeln von Daten bspw. mit den oben beschriebenen Fragen.
3. Gewinnen von Einsichten, also die Klärung von „Wieso, Weshalb, Warum“.
4. Festlegen von Maßnahmen.
5. Retrospektive der Retrospektive mit der Frage, was beim nächsten Treffen wiederholt oder verändert werden sollte.

Für diesen Ablauf ist der Scrum Master verantwortlich. Meist ist er Moderator und evtl. auch Motivator. Er wählt die Methoden aus und sorgt für eine offene, vertrauensvolle und ehrliche Kommunikation. Wichtig ist zu wissen, dass der Scrum Master kein Vorgesetzter ist, er ist Teil des Teams und versucht daher ebenfalls seine Leistung und das Miteinander mit den Entwicklern zu optimieren. Sollte der Product Owner am Meeting teilnehmen, sorgt der Scrum Master dafür, dass dieser nicht mehr Redeanteile als andere erhält. Er benötigt also ein Standing im Team und gegenüber dem Product Owner.

Bei der Durchführung von Retrospektiven sind zwei einfache, aber wichtige Regeln zu beachten:

1. Vegas-Regel: „What happens in Vegas, stays in Vegas.“ Alles was in der Retrospektive besprochen wird, bleibt den Teilnehmern der Retrospektive vorbehalten. Das Treffen zur Verbesserung der Zusammenarbeit im Team und der gemeinsamen Entwicklungsleistung basiert auf Vertrauen. Gerade beim Feedback untereinander ist dieses Vertrauen elementar.
2. Goldene Regel: Alle Teilnehmer behandeln sich mit Respekt und Wertschätzung. Es geht nicht um Schuldzuweisungen und Besserwisserei, es geht um das Team, in das sich jeder mit bestem Wissen und Fähigkeiten einbringt.

Die Retrospektive als Treffen des Scrum Teams verfolgt verschiedene Ziele:

- Die Verbesserung der Zusammenarbeit im Team und somit um die Verbesserung von Abläufen und Inhalten. Es geht auch um das Zusammenspiel zwischen einzelnen Entwicklern, um das Wirken des Scrum Masters und die Kommunikation mit dem Product Owner. Damit ist die Retrospektive wichtiger Teil eines kontinuierlichen Verbesserungsprozesses (KVP).
- Die Festlegung von Maßnahmen, von Dos und Don'ts, basierend auf den gemeinsam gewonnen Erkenntnissen und Überprüfung der vereinbarten Maßnahmen der vergangenen Retrospektive.
- Raum und Gelegenheit für offenes Feedback im Team, zur Vermeidung von Frust und Beseitigung von Missverständnissen.

Es gibt verschiedene Methoden, um Retrospektiven erfolgreich zu gestalten:

- Die *Starfish Retrospektive* hilft den Teilnehmern über verschiedene Grade der Zusammenarbeit nachzudenken und Aspekte entsprechend in einem Diagramm zu visualisieren: Start Doing (ab sofort nutzen), More of (mehr davon), Less of (weniger davon), Keep doing (weiterhin nutzen) und Stop Doing (nicht mehr nutzen).
- Die *4L Retrospektive* verfolgt einen ähnlichen Ansatz, jedoch nur mit 4 Kategorien: Loved, Learned, Lacked (was habe ich vermisst?) und Longed for (wonach habe ich mich gesehnt?).
- Die *3S Retrospektive* kennt 3 Kategorien: Start, Spice up, Stop. Ähnliche 3-teilige Gliederungen sind *Mad, Sad, Glad* oder *Plus, Minus, Interesting* oder *Anfangen, Aufhören, Weitermachen*.
- Bei der *BYOSM Retrospektive* beschäftigt sich das Team mit dem idealen Scrum Master, den sich das Team wünschen würde: *Build Your Own Scrum Master*. Dazu werden drei Fragen gestellt: Wie sieht unser perfekter Scrum Master aus? Was muss er über uns wissen? Wie können wir ihm helfen?
- Der Wetterbericht bietet die Möglichkeit, Stimmungen auf einem Flipchart in den Kategorien Sonnenschein, Bewölkung, Regen und Gewitter zu dokumentieren. Ähnlich funktionieren *Steh zu Deiner Meinung*, *Line Dance*, *Happiness Histogramm* und *Der perfekte Sprint*.
- Beim *Lean Coffee* werden Themen gesammelt, gruppiert, bewertet und entsprechend der Bewertung in einer festen Zeitvorgabe diskutiert. Ist das Thema nicht abschließend besprochen, kann die Zeitvorgabe verlängert werden.
- Beim *Pessimieren* wird zu Beginn eine negativ formulierte Frage gestellt (z.B. Was hätten wir getan, wenn wir das Sprintziel verfehlt hätten?), die Antworten auf die Fragen ins Gegenteil verkehrt, um anschließend die Fragen zu beantworten „Was hält uns davon ab?“ und „Warum ist das so?“ Ähnlich funktioniert Das Schlimmste, was wir machen können.
- Bei *Wünsch Dir was* geht es um die Beschäftigung mit der größten Herausforderung bei der Arbeit, dem Wunsch diese zu meistern und der Frage „Woran merkst Du, dass sich Dein Wunsch erfüllt hat?“.
- Bei *Verdeckter Boss* nimmt das Team den Blickwinkel eines Vorgesetzten ein und überlegt sich, was der ändern würde, hätte er unerkannt den letzten Sprint beobachten können.

Es gibt noch viele weitere Methoden und Möglichkeiten, um die Eindrücke und Meinungen der Teammitglieder in Erfahrung zu bringen. Es obliegt dem Scrum Master, hier einerseits neue Methoden einzubringen, andererseits an bewährten Techniken festzuhalten. Eine Retrospektive darf trotz aller Ernsthaftigkeit auch Spaß machen.

Neben den verschiedenen Methoden, die sich gut für den Meinungsaustausch im Team eignen, besteht auch die Möglichkeit, viele einzelne Fragen zu stellen, um Befindlichkeiten, Werte, Ziele, Vorgehensweisen etc. zu erkennen:

- Was kannst Du von mir erwarten?
- Was erwarte ich vom Team / von Dir?
- Was wünsche ich mir vom Team / von Dir?
- Was möchte ich ab sofort anders / besser tun?
- Was schätze ich an Dir und an Deiner Arbeit?
- Was wünsche ich Dir, dass Dir besser gelingt?
- Worauf kann das Team / kannst Du Stolz sein?
- Was waren positive / negative Momente im letzten Sprint?
- Was war großartig für Dich / das Team im letzten Sprint?
- Was hat Dich / das Team angetrieben / gebremst?
- Was hast Du am letzten Sprint gehasst?

- Wie findest Du das Zusammenspiel und die definierten Prozesse im Team und was würdest Du ändern?
- Wie finden die anderen Teammitglieder die Zusammenarbeit und die Beziehungen untereinander?
- Wo waren für Dich / das Team die Stolpersteine im letzten Sprint?
- Was behindert Dich / das Team bei der Arbeit?
- Was würdest Du gerne mal ausprobieren?
- Was lief gut und müssen wir uns merken, damit wir es nicht vergessen?
- Wie wird unser Team von außen wahrgenommen?
- Was hast Du / hat das Team im letzten Sprint gelernt?
- Welche Skills möchtest Du /sollte das Team verbessern?
- Wie hast Du Dich / glaubst Du haben sich Deine Kollegen gefühlt?
- Wo hättest Du mehr Hilfe benötigt / geben können?
- Was hat Dich / das Team überrascht?
- Was sollten wir im nächsten Sprint unbedingt verbessern?
- Was wünschst Du Dir vom Scrum Master / Product Owner / Manager?
- Wie findest Du denn Einsatz der Tools und was würdest Du verändern?
- Was nimmst Du aus der Retrospektive mit?
- Wie findest Du die Retrospektive im Sinne von Offenheit, Ehrlichkeit und Respekt?

Eine Retrospektive bietet viel Raum zur Beantwortung der unterschiedlichsten Fragen. Zwei Fragen sind zum Ende einer Retrospektive wichtig zu beantworten: 1. Wie nützlich war die Retrospektive für Dich (auf einer Skala von 1-5)? Und 2. Was kann der Moderator beim nächsten Mal besser machen?

Backlog Refinement

Das Backlog Refinement ist kein offiziell definierter Bestandteil von Scrum wie bspw. das Daily Scrum oder das Sprint Planning, allerdings empfiehlt es sich, es als regelmäßige Aktivität oder Meeting durchzuführen. Der Scrum Guide formuliert es wie folgt: „Product Backlog refinement is the act of **adding detail**, **estimates**, and **order** to items in the Product Backlog. This is an **ongoing process** in which the **Product Owner and the Development Team** collaborate on the details of Product Backlog items. During Product Backlog refinement, items are **reviewed** and **revised**. The Scrum Team decides how and when refinement is done. Refinement usually consumes no more than **10% of the capacity of the Development Team**. However, Product Backlog **items can be updated at any time by the Product Owner** or at the Product Owner's discretion."

