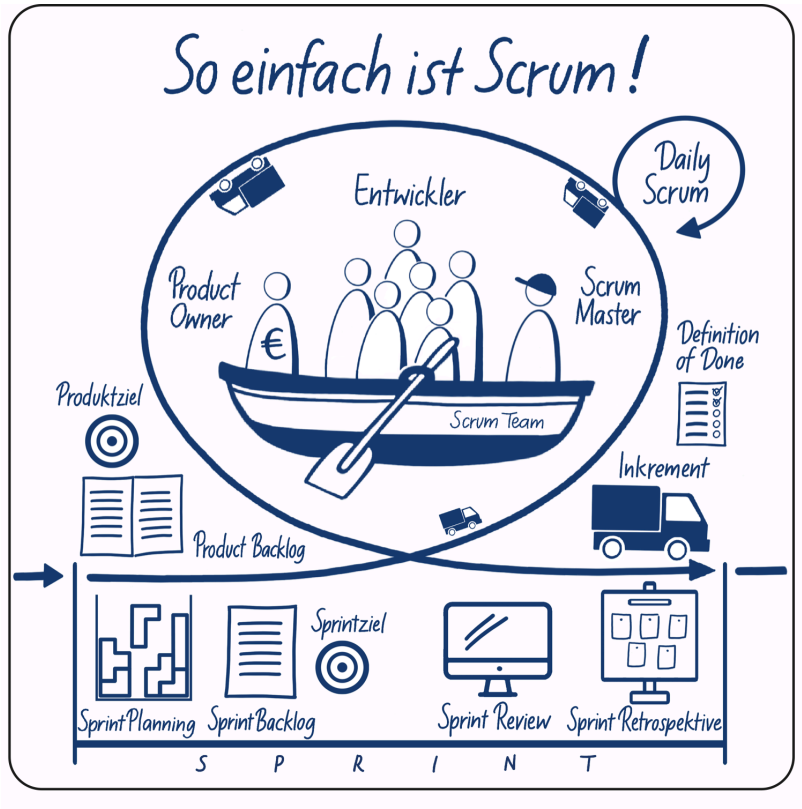
Scrum – auf dem Bierdeckel erklärt

Begriffe, Konzepte, Grundverständnis



März 2021

Vorwort

Dieses Paper führt in Scrum ein. Es erläutert die grundlegenden Begriffe und Konzepte und erklärt, wie diese zusammenhängen. Die Scrum-Mechanik ist dabei nur die eine Seite der Medaille. Die dahinterstehenden Werte und Prinzipien sind mindestens genauso wichtig. Daher findet sich nach der Scrum-Einführung eine Beschreibung des agilen Manifestes, das die agilen Prinzipien definiert. Danach findet sich eine Übersicht über die Scrum-Rollen, -Artefakte und –Events, die zum Nachschlagen geeignet ist.

Dieses Paper hilft dem Scrum-Neuling, sich einen ersten Überblick über die Funktionsweise von Scrum zu verschaffen. Dieses Grundverständnis dient als Orientierungshilfe für die weitere Vertiefung in Scrum, die durch Scrum-Einführungsbücher[1] oder Schulungen[2] erfolgen kann. Auf keinen Fall sind Sie nach der Lektüre dieses kurzen Artikels in der Lage, Scrum einzuführen.

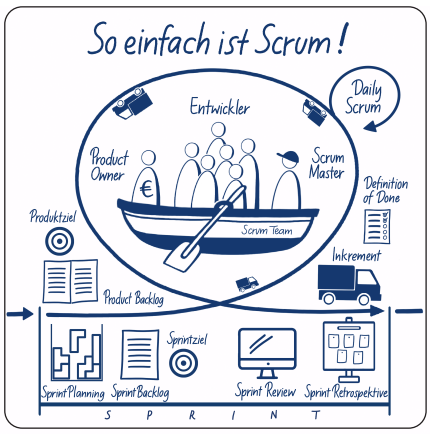
Inhalt

[1] z.B. Stefan Roock, Henning Wolf: „Scrum - verstehen und erfolgreich einsetzen“
[2] z.B. http://www.it-agile.de/schulungen

Scrum – Drei Perspektiven

Man kann Scrum in einem Satz beschreiben:

Scrum bedeutet:
Autonome Teams mit Business-Fokus,
die ihren Prozess in Verantwortung nehmen und
kontinuierlich verbessern.



In diesem Satz werden drei Perspektiven sichtbar, aus denen man Scrum betrachten kann:

1. Die *Produktperspektive* (Business-Fokus) beleuchtet, wie Produkte definiert und verbessert werden.
2. Die *Entwicklungsperspektive* beleuchtet, wie Entwickler/innen im teilautonomen Team Produkte entwickeln.
3. Die *Verbesserungsperspektive* (Prozess in Verantwortung nehmen) beleuchtet, wie Zusammenarbeit und Prozesse verbessert werden.

Diese drei Perspektiven werden in das Scrum-Framework integriert, das so einfach ist, dass es auf einen Bierdeckel passt (siehe Abbildung rechts). Jede Perspektive geht mit einer eigenen Ergebnisverantwortung innerhalb des Scrum Teams einher: Product Owner, Entwickler/innen, Scrum Master.

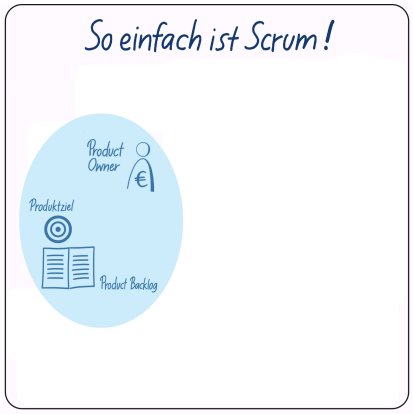
Wir beschreiben die drei dargestellten Perspektiven in den folgenden Abschnitten ausführlicher.

Produktperspektive

Die Produktperspektive beginnt mit der *Product-Owner-Verantwortung*, die immer von genau einer Person wahrgenommen wird. Diese Person sorgt für den Produkterfolg, indem sie den Produktnutzen durch die Priorisierung der Produkteigenschaften optimiert. Folglich darf sie Entscheidungen über das Produkt treffen. Man kann sich den Product Owner auch als Unternehmer/in im Unternehmen vorstellen.

In Scrum soll die Person mit der Product Owner-Verantwortung mit *einer* Stimme gegenüber dem Team und den Stakeholdern sprechen und Entscheidungen schnell fällen können. Daher kann die Verantwortung in Scrum nicht von mehreren Personen geteilt wahrgenommen werden und schon gar nicht durch ein Komitee. Es gilt für die Product Owner -Verantwortung also das Highlander-Prinzip: „Es kann nur einen geben."

