30-seitiges kostenloses E-Book: dpunkt.de/scrum-bierdeckel

Scrum –

auf dem Bierdeckel erklärt

Begriffe, Konzepte, Grundverständnis

2., aktualisierte Auflage 2022

it-agile

dpunkt.verlag

Über diese Broschüre

Diese Broschüre führt in Scrum ein. Sie erklärt die grundlegenden Begriffe und Konzepte und erläutert, wie diese zusammenhängen. Die Scrum-Mechanik ist nur die eine Seite der Medaille. Die dahinterstehenden Werte und Prinzipien sind mindestens genauso wichtig. Daher folgt nach der Scrum-Einführung eine Beschreibung des Agilen Manifestes, das die agilen Prinzipien definiert. Daran schließt sich eine Übersicht über die Scrum-Verantwortlichkeiten, -Meetings und -Artefakte an, die zum Nachschlagen geeignet ist. Den Abschluss der Broschüre bildet ein Abschnitt zu den Herausforderungen bei der Scrum-Einführung.

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

Stefan Roock, Januar 2022
Scrum-Trainer bei it-agile
stefan.roock@it-agile.de, Tel. 0172/429 76 17

1 z.B. Henning Wolf, Stefan Roock: Scrum – verstehen und erfolgreich einsetzen. dpunkt.verlag, 3. Auflage, 2021.
2 z.B. *https://www.it-agile.de/schulungen/scrum/*

dpunkt.verlag

it-agile

Inhalt

Scrum – drei Perspektiven

Man kann Scrum in einem Satz beschreiben:

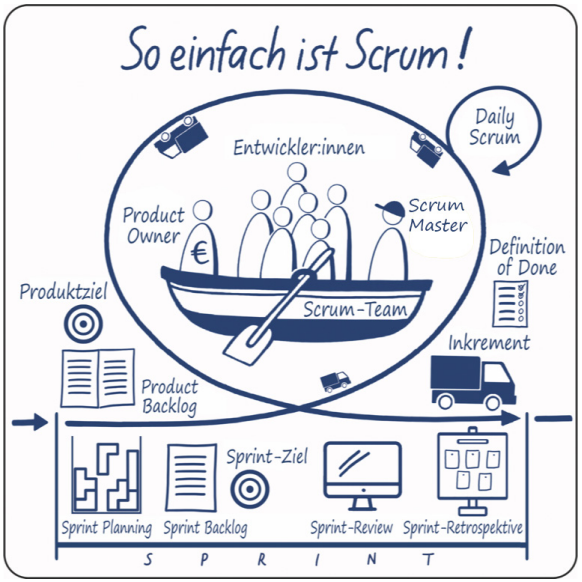
Scrum bedeutet:
Autonome Teams mit Businessfokus, die Verantwortung für ihren Prozess übernehmen.

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

1. Die *Produktperspektive* (Businessfokus) beleuchtet, wie Produkte definiert und verbessert werden.
2. Die *Entwicklungsperspektive* (autonome Teams) beleuchtet, wie Teams Produkte entwickeln.
3. Die *Verbesserungsperspektive* (Verantwortung für den Prozess übernehmen) beleuchtet, wie Zusammenarbeit und Prozesse verbessert werden.

Diese drei Perspektiven finden sich im Scrum-Framework wieder, das so einfach ist, dass es auf einen Bierdeckel passt (siehe Abbildung rechts)[1].

Die folgenden Abschnitte beschreiben die drei dargestellten Perspektiven ausführlicher.



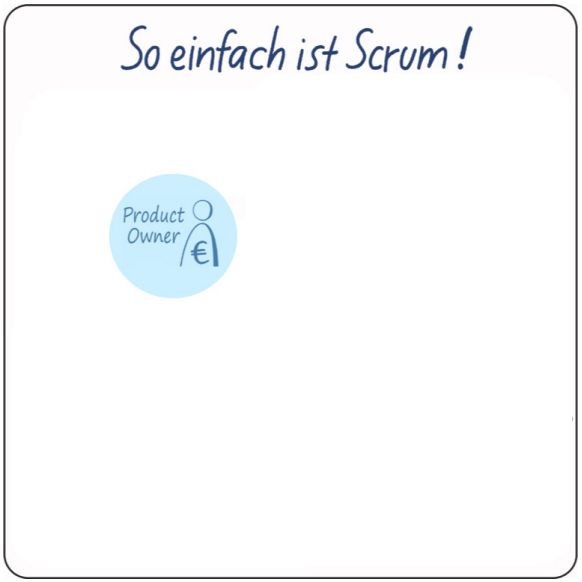
1 Den dargestellten Scrum-Bierdeckel gibt es im it-agile-Shop: *https://www.itagileshop.de/inspirieren.*

Produktperspektive

Die Produktperspektive beginnt mit der *Product-Owner-Verantwortlichkeit*[2]. Der Product Owner verantwortet den Produkterfolg, indem er den Produktnutzen durch die Priorisierung der Produkt-Features optimiert. Der Product Owner ist die Person, die das Produkt entwickelt haben möchte.

Wenn jemand ein Produkt entwickelt haben möchte, aber die Product-Owner-Verantwortlichkeit nicht übernehmen kann oder möchte, kann er die Verantwortlichkeit *vollständig* an eine andere Person abgeben. Auf jeden Fall muss der Product Owner aber bevollmächtigt sein, die Produktentscheidungen zu treffen. Man kann sich den Product Owner auch als Unternehmer im Unternehmen vorstellen.

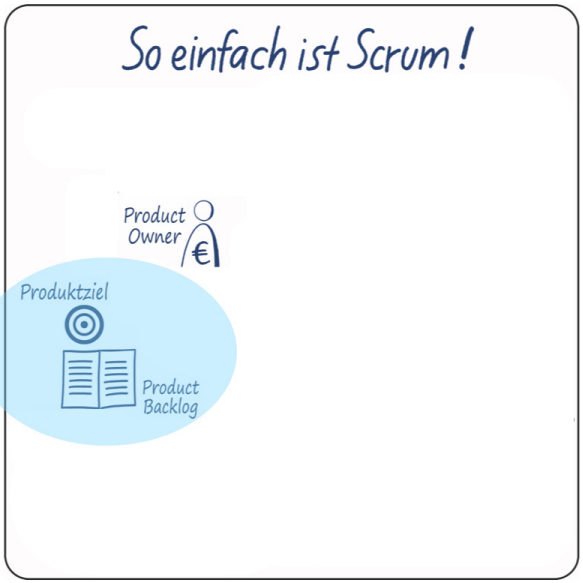
Für den Product Owner gilt das Highlander-Prinzip: »Es kann nur einen geben.« Die Verantwortlichkeit kann in Scrum nicht von mehreren Personen geteilt wahrgenommen werden und schon gar nicht durch ein Komitee. Man möchte in Scrum, dass der Product Owner mit *einer* Stimme gegenüber dem Team und den Stakeholdern spricht und Entscheidungen schnell fällen kann.



»Der Product Owner optimiert den Produktnutzen durch Priorisierung der Produkteigenschaften.«

2 In früheren Versionen des Scrum Guide war von »Rollen« die Rede.

Der Product Owner verfolgt immer genau ein *Produktziel*. Passend zum Produktziel pflegt der Product Owner ein *Product Backlog*, in dem die Produkteigenschaften beschrieben sind, die für das Produktziel notwendig erscheinen. Das Product Backlog wird durch den Product Owner *geordnet*. In der Praxis bedeutet diese Ordnung meist Priorisierung nach Geschäftswert. Die Entwickler:innen *schätzen* bei Bedarf das Product Backlog.

