edv-buchversand.de

Scrum

verstehen und erfolgreich einsetzen

» Hier geht's direkt zum Buch

DIE LESEPROBE

Diese Leseprobe wird Ihnen von www.edv-buchversand.de zur Verfügung gestellt.

2 Überblick über die Verantwortungen, Meetings und Artefakte in Scrum

»Scrum will help you to fail in thirty days or less.«

Ken Schwaber[1]

Dieses Kapitel gibt einen Überblick über Scrum. Es führt die Begriffe des Scrum-Frameworks ein, erklärt den Ablauf (Flow), die Verantwortungen, die verwendeten Artefakte und die Meetings (Events), wie sie im Scrum Guide (siehe [Schwaber-Sutherland2020]) definiert sind.

Auch wenn wir hier mit der »Mechanik« von Scrum beginnen, sollte Scrum nicht rein mechanisch verstanden werden. Es handelt sich bei Scrum um ein Management-Rahmenwerk, nicht um einen Prozess, dem man blind zu folgen hat. Vielmehr bietet Scrum einen Rahmen an, um bestimmte Prinzipien in der Projekt- oder Produktentwicklungsarbeit zu leben. Auf die Werte und Prinzipien von Scrum gehen wir gesondert in Kapitel 5 ein. Bis dahin haben wir grundlegende Kenntnisse über die Funktionsweise von Scrum geschaffen und können auf dieser Basis die Scrum-Werte und -Prinzipien konkret diskutieren.

2.1 Scrum-Flow

Beginnen wir mit dem generellen Ablauf von Scrum (Scrum-Flow). Im Gegensatz zur Steuererklärung passt dieser tatsächlich auf einen Bierdeckel. Den Beweis liefert Abbildung 2.1. Die zugehörigen Verantwortungen, Meetings[2] und Artefakte führen wir knapp in diesem Kapitel ein und beschäftigen uns im Rest des Buchs noch ausführlicher mit ihnen.

1. Siehe [Mayer2013].
2. Der Scrum Guide spricht hier von Events und versteht darunter neben den Meetings auch den Sprint. Wir weichen davon in diesem Kapitel zugunsten besserer Verständlichkeit ab.