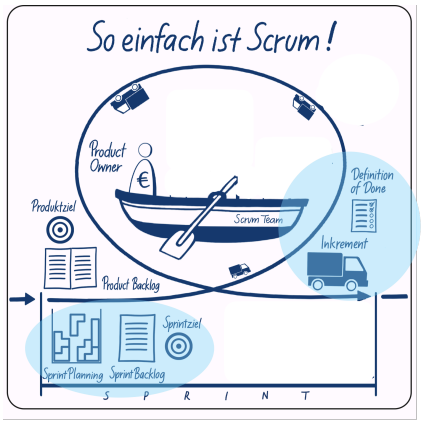
Der Product Owner verfolgt ein *Produktziel*. Scrum schreibt nicht vor, wie das Produktziel auszusehen hat. Je nach Kontext könnte es eine Produktvision geben, die allerdings nicht von Scrum gefordert wird. Diese Produktvision könnte als Produktziel dienen. Häufiger wird das Produktziel aber der Produktvision untergeordnet sein.



Passend zum Produktziel pflegt der Product Owner das *Product Backlog*, in dem die Produkteigenschaften beschrieben sind, die für den Produkterfolg notwendig erscheinen. Das Product Backlog wird durch den Product Owner passend zum Produktziel *priorisiert*.

Scrum legt nicht fest, wie genau die Einträge des Product Backlogs gestaltet sind. Viele Scrum Teams machen gute Erfahrungen mit User Stories: Exemplarische Benutzungsszenarien aus Sicht der Benutzer. User Stories haben einen anderen Fokus als klassische Anforderungen. Bei User Stories geht es darum, ein gemeinsames Verständnis bei allen Beteiligten zu erzeugen und nicht darum, dass die Beschreibung vollständig, widerspruchsfrei und korrekt ist.

Die Entwicklung erfolgt in Iterationen, die in Scrum *Sprints* heißen. Sprints haben eine immer gleiche Länge von max. 4 Wochen.

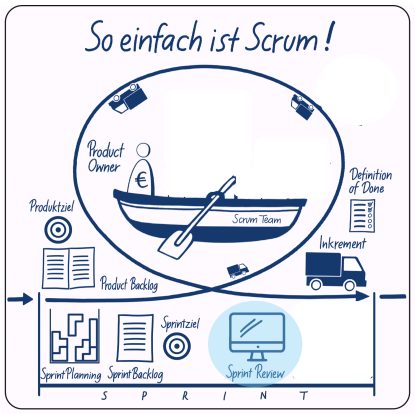


Was im Sprint entwickelt wird, wird im *Sprint Planning* festgelegt: Es wird ein *Sprintziel* definiert und dazu passend hochpriorisierte Einträge aus dem Product Backlog ausgewählt. Dabei legen die Entwickler/innen fest, wieviele Einträge sie in den Sprint ziehen und wie sie planen, die Einträge umzusetzen. Sprintziel, ausgewählte Einträge des Product Backlogs und Umsetzungsplan zusammen bilden das *Sprint Backlog*.

Das Ergebnis des Sprints ist ein auslieferbares *Produktinkrement*, das der *Definition of Done* genügt. Die Definition of Done ist eine Vereinbarung im Scrum Team darüber, wann etwas „fertig" ist. Ob das Produktinkrement dann tatsächlich an Kunden ausgeliefert wird, entscheidet der Product Owner. Die im

Produktinkrement implementierten Product Backlog-Einträge müssen auf jeden Fall produktionsreif sein (mind. entwickelt und qualitätsgesichert).

Die Entwickler/innen demonstrieren am Ende jedes Sprints das Produktinkrement im *Sprint Review* den Stakeholdern[3], damit sie Feedback zum Produkt geben können. Das Feedback wird vom Product Owner entgegengenommen und nach seinem Ermessen in das Product Backlog integriert.



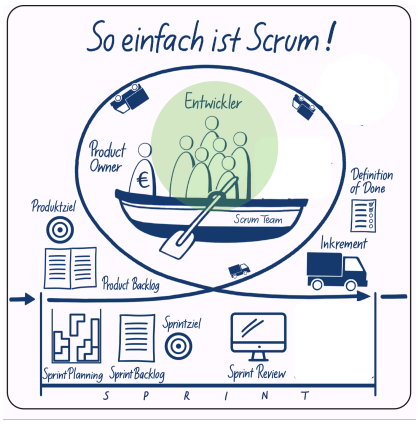
Gute Fragen, um nützliches Feedback im Sprint Review zu erhalten, sind:

- „Glauben wir, dass mit dem gezeigten Produktinkrement die angestrebte Wirkung für Kunden und das eigene Unternehmen erzielt werden kann? Was fehlt dafür noch?“
- „Was hindert uns daran, das vorliegende Produktinkrement produktiv zu benutzen?“
- „Wie kann das vorliegende Produktinkrement noch wertvoller für Kunden und das eigene Unternehmen gestaltet werden?“

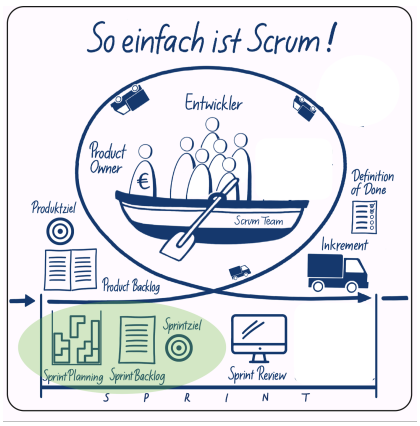
[3] Stakeholder ist in Scrum jeder, der Interesse am Produkt oder Einfluss auf die Entwicklung hat: Kunden, Anwender, Sponsoren, Manager, Betriebsrat etc.

Entwicklungsperspektive

Die Entwicklungsverantwortung liegt bei den *Entwicker/innen* [4]. Diese entwickeln ausgehend vom Sprint Backlog ein auslieferbares Produktinkrement. Ein Scrum Team besteht neben Product Owner und Scrum Master aus 3-9 Entwickler/innen, die alle Fähigkeiten vereinen, die notwendig sind, um das Sprint Backlog in das Produktinkrement zu überführen. Entwickler/innen können je nach Kontext sehr unterschiedliche Spezialisierungen haben: Programmierung, UX-Design, Oberflächengestaltung, Handbuch-Autorenschaft, Testen etc. Entwickler/innen sind also nicht alle gleich.