Es geht also beim Backlog Refinement bzw. Estimation um folgende Aspekte:

- Detaillierung des Product Backlogs
- Schätzung der Backlog Items
- Reihenfolge der Backlog Items

- Kontinuierlicher Prozess mit Product Owner und Entwicklungsteam
- Begutachtung und Überarbeitung der Backlog Items
- Aufwandsbegrenzung auf nicht mehr als 10% der Gesamtkapazität des Entwicklungsteams
- Rechte des Product Owners, jederzeit Informationen zu aktualisieren

Das Ziel beim Backlog Refinement ist es, die höher priorisierten Items im Backlog so vorzubereiten, dass sie sich gut für das Sprint Planning nutzen lassen. Dies führt in der Folge zur Verkürzung von Planungsmeetings bei gleichzeitiger Berücksichtigung aktueller Informationen. Der Zweck von Timeboxing ist es, für Projekte, Vorgänge und Aktivitäten Zeiten festzulegen und diese zu begrenzen. Wichtig ist dabei, die Dauer einer Timebox sinnvoll zu bestimmen. Beim Backlog Refinement steht aber die Aktualität der Backlog Items über allem, denn diese hat viele Auswirkungen auf die weitere Entwicklung. Es ist schwer vorstellbar, dass diese wichtige Tätigkeit beendet wird, nur weil das Ende einer Timebox erreicht wird.

Auch wenn der Scrum Guide explizit von Product Owner und Entwicklungsteam spricht, empfiehlt es sich, dass auch der Scrum Master an dem Backlog Refinement bzw. Estimation teilnimmt. Er kann steuernd eingreifen, wenn Diskussionen länger als beabsichtigt dauern – hier bietet sich das Diskutieren mit zeitlichen Begrenzungen pro User Story an. Bei der Definition einer guten User Story leistet der INVEST Ansatz von William Wake gute Dienste. Für das Team ist es wichtig, die User Storys gut sehen zu können, bspw. als Ausdruck an einer Wand. Hier hat sich das Arbeiten mit User Story Maps bewährt.

Gemeinsam erörtern die Beteiligten Akzeptanzkriterien und sorgen für die Vollständigkeit der Storys, die demnächst zur Umsetzung anstehen. Auch die gemeinsame Schätzung – gerne per Story Points als Maß für die Komplexität einer User Story – ist sehr wichtig. Ergeben sich Diskrepanzen in der Aufwandsschätzung könnte dies am unterschiedlichen Verständnis oder verschiedenen Realisierungsoptionen liegen. Dies würde zu erneuten Schätzungen oder der vorherigen Definition von Technical Storys oder Spike Storys führen.

Unternehmen stehen vor der Herausforderung, Backlog Refinements effizient zu planen und durchzuführen. Hierfür gibt es verschiedene Tipps:

- Die Vorbereitung

  Die Vorbereitung ist Aufgabe des Product Owners. Idealerweise sind die demnächst anstehenden User Storys bereits periodisiert, denn dies spart allen Beteiligten viel Zeit. Der Product Owner muss das Meeting gut vorbereiten. Die wichtigen anstehenden Storys des Backlogs müssen priorisiert, klar verständlich und detailliert genug beschrieben sein (idealerweise nach dem INVEST-Ansatz).

- Die Entwicklung von Akzeptanzkriterien

  Es empfiehlt sich, Akzeptanzkriterien gemeinsam im Team zu definieren, denn dies führt zu einem besseren, gemeinsamen Verständnis der Beteiligten. Alternativ könnten die Kriterien auch nur vom Product Owner definiert werden; so ließe sich zwar der Aufwand reduzieren, aber Verständnislücken könnten unter Umständen unentdeckt bleiben.

- Den Fokus im Blick

  Der Fokus im Backlog Refinement liegt nicht in der Aufwandsschätzung, sie ist lediglich ein Bestandteil von mehreren wie bspw. die Aufnahme von neuen User Storys, die Eliminierung von vorhandenen Items oder das Splitten von einzelnen User Storys.

- Die Frequenz

  Häufig liest man von wöchentlichen Backlog Refinements, doch dies macht nicht für jede Organisation Sinn. Besser wäre hier eine Orientierung an den eigenen Möglichkeiten und Vorstellungen. Dauert ein Sprint bspw. 4 Wochen, könnte es sinnvoll sein, alle 2 Wochen ein entsprechendes Meeting durchzuführen. Unabhängig von der Frequenz muss das Backlog Refinement natürlich immer vor dem nächsten Sprint Planning durchgeführt werden.

- Die Dauer eines Refinements

  Definieren Sie eine feste Dauer und nutzen Sie Ihren Scrum Master, denn er kann auf die Einhaltung der vereinbarten Regeln achten und sorgt so bspw. dafür, dass Meetings und Diskussionen in den Meetings nicht zu viel Zeit beanspruchen. Dauert das Backlog Refinement länger als ursprünglich geplant, ist es meist besser, einen neuen Termin zu vereinbaren.

- Die digitale Nachbereitung

  Für viele Unternehmen ist Traceability und Revisionssicherheit beim Arbeiten mit User Storys wichtig. Hier gibt es verschiedene Lösungen von Softwareherstellern, die Ihnen beim Digitalisieren Ihrer Ergebnisse helfen können. Es ist Geschmacksache, ob Sie den gesamten Prozess von Anfang an Digitalisieren oder mit haptischen User Storys und Story Mapping arbeiten wollen.

Scrum Begriffe

In Scrum gibt es verschiedene Begriffe, die regelmäßig genutzt werden. Dazu gehören u.a. das Backlock, die Timebox, die User Story, das Taskboard, die Definition of Done und Scrum of Scrums.

Backlog

Die Verwaltung von Informationen in Backlogs ist in Scrum sehr populär. Ein Backlog ist eine Sammlung von Dingen, insbesondere unvollständiger Arbeit oder Angelegenheiten, die erledigt werden müssen. Oder einfach ausgedrückt: Ein Backlog ist eine Liste von Aufgaben bzw. Anforderungen, die abgearbeitet bzw. realisiert werden sollen. Obwohl Scrum die Verwendung von Product Backlog und Sprint Backlog (neben dem Product Inkrement) als Artefakte definiert, ist die Organisation von Backlogs in Unternehmen unterschiedlich organisiert. So finden sich häufig Release Backlogs und Team Backlogs. Beim Sprint Planning 1 definiert der Product Owner sein Selected Backlog als Wunschliste – die abgestimmten User Storys wandern anschließend in das Sprint Backlog oder werden entsprechend referenziert. Der Scrum Master pflegt in einem Impediment Backlog die Hindernisse, die im Zuge der Daily Scrums kommuniziert wurden. Darüber hinaus können Entwickler bzw. Mitarbeiter auch persönliche Backlogs pflegen. Generell ist die Pflege und Priorisierung der Backlog Items eine wichtige und kontinuierliche Aufgabe.

Timebox

Timeboxing ist eine Technik zur Terminierung von Projekten, Vorgängen und Aktivitäten. Sie definiert einen Zeitrahmen und gewichtet den Faktor Zeit höher als die Faktoren Ressourcen und Inhalte. Im Gegensatz zum klassischen Projektmanagement mit der Definition von Arbeitspaketen und der Fixierung von zu erbringenden Leistungen, legt eine Timebox die Dauer und den Zeitrahmen für eine Arbeit fest. Zur Einhaltung des vorgegebenen Zeitrahmens, können sich Inhalte und Umfänge im Laufe einer Timebox verändern. Vorgänge werden somit nach einer festgelegten Dauer zum definierten Zeitpunkt beendet, selbst wenn nicht alle geplanten Inhalte des Vorgangs realisiert werden konnten. Nicht realisierte Inhalte werden auf nachfolgende Timeboxes verschoben oder ggfs. aufgrund neuer Erkenntnisse gestrichen. Der große Vorteil von Timeboxing besteht darin, dass es eine Planung unterstützt, auch wenn Unternehmen in einem dynamischen Umfeld agieren und gewünschte Ergebnisse sich nicht immer vorab und vollständig spezifizieren lassen.