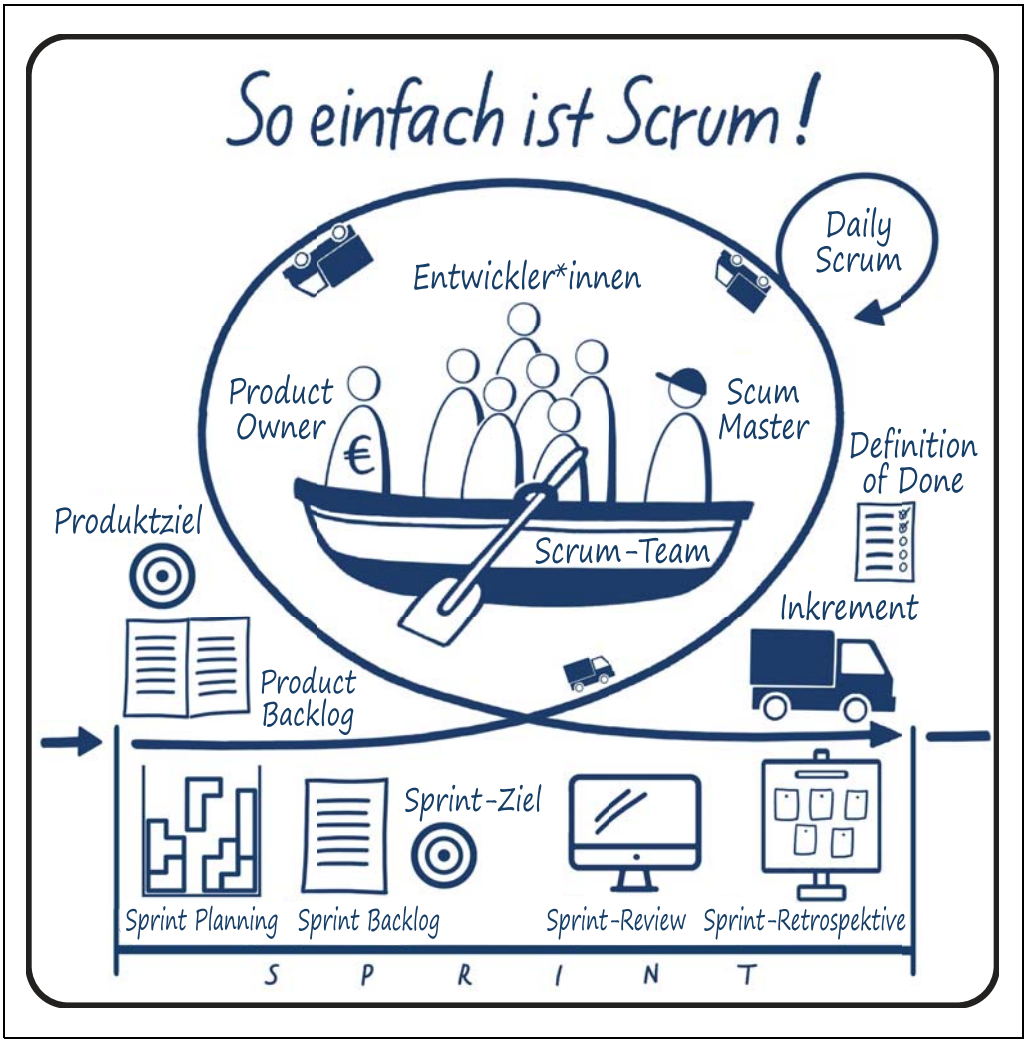

Abb. 2.1 *Der Scrum-Ablauf passt auf einen Bierdeckel.*

Am Anfang steht ein Produktziel, das beschreibt, wozu überhaupt ein Produkt entstehen (oder etwas an einem vorhandenen Produkt geändert werden) soll. Aus dem Produktziel werden konkrete Eigenschaften des Produkts abgeleitet, die der *Product Owner* im *Product Backlog* organisiert und priorisiert. Die Einträge im Product Backlog werden *Product Backlog Items* genannt. Der Product Owner ist für den wirtschaftlichen Erfolg des Produkts verantwortlich. Dieser Verantwortung folgend, wird er die Items im Product Backlog nach Geschäftswert priorisieren und sie in eine klare Reihenfolge bringen.

Die Entwicklung erfolgt bei Scrum in Iterationen fester und immer gleicher Länge, die *Sprints* heißen. Sprints sind maximal 30 Tage lang. Die *Entwickler* sind für die Umsetzung der Product Backlog Items verantwortlich. Der Product Owner ist damit für das »Was« und die Entwickler für das »Wie« der Entwicklung verantwortlich.

Zu Beginn jedes Sprints führen Product Owner und Entwickler ein *Sprint Planning* durch, das der Scrum Master moderiert. Im Sprint Planning wird festgelegt, welche Product Backlog Items in das Sprint Backlog übernommen werden und damit für diesen Sprint eingeplant werden. Dies erfolgt derart, dass der Product

Owner die Reihenfolge der Items vorgibt und die Entwickler entscheiden, wie viele Items in den Sprint passen. Am Ende des Sprint Planning legt das Scrum-Team gemeinsam das Sprint-Ziel fest als eine inhaltliche Klammer für die Arbeit im Sprint.

Direkt nach dem Sprint Planning beginnt die Entwicklungsarbeit im Sprint, bei der die Entwickler sich selbst organisieren, also z. B. selbst entscheiden, wer als Nächstes welche Aufgabe angeht. Der Scrum Master unterstützt das Team bei der Selbstorganisation und sorgt dafür, dass Scrum effektiv angewendet wird. Dazu gehört unter anderem, dass der Scrum Master eine störungsfreie Umgebung schafft, in der die Entwickler fokussiert arbeiten können.