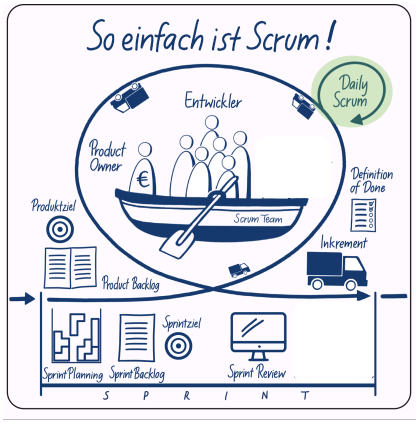


Die Entwickler/innen organisieren sich selbst. Unter ihnen gibt es weder eine formelle Hierarchie noch herausgehobene Rollen oder Positionen.



Im *Sprint Planning* wenden die Entwickler/innen das Pull-Prinzip an: Die Entwickler/innen bestimmen, *wieviel* Arbeit sie in den Sprint aufnehmen, nicht der Product Owner. Für die ausgewählten Einträge aus dem Product Backlog erstellen die Entwickler/innen einen Plan für die Umsetzung im Sprint. Wie genau dieser Plan aussieht, lässt Scrum offen. Viele Teams erstellen technische Tasks, um den Umsetzungsplan zu definieren. Die ausgewählten Einträge aus dem Product Backlog bilden zusammen mit dem Sprintziel und dem Umsetzungsplan das *Sprint Backlog*.

Die Entwickler/innen machen so eine Vorhersage (Forecast) darüber, was sie im Sprint schaffen können. Diese Vorhersage soll die Qualität einer Wettervorhersage haben. In der Regel sollten die Entwickler/innen das liefern, was sie geplant haben. Es sollte aber niemand übermäßig überrascht sein, wenn das hin und wieder nicht klappt.



Während des Sprints treffen sich die Entwickler/innen täglich zum Daily Scrum, um sich über den Arbeitsfortschritt und die nächsten Aufgaben im Sprint abzustimmen. Dazu kommen die Entwickler/innen an jedem Arbeitstag zur gleichen Uhrzeit am gleichen Ort

[4] Im deutschen Scrum Guide ist von „Developer“ die Rede.

für maximal 15 Minuten zusammen und beantworten drei Fragen:

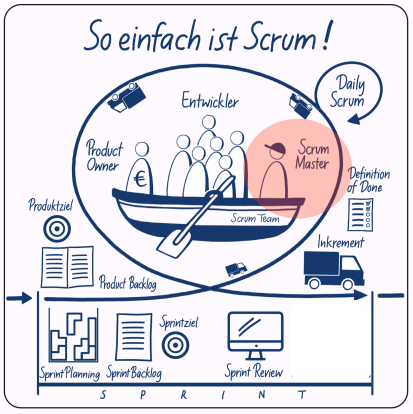
1. Was wurde seit dem letzten Daily Scrum erledigt, was uns dem Sprintziel nähergebracht hat?
2. Welche Hindernisse sehen wir auf dem Weg zum Sprintziel?
3. Was planen wir bis zum nächsten Daily Scrum zu erledigen, was uns dem Sprintziel näherbringt?

Der Product Owner ist ein optionaler Teilnehmer am Daily Scrum.

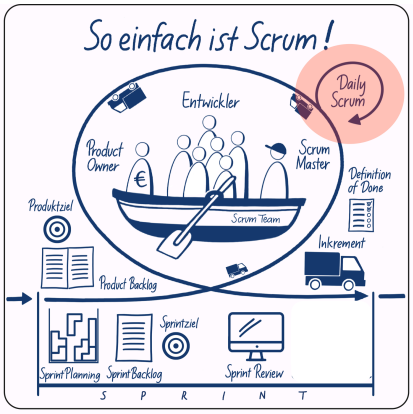
Verbesserungsperspektive

Die *Scrum Master*-Verantwortung sorgt dafür, dass ein hocheffektives Scrum-Team entsteht. Scrum sieht den Scrum Master in der (Mit-)Verantwortung für die Lieferung wertvoller Produktinkremente. Scrum soll so angewendet werden, dass Wertschöpfung maximiert wird.

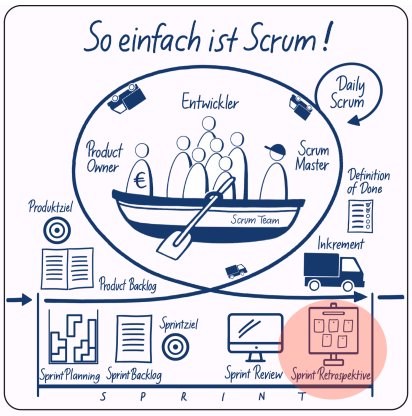
Dazu stellt der Scrum Master sicher, dass Product Owner, die Entwickler/innen und Stakeholder verstehen, wie Scrum funktioniert und Scrum effektiv anwenden. Für die Entwickler/innen schafft der Scrum Master einen Rahmen, in dem sie sich selbst organisieren können und hält ihnen immer wieder den Spiegel vor. Der Scrum Master kümmert sich außerdem darum, dass Impediments (Hindernisse) identifiziert und beseitigt werden. Er moderiert in der Regel die Scrum-Events.



Die kontinuierliche Verbesserung des Entwicklungsprozesses ist bei Scrum in zwei Events verankert. Zum einen nehmen Verbesserungen ihren Ausgang im *Daily Scrum*, wenn Hindernisse identifiziert werden. Hindernisse sind in Scrum alles, was die Arbeit an aktuellen Aufgaben blockiert oder verlangsamt. Der Scrum Master kümmert sich um die Beseitigung der Hindernisse. Das bedeutet nur selten, dass er die Hindernisse alleine aus der Welt schafft. Er wird dazu in der Regel mit weiteren Parteien im Unternehmen kooperieren müssen (z.B. um finanzielle Mittel für schnellere Rechner zu beschaffen).



Zum anderen findet am Ende des Sprints die *Sprint Retrospektive* statt. Hier reflektiert das ganze Scrum Team darüber, was im letzten Sprint gut und was weniger gut gelaufen ist. Auf dieser Basis definiert es Verbesserungsmaßnahmen, die es im nächsten Sprint umsetzt. Mindestens eine Maßnahme wird Teil des Sprint Backlogs des nächsten Sprints. Die Sprint Retrospektive wird in der Regel durch den Scrum Master moderiert.