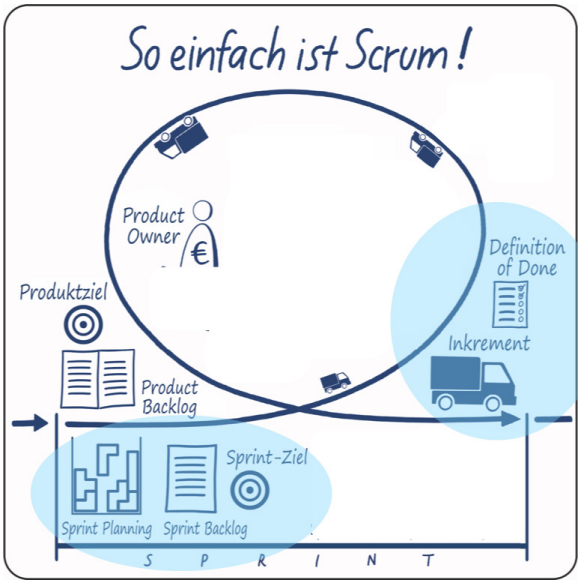


Scrum legt nicht fest, wie genau die Einträge des Product Backlog gestaltet sind. Viele Teams machen gute Erfahrungen mit User Stories: exemplarische Benutzungsszenarien aus Sicht eines Benutzers. User Stories haben eine andere Qualität als klassische Anforderungen. Bei User Stories liegt der Fokus darauf, ein gemeinsames Verständnis bei allen Beteiligten zu erzeugen, und nicht darauf, dass die Beschreibung vollständig, widerspruchsfrei und korrekt ist.

»Das Product Backlog ist ein dynamisches Artefakt mit den Produkteigenschaften, die der Product Owner für entscheidend hält.«

Die Entwicklung erfolgt in Iterationen, die in Scrum *Sprints* heißen. Sprints haben eine immer gleiche Länge von maximal 4 Wochen. Was warum im Sprint entwickelt wird, wird im *Sprint Planning* festgelegt. Dort wird das Sprint-Ziel definiert und hoch priorisierte Einträge aus dem Product Backlog ausgewählt, von denen die Entwickler:innen meinen, dass sie sie im Sprint umsetzen können. Sprint-Ziel und ausgewählte Einträge aus dem Product Backlog werden im *Sprint Backlog* festgehalten.

Das Ergebnis eines jeden Sprints ist ein lieferbares *Produktinkrement*. Ob das Produktinkrement tatsächlich an Kunden ausgeliefert wird, entscheidet der Product Owner. Die im Produktinkrement implementierten Produkteigenschaften müssen aber auf jeden Fall produktreif sein (mind. entwickelt und qualitätsgesichert). Was genau »produktreif« bedeutet, wird in der *Definition of Done* festgelegt.



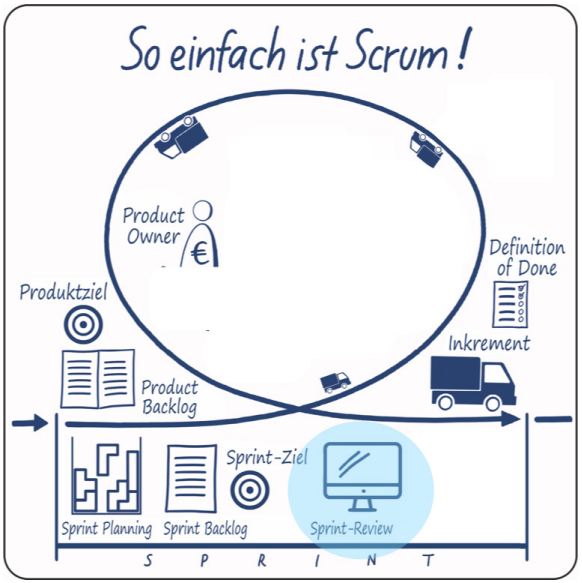
»Am Ende des Sprints steht ein nützliches, lieferbares Produktinkrement.«

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

Beim Feedback sollte die Frage im Vordergrund stehen, was das Produktinkrement im Einsatz für Kunden bewirken würde. Daher sollten am Sprint-Review auch stets Kunden bzw. Benutzerinnen teilnehmen.

Gute Fragen, um nützliches Feedback zu erhalten, sind:

- »Was hindert uns daran, das vorliegende Produktinkrement produktiv zu nutzen?«
- »Wie kann das vorliegende Produktinkrement noch wertvoller gestaltet werden?«



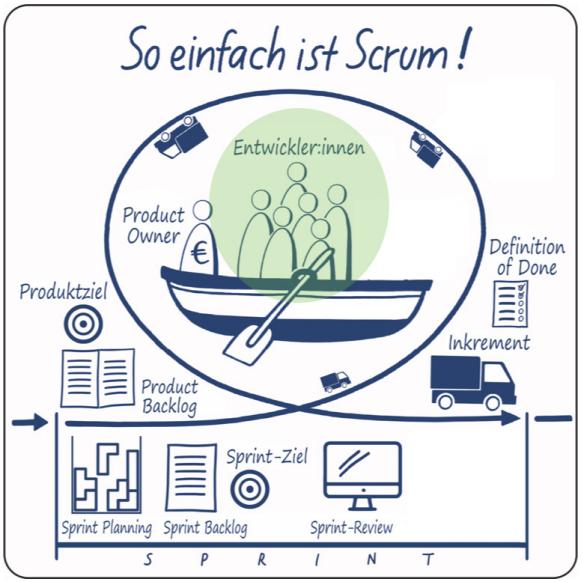
»Im Sprint-Review wird Feedback zum Produktinkrement eingesammelt, um das Produkt zu optimieren.«

3 Stakeholder ist in Scrum jede Person, die Interesse am Produkt oder Einfluss auf die Entwicklung hat: Kunden, Benutzerinnen, Sponsoren, Management, Betriebsrat etc.

Entwicklungsperspektive

Die Entwickler:innen entwickeln ausgehend vom Sprint Backlog ein lieferbares Produktinkrement. Im Scrum-Team arbeiten 3–9 Personen, die gemeinsam alle Fähigkeiten besitzen, die notwendig sind, um das Sprint Backlog in das Produktinkrement zu überführen. Dazu gehören demnach nicht nur solche, die programmieren. Je nach Kontext gehören auch Personen mit Expertise in User Experience Design (UX Design), Dokumentation oder Qualitätssicherung dazu.

Die Entwickler:innen managen sich selbst. Es gibt weder eine formelle Hierarchie noch herausgehobene Verantwortlichkeiten oder Positionen unter ihnen.

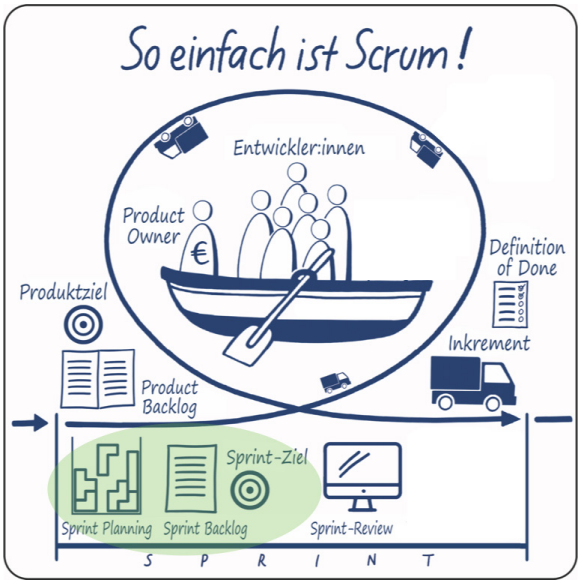


»Die Gruppe der Entwickler:innen ist cross-funktional besetzt und managed sich selbst.«

Die Entwickler:innen bestimmen im *Sprint Planning*, wie viel Arbeit sie in den Sprint aufnehmen. Sie wenden das Pull-Prinzip an (sie »ziehen« Arbeit in den Sprint). Das steht im Gegensatz zum Push-Prinzip, bei dem eine andere Person außerhalb festlegt, wie viel Arbeit die Entwickler:innen in einem vorgegebenen Zeitraum zu erledigen haben.