Timeboxing ist eine zentrale Technik in Scrum. Sie ist Bestandteil sowohl beim Sprint Planning, als auch im Sprint selbst, sie findet Anwendung bei den Daily Scrums, dem Sprint Review und der Sprint Retrospektive. Es gibt sogar Organisationen, da wird Timeboxing nicht nur für Sprints, sondern sogar innerhalb einzelner Sprints verwendet, bspw. zur Beseitigung von Unklarheiten, zur Durchführung von Recherchen oder beim Arbeiten mit Spike Storys. Ein wichtiger Gedanke beim Timeboxing ist die klare Definition of Done, durch die Inhalte, die nicht allen Akzeptanz- und Fertigstellungskriterien entsprechen, in zukünftigen Timeboxes behandelt werden. Zusätzlich ermutigt das Arbeiten mit Timeboxing Teams auch, zügig mit der Arbeit zu beginnen, denn so erhalten sie frühzeitig Feedback und können Lieferobjekte entsprechend schnell anpassen.

Folgende Timeboxes werden für die verschiedenen Scrum Events empfohlen:

- Die Realisierung von Anforderungen erfolgt in Scrum im Rahmen der Sprints. Im Gegensatz zu einer klassischen Iteration sind Sprints mit einer Dauer von einer bis maximal vier Wochen sehr kurz. Welche Timebox für ein Unternehmen am besten passt, lässt sich nicht allgemeingültig beantworten. Bei der Festlegung können sich Unternehmen an der geplanten Gesamtlaufzeit der Entwicklung, an der Unternehmenskultur und/oder dem Entwicklungsgegenstand orientieren.
- Bei einem einmonatigen Sprint eine Timebox für das Sprint Planning von 8 oder weniger Stunden. Je kürzer der Sprint wird, desto kürzer sollte der Zeitrahmen für das Sprint Planning sein. Bei einer Sprintdauer von einer Woche wären also 2 oder weniger Stunden angemessen.
- Das Daily Scrum ist eine Timebox von 15 Minuten, in dem das Team Aktivitäten untereinander synchronisieren, Hindernisse für das Erreichen der Ziele sowie anstehende Tasks bespricht.
- Für einmonatige Sprints empfiehlt sich eine Timebox von 4 Stunden oder weniger für ein Sprint Review, bei einem dreiwöchigen Sprint von 3 Stunden oder weniger, usw.
- Bei einem einmonatigen Sprint eignen sich meist 2 Stunden als Timebox für die Retrospektive.
- Das Backlog Refinement ist kein offiziell definierter Bestandteil von Scrum, dennoch legt der Scrum Guide die Durchführung von regelmäßigen Meetings zwischen dem Product Owner und dem Entwicklungsteam nahe. Als Orientierung für eine Timebox nennt der Scrum Guide „no more than 10% of the capacity of the Development Team“.

User Story

Der Begriff „User Story“ stammt aus dem Englischen. Das Wort *user* bedeutet Anwender oder Benutzer, *story* bedeutet Geschichte. Im wörtlichen Sinne beschreibt eine User Story also eine Geschichte eines Anwenders. Im agilen Projektmanagement und der Softwareentwicklung ist eine User Story ein Werkzeug, um gewünschte Funktionalitäten eines Systems aus Sicht des Anwenders zu beschreiben. Dabei bietet eine User Story vor allem drei Vorteile:

- sie ist leicht zu verstehen und vermittelt die Wünsche der Anwender
- sie ist schnell erstellt und erleichtert die Schätzung des Aufwands zur Realisierung
- sie lässt sich schrittweise detaillieren und unterstützt so die iterative Entwicklung

Im Gegensatz zu einer Geschichte, die Sie an einem Lagerfeuer erzählen, sollte eine User Story in kurzen Sätzen und mit einfachen Worten beschrieben werden. So stellen Sie sicher, dass alle an Ihrer Entwicklung beteiligten Kollegen sie auch ohne technischen Hintergrund verstehen. Es geht vor allem um *Wer*, *Was* und *Warum*, also wer möchte was von einem System, um welchen Nutzen davon zu haben. Wie die gewünschte Funktionalität später umgesetzt wird, ist für eine User Story unerheblich. Beim Schreiben einer User Story empfiehlt sich folgende Satzstruktur:

Als (Rolle) möchte ich (Funktionalität), um (Nutzen) zu erreichen.

Achten Sie bei der Formulierung darauf, sie aus der Sicht des Anwenders, Benutzers oder Kunden zu schreiben und einen wirklichen Nutzen für ihn zu erzeugen.

Hier finden Sie einige Beispiele für User Storys:

- Als Filmliebhaber möchte ich über neue Filme informiert werden, um zu wissen, welche Filme als nächstes im Kino laufen.
- Als Filmliebhaber möchte ich einmal pro Woche einen Newsletter erhalten, um zu wissen, welche Filme als nächstes im Kino laufen.
- Als Filmliebhaber möchte ich einmal pro Woche per Mail über neue Science Fiction Filme informiert werden, die im ABC Kino laufen, um mir für dieses Kino Tickets online buchen zu können.

Selbst mit der empfohlenen Satzstruktur können User Storys unterschiedlich detailliert sein. Sie können sie relativ grob und nach und nach mit zusätzlichen Details verstehen, so lange, bis sie detailliert genug ist, um sie umzusetzen.

Der unterschiedliche Umfang von Funktionalitäten eines Systems und Nutzen aus Sicht eines Anwenders führt häufig zu den Kategorisierungen *Epic*, *Feature* und *User Story*. Die Unterscheidung in diese drei Kategorien ist jedoch nicht standardisiert, d.h. in manchen Unternehmen sind Features umfangreicher als Epics. Wichtig ist daher, dass es im Unternehmen ein einheitliches Verständnis gibt. Die Unterscheidung zwischen Epics, Features und User Storys könnte bspw. wie folgt aussehen:

- Epics beschreiben Funktionalitäten auf höchstem fachlichen Niveau.
- Features verfeinern die Funktionalitäten von Epics und lassen sich für die Release-Planung verwenden. Sie sind aber noch zu umfangreich, um in einer Iteration bzw. einem Sprint umgesetzt zu werden.

- User Storys verfeinern Features und lassen sich in Sprints einplanen und umsetzen.

Es kommt vor, dass von Softwareentwicklern geschriebene Pflichtenhefte für Fachabteilungen schwer verständlich sind. Solche Kommunikationsprobleme werden mit agilen Methoden wie Scrum adressiert. In Scrum hat der Product Owner die Aufgabe, aus den Epics, Features und User Storys genauso viele Anforderungen zu generieren, wie im nächsten Sprint realisiert werden können. Im Gegensatz zur Formulierung eines Lastenhefts wird die Vorlaufzeit zur Planung und Realisierung reduziert und die inhaltliche Flexibilität erhöht. Dieser Ansatz ist sehr pragmatisch, stellt aber Anforderungen auf anderen Ebenen wie bspw. Vertragsgestaltung, Aufwandsschätzung und Leistungsabrechnung.

Mit einem *Use Case* beschreiben Sie, wie ein Akteur mit einem zu entwickelnden System interagiert. Use Cases sind sehr beliebt, denn sie ermöglichen ebenso wie User Storys die Erhebung von Anforderungen an ein System. Und beide verfolgen das Ziel, einen Mehrwert für den Anwender zu schaffen. Worin liegen aber die Unterschiede zwischen Use Cases und User Storys? Ein Use Case ist ist größer und detaillierter als eine User Story. Er deckt einen Kontext ab und ist damit umfangreicher. Er umfasst somit mehrere User Storys. Und er ist deutlich langlebiger, d.h. er wird über die gesamte Systementwicklung gepflegt, während die User Story mit ihrer Erledigung in einem Sprint praktisch verschwindet.

Use Cases und User Storys ergänzen sich sehr gut. Durch die Kombination beider Methoden lassen sich Anforderungen genauer verstehen. Um Use Cases wie User Storys in eine Sprint-Planung aufzunehmen, muss man sie zu *Use Cases Slices* zerschneiden. Diese Technik nennt sich *Use Case 2.0*.

Entwicklungsteams nutzen neben User Storys oft auch sogenannte *Technical Storys*. Mit ihnen werden weder Funktionalitäten noch ein Nutzen eines Anwenders beschrieben, sondern Kriterien, die den technischen Aufwand hinter einer User Story festhalten. So versuchen Entwicklungsteams nicht-funktionale Anforderungen bspw. bezüglich Performance, Skalierung, Sicherheit oder Verfügbarkeit zu definieren. Wichtig ist bei der Erfassung, dass es eine eindeutige Trennung von Technical Storys und User Storys gibt. Durch eine entsprechende Kennzeichnung vermeiden Sie Missverständnisse bei der Priorisierung und Sprint-Planung bspw. im Rahmen eines *User Story Mappings*.