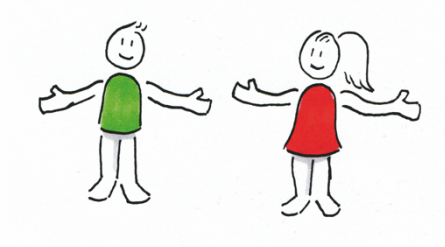
Die Scrum-Werte

Scrum definiert ein Rahmen, der von Teams ausgefüllt werden muss. Die Scrum-Werte geben Orientierung, welche der selbst gewählten Lösungen gut zu Scrum passen und welche eher nicht. Die Werte definieren damit das Scrum-Mindset.

Commitment

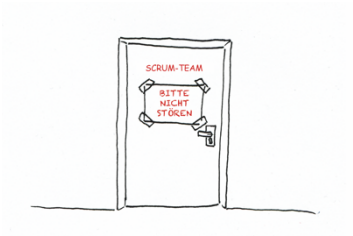
Commitment ins Deutsche zu übersetzen ist eher sperrig. Dem am nächsten käme wohl der Begriff (Selbst-)Verpflichtung. Commitment ist für das Scrum Team (Product Owner, Scrum Master und Entwickler/innen) wichtig, vor allem im Sinne einer Selbstverpflichtung jedes Einzelnen, weil man Scrum eben nicht verordnen kann. Das Team muss sowohl die Aufgabe als auch die Zusammenarbeit bewusst annehmen. Nur so können die Aufgaben erfolgreich erledigt werden. Commitment bedeutet, dass die Mitglieder mit ganzem Herzen dabei sind und dass es ihnen nicht egal ist, ob die Aufgabe gelingt. Alle im Scrum Team müssen wirklich wollen, dass die Aufgabe gelingt, und sich dafür einsetzen.

Offenheit



Offenheit bedeutet, vorurteilsfrei offen für die Realität zu sein. Alle im Team sollen sowohl gehört werden als auch alle Informationen haben, um beste Ergebnisse zu erzielen. Nur so lässt sich eine vernünftige Planung und eine realistische Vorhersage erreichen. Das ist besonders dann wichtig, wenn die Arbeit des Teams nicht direkt sichtbar ist (z. B. Softwareentwicklung). Natürlich geht es bei Offenheit auch darum, zu persönlichen Schwächen oder Fehlern zu stehen und diese nicht zu vertuschen oder zu verbergen, damit gemeinsam daraus gelernt werden kann und ein Umgang mit dem Problem für die Zukunft gefunden wird.

Fokus



Die Kraft, die im Wert Fokus liegt, leuchtet den Meisten intuitiv sofort ein. Das macht fokussiertes Arbeiten aber noch lange nicht einfach und selbstverständlich. Die Idee von Jeff Sutherland, dass Scrum Teams die doppelte Menge an Arbeit in der Hälfte der Zeit erledigen, ist ohne Fokus niemals möglich. Zudem gehört dazu, dass alle Mitglieder des Scrum Teams voll dabei sind und nicht noch andere Aufgaben wahrnehmen.

Mut

Es braucht Mut für Offenheit, es braucht Mut für Veränderungen und es braucht Mut, Probleme anzusprechen. Diesen Mut benötigen sowohl die Teams als auch die Führungskräfte, die sich mit Scrum auf ein anderes Arbeiten einlassen. Damit wird auch deutlich, dass Mut ein zentraler Wert ist, der das Leben der anderen Werte erst ermöglicht.

Respekt

Respekt ist eine wesentliche Voraussetzung für Offenheit. Nur mit Respekt kann man gemeinsam daran arbeiten, besser zu werden. Dafür ist es wichtig, nichts und niemanden zu verurteilen. Viele Unternehmen haben eine ausgeprägte Anschuldigungskultur, suchen also ständig den Schuldigen. Das ist aber keine erfolgreiche Strategie und löst nicht die Probleme! In diesem Sinne trägt Scrum mit dem Wert Respekt dazu bei, Organisationen (wieder) erfolgreich zu machen. Das Leben dieses Wertes schafft in der Organisation eine Umgebung, die mehr Möglichkeiten eröffnet.

Das agile Manifest

Scrum und seine Werte haben dazu beigetragen Anfang des 21. Jahrhunderts den Begriff „Agilität“ zu prägen. In der Zusammenarbeit mit anderen Vorgehensideen (z. B. Extreme Programming) ist das Agile Manifest entstanden. Dieses Leitbild für Agilität beginnt mit vier Wertaussagen.

Wertaussagen

Die Abbildung rechts zeigt die Wertaussagen des agilen Manifests. Obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein. Häufig hilft die Perspektive, dass wir die Dinge auf der rechten Seite nur soweit verfolgen, wie sie uns helfen, die Dinge auf der linken Seite besser zu erreichen.

Individuen und Interaktionen	wichtiger als	Prozesse und Tools
Laufende Software	wichtiger als	Ausführliche Dokumentation
Zusammenarbeit mit dem Kunden	wichtiger als	Vertrags-verhandlungen
Reagieren auf Veränderungen	wichtiger als	Planbefolgung

In klassischen Kontexten generieren die Dinge auf der rechten Seite *subjektiv wahrgenommene Sicherheit*. Wer sich an die Prozesse hält und die vorgeschriebenen Tools einsetzt, wer jede seiner Tätigkeiten haarklein dokumentiert, wer alle Eventualitäten in Verträgen berücksichtigt und wer sich an den Plan hält, kann bei Problemen nachweisen, dass er nicht schuld ist. Leider generieren wir auf diese Weise in komplexen Märkten keinen Geschäftswert. In dynamischen Märkten brauchen wir die Flexibilität, die uns die Dinge auf der linken Seite geben.

Dieser Gegensatz erklärt, warum die Einführung agiler Verfahren in der Praxis häufig so schwierig ist. Alle Beteiligten müssen ein Stück dieser „Sicherheit durch Statik“ loslassen, um auf den Kunden und den Geschäftswert fokussieren zu können.

Prinzipien

Ergänzt werden die vier Wertaussagen durch zwölf Prinzipien, die konkretisieren, wie die Werte sich auf die tägliche Arbeit auswirken:

1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software[5] zufrieden zu stellen.
2. Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
3. Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
4. Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
5. Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
6. Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
7. Funktionierende Software ist das wichtigste Fortschrittsmaß.
8. Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
9. Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
10. Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell.
11. Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
12. In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

[5] Außerhalb der IT kann der Begriff Software problemlos durch Produkt oder Service ersetzt werden.

Überblick über die Scrum-Verantwortungen, -Events und -Artefakte

Dieser Abschnitt gibt prägnante Übersichten und Checklisten für die Verantwortungen, Events und Artefakte von Scrum. Diese sollten auf keinen Fall dogmatisch verwendet werden. Es gibt nicht die eine richtige Form, die Events durchzuführen, die Verantwortungen auszuleben oder die Artefakte zu gestalten. Dieser Abschnitt kann aber als Kurzreferenz helfen sowie als Startpunkt, um überhaupt einmal mit irgendetwas anzufangen.