Zur Abstimmung unter den Entwicklern findet jeden Werktag während des Sprints das *Daily Scrum* statt. Im Daily Scrum treffen sich die Entwickler im Stehen für maximal 15 Minuten, um den Fortschritt im Sprint zu begutachten und die Arbeit im Team bis zum nächsten Daily Scrum zu organisieren.

Am Ende des Sprints liefern die Entwickler ein *Produktinkrement* ab. Das Produktinkrement soll auslieferbar sein (Shippable Product Increment). Es muss nicht zwingend ausgeliefert werden, muss aber die Qualität haben, dass es ausgeliefert werden könnte. Dieser Qualitätsanspruch wird in der Definition of Done beschrieben. Im *Sprint-Review* präsentieren die Entwickler die neuen Produkteigenschaften, um Transparenz über den Entwicklungsfortschritt zu schaffen und Feedback zum Produkt zu erhalten. Dazu sind neben dem Product Owner die relevanten Stakeholder (Kunden, Anwender, Management etc.) anwesend. Das Feedback der Stakeholder führt zu Änderungen am Product Backlog: Neue Einträge entstehen, existierende Einträge werden neu priorisiert oder aus dem Product Backlog entfernt.

Nach dem Sprint-Review findet die *Sprint-Retrospektive* statt, in der Scrum Master, Entwickler und Product Owner daran arbeiten, die Zusammenarbeit und den Prozess kontinuierlich zu verbessern.

Direkt nach einem Sprint beginnt der nächste Sprint. Es gibt bei Scrum keine Zeiten zwischen zwei Sprints. Auf die Sprint-Retrospektive eines Sprints folgt sofort das Sprint Planning des nächsten Sprints.

2.2 Die Verantwortlichkeiten

Im Kern steht bei Scrum das Scrum-Team, das gemeinsam die Ergebnisse verantwortet. Innerhalb des Scrum-Teams werden drei Verantwortlichkeiten unterschieden: *Product Owner*, *Entwickler* und *Scrum Master*. Diese werden in den folgenden Kapiteln noch ausführlich beschrieben. Hier geben wir einen ersten kleinen Einblick.

In früheren Versionen des Scrum Guide wurden Rollen beschrieben und nicht Verantwortlichkeiten gebündelt und benannt. Das ist ein für die Praxis vermutlich nicht besonders relevanter Unterschied. Aus Sicht von Scrum als Rahmenwerk jedoch wird man dadurch noch generischer, weil eben z. B. nicht zwingend eine Rolle Product Owner besetzt werden muss. Ein Abteilungsleiter oder eine Produktmanagerin kann die Verantwortlichkeiten des Product Owner bei Scrum wahrnehmen.

Andererseits ändert es nichts an unserer Praxiserfahrung, dass Scrum-Teams erfolgreicher sind, wenn keine weiteren anderen Rollen oder Verantwortlichkeiten neben denen im Scrum-Team übernommen werden.

2.2.1 Product Owner

Der *Product Owner* ist für den (meist wirtschaftlichen) Erfolg des Produkts verantwortlich. Er kommt dieser Verantwortung in erster Linie dadurch nach, dass er den Produktnutzen durch die Priorisierung von Produkteigenschaften optimiert. Zu entscheiden, welche Produkteigenschaften überhaupt umgesetzt werden, gehört zur Priorisierung. Der Product Owner entscheidet also auch darüber, welche Produkteigenschaften nicht umgesetzt werden.

Zu seinen Aufgaben gehören laut Scrum Guide:

- Produktwert und damit den Wert der Arbeit der Entwickler maximieren
- Produktziel entwickeln und explizit kommunizieren
- Product Backlog Items erstellen und klar kommunizieren
- Product Backlog Items priorisieren/sortieren
- Product Backlog sichtbar und transparent machen
- Sicherstellen, dass alle Beteiligten die Product Backlog Items verstehen

Die Aufgaben kann der Product Owner selbst durchführen oder die Umsetzungsverantwortung an andere delegieren, er bleibt aber in jedem Fall ergebnisverantwortlich.