Eine *Spike Story* ist eine Art User Story, die verwendet wird, um eine funktionale oder technische Anforderung besser zu verstehen, und so Unsicherheiten zu beseitigen. Sie ist ein Auftrag für eine Analyse, mit der eine Frage beantwortet, Informationen gesammelt oder Projektrisiken adressiert werden. Im Gegensatz zu einer User Story wird sie nicht geschätzt. Das Entwicklungsteam verpflichtet sich, eine bestimmte Zeit zu investieren, um die notwendige Analyse durchzuführen. Das Ergebnis führt zur Überarbeitung der User Story im Sinne einer Verfeinerung, einer Aufteilung in mehrere User Storys oder der Beschreibung einer gänzlich neuen User Story.

Wie schätzen Sie den Aufwand einer User Story? Die Antwortet lautet: in Personentagen oder mit *Story Points*. Beim Arbeiten mit Story Points bestimmt das Entwicklungsteam ein Kriterium oder eine Verknüpfung von mehreren Kriterien, um die Größe der User Story festzulegen. Ein solches Kriterium könnte *Komplexität* sein, bspw. in Bezug auf die Verwendung von verschiedenen Schichten des Architekturmodells. Es geht also bei Story Points nicht um die Zeit, die zur Umsetzung der User Story benötigt wird, sondern um die strukturellen Eigenschaften einer User Story. Natürlich kann ein erfahrenes Entwicklungsteam im Vergleich zu einem weniger erfahrenen Team größere User Storys in einem Sprint

umsetzen, doch die Größe der Story bleibt davon unberührt. In anderen Worten: die Eigenschaften einer User Story hängen nicht von den Fähigkeiten des Teams ab.

In Scrum drückt die sogenannte *Velocity* aus, wie viele Story Points ein Team pro Iteration umsetzt, so dass auch beim Arbeiten mit Story Points Aussagen getroffen werden können, in welcher Iteration ein Feature geliefert wird. In der Praxis können Merkmale einer User Story häufig ohne detaillierte Tätigkeitenanalyse identifiziert werden. Dies ist ein wesentlicher Unterschied zu der Aufwandsschätzung in Personentagen, bei der alle notwendigen Tätigkeiten identifiziert und summiert werden. Oft wird eine User Story mit mittlerer Größe als Referenz ausgewählt und die Schätzung im Vergleich zu dieser Referenz durchgeführt. Als Wertebereich dient eine Fibonacci-Reihe bis 13, ergänzt mit 20, 40 und 100, wobei 1 einer Aufgabe mit niedriger Komplexität und 100 einer noch nicht einschätzbare Aufgabe entspricht.

Wie stellen Sie fest, ob Sie eine qualitativ gute User Story definiert haben? William Wake bietet mit seinem INVEST-Prinzip Hilfe bei der Formulierung von User Storys. INVEST ist ein Akronym:

- Indepentent: Die User Story steht für sich und ist unabhängig von anderen Storys.
- Negotiable: Der Inhalt eine User Story ist verhandelbar und wird nach und nach detaillierter beschrieben, bis sie sich umsetzen lässt.
- Valuable: Die User Story ist wertvoll und bietet dem Anwender oder Kunden einen Mehrwert bzw. Nutzen.
- Estimable: Der Aufwand der User Story muss sich durch die Entwickler schätzen lassen.
- Small: Die User Story ist so klein, dass sie innerhalb einer Iteration bzw. eines Sprints realisierbar ist.
- Testable: Die User Story verfügt über *Akzeptanzkriterien* und ist prüfbar.

Wann wissen Sie, ob eine User Story vollständig implementiert wurde? Hier helfen Akzeptanzkriterien. Mit Akzeptanzkriterien legen Sie stichpunktartig Ergebnisse fest, die durch die User Story erfüllt werden müssen.

Akzeptanzkriterien gelten als Bindeglied zwischen User Storys und Testfällen. Eine Möglichkeit, Akzeptanzkriterien zu definieren, ist das Hinterfragen von Schlüsselwörtern, also Verben, Adjektiven und Substantiven. Beispiel:

"Als Kinobesucher möchte ich meine gekauften Tickets in meinem Profil speichern, damit ich nachvollziehen kann, welche Filme ich gesehen habe."

- Wer speichert was, wann, wo?
- Was geschieht bei der Speicherung eines neuen Tickets mit bereits gespeicherten Informationen?
- Wie viele Tickets sollen gespeichert werden können?

Mit solchen W-Fragen finden Sie leicht Akzeptanzkriterien. Als Checkliste sind sie eine gute Basis für die Entwicklung von umfangreichen Testfällen.

Taskboard

Ein zentrales Element beim Arbeiten mit Scrum ist die Visualisierung der Tasks in Form eines Taskboards. Die verschiedenen Tasks zur Umsetzung einer User Story wandern von *To Do*, über *Work in Progress* und *To Verify* zu *Done*. Gibt es Impediments in Bezug auf eine einzelne Aufgabe lässt sich dies leicht mit Punkten visualisieren. Durch die gemeinsame Darstellung der Aufgaben und des Fortschritts im Daily Scrum weiß jedes Teammitglied, was noch getan werden muss, um das gemeinsam vereinbarte Sprint Goal zu erreichen. Die Transparenz und das Bewusstsein um den Status quo werden gesteigert. Zusätzlich zur permanenten Visualisierung der Aufgaben im Büro der Entwickler fördert das manuelle Verschieben der Tasks die Motivation der Mitarbeiter, die gesteckten Ziele zu erreichen. Beim Einsatz von elektronischen Taskboards ist der Effekt weniger stark vorhanden. Und auch das Aktualisieren des Burn-Down Charts hat nicht diesen Effekt.

Definition of Done

Die Definition of Done ist eine Checkliste mit Qualitätskriterien, die beschreibt, welche Kriterien erfüllt sein müssen, damit die Erstellung eines Produkts als erledigt betrachtet werden kann. Da Menschen unterschiedliche Vorstellungen von Qualität haben, ist es wichtig, sich gemeinsam im Team auf eine Definition of Done (DoD) festzulegen. Damit liegt die Verantwortung für die DoD auch beim Team und sie wirkt als Selbstverpflichtung für das Team. Anhand der definierten Kriterien lässt sich fortan feststellen, ob ein Backlog Item, ein Feature oder eine User Story tatsächlich *fertig* bzw. *done* ist.

„Ich bin fast fertig." Oder: „Die Aufgabe ist zu 90% erledigt." Vielleicht haben Sie solche Aussagen auch schon einmal gehört, die Hinweise zum Fortschritt einer Aufgabe liefern, aber nicht sonderlich konkret sind. Was bedeutet „fast fertig"? Welche 10% fehlen noch zur Fertigstellung? Mit einer Definition of Done erzeugen Sie Klarheit. Durch die Verwendung einer DoD wissen Sie, was benötigt wird, damit etwas als „fertig" gelten kann. Diese Klarheit hängt nicht von persönlichen Einschätzungen ab; sie ist das Ergebnis von vereinbarten Kriterien, die allesamt erfüllt sein müssen, so dass aus einem „fast fertig" ein „Fertig" bzw. „Done" wird. Darüber hinaus steht eine Definition of Done für einen Qualitätsanspruch, für eine Ausrichtung auf Werte und Prozeduren.

Sie können eine Definition of Done ausformulieren oder als Kriterienliste anlegen. Wenn Sie Software entwickeln, könnte eine ausformulierte DoD wie folgt lauten:

„Eine User Story gilt als fertig, sofern die Akzeptanzkriterien erfüllt sind, die Codeerzeugung nach Standard XYZ erfolgt ist, der Code in der Versionsverwaltung gesichert, die manuelle Überprüfung durch ein Teammitglied und die automatisierten Tests keine Fehler ergeben haben und die Dokumentation angepasst wurde."

Häufig entscheiden sich Organisationen dazu, ihre Definition of Done als Kriterienliste anzulegen, denn dies bietet die Option, einzelne Kriterien separat abzuhaken. Eine solche Liste könnte folgende Punkte beinhalten:

- Alle Akzeptanzkriterien werden erfüllt
- Der Code ist vollständig implementiert und kommentiert

- Der Code wurde im Pair Programming erarbeitet
- Die Coding Standards von XYZ und die internen Konventionen ABC wurden eingehalten
- Der Code steht unter Versionsverwaltung
- Ein Code Review wurde durchgeführt
- Die Unit-Tests wurden durchgeführt und bestanden
- Eine Dokumentation wurde erstellt
- Ein Eintrag im Change Log wurde angelegt

Bei der Erstellung einer Definition of Done sollten Teams darauf achten, dass die festgelegten Kriterien überprüft werden müssen. Je mehr Kriterien definiert werden, desto mehr Kriterien gilt es zu überprüfen. Würde die Überprüfung nicht stattfinden, wäre schon bald der Sinn der DoD hinfällig. Anspruch und Realität müssen also zusammenpassen.