Verantwortlichkeit: Scrum Master

Die Events machen in Scrum in der Summe ca. 10 % der Zeit aus. Rechnen wir für die Vor- und Nachbereitung noch einmal dieselbe Zeit, verbleibt doch ein erheblicher Anteil Arbeitszeit, in denen der Scrum Master sich auf andere Weise nützlich macht. Welche Aufgaben Scrum Master in der Praxis übernehmen, hängt vom Unternehmen, vom Projekt und von der Reife des Teams ab. Im Folgenden findet sich eine Liste mit Beispielen aus der Praxis. Die **fett** gesetzten Punkte sind die Aufgaben, die der Scrum Master auf jeden Fall wahrnehmen muss.

Teamebene

1. **Gemeinsam mit dem Team Retrospektiven-Maßnahmen umsetzen**
2. Die Entwickler/innen unterstützen, ein besseres technisches Verständnis zu erwerben, dabei ggf. an Entwicklermeetings teilnehmen und agile Entwicklungspraktiken einführen (testgetriebene Entwicklung, kontinuierliche Integration, Pair Programming)
3. **Gerade für neue Scrum-Teams: dem Team beim Umgang mit Veränderungen helfen, die beim Umstieg auf Scrum anstehen**
4. Materialnachschub fürs Taskboard organisieren
5. **Impediments aufnehmen und bei der Behebung unterstützen**: Diese können konkret aus einzelnen Entwicklungsaufgaben resultieren, Teamprobleme sein, Kommunikationsprobleme im Team, mit dem Product Owner oder zu Stakeholdern, sich aber auch auf Organisationsebene befinden. (Was darf das Team, wer stört das Team?)
6. **Für Festlegung von Teamspielregeln sorgen und diese gut sichtbar machen**
7. **Teammitglieder an die vereinbarten Spielregeln erinnern**
8. Einzelgespräche mit Teamfokus: Was brauchst du im/vom Team? Wie geht es dir gerade? Wie zufrieden bist du? Feedback an dich, Feedback von dir? Wie sehe ich deine Rolle im Team und deinen Beitrag fürs Team? Wo stehst du dem Team im Weg? Wo könntest du dich mehr einbringen? (Empfehlung: Einzelgespräche alle zwei bis drei Wochen mit jedem Teammitglied inklusive Product Owner führen.)
9. **Aha-Momente oder Leidensdruck erkennen als Initialpunkt für sofortige Veränderung** (nicht immer nur Input für Retrospektiven, oft auch direkt umsetzbar)
10. Netzwerk durchforsten für Ideen, wie man Probleme/Herausforderungen des Teams lösen könnte (Optionen schaffen)
11. Beurteilung der Situation mit Scrum Master-Kollegen, Coaches oder Netzwerk diskutieren, um wach zu bleiben und nicht auf die eigene Perspektive beschränkt zu bleiben
12. Informativen Workspace gestalten bzw. das Team anregen, ihn zu gestalten sowie aktuell und hilfreich zu halten
13. Organisatorische Aufgaben wie das Buchen von Meetingräumen etc. (aber nicht so, dass die Teammitglieder das nicht mehr selbst können)
14. Social Events für das Team-Building organisieren (das kann auch das Bestärken von Kollegen sein, die dann die Organisation übernehmen)
15. **Auf Missstände hinweisen, selbst wenn diese erst mal für das Team kein großes Problem zu sein scheinen** (Beispiel: Sprints werden nicht geschafft, was das Team zwar nicht so dramatisch findet, der Scrum Master oder die Stakeholder aber schon.)
16. **Konflikte moderieren**
17. **Beteiligung an Diskussionen, insbesondere um zu helfen, mehr**

Optionen zu schaffen und auf Daten aufmerksam zu machen sowie Beobachtungen wiederzugeben (auch mal auf Gutes hinweisen, also Dinge, die schon gut laufen)

18. Sessions zum Thema Eigenverantwortlichkeit organisieren
19. Dem Team helfen, Akzeptanzkriterien direkt in testbare Form zu bringen und dann entsprechend automatisiert zu testen
20. In Konfliktsituationen Einzelgespräche mit Teammitgliedern führen
21. **Das Team vor unerwünschten Einflüssen von außen schützen**, also z.B. Teammitgliedern den Rücken stärken, die von ihrem Chef für nicht vereinbarte zusätzliche Aufgaben abgezogen werden sollen

Teamübergreifende Organisationsebene

22. Unterstützung bei der Organisation von teamübergreifendem Wissenstransfer zwischen Programmierer/innen, Tester/innen etc., beispielsweise in *Communities of Practice* (CoP)
23. Austausch mit anderen Scrum Mastern (z.B. in einer Scrum Master-CoP, aber auch über Community-Events), um über Herausforderungen und Verbesserungen zu sprechen und um neue Ideen für Verbesserungsmaßnahmen zu bekommen
24. Neue Scrum Master ausbilden
25. Teilnahme an Meetings und Gesprächen mit Zulieferern des Teams oder Empfängern von Teamergebnissen gemeinsam mit Entwickler/innen und dem Product Owner, damit das Team optimal in die Gesamtprozesse eingebunden ist und immer alle nötigen Informationen hat (und weitergibt)
26. **Scrum erklären: Verantwortlichkeiten, Events, Artefakte und Werte erklären für das Team, aber auch für weitere Personen im Unternehmen oder bei Kunden**
27. Wenn es schon halbwegs läuft mit Scrum, an Organisationsmeetings teilnehmen, die das Team betreffen (könnten), um Anregungen für mehr oder konsequenteres Scrum zu geben, die Teambedürfnisse zu kommunizieren und um direktere Kommunikation mit dem Team herzustellen
28. Teamübergreifenden Austausch anregen (auf Product Owner- und Teamebene)
29. **Mit Rat und Tat Fragen zu Scrum beantworten für das Team und Außenstehende**
30. **Mit Managern, Projektleitern, Teamleitern etc. über Rechte und Pflichten der Teams sprechen und darüber, wie die Teams gestärkt werden können**
31. Scrum/agil der Personalabteilung erklären
32. **Zusammenspiel/Abstimmung zwischen Teams verbessern**
33. Manager dabei unterstützen, das Team für schwierige personelle Situationen Lösungen finden zu lassen, anstatt selbst Lösungen vorzugeben
34. Die anderen Scrum Master unterstützen und coachen
35. Änderungen der Team-Zusammensetzung moderieren
36. Das Controlling mit der neuen Scrum-Welt in Verbindung bringen
37. Die unternehmensinterne Vernetzung der Scrum Master und „Agilen“ über Sparten hinaus begleiten

Anforderungsebene und Product Owner unterstützen

38. **Bei Story-Schnitt und Backlog-Organisation den Product Owner unterstützen**
39. **Den Product Owner beim Stakeholder-Management unterstützen**
40. **Mit dem Product Owner und mit den Entwickler/innen das Schreiben von User Stories üben**
41. **Den Product Owner dabei unterstützen, die Anforderungsflut strukturierter zu bewältigen**
42. Die Prozessfindung beim Portfoliomanagement der Product Owner und Stakeholder begleiten

Verantwortlichkeit: Product Owner

Die Aufgaben des Product Owners variieren abhängig von Unternehmen und Projekt. Die folgende Liste enthält Beispiele von Product Owner-Aufgaben aus der Praxis. Die **fett** gesetzten Punkte sind die Aufgaben, die der Product Owner auf jeden Fall wahrnehmen muss.