Die Entwickler:innen machen eine Vorhersage (Forecast) darüber, was sie im Sprint schaffen können. Diese Vorhersage soll die Qualität einer Wettervorhersage haben. In der Regel sollte das geliefert werden, was eingeplant wurde. Es sollte aber niemand übermäßig überrascht sein, wenn das hin und wieder nicht der Fall ist.

Für die ausgewählten Einträge aus dem Product Backlog erstellen die Entwickler:innen einen Plan für die Umsetzung im Sprint. Das *Sprint-Ziel* bildet zusammen mit den ausgewählten Einträgen aus dem Product Backlog und dem Umsetzungsplan das *Sprint Backlog*.

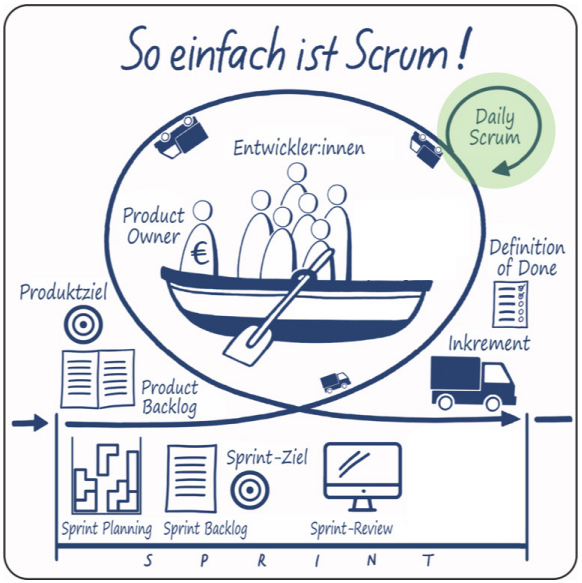


»Die Entwickler:innen ziehen Arbeit in den Sprint und erstellen einen Forecast, wie viel sie schaffen können.«

Während des Sprints treffen sich die Entwickler:innen werktäglich zum *Daily Scrum*, um sich über den Arbeitsfortschritt und die nächsten Aufgaben im Sprint abzustimmen. Dazu kommen sie jeden Werktag zur gleichen Uhrzeit am immer gleichen Ort für maximal 15 Minuten zusammen und beantworten drei Fragen:

1. Was habe ich seit dem letzten Daily Scrum erledigt, das uns dem Sprint-Ziel näherbringt?
2. Welche Hindernisse sehe ich auf dem Weg zum Sprint-Ziel?
3. Was plane ich, bis zum nächsten Daily Scrum zu erledigen, das uns dem Sprint-Ziel näherbringt?

Der Product Owner nimmt optional am Daily Scrum teil.

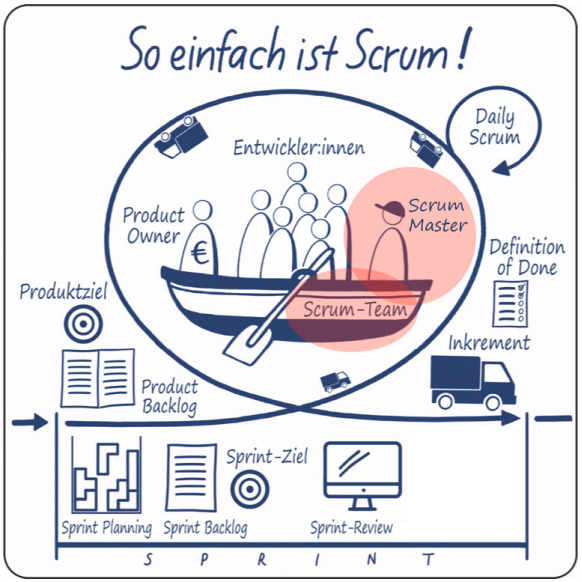


»Im Daily Scrum nehmen die Entwickler:innen die Einsatzplanung für den Tag vor.«

Verbesserungsperspektive

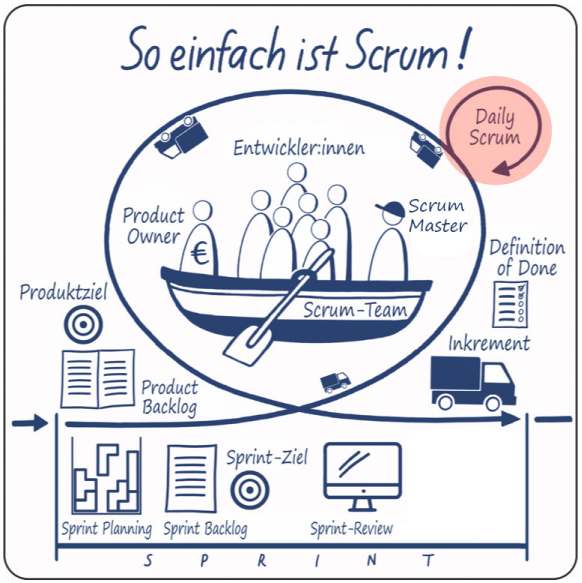
Der *Scrum Master* sorgt für ein hocheffektives *Scrum-Team*, bestehend aus den Entwickler:innen, dem Product Owner und dem Scrum Master.

Er sorgt dafür, dass Product Owner, die Entwickler:innen und Stakeholder (inkl. Management) verstehen, wie Scrum funktioniert, und hilft ihnen, Scrum effektiv anzuwenden. Gegenüber den Entwickler:innen schafft er einen Rahmen, in dem sie sich selbst organisieren können. Er hält dem ganzen Scrum-Team immer wieder den Spiegel vor, damit dieses sich weiter verbessern kann. Der Scrum Master kümmert sich außerdem darum, dass Hindernisse identifiziert und beseitigt werden. Er moderiert die Scrum-Meetings (insbesondere Sprint-Planning, Daily Scrum, Sprint-Review und Sprint-Retrospektive).



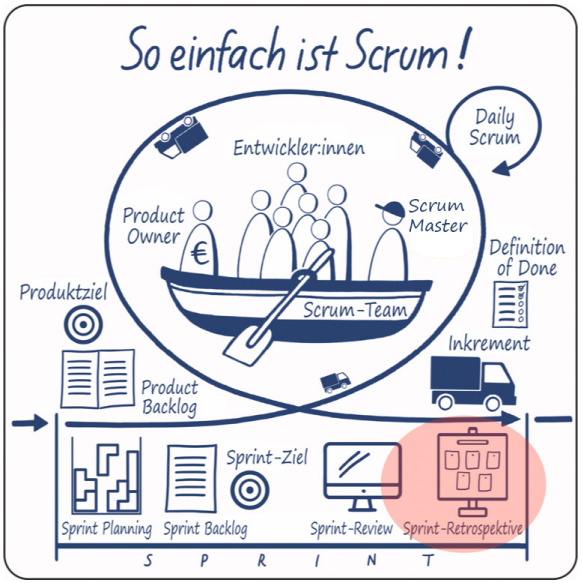
»Der Scrum Master sorgt für ein hocheffektives Scrum-Team.«

Der kontinuierliche Verbesserungsprozess ist bei Scrum in zwei Meetings verankert: *Daily Scrum* und *Sprint-Retrospektive*. Verbesserungen nehmen 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 das Hindernis alleine aus der Welt schafft. Er wird dazu in der Regel mit weiteren Parteien im Unternehmen interagieren müssen (z. B. um finanzielle Mittel für schnellere Rechner zu beschaffen).



»Im Daily Scrum werden Hindernisse identifiziert.«

Am Ende des Sprints findet nach dem Sprint-Review die *Sprint-Retrospektive* statt. In dieser reflektiert das 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 umsetzen will. Die Sprint-Retrospektive wird durch den Scrum Master moderiert.