Es gibt einen wesentlichen Unterschied zwischen Akzeptanzkriterien und einer Definition of Done: Akzeptanzkriterien beziehen sich immer konkret auf ein Item wie bspw. eine User Story. Sie sorgen für mehr Klarheit, was im Rahmen der User Story umzusetzen ist, und sind sie Quelle für Akzeptanztests. Die Definition of Done hingegen bezieht sich bspw. auf das generelle Arbeiten mit User Storys, d.h. die dort formulierten Kriterien sind invariant. Müssen bspw. Entwicklungen vor der Freigabe einer Peer-Prüfung und einem QA-Test unterzogen werden, finden sich diese Kriterien in der DoD und nicht in den Akzeptanzkriterien der User Story wieder. Zum Fertigstellen einer User Story reicht also nicht, lediglich die Akzeptanzkriterien zu erfüllen, auch die Kriterien der DoD müssen erfüllt sein. Grundsätzlich kann die Erfüllung der Akzeptanzkriterien auch ein Kriterium bei der DoD darstellen.

Das Arbeiten mit einer Definition of Done bietet verschiedene Vorteile:

- Es gibt ein gemeinsames Verständnis von Qualität und das Commitment des Teams, diese Qualität zu liefern. Dabei ist der Qualitätsgedanke umfassender, denn er bezieht sich auf mehr als nur eine Aufgabe oder eine User Story.
- Jeder im Team weiß, was von jedem erwartet wird und was das Team als Einheit zu liefern hat.
- Der Anspruch im Team, qualitative Arbeit zu leisten, steigt.
- Die vereinbarten Kriterien lassen sich überprüfen, d.h. die DoD ist ein Arbeitsmittel.
- Es lassen sich für verschiedene Aspekte DoDs definieren, bspw. für User Storys, Features, Sprints oder Releases. So entstehen zu Themen passende DoDs, die sich auch gemeinsam im Team weiterentwickeln lassen.
- Es entsteht Klarheit darüber, ob ein Backlog Item, ein Feature oder eine User Story tatsächlich „done“ ist.

Unternehmen stehen bei der Nutzung einer Definition of Done vor mehreren Herausforderungen:

- Wie gelingt die konsequente Anwendung einer Definition of Done?
- Wie wird das Zusammenspiel zwischen verschiedenen DoDs organisiert?
- Wie lässt sich die Definition of Done weiterentwickeln?

Bei der Nutzung einer Definition of Done stellt sich Organisationen früher oder später die Frage, ob es akzeptabel ist, nicht immer alle Kriterien vollständig zu erfüllen. Vielleicht sind die definierten Kriterien

doch umfangreicher als ursprünglich beabsichtigt, vielleicht ist es in Einzelfällen in Ordnung, technische Schulden zu akzeptieren? Eine allgemeingültige Antwort auf die Frage gibt es nicht; sie liegt in einer Grauzone. Für eine Organisation ist es aber wichtig zu unterscheiden, ob es eine einmalige Abweichung sein wird oder ob die DoD dauerhaft verändert werden soll. Die Weiterentwicklung der Kriterien kann so zur Reduzierung oder im umgekehrten Fall zur Aufstockung von Kriterien führen.

Die Verwendung mehrerer DoDs auf verschiedenen Ebenen kann zu einem Overhead und einem Verlust an Transparenz führen. Sind die Akzeptanzkriterien einer User Story erfüllt, gilt es die Definition of Done zu überprüfen. Die DoD für die Entwicklung von User Storys ist naheliegend. Ergänzend greift bspw. die Definition of Done des Sprints und dann evtl. die des Releases. Wo verwalten Organisationen die entsprechenden Informationen? Wie erfolgt die Abnahme einer User Story, wenn auf einer anderen Ebene ein Kriterium nicht erfüllt wurde? Führt das Arbeiten mit mehreren DoDs in der Konsequenz nicht dazu, dass Kriterien der höheren Ebenen schrittweise in Richtung der User Storys verlagert werden? Auch diese Fragen lassen sich nicht allgemeingültig beantworten. In anderen Worten: Organisationen müssen einen pragmatischen Weg im Umgang mit den DoDs der verschiedenen Ebenen finden und gegebenenfalls nicht nur eine einzelne Definition of Done sondern auch das Zusammenspiel verschiedener DoDs weiterentwickeln.

Scrum of Scrums

Das Scrum of Scrums bezeichnet ein regelmäßiges Treffen von Vertretern einzelner Scrum-Teams. Es dient dem Zweck, sich gegenseitig über den Status quo der einzelnen Teams, über anstehende Tätigkeiten und mögliche Hindernisse bei der Entwicklung auszutauschen. Ziel des Scrum of Scrums ist es, die Arbeit der verschiedenen Scrum-Teams zu synchronisieren und Entwicklungen von Teams zu identifizieren, die andere Teams bei der Umsetzung ihrer Anforderungen beeinflussen. Idealerweise schickt jedes Team einen Vertreter zum Scrum-of-Scrums-Meeting, so dass alle Teams gleichgewichtig repräsentiert werden. Die Arbeit der Scrum-of-Scrums Meetings kann auch auf einer höheren Ebene – mit Vertretern der verschiedenen Scrum-of-Scrums-Runden – fortgeführt werden. Auch wenn Formulierungen wie *Scrum-of-Scrum-of-Scrums* leicht verständlich sind, haben sie sich in der Scrum-Community nicht durchgesetzt.

Daily Scrum und Scrum of Scrums sind sehr ähnlich. Beim Daily Scrum darf jedes Teammitglied drei Fragen beantworten:

- Was habe ich seit gestern getan?
- Was mache ich bis morgen?
- Was behindert mich bei meiner Arbeit?

Da beim Scrum of Scrums jeder Teilnehmer sein als Team-Botschafter auftritt, werden die Fragen umformuliert:

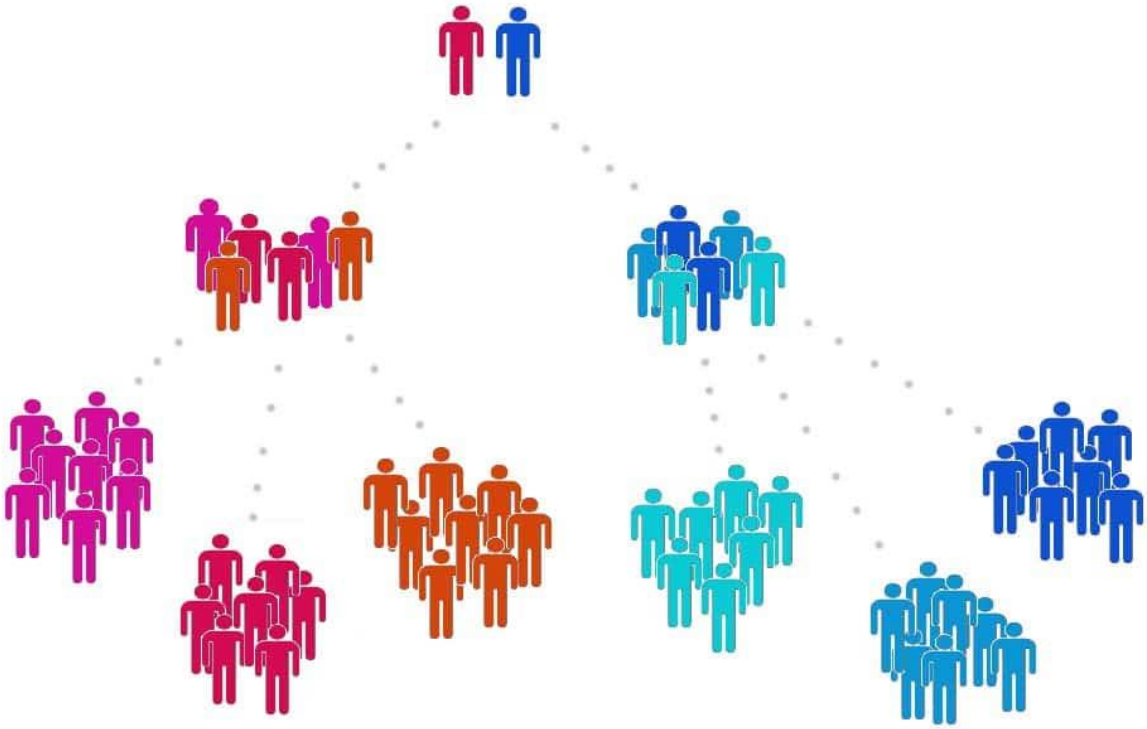
- Was hat mein Team geschafft, seit wir uns das letzte Mal getroffen haben?
- Was wird mein Team bis zum nächsten Treffen erledigen?
- Welche Hindernisse behindern mein Team bei der Arbeit?
- Könnte eine Tätigkeit meines Teams ein anderes Team beeinflussen oder behindern?