Der Product Owner soll eine Person sein und kein Komitee. Damit diese Person ihre Arbeit gut ausüben kann, müssen ihre Produktentscheidungen von der Organisation respektiert werden. Nur so kann sie unabhängig im Sinne des Produkts und seiner Wirtschaftlichkeit die besten Entscheidungen treffen.

2.2.2 Entwickler

Entwickler nennen wir die Scrum-Team-Mitglieder, die jeden Sprint ein nutzbares Produktinkrement erschaffen. Inklusive Scrum Master und Product Owner gehen wir meist von maximal zehn Personen in einem Scrum-Team aus. Die Gruppe der Entwickler ist cross-funktional, hat also in Summe alle Fähigkeiten (Skills), die nötig sind, um das Sprint Backlog in ein lieferbares Produktinkrement umzuwandeln. Insofern sind meist neben Programmierern auch andere Experten wie Tester, Designer oder Mitarbeiter aus dem Betrieb als Entwickler ins Scrum-Team eingebunden.

Die Hauptverantwortung der Entwickler besteht darin, ein technisch erfolgreiches und wartbares Produkt zu erstellen. In dieser Verantwortung dürfen und müssen die Entwickler dem Wunsch des Product Owners nach mehr Backlog Items im Sprint widersprechen, wenn dadurch die Qualität leiden würde.

Die Entwickler organisieren sich und ihre Arbeit selbst. Dabei sind alle gleich, es soll keine Titel und auch keine Subteams (die Entwickler, die Tester, die Frontend-

Entwickler etc.) geben. Von der Selbstorganisation verspricht man sich bessere Entscheidungen, die die Qualität des Produkts auf Dauer gewährleisten. Natürlich wird es auch in Scrum-Teams Spezialisten mit besonderen Skills geben. Trotzdem wird die Verantwortung gemeinsam getragen.

Der Scrum Guide sieht die Entwickler ergebnisverantwortlich dafür,

- mit dem Sprint Backlog einen Plan für den Sprint zu erstellen,
- Qualität im Sinne der Definition of Done zu erbringen,
- täglich darauf zu schauen, ob das Sprint-Ziel noch erreicht werden kann, und seinen Plan dahingehend anzupassen sowie
- sich wechselseitig als Experten zur Verantwortung zu ziehen.

Bei Überlastung des Product Owners übernehmen die Entwickler auch mal Aufgaben wie das Erstellen oder Verfeinern von Product Backlog Items.

2.2.3 Scrum Master

Der *Scrum Master* sorgt für die erfolgreiche Anwendung von Scrum. Das bedeutet deutlich mehr, als nur den Zeigefinger zu erheben, wenn sich irgendwer nicht »Scrum-konform« verhält. Stattdessen hilft er dabei, Scrum-Theorie und -Praxis zu verstehen, sowohl innerhalb des Scrum-Teams als auch innerhalb der Organisation. Der Scrum Master ist ergebnisverantwortlich für die Effektivität des Scrum-Teams. Scrum Master sind Leader, die dem Scrum-Team und der Gesamtorganisation dienen.

Der Scrum Master dient dem Scrum-Team auf unterschiedliche Weise, unter anderem:

- Er coacht die Teammitglieder in Selbstmanagement und interdisziplinärer Zusammenarbeit.
- Er hilft dem Scrum-Team, fokussiert an der Erstellung hochwertiger Produktinkremente zu arbeiten.
- Er unterstützt dabei, dass alle Hindernisse (Impediments) aus dem Weg geräumt werden, damit das Scrum-Team Fortschritte erzielen kann.
- Er stellt sicher, dass alle Scrum-Meetings stattfinden und dabei positiv und produktiv sind und innerhalb der vorgegebenen Timebox bleiben.