»In der Sprint-Retrospektive vereinbart das Scrum-Team Verbesserungsmaßnahmen für den nächsten Sprint.«

Das Agile Manifest

Für agile Entwicklung gibt es mit dem *Agilen Manifest*[4] ein Leitbild dafür, was Agilität bedeutet.

»Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen.

Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

- Individuen und Interaktionen mehr als Prozesse und Werkzeuge
- Funktionierende Software mehr als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
- Reagieren auf Veränderung mehr als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.«

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 Kontexten keinen Geschäftswert. In dynamischen Kontexten brauchen wir die Flexibilität, die uns die Dinge auf der linken Seite geben.

4 *http://agilemanifesto.org*

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.

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 zufriedenzustellen.
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.

Das Agile Manifest ist vor dem Hintergrund agiler Softwareentwicklung entstanden. Wenn man überall »Software« durch »Produkt« ersetzt, funktioniert es aber für jegliche andere Produktentwicklung ebenso.

Überblick über die Scrum-Verantwortlichkeiten, -Meetings und -Artefakte

Dieser Abschnitt enthält kurze, griffige Übersichten und Checklisten für die Verantwortlichkeiten, Meetings und Artefakte von Scrum[5]. Diese sollten auf keinen Fall dogmatisch verwendet werden. Es gibt nicht die eine richtige Form, die Events durchzuführen, die Verantwortlichkeiten auszuleben oder die Artefakte zu gestalten. Dieser Abschnitt kann aber als Kurzreferenz dienen sowie als Startpunkt, um überhaupt einmal mit irgendetwas anzufangen. Wer Scrum allerdings nach fünf Sprints immer noch genauso praktiziert, wie es in den Übersichten und Checklisten dargestellt ist, macht etwas falsch: Inspektion und Adaption (Inspect&Adapt) des Prozesses fehlt!

Scrum-Master-Aufgaben

Die Meetings machen in Scrum in der Summe ca. 10 % der Zeit aus. Rechnen wir für die Vor- und Nachbereitung noch einmal dieselbe Zeit, verbleibt ein erheblicher Anteil Arbeitszeit, in der 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 Scrum-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 Scrum-Team Retrospektiven-Maßnahmen umsetzen**
2. Die Entwickler:innen unterstützen, ein besseres technisches Verständnis zu erwerben, dabei ggf. an Entwicklungs-meetings teilnehmen und agile Entwicklungspraktiken einführen (testgetriebene Entwicklung, kontinuierliche Integration, Pair Programming etc.)
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. Einzelgespräche mit Entwickler:innen und dem Product Owner führen; generell ein Ohr am Scrum-Team haben, um mitzubekommen, was los ist. Mögliche Fragen: Was brauchst du im/vom Team? Wie geht es dir gerade? Wie zufrieden bist du? Feedback an dich, Feedback von dir? Wie siehst du 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.)
6. **Hindernisse 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?)
7. **Für Festlegung von Spielregeln im Scrum-Team sorgen und diese gut sichtbar machen**
8. **Die Mitglieder des Scrum-Teams an die vereinbarten Spielregeln erinnern**

5 Der Abschnitt ist eine Variation des Textes aus Henning Wolf, Stefan Roock: Scrum – verstehen und erfolgreich einsetzen, dpunkt.verlag, 3. Auflage, 2021.

9. **Aha-Momente oder Leidensdruck erkennen als Initialpunkt für sofortige Veränderung** (nicht immer nur Input für Retrospektiven, oft auch direkt umsetzbar)
10. Netzwerk durchforsten nach Ideen, wie man Probleme/Herausforderungen des Scrum-Teams lösen könnte (Optionen schaffen)
11. Beurteilung der Situation mit anderen Scrum Mastern, Coaches oder Netzwerk diskutieren, um wach zu bleiben und nicht auf die eigene Perspektive beschränkt zu sein
12. Informativen Workspace gestalten bzw. das Scrum-Team anregen, ihn zu gestalten sowie aktuell und hilfreich zu halten
13. Organisatorische Aufgaben wie das Buchen von Meetingräumen etc. (die Mitglieder des Scrum-Teams sollten das aber auch selbst können)
14. Social Events für das Team-Building organisieren (das kann auch darin bestehen, Teammitglieder zu bestärken, die Organisation selbst zu übernehmen)
15. **Auf Missstände hinweisen, selbst wenn diese erst einmal für das Scrum-Team kein großes Problem zu sein scheinen** (Beispiel: Sprints werden nicht geschafft, was die Entwickler:innen zwar nicht so dramatisch finden, 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 auf Dinge, die schon gut laufen)
18. Sessions zum Thema Eigenverantwortlichkeit organisieren
19. **Die Erstellung eines Teamvertrags moderieren**
20. Dem Scrum-Team helfen, Akzeptanzkriterien direkt in testbare Form zu bringen und dann entsprechend automatisiert zu testen
21. In Konfliktsituationen Einzelgespräche mit Teammitgliedern führen
22. **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

23. Unterstützung bei der Organisation von teamübergreifendem Wissenstransfer zwischen Entwickler:innen, Tester:innen etc., beispielsweise in *Communities of Practice* (CoP)
24. 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
25. Neue Scrum Master ausbilden
26. Teilnahme an Meetings und Gesprächen mit Zulieferern des Teams oder Empfängern von Teamergebnissen gemeinsam mit Entwickler:innen und dem Product Owner, damit das Scrum-Team optimal in die Gesamtprozesse eingebunden ist und immer alle nötigen Informationen hat (und weitergibt)
27. **Scrum erklären: Verantwortlichkeiten, Meetings (Events), Artefakte und Werte für das Team erklären, aber auch für weitere Personen im Unternehmen oder bei Kunden**

28. Wenn Scrum schon halbwegs läuft, an Organisationsmeetings teilnehmen, die das Scrum-Team betreffen (könnten), um Anregungen für mehr oder konsequenteres Scrum zu geben, die Teambedürfnisse zu kommunizieren und um direktere Kommunikation mit dem Team herzustellen
29. Teamübergreifenden Austausch anregen (auf Ebene der Product-Owner und Entwickler:innen)
30. **Mit Rat und Tat Fragen zu Scrum für das Scrum-Team und Außenstehende beantworten**
31. **Mit Management, Projektleitung, Teamleitung etc. über Rechte und Pflichten der Scrum-Teams sprechen und darüber, wie diese gestärkt werden können**
32. Scrum/agile Methoden der Personalabteilung erklären
33. **Zusammenspiel/Abstimmung zwischen Teams verbessern**
34. Das Management dabei unterstützen, das Team für schwierige Situationen Lösungen finden zu lassen, anstatt selbst Lösungen vorzugeben
35. Andere Scrum Master unterstützen und coachen
36. Änderungen der Teamzusammensetzung moderieren
37. Das Controlling mit der neuen Scrum-Welt in Verbindung bringen
38. Die unternehmensinterne Vernetzung der Scrum Master und anderen »Agilisten« über Sparten hinaus begleiten

Anforderungsebene und Product Owner unterstützen

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

Product-Owner-Aufgaben

Die Aufgaben des Product Owners variieren abhängig von Unternehmen und Produkt/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. **Produktziel definieren**
3. **Produktziel und soweit vorhanden Produktvision an Stakeholder und Entwickler:innen kommunizieren**
4. **Schreiben von User Stories** (allein, mit Stakeholdern, mit den Entwickler:innen)
5. **Akzeptanzkriterien für User Stories formulieren** (in der Regel zusammen mit den Entwickler:innen)
6. **Ordnen/Priorisieren des Product Backlog** (inkl. Entscheidung, was entwickelt wird und was nicht)