Die Antwort auf die letzte Frage ist besonders wichtig, denn die Erkenntnisse müssen zwingend von den Vertretern an die eigenen Teams weitergegeben werden.

Die Meinungen zu den Häufigkeiten eines Scrum-of-Scrums variieren. Oft wird ein tägliches Treffen empfohlen, doch auch andere Rhythmen wie einmal oder zweimal pro Woche können ausreichen. Wichtig ist, dass nicht jedesmal ein neuer Termin mit den Teilnehmern im Scrum of Scrums abgestimmt wird, sondern Termine im Vorfeld feststehen. So könnten sich die Vertreter bspw. jeden Dienstag und Donnerstag treffen und ein Scrum-of-Scrum-of-Scrums jeden Freitag stattfinden.

Wie beim Daily Scrum empfiehlt sich auch beim Scrum of Scrums mit einer Timebox zu arbeiten. Die Dauer sollten die Vertreter gemeinsam und in Abhängigkeit der Teilnehmeranzahl festlegen. Manche Organisationen entscheiden sich bewusst gegen ein tägliches Treffen von 15 Minuten, sondern setzen auf längere Meetings von 45 oder 60 Minuten. Dies macht insbesondere dann Sinn, wenn Probleme, die alle Teams und damit alle Mitarbeiter betreffen, in den Meetings besprochen und sofern möglich auch direkt gelöst werden sollen.

Wer sich in einem Scrum of Scrums-Meeting trifft, ist nicht standardisiert. In manchen Organisationen treffen sich stets die Product Owner, in anderen hingegen die Scrum Master. Häufig werden auch Fachkräfte entsendet, die an den konkreten Umsetzungen von Anforderungen beteiligt sind, also bspw. ein Softwarearchitekt, ein Entwickler oder ein Tester. Auch ein Wechsel der Team-Botschafter ist denkbar, so dass ein UX-Experte zu Projektbeginn öfter an den Meetings teilnehmen könnte, da in dieser frühen Projektphase viele Design-Entscheidungen zu treffen sind. Im Verlauf des Projekts könnte dann ein Tester häufiger zu den Meetings entsandt werden. Grundsätzlich sollten nicht mehr als neun Teilnehmer an den Treffen teilnehmen - dies führt bspw. bei 10 oder mehr Teams zu mehreren Scrum of Scrums.



Arbeiten verschiedene Teams - evtl. sogar an unterschiedlichen Standorten - an der Entwicklung einer gemeinsamen Lösung, sollten Abhängigkeiten zwischen den Teams idealerweise reduziert werden. Dazu empfiehlt sich

- jedes Team cross-functional zu besetzen, so dass die gestellten Aufgaben im Team möglichst autark gelöst werden können.
- User Storys nach dem INVEST-Prinzip zu beschreiben, so dass jede Story unabhängig von anderen Storys realisiert werden kann.
- User Storys so zu schneiden, dass alle Schichten der Architektur beachtet werden, so dass realisierte Funktionen auch potenziell lieferbar sein können. Hier ist das User Story Mapping eine gute Methode.
- Die Pflege des Product Backlogs konsequent zu betreiben, so dass die Verteilung von Backlog-Items auf die einzelnen Teams zügig und strukturiert erfolgen kann.
- die Sprintlängen und Sprinttermine zu synchronisieren, so dass Teams nicht unnötig aufeinander warten müssen oder die Integration von Lösungsteilen nicht verzögert wird.

Bei der Durchführung der Meetings empfiehlt es sich, einige Aspekte zu beachten:

- Die Vertreter der Teams sollten keine Namen von Teammitgliedern nennen, denn sie repräsentieren ihr gesamtes Team und nicht einzelne Kollegen. Zudem bleiben Beschreibungen und spätere Problemdiskussionen dadurch weitestgehend auf einer fachlichen Ebene.
- In der Kürze liegt die Würze, d.h. jeder Teilnehmer sollte sich an seine Zeitvorgabe halten, zumal so die Wichtigkeit von Themen nicht durch die Redelänge beeinflusst wird.
- Im Gegensatz zum Daily Scrum können auch Probleme in den Meetings thematisiert werden. Lösungen sollten aber erst besprochen werden, nachdem jeder Team-Botschafter seine Fragen beantwortet hat.

Scrum of Scrums bietet eine Reihe von Vorteilen:

- Die regelmäßige Kommunikation und Zusammenarbeit zwischen den verschiedenen Teams wird gefördert.
- Jedes Team kennt den Status quo der eigenen Entwicklung, weiß welche Aufgaben anstehen und welche Hindernisse aus dem Weg geräumt werden müssen.
- Der Projektfortschritt wird für alle Beteiligten sichtbar.
- Abhängigkeiten zwischen Teams werden reduziert.
- Risiken werden reduziert, indem Teams darauf achten, mit ihren Entwicklungen andere Teams nicht zu behindern.
- Probleme werden klar angesprochen und Lösungen gemeinsam gesucht. Auch die konkrete Unterstützung über Teamgrenzen hinweg zur Lösung von konkreten Herausforderungen ist denkbar.
- Am Ende des Sprints liefert jedes Scrum-Team einen Teil des Produktinkrements. Das zu entwickelnde System wächst somit inkrementell durch die Ergebnisse der einzelnen Teams. Die Koordination erfolgt über das Scrum of Scrums.

Organisationen, die mit mehreren Teams arbeiten, stehen vor einigen Herausforderungen:

- Arbeiten die Teams an verschiedenen Standorten lässt sich meist das Scrum-of-Scrums-Meeting nicht an einem gemeinsamen Ort durchführen. Hier können digitale Hilfsmittel wie Videokonferenzen helfen.
- Ist es notwendig, Ergebnisse und Erkenntnisse zu dokumentieren, müssen Organisationen Wege finden, diese Dokumente einfach und schnell zu erzeugen, sie gegebenenfalls zu versionieren und dann in der Folge den Team-Botschaftern bzw. den einzelnen Teams zur Verfügung zu stellen. Damit eine solche Dokumentation möglichst großen Nutzen entfaltet, muss sie im nächsten Scrum of Scrums wieder als Arbeitsmittel bereitstehen.
- Die passende Frequenz muss definiert werden - einerseits wollen Unternehmen regelmäßigen Austausch, andererseits verursacht die Durchführung von Meetings Aufwände. Es ist also wichtig, das Scrum of Scrums so kurz und sinnvoll wie möglich zu gestalten, und gleichzeitig für die Teams relevante Aspekte wie bspw. Probleme und mögliche Problemlösungen zu thematisieren.
- Auch die Durchführung von Scrum of Scrums lässt sich optimieren und weiterentwickeln. Unternehmen sollten daher Feedback der beteiligten Vertreter einholen - aber natürlich nicht während eines Scrum-of-Scrums-Meetings und vermutlich auch nicht mit einer Timebox.
- Wenn es ein Scrum of Scrum gibt, kann es auch eine *Retrospective of Retrospectives* geben, die es entsprechend zu terminieren und mit Vertretern der einzelnen Teams zu gestalten gilt.

ScrumBut

Scrum definiert ein Set aus Rollen, Regeln und Ritualen zur Entwicklung von Software, Produkten oder Services. Dabei sieht Scrum vor, dass im Zuge einer Retrospektive der Prozess der Entwicklung an die Organisation oder die Situation, in der sich die Organisation befindet, angepasst werden kann. ScrumBut bezeichnet den Verzicht auf Teile der ursprünglich festgelegten Rollen, Regeln und Rituale. Der Ausdruck geht auf „We use Scrum, but …“ zurück, auf Deutsch also „Wir nutzen Scrum, aber …“.