Produkteigenschaften

1. Produktvision erstellen

2. Produktvision an Stakeholder und im Scrum Team kommunizieren
3. **Produktziel definieren**
4. **Produktziel an Stakeholder und im Scrum Team kommunizieren**
5. **Product Backlog-Einträge erstellen** (allein, mit Stakeholdern, mit den Entwickler/innen)
6. **Akzeptanzkriterien für Product Backlog Items formulieren** (in der Regel zusammen mit den Entwickler/innen)
7. **Ordnen/Priorisieren des Product Backlogs** (inkl. Entscheidung, was entwickelt wird und was nicht)
8. **Die bereits entwickelten Produktinkremente kennen**
9. **Mit den bereits entwickelten Produktinkrementen „herumspielen“**
10. Die Wertschöpfung des Produkts definieren
11. **Die Wertschöpfung des Produkts kennen, messen und optimieren**
12. Produktbezogene Feedbackschleifen installieren und verkürzen

Zusammenarbeit mit dem Team

13. **Refinement des Product Backlogs** (in der Regel zusammen mit den Entwickler/innen)
14. **Zu große Product Backlog Items aufsplitten** (in der Regel zusammen mit den Entwickler/innen), sodass sie in Sprints passen
15. **Eine Sprintziel-Skizze in das Sprint Planning mitbringen**
16. **Hoch priorisierte, gut ausgearbeitete Product Backlog-Einträge in das Sprint Planning mitbringen**
17. **Mitarbeit im Sprint Planning**
18. **Beantwortung fachlicher Fragen der Entwickler/innen im Sprint Planning und während des Sprints**
19. Teilnahme an Daily Scrums
20. **Teilnahme an und Mitarbeit in Sprint-Retrospektiven**
21. **Dem Scrum Team helfen, seinen Prozess zu verbessern**
22. Definition der *Definition of Ready* zusammen mit Entwickler/innen und Scrum Master
23. **Definition der *Definition of Done* zusammen mit den Entwickler/innen und Scrum Master**
24. **Feedback zu implementierten Product Backlog-Einträgen an das Team im Sprint oder im Sprint Review**
25. **Den Entwickler/innen eigene und Stakeholder-Unzufriedenheiten deutlich machen und erklären,** Mitarbeit bei der Suche nach Lösungen
26. Im Scrum Team die relevanten Geschäftszahlen/KPIs transparent machen
27. Im Scrum Team verdeutlichen, wie das Produkt auf dem Markt bzw. bei den Kunden ankommt

Kunden/Anwender

28. **Kundenbedürfnisse verstehen** (mit Kunden/Anwendern sprechen)
29. **Den Markt verstehen**
30. **Ausgewählte Kunden/Anwender in die Sprint Reviews integrieren**
31. **Aufsetzen und Durchführen geeigneter Erfolgsmetriken** (z.B. Kundenzufriedenheit über den *Net Promoter Score* messen)
32. **Risikomanagement über die Ordnung/Priorisierung des Product Backlogs**
33. **Annahmen über Kunden/Anwender/Märkte testen** (z.B. mit einem *Minimum Viable Product*)

Management sonstiger Stakeholder

34. **Dafür sorgen, dass die richtigen Stakeholder zum Sprint Review kommen**
35. Erstellung und Aktualisierung des Releaseplans
36. Aktualisierung des Release Burnup Charts
37. Kommunikation von Status, Fortschritt und Planung an die Stakeholder (z.B. mit Release Burnups, Cumulative Flow Diagrams oder Parking Lot Diagrammen)
38. Stakeholder über neue Produkteigenschaften informieren
39. **Budgetkontrolle**

Verantwortlichkeit: Entwickler/innen

Die Aufgaben der Entwickler/innen variieren abhängig von Unternehmen und Projekt. Was zu den Aufgaben der Entwickler/innen gehört und was nicht, wird zum Großteil über die *Definition of Ready* und die *Definition of Done* formuliert. Die folgende Liste enthält Beispiele von Aufgaben der Entwickler/innen aus der Praxis. Die **fett** gesetzten Punkte sind die Aufgaben, die die Entwickler/innen auf jeden Fall in Scrum wahrnehmen müssen.

Arbeitsorganisation

1. **Festlegen, wieviele Product Backlog-Einträge in den Sprint gezogen werden**
2. **Umsetzungsplan im Sprint Planning erstellen**
3. **Organisation der Teamarbeit im Daily Scrum**
4. Pair Programming mit Teammitgliedern
5. **Einarbeitung neuer Teammitglieder**

Technisch

6. **Produktinkremente programmieren, testen und dokumentieren**
7. Automatisierte Tests (Unit Tests, Integrations-, Last-, Akzeptanztests) erstellen und kontinuierlich durchführen
8. **System- und Softwarearchitektur erstellen**
9. **Softwaretechnischer Entwurf**
10. **Auswahl geeigneter Technologien für die Umsetzung**
11. **Umgebung für Continuous Integration aufsetzen und am Laufen halten**
12. Betrieb und Support der entwickelten Software

Bezogen auf Stakeholder

13. Usability-Tests durchführen
14. User Acceptance Tests durchführen
15. **Produktinkremente im Sprint-Review demonstrieren**
16. User Experience gestalten
17. **Bugs beseitigen**

Unterstützung des Product Owners

18. Schätzung des Product Backlogs
19. **Den Product Owner bei der Konzeption unterstützen.**
20. **Zusammen mit dem Product Owner Product Backlog-Einträge erstellen und im Refinement verfeinern**
21. **Zusammen mit dem Product Owner Akzeptanzkriterien für Product Backlog-Einträge erstellen**