Der Scrum Master dient dem Product Owner unter anderem wie folgt:

- Er unterstützt bei der Suche nach effektiven Techniken zur Definition des Produktziels und beim Management des Product Backlog.
- Er hilft dem ganzen Scrum-Team dabei, die Notwendigkeit klarer und präziser Product Backlog Items zu verstehen.
- Er sorgt für die Etablierung einer empirischen Produktplanung für ein komplexes Umfeld.
- Er hilft nach Wunsch oder Bedarf bei der Zusammenarbeit mit Stakeholdern, z. B. durch Moderation.

Der Scrum Master dient der gesamten Organisation unter anderem auf diesen Wegen:

- Er führt, coacht und schult bei der Einführung von Scrum in der Organisation,
- Er empfiehlt und plant die Einführungen von Scrum in der Organisation.
- Er unterstützt Mitarbeiter und Stakeholder beim Verständnis und der Umsetzung eines empirischen Ansatzes für komplexe Arbeit.
- Er beseitigt Barrieren zwischen Stakeholdern und Scrum-Teams.

2.2.4 Scrum-Team

Das gesamte *Scrum-Team* ist ergebnisverantwortlich, in jedem Sprint ein wertvolles, nützliches Produktinkrement zu erstellen. Dies macht insbesondere deutlich, dass die Scrum-Verantwortungen nur gemeinsam ein erfolgreiches Produkt erstellen können. Insbesondere soll auch dem Eindruck entgegengewirkt werden, der Product Owner würde bei den Entwicklern etwas beauftragen und diese würden am Ende des Sprints »liefern wie bestellt«. Das Scrum-Team entwickelt und liefert gemeinsam ein wertvolles Produkt.

Scrum-Teams sollen möglichst unabhängig von anderen Teams sein. Wir werden in Kapitel 7 skizzieren, wie diesem Grundsatz bei größeren Entwicklungsvorhaben mit mehreren Teams Rechnung getragen werden kann.

2.2.5 Kein Projektleiter in Scrum

Es gibt in Scrum-Teams keinen Projektleiter, der das Gesamtvorhaben in Arbeitspakete zerlegt und deren Umsetzung überwacht. Eine solche Rolle würde die Selbstorganisation der Entwickler erheblich behindern.

Die Aufgaben klassischer Projektleitung verteilen sich auf die Verantwortlichkeiten Product Owner, Scrum Master, Entwickler und Scrum-Team. Man braucht in Scrum also immer noch Projektleitungstätigkeiten, aber keine Projektleiterrolle oder -position.

2.3 Meetings (Events)

Betrachten wir im Folgenden die *Meetings* von Scrum. Es fällt dabei auf, dass der Sprint mit einem Meeting beginnt (Sprint Planning) und mit zwei Meetings endet (Sprint-Review und Sprint-Retrospektive). Dazwischen ist lediglich das Daily Scrum vorgesehen. Bei Bedarf können sich Entwickler oder Scrum-Teams natürlich auch zusätzlich im Sprint zusammensetzen und besprechen. Für das Funktionieren der Entwicklung nach Scrum sind aber keine weiteren Meetings notwendig.

2.3.1 Sprint Planning

Das Ziel des *Sprint Planning* ist es, für den Sprint drei Fragen zu beantworten:

1. Warum ist dieser Sprint wertvoll? (Sprint-Ziel)
2. Was werden wir in diesem Sprint schaffen? (Product Backlog Items)
3. Wie werden wir es tun? (Plan für die Umsetzung)