Die Intention eines ScrumButs liegt in der Optimierung der Arbeit nach Scrum in einer und für eine konkrete Organisation. Der Verzicht auf Teile von Scrum kann sich auf Rollen, auf die Durchführung der typischen Aktivitäten wie Sprints, Sprint Planning, Daily Scrum, Sprint Review, Retrospektive oder Backlog Refinement, oder auf die Verwendung der Artefakte wie Backlogs und Inkremente beziehen. Die Bandbreite für mögliche ScrumButs ist sehr groß, sie reicht damit von der Interpretation der Rollen („Für uns ist der Product Owner nicht so wichtig.“) über die Dauer und Frequenz einzelner Events („Wir führen einmal zu Beginn des Projekts ein Sprint Planning durch.“) bis hin zu den Artefakten („Anstelle von Product oder Sprint Backlogs verwenden wir unsere internen Projektdokumente.“)

In der Praxis von Scrum kommt es auch zu ScrumButs, die nicht der positiven Intention entsprechen. Die Durchführung von Workshops zur Ermittlung von Anforderungen wird in Scrum nicht definiert, auch wenn solche Workshops sinnvoll sein können. Etwas nicht zu tun, obwohl es positiv für die Entwicklung einer Lösung wäre, ist nicht im Sinne von Scrum, so dass auch hier von einem ScrumBut gesprochen werden kann.

ScrumBut erkennt man meist an einer typischen Syntax: *ScrumBut, Grund, Workaround*. Beispiel:

In manchen Publikationen werden alternative Beispiele verwendet: „Wir sprinten 6 bis 12 Wochen, denn so vermeiden wir Probleme“. Auch dies ist ein ScrumBut, also eine Abweichung der empfohlenen Sprintlänge von einer bis vier Wochen. Der Satz könnte auch „Wir nutzen Scrum, aber oft tun wir uns mit Aufwandsschätzungen schwer, also verwenden wir Sprints mit einer Dauer von 12 Wochen“ lauten.

Gründe für ScrumButs

Es gibt verschiedene Gründe für die Verwendung von ScrumButs. Manche Teams nutzen ScrumButs, um kurzfristig Probleme zu lösen. Oft stehen sie unter Zeitdruck und eine Modifizierung hilft scheinbar, ein Problem zu beseitigen. Um Zeit zu sparen, könnten bspw. einzelne Daily Scrums entfallen, da die Sorge besteht, ein Sprintziel zu verpassen. Ein akutes Problem ließe sich so lösen, dauerhaft sollte dies aber nicht mit einer solchen Begründung zur Reduzierung von Daily Scrums führen. Ursächliche Probleme - bspw. die falsche Schätzung von Aufwänden - tauchen oft zu späteren Zeitpunkten wieder auf.

Ein häufig genannter Grund zur Anpassung von Scrum ist das Umfeld einer Organisation: in manchen Branchen gelten besondere Nachweispflichten, in anderen ist das Arbeiten mit Lasten- und Pflichtenheften etabliert, in weiteren werden umfangreiche Vertragswerke ausgehandelt. Dennoch lässt sich Scrum - mit einigen Anpassungen als ScrumBut oder ScrumAnd - sinnvoll anwenden. Es gibt auch Ansätze, Transitionen von Unternehmen hin zu Scrum als ScrumBut zu verstehen, denn die Rollen, Aktivitäten und Artefakte sind zu Beginn der Einführung noch nicht etabliert, so dass sie noch nicht wie im Scrum Guide beschrieben zum Einsatz kommen (können).

Das agile Manifest

Scrum folgt den Werten, die im agilen Manifest festgehalten sind. Dieses Manifest sagt unter anderem, dass funktionierende Software wichtiger sei als eine umfassende Dokumentation. Das bedeutet nicht, dass Dokumentation unwichtig ist, sie ist im Verhältnis zu einer funktionierenden Software nur weniger wichtig. Nun gibt es Organisationen, die für jeden Sprint eine Spezifikation erstellen, die sich aus Items des Sprint Backlogs und Tasks ergeben, die das Team definiert. Scrum sieht ein solches Spezifikationsdokument nicht vor, dennoch können solche Anpassungen für Organisationen sinnvoll sein. Ken Schwaber, einer der Verfasser des agilen Manifests, hat daher schon frühzeitig erkannt, dass die Abweichung von den definierten Prinzipien legitim ist. Etwas weiter gedacht kann es sogar für Organisationen notwendig sein, von den definierten Ansätzen abzuweichen. Muss eine Retrospektive nach jedem Sprint stattfinden oder reicht es auch zum Ende eines Release? Darf eine Timebox auch 20 anstatt nur 15 Minuten dauern? Sollte es Bestrafungen für verspätetes Erscheinen zum Daily Scrum geben? Entscheidend für Organisationen ist die bewusste Beantwortung solcher Fragen anhand konkreter Herausforderungen; das agile Manifest liefert hier lediglich Empfehlungen.

Sind ScrumButs gut oder schlecht?

In manchen Diskussionen entsteht der Eindruck, die Anpassung von Scrum sei eine schlechte Idee - dem ist nicht so. Die Anpassung einer allgemeinen Methodik auf individuelle Situationen ist sogar zwingend notwendig. Auch wenn die 17 Autoren des agilen Manifests viel Erfahrung bei der Entwicklung von Produkten und der Zusammenarbeit in Teams besitzen, sie kennen nicht alle individuellen Situationen und Herausforderungen.

Bei einer Anpassung von Scrum ist die bewusste Auseinandersetzung mit einer konkreten Herausforderung wichtig. Das setzt voraus, dass es sich bei der Anpassung nicht nur um eine kurzfristige Beseitigung eines Problems als Quick Fix handelt. Mit ScrumBut sollten Probleme gelöst und nicht verschleiert werden. Muss ein Product Owner bspw. mehrere Produkte betreuen und kann daher nicht an Meetings teilnehmen, wäre es keine gute Lösung, ihn nicht mehr zu den Meetings einzuladen; es wäre sinnvoller, seine Arbeitslast auf andere Mitarbeiter zu verteilen, so dass er seiner Verantwortung auch tatsächlich nachkommen kann.

ScrumBut und ScrumAnd

Der 2017 State of Scrum Report der Scrum Alliance stellt fest, dass 86% der Teams ein Daily Scrum durchführen, 11% führen es mehrmals pro Woche aber nicht täglich, 2% nach Bedarf und 1% überhaupt nicht durch. Dieses Beispiel zeigt, dass 14% mit ScrumBut arbeiten. Bei ScrumBut werden je nach individueller Situation eine oder mehrere Praktiken weggelassen oder adaptiert. ScrumAnd hingegen ergänzt Scrum um die eine oder andere Praktik, also "We use Scrum and ...".

ScrumAnds lassen sich relativ häufig in Organisationen finden. Der Scrum Guide sagt bspw. nichts von User Storys, Epics oder Taskboards, obwohl diese Elemente in der agilen Welt weit verbreitet sind. Die Verwendung dieser Aspekte verursacht zwar etwas Aufwand, aber sie gelten allgemein als sehr nützlich und werden durchweg positiv beurteilt. ScrumButs hingegen können gefährlich sein, weil hier etablierte Praktiken bewusst oder unbewusst weggelassen werden. Es empfiehlt sich daher, ScrumButs vorsichtig zu entwickeln, konkrete Erfahrungen - wie bei der Verwendung von Scrum - zu sammeln und gegebenenfalls weitere Anpassungen vorzunehmen.

ScrumBut Beispiele

ScrumButs bei Rollen

Es gibt ScrumButs, die beziehen sich auf die Rollen oder die Interpretation der Rollen. Hier finden Sie einige Beispiele. We use Scrum, but ...

- wir sind sehr effektiv, also arbeiten wir ohne Scrum Master.
- unsere Stakeholder sind zu beschäftigt um an den Sprint Reviews teilzunehmen, also laden wir sie gar nicht mehr ein.
- wir entwickeln verschiedene Produkte parallel, also haben wir eine separate Abteilung mit Testern.
- wir sind eine kleine Firma, also besteht unsere Entwicklung aus einem Mitarbeiter.

- immer auf die Entwickler aufzupassen ist auf Dauer langweilig, also übernimmt jeder einmal die Rolle des Scrum Masters.
- bei uns klappt die Selbstorganisation nicht richtig, also weist der Product Owner den Entwicklern User Storys zu.
- der Product Owner hat nicht viel Zeit, also formuliert er die Anforderungen so detailliert, dass er nicht am Sprint Planning teilnehmen muss.
- wir haben so viele Entwicklungsaufgaben, also entwickelt auch unser Scrum Master als Teil des Teams Software.
- unser Scrum Master hat schon in vielen Firmen gearbeitet, so dass er mit seiner Erfahrungen klare Anweisungen geben kann.
- unsere Projektmanager sind sehr gut, also unterstützen sie unsere Scrum Master beim Managen der jeweiligen Teams.

ScrumButs bei Aktivitäten

Es gibt ScrumButs, die beziehen sich auf die Aktivitäten, auf die Dauer, die Frequenz und die Art der Durchführung. We use Scrum, but ...