Verbesserung

22. **Sich selbst bezüglich Technologien, Vorgehen und Fachwissen weiterentwickeln**
23. Zusammen mit dem Product Owner und Scrum Master die *Definition of Ready* formulieren
24. **Zusammen mit dem Product Owner und Scrum Master die *Definition of Done* formulieren**

Event: Sprint Planning

- Ergebnisse: Sprintziel, selektierte Einträge aus dem Product Backlog, Plan für die Umsetzung
- Dauer: maximal zwei Stunden pro Sprint-Woche (also vier Stunden für einen zweiwöchigen Sprint)
- Teilnehmer: Product Owner, Scrum Master, Entwickler/innen, bei Bedarf eingeladene Fachexperten für spezifische anstehende Fachfragen
- Mögliches Vorgehen:
  1. Der Scrum Master fragt das Team, welche Dinge innerhalb der Zeitspanne passieren, die seine Kapazität beeinflussen.
  2. Der Product Owner stellt seine Idee für ein Sprintziel vor sowie die hoch priorisierten Product Backlog Items.
  3. Der Scrum Master fragt die Entwickler/innen, ob das erste Product Backlog Item in den Sprint passt. Beantwortet die Entwickler/innen die Frage positiv, fragt der Scrum Master, ob das zweite Product Backlog Item zusätzlich in den Sprint passt. Dieses Verfahren wird so lange wiederholt, bis die Entwickler/innen Zweifel haben, ob es noch mehr schaffen kann.
  4. Jetzt wird das Sprintziel überarbeitet und finalisiert. Der Product Owner schätzt ab, ob der Sprint einen positiven ROI (Return on Investment) hat, wenn die gewählten Backlog Items umgesetzt werden können. Wenn dies nicht der Fall ist, geht das Scrum Team zurück zum ersten Schritt.
  5. Dann wird der sogenannte Task-Breakdown durch die Entwickler/innen eingeleitet. Dazu werden Kleingruppen von jeweils zwei bis drei Entwickler/innen gebildet. Jede Kleingruppe wählt einen Teil der Backlog Items aus und erstellt die technischen Tasks für die Umsetzung.
  6. Die erstellten Tasks werden anschließend im Plenum vorgestellt, und es wird Feedback eingesammelt. Gegebenenfalls wird eine zweite Runde Kleingruppenarbeit angeschlossen.

7. Es wird auf Basis der erstellten Tasks geprüft, ob die ausgewählten Backlog Items tatsächlich im Sprint umgesetzt werden können.
8. Der Product Owner wird über das Ergebnis der Abschätzung informiert. Gegebenenfalls wird ein Backlog Item aus dem Sprint Backlog entfernt oder eine weiteres hinzugefügt. Wenn notwendig, wird das Sprintziel erneut angepasst.

- Empfehlungen:
  - Der Beamer bleibt aus. Der Product Owner bringt die Product Backlog Items auf Papier mit. Die Tasks werden ebenfalls auf Papier erstellt. Wird das Sprint Planning remote durchgeführt, empfiehlt sich ein digitales Whiteboard.
  - Der Product Owner bleibt während des Task-Breakdown im Raum. (Häufig treten bei dieser Tätigkeit weitere fachliche Rückfragen auf.)
  - Für die Tasks gilt die Regel, dass sie maximal einen Personentag an Aufwand groß sein dürfen. Tasks müssen also entsprechend klein gestaltet sein.
  - Für fortgeschrittene Teams: Task-Breakdown durch immer kleinere User Stories ersetzen.

Event: Daily Scrum

- Ergebnis: Einsatzplanung für das Team für den Tag, so dass das Team Wertschöpfung maximiert
- Dauer: max. 15 Minuten (jeden Werktag zur selben Uhrzeit am selben Ort)
- Teilnehmer: Entwickler/innen und Scrum Master; Product Owner optional; Stakeholder optional (Stakeholder dürfen zuhören, aber nicht sprechen)
- Mögliches Vorgehen (die Entwickler/innen beantworten drei Fragen):
  1. Was habe ich gestern erledigt, was uns hilft, das Sprintziel zu erreichen? Ggfs. Sprint Burndown aktualisieren.
  2. Habe ich Hindernisse wahrgenommen, die mich oder uns daran hindern, das Sprintziel zu erreichen?
  3. Was werde ich heute erledigen, damit wir gemeinsam das Sprintziel erreichen?
- Empfehlungen:
  - Das Daily Scrum findet vor einem physikalischen Taskboard statt.
  - Die ersten beiden der obigen Fragen werden einzeln von den Entwickler/innen beantwortet. Wenn diese beiden Fragen von allen beantwortet wurden, wird die dritte Frage gemeinsam im Team beantwortet.
  - Hindernisse, die die Weiterarbeit an einem Sprint Backlog Item oder einem Task blockieren, werden mit roten Haftnotizen direkt auf den zugehörigen User Stories bzw. Tasks kenntlich gemacht.
  - Andere Hindernisse werden in der Nähe des Taskboards visualisiert.

Event: Sprint Review

- Ergebnisse: Klarheit darüber, was am Produkt mit hoher Priorität noch zu tun ist; Änderungen am Product Backlog; ggf. Fortschreibung des Releaseplans
- Dauer: ca. eine Stunde pro Sprint-Woche (also zwei Stunden für einen zweiwöchigen Sprint)
- Teilnehmer: Product Owner, Scrum Master (Moderation), Entwickler/innen, Stakeholder (insbesondere Kunden und Anwender)
- Mögliches Vorgehen:
  1. Demonstration des Produktinkrements durch die Entwickler/innen. Die Demonstration erfolgt auf einer vorher vereinbarten Test- und Integrationsumgebung und nicht auf einem Entwicklungsrechner. Es darf nur gezeigt werden, was gemäß der Definition of Done komplett erledigt ist.
  2. Gegebenenfalls Akzeptanz des Produktinkrementes durch den Product Owner (wenn nicht bereits im Sprint erfolgt)
  3. Gegebenenfalls Aktualisierung des Release Burnup Charts o.Ä. (siehe unten)
  4. Feststellung durch den Product Owner, ob bzw. inwieweit das Sprintziel erreicht wurde
  5. Sammeln von Feedback zum Produkt; Festhalten des Feedbacks durch den Product Owner
  6. Feststellen, welches Feedback besonders dringlich ist. Anpassung des Product Backlogs bezüglich dieses dringlichen Feedbacks durch den Product Owner
  7. Gegebenenfalls Anpassung des Releaseplans

- Empfehlungen:
  - Der Product Owner sorgt dafür, dass die richtigen Stakeholder beim Sprint-Review anwesend sind.
  - Es sind Kunden und Anwender anwesend.
  - Der Product Owner bettet den aktuellen Sprint in das Gesamtvorhaben ein.
  - Die Demonstration des Produktinkrements basiert auf dem Sprintziel und erzählt eine Geschichte, die es den Stakeholdern erleichtert, das Gezeigte in einen geeigneten Kontext zu setzen.
  - Der Scrum Master sorgt durch Moderation dafür, dass die Stakeholder nützliches Feedback zum Produkt geben.
  - Bei vielen Stakeholdern im Sprint Review sorgt der Scrum Master durch geeignete Techniken der Großgruppenmoderation für die effektive Durchführung des Sprint Reviews.