7. **Die bereits entwickelten Produktinkremente kennen**
8. Mit den bereits entwickelten Produktinkrementen »herumspielen«
9. Die Wertschöpfung des Produkts definieren
10. **Die Wertschöpfung des Produkts kennen, messen und optimieren**
11. Produktbezogene Feedbackschleifen installieren und verkürzen

Zusammenarbeit mit den Entwickler:innen

12. **Gemeinsames Refinement des Product Backlog**
13. **Zu große User Stories gemeinsam aufsplitten**
14. **Eine Sprint-Ziel-Skizze in das Sprint Planning mitbringen**
15. **Hoch priorisierte, gut ausgearbeitete Product-Backlog-Einträge in das Sprint Planning mitbringen**
16. **Mitarbeit im Sprint Planning**
17. **Beantwortung fachlicher Fragen der Entwickler:innen im Sprint Planning und während des Sprints**
18. Teilnahme an Daily Scrums
19. **Mitarbeit in Sprint-Retrospektiven**
20. Im Scrum-Team dabei helfen, den Entwicklungsprozess zu verbessern
21. Definition der *Definition of Ready* zusammen mit den Entwickler:innen und dem Scrum Master
22. **Definition der Definition of Done zusammen mit den Entwickler:innen und dem Scrum Master**
23. **Feedback zu implementierten Produkteigenschaften an die Entwickler:innen im Sprint oder im Sprint-Review**
24. **Den Entwickler:innen eigene Unzufriedenheiten deutlich machen und erklären;** Mitarbeit bei der Suche nach Lösungen
25. Den Entwickler:innen die relevanten Geschäftszahlen/KPIs transparent machen
26. Den Entwickler:innen verdeutlichen, wie das Produkt auf dem Markt bzw. bei den Kunden ankommt

Kunden/Anwenderinnen

27. **Kundenbedürfnisse verstehen** (mit Kunden/Anwenderinnen sprechen)
28. **Den Markt inkl. Wettbewerber verstehen**
29. **Ausgewählte Kunden/Anwenderinnen in die Sprint-Reviews integrieren**
30. **Aufsetzen und Durchführen geeigneter Erfolgsmetriken** (z. B. Kundenzufriedenheit über den *Net Promoter Score* oder *Fit-For-Purpose* messen)
31. **Risikomanagement über die Ordnung/Priorisierung des Product Backlog**
32. **Annahmen über Kunden/Anwenderinnen/Märkte testen** (z. B. mit einem *Minimum Viable Product*)

Management sonstiger Stakeholder

33. **Dafür sorgen, dass die richtigen Stakeholder zum Sprint-Review kommen**
34. Erstellung und Aktualisierung des Releaseplans
35. Aktualisierung des Value-Creation-Charts und Release-Burnup-Charts
36. Kommunikation von Releaseplan, Value-Creation-Chart und Release-Burnup-Chart an die Stakeholder

37. Stakeholder über neue Produkteigenschaften informieren
38. **Budgetkontrolle**

Aufgaben der Entwickler:innen

Die Aufgaben der Entwickler:innen variieren abhängig von Unternehmen und Projekt. Was zu ihren Aufgaben 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 von ihnen auf jeden Fall in Scrum wahrgenommen werden müssen.

Arbeitsorganisation

1. **Umsetzungsplan im Sprint Planning erstellen**
2. **Organisation der Teamarbeit im Daily Scrum**
3. Pair Programming mit Teammitgliedern
4. **Einarbeitung neuer Teammitglieder**

Technisch

5. **Produktinkremente erstellen, qualitätssichern und dokumentieren**
6. Automatisierte Tests (Unit Tests, Integrations-, Last-, Akzeptanztests) erstellen und kontinuierlich durchführen
7. **System- und Softwarearchitektur erstellen**
8. **Softwaretechnischer Entwurf**
9. **Auswahl geeigneter Technologien für die Umsetzung**
10. Betrieb und Support des entwickelten Produkts

Bezogen auf Stakeholder

11. Usability-Tests durchführen
12. Benutzerakzeptanztests durchführen
13. Umgebung für Continuous Integration aufsetzen und am Laufen halten
14. **Produktinkremente im Sprint-Review demonstrieren**
15. User Experience gestalten
16. **Bugs beseitigen**

Unterstützung des Product Owners

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

Verbesserung

20. **Sich selbst bzw. Technologien, das Vorgehen und die fachliche Domäne weiterentwickeln**
21. Zusammen mit dem Product Owner und dem Scrum Master die *Definition of Ready* formulieren
22. **Zusammen mit dem Product Owner und dem Scrum Master die *Definition of Done* formulieren**

Daily Scrum

- Ergebnis: Einsatzplanung für das Scrum-Team für den Tag
- Dauer: maximal 15 Minuten (jeden Werktag zur selben Uhrzeit am selben Ort)

- Teilnehmende: Entwickler:innen und Scrum Master; Product Owner optional; Stakeholder optional (Stakeholder dürfen zuhören, aber nicht sprechen)
- Vorgehen: Vonseiten der Entwickler:innen werden drei Fragen beantwortet:
  - Was habe ich gestern erledigt, das dem Scrum-Team geholfen hat, das Sprint-Ziel zu erreichen?
  - Habe ich Hindernisse gesehen, die uns daran hindern, das Sprint-Ziel zu erreichen?
  - Was werde ich heute erledigen, damit wir das Sprint-Ziel möglichst effektiv erreichen?
- Empfehlungen:
  - Bei Präsenzarbeit: Das Daily Scrum findet vor einem physikalischen Taskboard statt.
  - Bei Remote-Arbeit: Das Daily Scrum findet vor einem virtuellen Taskboard statt, das einem physikalischen Taskboard nachempfunden wurde.
  - Die ersten beiden der obigen Fragen werden einzeln von den Entwickler:innen beantwortet. Wenn diese sich dazu geäußert haben, wird die dritte Frage gemeinsam beantwortet. Wer was als Nächstes macht, sollte eine Teamentscheidung sein.
  - Hindernisse, die die Weiterarbeit an einer User Story 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.

Sprint Planning

- Ergebnisse: Sprint-Ziel, selektierte Einträge aus dem Product Backlog, Plan für die Umsetzung
- Dauer: maximal zwei Stunden pro Sprint-Woche (also vier Stunden für einen zweiwöchigen Sprint)
- Teilnehmende: Product Owner, Scrum Master, Entwickler:innen, bei Bedarf eingeladene Fachleute für spezifische anstehende Fragen
- Vorgehen:
  - Der Scrum Master fragt bei den Entwickler:innen die Anzahl der für den Sprint verfügbaren Personentage ab.
  - Der Product Owner stellt seine Idee für ein Sprint-Ziel vor sowie die hoch priorisierten User Stories.
  - Der Scrum Master fragt die Entwickler:innen, ob die erste User Story in den Sprint passt. Wird diese Frage positiv beantwortet, fragt der Scrum Master, ob die zweite User Story zusätzlich in den Sprint passt. Dieses Verfahren wird so lange wiederholt, bis die Entwickler:innen Zweifel haben, ob sie noch mehr schaffen können.
  - Jetzt wird das Sprint-Ziel überarbeitet und finalisiert. Der Product Owner schätzt ab, ob der Sprint einen positiven ROI (Return on Investment) hat, wenn die gewählten User Stories umgesetzt werden können. Wenn dies nicht der Fall ist, geht das Scrum-Team zurück zum ersten Schritt.
  - Dann wird der sogenannte Task-Breakdown durch die Entwickler:innen eingeleitet. Dazu werden Kleingruppen von jeweils zwei bis drei Personen gebildet. Jede Kleingruppe wählt einen Teil der User Stories aus und erstellt die Tasks für die Umsetzung.
  - Die erstellten Tasks werden anschließend im Plenum vorgestellt, und es wird Feedback eingesammelt. Gegebenenfalls wird eine zweite Runde Kleingruppenarbeit angeschlossen.
  - Es wird auf Basis der erstellten Tasks geprüft, ob die ausgewählten User Stories tatsächlich im Sprint umgesetzt werden können.