- Retrospektiven sind nicht effektiv, also machen wir nur eine zum Ende des Releases.
- wir arbeiten mit einem Wasserfall im Sprint, also testen wir alle realisierten User Storys sicherheitshalber erst am Ende des Sprints.
- bei uns dauert das Sprint Planning zu lange, also macht der Product Owner die Ansagen, wer was wie entwickeln soll.
- bei uns kommunizieren die Entwickler permanent miteinander, also machen wir das Daily Scrum nur einmal pro Woche.
- meist haben wir unser Sprintziel in der Kürze der Sprints nicht erreicht, also verzichten wir nun auf eine Definition der Sprintlängen und arbeiten so lange, bis wir alle User Storys realisiert haben.
- wir tun uns mit der Aufwandsschätzung und technischen Zusammenhängen schwer, also dauern unsere Sprints sicherheitshalber 12 Wochen.
- bei uns machen zwei Sessions im Sprint Planning wenig Sinn, also machen wir eine gemeinsame Planung in der Hälfte der Zeit.
- wir glauben bei den vielen Meetings nicht an Timeboxing, also lassen wir es mit Ausnahme der Sprints einfach weg.
- wenn wir immer nur für das nächste Release oder den nächsten Sprint planen verlieren wir den Überblick, also planen wir immer für drei Releases im Voraus.

ScrumButs bei Artefakten

Es gibt ScrumButs, die beziehen sich auf die Artefakte. Hier finden Sie einige Beispiele. We use Scrum, but ...

- unser Auftraggeber möchte gerne ein Lastenheft von uns sehen, also erstellen wir zu jedem Sprint für ihn eins.

- wir wollen eine klare Verantwortung zwischen Entwicklung und Testing, also teilen wir die User Storys entsprechend in Entwicklungs- oder Test-Storys auf.
- die Aufwandsschätzung ist uns zu ungenau, also schätzen wir zusätzlich pro User Story die Aufwände für die Entwicklung von Backend, Frontend, Integration und Testing.
- wir haben gute Erfahrung mit dem horizontalen Verfeinern von User Storys gemacht, also verzichten wir auf das vertikale Verfeinern auch wenn wir am Ende des Sprints kein Inkrement liefern können.
- da wir lediglich unsere Releases unseren Kunden zur Verfügung stellen, produzieren wir das Lieferergebnis nur auf Zuruf oder einmalig zum Releasetermin.
- da alle Mitarbeiter in verschiedenen Projekten aktiv sind und der Zeitdruck hoch ist, verzichten wir manchmal auf die Einhaltung der Definition of Done.
- wir brauchen kein separates Sprint Backlog, also nutzen wir einfach die Informationen, die im Product Backlog stehen.
- da unser Product Backlog so groß ist, verwalten wir zusätzlich Backlogs für Epics, Features und Kundenwünsche.
- unsere höchste Priorität ist die Einhaltung unseres initialen Releaseplans, also machen wir häufig Überstunden um diesen einzuhalten.

ScrumButs durch Unwissenheit

ScrumButs können durchaus eine Berechtigung haben - sofern sie bewusst als Abweichung von Scrum verstanden, eingeführt, überprüft und gegebenenfalls adaptiert werden. ScrumButs, die auf Unwissenheit zurückgehen, sorgen aber in den seltensten Fällen für positive Effekte. Sie beruhen häufig auf unbewiesenen Annahmen, auf Vorurteilen oder Meinungen einzelner Mitarbeiter. Hier finden Sie mögliche Beispiele für ScrumButs durch Unwissenheit:

- Das Management hat gehört, dass „agil“ so einfach und gut ist, also machen wir nun größtenteils Scrum.
- Scrum besteht nur aus wenigen Rollen, Aktivitäten und Artefakten, da ist doch kein Training notwendig.
- Wir haben 15 Entwickler, die wollen wir nicht in zwei ungleich große Teams aufteilen, also arbeiten wir einfach wie gehabt mit einem gemeinsamen Team.
- Wir machen 100% Scrum, also alle Aspekte, die nicht im Scrum Guide beschrieben werden, lassen wir einfach weg.
- Warum sollten wir Mitarbeiter ausbilden oder gar zertifizieren?
- Scrum steht für Offenheit und Feedback, also dokumentieren wir dies auch öffentlich.
- Wir nehmen jetzt einfach mal die besten Elemente aus Scrum heraus und wenn es nicht klappt, dann lassen wir es wieder sein.
- Vertrauen ist ja schön und gut, aber mit klaren Ansagen erreichen wir unsere Ziele schneller.
- Warum sollten wir miteinander über unsere Erfahrungen sprechen - jeder Mitarbeiter macht doch dieselben.

Der bewusste Umgang mit ScrumButs

Die Entwicklung von Software, Produkten und Services mit Scrum ist nicht immer so leicht, wie es vielleicht auf den ersten Blick klingen mag. Für Unternehmen kann es daher sinnvoll sein, das Vorgehen auf die konkreten Herausforderungen und Möglichkeiten anzupassen. Niemand erhält ein Sternchen im Notenheft, wenn eine Entwicklung 100% nach Scrum durchgeführt wird. Wenn sich Organisationen mit der Anpassung von Scrum beschäftigen, sollten sie ein tieferes Verständnis von Scrum erlangt haben. Ein solches Verständnis entwickelt sich über die Zeit und durch die Einhaltung und Anwendung der definierten Rollen, Regeln und Rituale. Erst in der Folge kann eine Anpassung - ein ScrumBut - seine gewünschte Wirkung erzielen. Eine Anpassung, die lediglich ein Problem verschleiert und nicht löst, ist nicht sinnvoll. Steht ein Product Owner bspw. nicht regelmäßig zur Verfügung und kann daher nicht an den notwendigen Meetings teilnehmen, sollte nicht die Verbesserung der Dokumentation von Anforderungen im Vordergrund stehen, sondern die Entlastung des Product Owners. Die fehlende Kommunikation ist der Schwachpunkt, da hilft ein ScrumBut vermutlich wenig. Es empfhielt sich daher - ähnlich wie bei Scrum an sich - ScrumButs immer wieder zu hinterfragen und anzupassen, denn so kann die Anpassung die bestmögliche Wirkung entfalten.

Herausforderungen für Unternehmen

Die Anzahl der Organisationen, die Scrum nutzen, wächst kontinuierlich. Unternehmen erhoffen sich mehr Flexibilität, schnellere Entwicklungszyklen oder niedrigere Kosten. Durch die Sprints, die Kommunikation zwischen Product Owner und Stakeholder und das gemeinsame Sprint Planning lässt sich die Flexibilität häufig steigern. Doch diese Flexibilität ist nicht kostenlos. Scrum erfordert regelmäßige Kommunikation zwischen den Beteiligten in Form von Daily Scrums, Sprint Plannings, Sprint Reviews, Retrospektiven und gegebenenfalls Scrum of Scrums. Hinzu kommt der Aufwand, Backlogs zu pflegen, Arbeitspakete zu definieren, Aufwände zu schätzen und den Fortschritt zu dokumentieren. In anderen Worten: Scrum ist keine Abkürzung zu Unternehmenserfolg. Technische Herausforderungen bleiben bestehen, sie lassen sich aber vermutlich im Laufe einer Entwicklung mit neuem Wissen besser beurteilen als zu Projektbeginn. Damit sinkt das Risiko falscher Projektpläne und Versprechungen. Damit dies funktioniert, müssen Organisationen aber nicht nur lernen, die Regeln von Scrum anzuwenden, und die Rollen mit geeigneten Mitarbeitern zu besetzen. Es ist notwendig, ein agiles Mindset zu entwickeln. Das Management muss sich von einem Command and Control – Führungsstil lösen. Das Team muss lernen mit Verantwortung umzugehen und sie bewusst in Anspruch zu nehmen. Der Product Owner muss verstehen, dass er nicht der Vorgesetzte des Entwicklungsteams ist. Und der Scrum Master darf sich nicht als Projektleiter verstehen. Erst mit einem solchen agilen Mindset entfaltet Scrum die größten Vorteile.

Gerne stellen wir Ihnen Wissen, Erfahrung und 100% Leidenschaft für Ihre Softwareentwicklung und Anforderungsanalyse zur Verfügung. Wir helfen Ihnen bei der Erhebung, Strukturierung und Verwaltung von Anforderungen und achten dabei auf Konsistenz, Vollständigkeit und Nachvollziehbarkeit. Wir unterstützen Sie beim Identifizieren von technischen Zusammenhängen und berücksichtigen Stakeholder, Ziele und Randbedingungen.

Sprechen Sie mit uns über Ihre Situation, Herausforderungen und Rahmenbedingungen. Vereinbaren Sie einfach einen Rückruf – wir melden uns so schnell wie möglich bei Ihnen. Sie erreichen uns unter https://t2informatik.de/ oder +49 (30) 419 58 981.