Event: Sprint-Retrospektive

- Ergebnisse: Verbesserungsmaßnahmen, die das Scrum Team im nächsten Sprint umsetzen will
- Dauer: ca. eine Stunde pro Sprintwoche (also zwei Stunden für einen zweiwöchigen Sprint)
- Teilnehmer: Scrum Master (als Moderator), Product Owner, Entwickler/innen
- Mögliches Vorgehen:
  - *Set the stage*: Der Scrum Master eröffnet die Retrospektive und stellt eine Arbeitsumgebung her, in der sich alle Teilnehmenden engagieren mögen.
  - *Gather data*: Die Teilnehmenden sammeln qualitative und quantitative Daten über den letzten Sprint.
  - *Generate insights*: Die Teilnehmenden gewinnen Einsichten darüber, warum bestimmte positive oder negative Effekte aufgetreten sind.
  - *Decide what to do*: Die Teilnehmenden entscheiden, was sie tun wollen, um negative Effekte zu beseitigen oder zu dämpfen oder um positive Effekte zu verstärken oder zu erhalten.
  - *Closing*: Der Scrum Master beendet die Retrospektive und sorgt dafür, dass sich jemand um die Ergebnisse kümmert.

- Empfehlungen:
  - Es sollten nur wenige Maßnahmen vereinbart werden, die das Scrum Team auch realistisch im nächsten Sprint umsetzen kann.
  - Es sollte auch über Stimmungen und Gefühle gesprochen werden.
  - Der Scrum Master sollte die verwendeten Techniken variieren.
  - Es sollte geprüft werden, ob die Maßnahmen der letzten Retrospektive umgesetzt wurden und welche Effekte die Maßnahmen gezeigt haben.

Artefakt: Product Backlog

- Zweck: Überblick über die noch ausstehenden Eigenschaften/Features des Produkts
- Eigenschaften:
  - Produktziel beschreibt das Warum
  - Einträge beschreiben das Was und nicht das Wie.
  - Geordnet (in der Regel nach Priorität)
  - Hoch priorisierte Einträge sind klein und detailliert ausgearbeitet.
  - Niedrig priorisierte Einträge sind groß und nur grob skizziert.
  - Transparent im Scrum Team und für die Stakeholder
- Verwendung:
  - Ordnung/Priorisierung durch den Product Owner
  - Schätzung durch die Entwickler/innen, falls für den Kontext notwendig
  - Basis für Releaseplanung und –Controlling
- Empfehlungen:
  - Das Product Backlog enthält gut sichtbar das Produktziel.
  - Das Product Backlog existiert physikalisch (Karteikarten/Haftnotizen an der Wand). Bei Remote Teams: virtuelles Whiteboard wie Miro, Mural oder Concept Board.
  - Beschränkung auf maximal 70–80 Einträge pro Release (noch besser: ein bis zwei Dutzend)
  - Release-Dauer max. 6 Monate, besser nur 3 Monate
  - Ist nur in dem Maße geschätzt, wie für die Release-Planung notwendig.
  - Unterscheidung in Einträge, die für das aktuelle Release geplant sind und solche für später (Ideen)
  - User Stories als Product Backlogs Items (wenn im Unternehmen nicht bereits ein anderes gut funktionierendes Format etabliert ist)

- Gruppierung der Einträge nach Themen, Epics oder über Story Mapping oder Impact Mapping.
- Dreistufige Gruppierungen vermeiden, SIE deuten darauf hin, dass zu früh zu detailliert geplant wird.
- Einen Bereich „Ready for Sprint" vorsehen mit den Einträgen, die für den nächsten Sprint vorbereitet sind.
- Bugs, die nicht sofort beseitigt werden, werden ins Product Backlog aufgenommen und durch den Product Owner priorisiert.

Artefakt: Sprint Backlog

- Zweck: Überblick über die noch ausstehenden Arbeiten im Sprint
- Eigenschaften:
  - Sprintziel beschreibt das Warum
  - Enthält die für den Sprint ausgewählten Product Backlog-Einträge sowie den Plan für die Umsetzung (oft technische Tasks)
  - Geordnet (in der Regel nach Priorität)
  - Zeigt den Zustand der Planumsetzung.
- Verwendung:
  - Wird im Sprint Planning durch die Entwickler/innen erstellt.
  - Aktualisierung durch die Entwickler/innen im Daily Scrum
- Empfehlungen:
  - Das Sprint Backlog existiert physikalisch in Form eines Taskboards (Karteikarten/Haftnotizen an der Wand) im Teamraum. Bei Remote-Arbeit: Die Entwickler/innen entscheiden über das eingesetzte Tool.
  - Die Reihenfolge auf dem Taskboard bildet die Priorisierung der Product Backlog-Einträge ab.
  - Darstellung von Sprintziel und Definition of Done auf dem Taskboard
  - User Stories und Tasks als Einträge (wenn im Unternehmen nicht bereits ein anderes gut funktionierendes Format etabliert ist)
  - Eigene Zeile oben auf dem Taskboard für ungeplante Bugs, die noch im Sprint erledigt werden müssen

Artefakt: auslieferbares Produktinkrement

- Zweck: Wertschöpfung für das Unternehmen und Kunden
- Eigenschaften:
  - Lieferbar (gemäß der Definition of Done), mindestens:
    - funktionsfähig unter Produktionsbedingungen
    - qualitätsgesichert
    - dokumentiert
- Verwendung:
  - Erstellung und Qualitätssicherung im Sprint durch die Entwickler/innen
  - Demonstration im Sprint Review auf einer vorher vereinbarten Test- und Integrationsumgebung, um Feedback zum Produkt zu bekommen
- Empfehlungen:
  - Automatisierte Unit Tests und Continuous Integration als Instrumente zur Qualitätssicherung (inkl. Regression)
  - Testgetriebene Entwicklung und Refactoring, um bei inkrementeller Entwicklung eine Erosion der Entwurfs- und Codequalität zu vermeiden
  - Notwendige Dokumentation über die Definition of Done vereinbaren