- Der Product Owner wird über das Ergebnis der Abschätzung informiert. Gegebenenfalls wird eine User Story aus dem Sprint Backlog entfernt oder eine weitere hinzugefügt. Wenn notwendig, wird das Sprint-Ziel angepasst.

- Empfehlungen:
  - Bei Präsenzarbeit: Der Beamer bleibt aus. Der Product Owner bringt die User Stories auf Papier mit. Die Tasks werden ebenfalls auf Papier erstellt.
  - Bei Remote-Arbeit: Es sollten Tools verwendet werden, die eine Partizipation der Teammitglieder auf Augenhöhe gut unterstützen und wo alle sofort sehen, was die anderen tun. Tools, die Post-its simulieren, sind dafür mitunter besser geeignet als Ticket-Systeme.
  - Der Product Owner bleibt während des Task-Breakdown anwesend. (Häufig treten bei dieser Tätigkeit weitere fachliche Rückfragen auf.)
  - Für die Tasks gilt die Regel, dass sie maximal einen Personentag an Aufwand erfordern dürfen. Tasks müssen also entsprechend klein gestaltet sein.
  - Mit so kleinen Tasks kann man für die finale Abschätzung, ob man die User Stories im Sprint schaffen kann, einfach die Tasks zählen und mit den verfügbaren Personentagen im Sprint vergleichen.

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)

- Teilnehmende: Product Owner, Scrum Master (Moderation), Entwickler:innen, Stakeholder (insbesondere Kunden und Anwenderinnen)

- Vorgehen:
  - Demonstration des Produktinkrements durch die Entwickler:innen. Die Demonstration erfolgt auf einer vorher vereinbarten Test- und Integrationsumgebung und nicht auf einem Entwicklerrechner. Es darf nur gezeigt werden, was gemäß der Definition of Done komplett erledigt ist. Der Scrum Master bestätigt, dass die Definition of Done eingehalten wurde.
  - Gegebenenfalls Akzeptanz des demonstrierten Produktinkrements durch den Product Owner (wenn nicht bereits im Sprint erfolgt)
  - Gegebenenfalls Aktualisierung des Value-Creation-Charts und des Release-Burnup-Charts (siehe unten)
  - Feststellung durch den Product Owner, ob bzw. inwieweit das Sprint-Ziel erreicht wurde
  - Sammeln von Feedback zum Produkt; Festhalten des Feedbacks durch den Product Owner
  - Feststellen, welches Feedback besonders dringlich ist; Anpassung des Product Backlog bezüglich dieses dringlichen Feedbacks durch den Product Owner
  - Gegebenenfalls Anpassung des Releaseplans

- Empfehlungen:
  - Der Product Owner sorgt dafür, dass die richtigen Stakeholder beim Sprint-Review anwesend sind.
  - Die Demonstration des Produktinkrements basiert auf dem Sprint-Ziel und erzählt eine Geschichte, die es den Stakeholdern erleichtert, das Gezeigte in einen geeigneten Kontext zu setzen.

- Das erfordert eine Vorbereitung des Sprint-Reviews durch die Entwickler:innen. Die Vorbereitung sollte nicht zu lange dauern. Das Sprint-Review ist *keine* Marketing-Veranstaltung.
- 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.

Sprint-Retrospektive

- Ergebnisse: Verbesserungsmaßnahmen, die das Scrum-Team im nächsten Sprint umsetzen will
- Dauer: ca. eine Stunde pro Sprint-Woche (also zwei Stunden für einen zweiwöchigen Sprint)
- Teilnehmende: Scrum Master (als Moderator), Product Owner, Entwickler:innen
- Vorgehen:
  - *Set the stage*: Der Scrum Master eröffnet die Retrospektive und stellt eine Arbeitsumgebung her, in der sich alle Teilnehmenden engagieren möchten.
  - *Gather data*: Die Teilnehmenden sammeln qualitative und quantitative Daten zum 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 und 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 Moderationstechniken variieren.
  - Es sollte geprüft werden, ob die Maßnahmen der letzten Retrospektive umgesetzt wurden und welche Effekte die Maßnahmen gezeigt haben.

Backlog Refinement

- Ergebnisse: Product Backlog in einem aktuellen, geordneten und aufgeräumten Zustand
- Dauer: ca. zwei Stunden pro Sprint-Woche (häufig jede Woche zwei Stunden am selben Wochentag zur selben Uhrzeit; z. B. donnerstags 10–12 Uhr)
- Teilnehmende: Scrum Master (als Moderator), Product Owner, Entwickler:innen, Fachleute (auf Einladung)
- Vorgehen:
  - Entfernen obsoleter Einträge aus dem Product Backlog
  - Hinzufügen neuer Einträge in das Product Backlog (Vorstellung der neuen Einträge durch den Product Owner)
  - Wert- und Aufwandsschätzung der neuen Einträge im Product Backlog
  - Wert- und Aufwandsschätzung der Einträge im Product Backlog, die einer Neuschätzung bedürfen
  - Überarbeitung der Priorisierung
  - Verfeinerung hoch priorisierter Product-Backlog-Einträge für die nächsten ein bis drei Sprints (Product-Backlog-Einträge auf eine angemessene Größe aufteilen; fachliche Details klären; Akzeptanzkriterien ergänzen)

- Empfehlungen:
  - Es handelt sich um einen *Workshop*, in dem Product Owner und Entwickler:innen gemeinsam die Verantwortung für die Vorbereitung der nächsten Sprints übernehmen.
  - Bei Präsenzarbeit: Der Beamer bleibt aus. Der Product Owner bringt die Product-Backlog-Einträge auf Papier mit.
  - Bei Remote-Arbeit: Es sollten Tools verwendet werden, die reichhaltige Annotationen inkl. Skizzen zu Epics und User Stories erlauben und nicht nur auf Text in einem Ticket beschränkt sind.
  - Beim Aufteilen von Product-Backlog-Einträgen arbeiten die Entwickler:innen mit. Auch neue Product-Backlog-Einträge können von ihnen erstellt werden.
  - Akzeptanzkriterien werden gemeinsam zwischen Product Owner und Entwickler:innen festgelegt.
  - Das Meeting ist optional und nicht in jedem Kontext notwendig bzw. sinnvoll. Experimentieren Sie ggf. mit verschiedenen Ansätzen.

Release Planning

- Ergebnisse: mit den Stakeholdern abgestimmter Releaseplan
- Dauer: ½ bis 1 Tag
- Teilnehmende: Scrum Master (als Moderator), Product Owner, Entwickler:innen (ggfs. nicht alle), Stakeholder
- Vorgehen:
  - Vor dem Release Planning wurde das initiale Product Backlog erstellt, geschätzt und priorisiert.
  - Der Product Owner stellt die Produktvision und das Produktziel vor.
  - Der Product Owner stellt das priorisierte Product Backlog vor.
  - Der Scrum Master oder die Entwickler:innen stellen die aktuell gemessene Entwicklungsgeschwindigkeit (Velocity) vor.
  - Der Product Owner legt Product-Backlog-Einträge grob auf die Zeitachse.
  - Die daraus resultierenden Konsequenzen werden gemeinsam diskutiert. Gegebenenfalls werden Änderungen an Releasedatum oder Product Backlog vorgenommen.
  - Es wird vereinbart, wie der Product Owner die Stakeholder über den Fortschritt im Release und Änderungen am Releaseplan informiert.
- Empfehlungen:
  - Bei Präsenzarbeit: Der Beamer bleibt aus.
  - Bei Remote-Arbeit: Tools mit guten Partizipationsmöglichkeiten, die elektronische Post-its unterstützen, funktionieren gut.
  - Das Meeting ist optional und nicht in jedem Kontext notwendig bzw. sinnvoll.

Product Backlog

- Zweck: Überblick über die noch ausstehenden Eigenschaften/Features des Produkts
- Eigenschaften:
  - Einträge beschreiben das Was und nicht das Wie.
  - Einträge sind geordnet (in der Regel nach Priorität)
  - Hoch priorisierte Einträge sind klein und detailliert ausgearbeitet.
  - Niedrig priorisierte Einträge sind groß und nur grob skizziert.
  - Einträge sind geschätzt bzgl. Wert und Aufwand
  - Transparent für die Entwickler:innen und die Stakeholder

- Enthält nur die noch offenen Eigenschaften/Features (und nicht die bereits abgeschlossenen)

- Verwendung:
  - Ordnung/Priorisierung durch den Product Owner
  - Aufwandsschätzung durch die Entwickler:innen
  - Basis für Releaseplanung und -Controlling
- Empfehlungen:
  - Bei Präsenzarbeit: Das Product Backlog existiert physikalisch (Karteikarten/Haftnotizen an der Wand).
  - Bei Remote-Arbeit: Das Product Backlog ist grafisch passend aufbereitet (Orientierung an physikalischer Aufbereitung ist häufig sinnvoll) und nicht eine endlose lineare Liste von Einträgen.
  - Beschränkung auf maximal 70–80 Einträge pro Release (noch besser: ein bis zwei Dutzend)
  - Unterscheidung in Einträge, die für das aktuelle Release geplant sind, und solche für später (Ideen)
  - User Stories und ggfs. Epics als Product-Backlog-Einträge (wenn im Unternehmen nicht bereits ein anderes gut funktionierendes Format etabliert ist)
  - Gruppierung der Einträge nach Themen
  - Bugs, die nicht sofort beseitigt werden, werden ins Product Backlog aufgenommen und durch den Product Owner priorisiert.

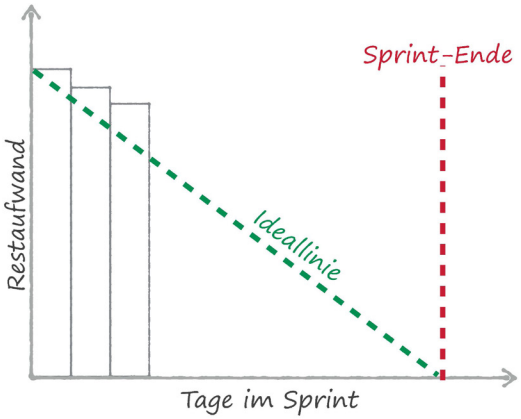
Sprint Backlog

- Zweck: Überblick über die noch ausstehenden Arbeiten im Sprint
- Eigenschaften:
  - Enthält das Sprint-Ziel, die für den Sprint ausgewählten Product-Backlog-Einträge sowie den Plan für die Umsetzung
  - Geordnet (in der Regel nach Priorität)
  - Zeigt den Zustand der Planumsetzung
- Verwendung:
  - Wird im Sprint Planning von den Entwickler:innen erstellt
  - Aktualisierung durch die Entwickler:innen im Daily Scrum
- Empfehlungen:
  - Bei Präsenzarbeit: Das Sprint Backlog existiert physikalisch in Form eines Taskboards (Karteikarten/Haftnotizen an der Wand) im Teamraum.
  - Bei Remote-Arbeit: Das Sprint Backlog ist ähnlich einem physikalischem Taskboard organisiert und visualisiert. Die meisten Ticket-Systeme bringen eine entsprechende Visualisierung mit. Viele Teams stellen aber auch fest, dass eine Darstellung anhand elektronischer Post-its auf einem digitalen Whiteboard besser funktioniert.
  - Die Reihenfolge auf dem Taskboard bildet die Priorisierung der Product-Backlog-Einträge ab.
  - Spalten: User Story, ToDo, Doing, Done
  - Darstellung von Sprint-Ziel 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

Produktinkrement

- Zweck: Wertschöpfung für das Unternehmen und den/die 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 Entwurfsqualität zu vermeiden
  - Notwendige Dokumentation über die Definition of Done vereinbaren

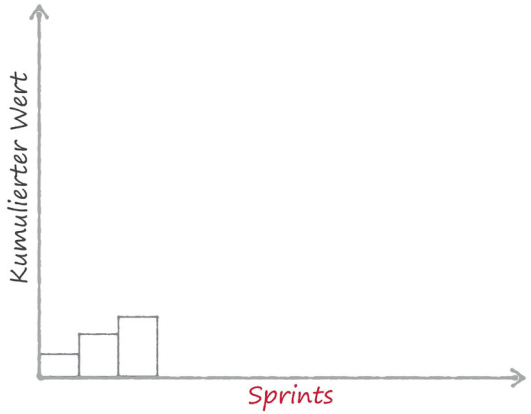
Sprint-Burndown-Chart



- Zweck: Frühe Einschätzung der Erfolgswahrscheinlichkeit des Sprints für die Entwickler:innen
- Eigenschaften:
  - Prognostiziert den weiteren Fortschritt im Sprint auf Basis der bereits im Sprint erledigten Arbeit
  - Visualisiert dazu bereits die im Sprint erledigte Arbeit und den angenommenen Fortschritt für den Rest des Sprints
  - Die optionale Ideallinie zeigt den idealen Arbeitsfortschritt, damit Abweichungen früh erkannt und diskutiert werden können.
- Verwendung:
  - Aktualisierung direkt vor oder im Daily Scrum durch die Entwickler:innen
  - Betrachtung im Daily Scrum, um das weitere Vorgehen im Sprint zu planen
- Empfehlungen:
  - Bei Präsenzarbeit: handgezeichnet auf DIN A3 oder Flipchart
  - Bei Remote-Arbeit werden meist die vom eingesetzten Tool generierten Graphen verwendet. Hier ist darauf zu achten, dass diese auch explizit in jedem Daily Scrum diskutiert werden.

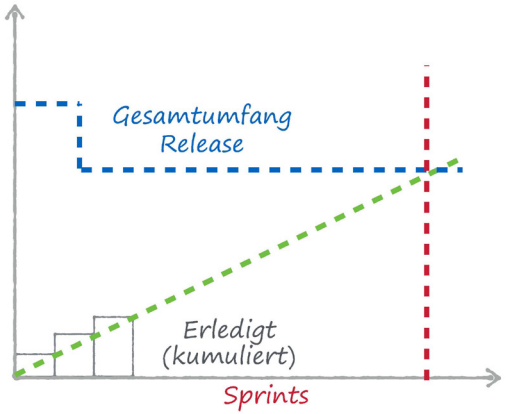
- Hängt direkt neben dem Taskboard
- Restaufwand basiert auf den Tasks (bzw. dem Umsetzungsplan für die Product-Backlog-Einträge).
- Restaufwand ermitteln durch Zählen von Tasks (ohne den Overhead, Reststunden zu schätzen)
- Fortgeschrittene Scrum-Teams, die sehr wenige Product-Backlog-Einträge parallel in Arbeit haben, können das Sprint-Burndown-Chart auf Basis erledigter Product-Backlog-Einträge (statt Tasks) führen.
- Das Sprint-Burndown-Chart ist ein optionales Artefakt. Es ist in langen Sprints sehr nützlich und weniger hilfreich in sehr kurzen Sprints.

Value-Creation-Chart



- Zweck: Transparenz über Wertschöpfung; gemeinsames Ziel im gesamten Scrum-Team etablieren; zielführende Diskussionen über Anpassungen oder Abbruch der Entwicklung führen
- Eigenschaften:
  - Visualisiert den kumuliert geschaffenen Wert
- Verwendung:
  - Aktualisierung im Sprint-Review
  - Betrachtung im Sprint-Review, um das weitere Vorgehen im Release zu planen
  - Der Wert wird meist auf Basis abstrakter Value Points ermittelt.
- Empfehlungen:
  - Bei Präsenzarbeit: handgezeichnet auf DIN A3 oder Flipchart
  - Bei Remote-Arbeit: Die meisten Tools unterstützen diese Art der Visualisierung nicht direkt. Dann kann man die Darstellung mit wenig Zusatzaufwand in einer Tabellenkalkulation erstellen.
  - Hängt direkt neben dem Product Backlog
  - Das Value-Creation-Chart ist ein optionales Artefakt. Es ist in den meisten Kontexten nützlich.

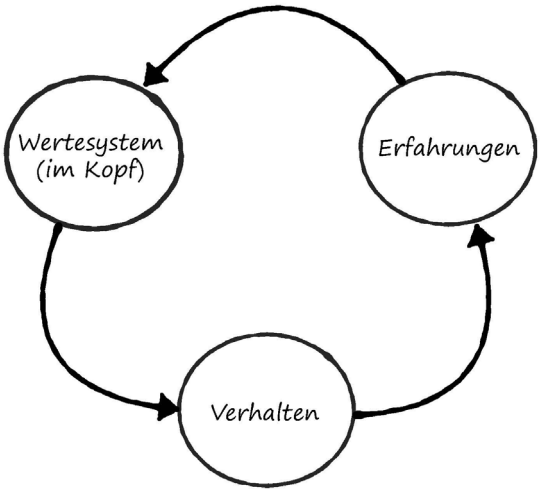
Release-Burnup-Chart



- Zweck: Frühe Einschätzung des Releaseumfangs bzw. des Releasetermins für den Product Owner und die Stakeholder

- Eigenschaften:
  - Prognostiziert den weiteren Fortschritt im Release auf Basis der bereits erledigten Product-Backlog-Einträge
  - Visualisiert dazu bereits erledigte Produkteigenschaften und den angenommenen Fortschritt für den Rest des Release
  - Die optionale Ideallinie (grün) zeigt den idealen Arbeitsfortschritt, damit Abweichungen früh erkannt und diskutiert werden können.
- Verwendung:
  - Aktualisierung im Sprint-Review
  - Betrachtung im Sprint-Review, um das weitere Vorgehen im Release zu planen
  - Der Restaufwand basiert auf den Schätzungen des Product Backlog (z.B. Story Points).
- Empfehlungen:
  - Bei Präsenzarbeit: handgezeichnet auf DIN A3 oder Flipchart
  - Bei Remote-Arbeit: Die meisten Ticket-Systeme können den Graphen generieren. Wichtig ist, den Graphen regelmäßig (z.B. in oder nach jedem Sprint-Review) mit dem Scrum-Team und Stakeholdern zu diskutieren.
  - Hängt direkt neben dem Product Backlog
  - Das Release-Burnup-Chart ist ein optionales Artefakt. Es ist in langen Releases sehr nützlich und weniger hilfreich in sehr kurzen Releases (und komplett überflüssig bei kontinuierlichen Releases).

Scrum einführen



Agile Entwicklung erfordert veränderte Verhaltens- und Denkweisen bei allen Beteiligten. Nur die Scrum-Mechanik zu installieren, reicht nicht aus.

Die notwendigen Verhaltensänderungen lassen sich nicht über Anweisungen herbeiführen, wie nebenstehende Abbildung zeigt. Wir alle haben Wertesysteme im Kopf. Ein Glaubenssatz in diesem Wertesystem könnte z.B. sein: »Vertrauen ist gut, Kontrolle ist besser.« Dieses Wertesystem prägt das konkrete Verhalten, das wir an den Tag legen, z.B.: »Herr Müller, ich vertraue Ihnen diese Aufgabe an und möchte, dass Sie mir morgen früh Bericht über den Fortschritt erstatten.« Dieses Verhalten erzeugt im gegebenen Kontext bestimmte Reaktionen und Ergebnisse und prägt damit die Erfahrungen, die wir machen. So erfahren wir vielleicht am nächsten Tag, dass Herr Müller mit der ihm anvertrauten Aufgabe noch nicht einmal angefangen hat. Diese Erfahrungen wirken zurück auf

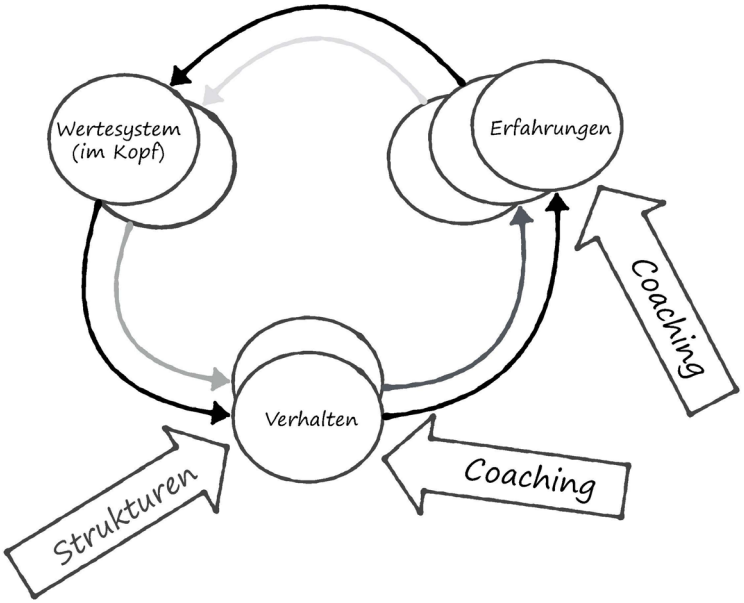
unser Wertesystem (»Gut, dass ich kontrolliert habe«). In der Regel haben sich Zyklen entwickelt, in denen sich Werte und Erfahrungen gegenseitig verstärken (»Nächstes Mal kontrolliere ich am besten halbtäglich«).

Die neuen Verhaltensweisen müssen schrittweise erlernt werden. Ein verändertes Verhalten erzeugt andere Erfahrungen und schließlich ändert sich unser Wertesystem im Kopf.

Wer schon mal versucht hat, abzunehmen oder mit dem Rauchen aufzuhören, weiß, wie schwer es ist, angelernte Verhaltensweisen abzulegen. So geraten wir immer wieder in Situationen, in denen wir uns wider besseres Wissen unpassend verhalten. Und selbst wenn wir die gewünschte Verhaltensweise in einer Schulung eingeübt haben, fallen wir in Stress-Situationen häufig wieder zurück in alte Verhaltensmuster.

Strukturen (wie das Scrum-Framework) zusammen mit Coaching (durch den Scrum Master oder einen externen Scrum-Coach) helfen dabei, Verhalten nachhaltig zu ändern. Coaching unterstützt außerdem dabei, die neu gemachten Erfahrungen passend einzuordnen und die positiven Effekte des neuen Verhaltens anzuerkennen. Dazu muss der Coach verstehen, was agile Entwicklung wirklich bedeutet, und er muss diese bereits praktiziert haben – ansonsten hat er den Prozess der Verhaltensänderung selbst noch nicht durchlaufen.

Für eine erfolgreiche Scrum-Einführung muss also das Wissen vermittelt *und* Verhaltensweisen geändert werden. Eine Kombination aus Schulungen und Coaching ist unabdingbar.



dpunkt.verlag

it-agile

Interessiert an weiteren Informationen?

Das Buch zur Broschüre

dpunkt.de/produkt/scrum-verstehen-und-erfolgreich-einsetzen-3/

it-agile

Entwicklung, Beratung, Schulung

Was wollen Sie erreichen und wofür?

direkt zum Angebot

Nehmen Sie jetzt Kontakt mit uns auf, um den nächsten Schritt zu wirkungsvoller Agilität zu gehen:

0800 / 482 4453 wirkungsvoll@it-agile.de