Die erste Frage wird oft vorläufig durch einen Vorschlag des Product Owners für ein Sprint-Ziel zu Beginn des Planning beantwortet, dann aber im Verlauf des Sprint Planning angepasst, wenn das Scrum-Team mehr über den möglichen Umfang des Sprint Backlog weiß. Die zweite Frage ergibt sich im Wesentlichen durch die feste Reihenfolge der priorisierten Backlog Items (vom Product Owner vorgegeben). Allerdings entscheiden die Entwickler, wie viele Items in den Sprint gezogen werden. So verhindern die Entwickler Überlastung und können zuverlässig in hoher Qualität lieferbare Produktinkremente entwickeln. Methodisch schätzen dazu die meisten Teams spätestens im Sprint Backlog die anstehenden Product Backlog Items. So bekommen sie ein besseres Gefühl dafür, welche Menge an Product Backlog Items für den Sprint realistisch ist. Man spricht davon, dass die Entwickler eine *Vorhersage* (Forecast) abgeben, wie viel sie schaffen können. Diese Vorhersage mag nicht immer stimmen, aber Scrum-Teams sollten anstreben, dass ihre Vorhersagen meistens zutreffen. Angestrebt ist hier eine Verlässlichkeit ähnlich der Wettervorhersage: In der Regel sollte sie stimmen. Alle Beteiligten wissen aber, dass die Wettervorhersage mit Unwägbarkeiten behaftet ist, und niemand wird den Meteorologen dafür bestrafen, wenn die Vorhersage mal nicht stimmte.

Für die Schätzung, aber auch für die spätere Erledigung der Sprint Backlog Items und damit für die Beantwortung der dritten Frage, wie die Product Backlog Items technisch umgesetzt werden, erstellen die Entwickler im Sprint Planning einen Plan. Viele Teams verwenden dazu einen *Task-Breakdown* für jedes in den Sprint gezogene Product Backlog Item. Wenn dieser Task-Breakdown von allen Entwicklern gemeinsam vorgenommen wird, erhöht sich seine Qualität, und die Wahrscheinlichkeit steigt, dass die Entwickler nichts Wesentliches übersehen haben.

Spätestens jetzt legt das Scrum-Team noch gemeinsam für den Sprint ein Sprint-Ziel fest. Damit soll eine zusammenfassende Beschreibung der wesentlichen wertschöpfenden Erweiterungen am Produkt gegeben werden. Mit einem guten Sprint-Ziel können Product Owner und Entwickler in einen sinnvollen Dialog einsteigen, falls während des Sprints offensichtlich wird, dass das Ziel mit den bisher geplanten Sprint Backlog Items nicht mehr erreichbar ist. Dann können Product Owner und Entwickler diskutieren, ob und wie man stattdessen das Sprint-Ziel erreichen kann.

Im Sprint Planning kooperiert das gesamte Scrum-Team unter Moderation durch den Scrum Master. Das Sprint Planning dauert für einen Sprint von einem Monat maximal acht Stunden und sollte bei kürzeren Sprints entsprechend kürzer sein.

Mehr Details zum Sprint Planning finden Sie in Kapitel 3 und Kapitel 4.

2.3.2 Daily Scrum

Das *Daily Scrum* findet, wie der Name sagt, werktäglich statt – am besten immer zur gleichen Zeit am gleichen Ort. Im Daily Scrum koordinieren die Entwickler ihre Arbeit bezogen auf das Sprint-Ziel.

Das Scrum-Team trifft sich für das Daily Scrum im Stehen, was den komprimierten und knappen Charakter dieses täglichen Treffens unterstreicht. Es dauert maximal 15 Minuten. Vielen Teams hilft reihum die Beantwortung der folgenden drei Fragen:

1. Was habe ich gestern gemacht, das uns hilft, das Sprint-Ziel zu erreichen?
2. Was werde ich heute tun, das uns hilft, das Sprint-Ziel zu erreichen?
3. Sehe ich irgendwelche Hindernisse, die uns davon abhalten könnten, das Sprint-Ziel zu erreichen?

Da es um die Koordination der Entwickler geht, muss der Product Owner nicht zwingend am Daily Scrum teilnehmen. Wir empfehlen dieses jedoch, weil er so einmal täglich im Anschluss an das Meeting den Teammitgliedern niedrigschwellig für Rückfragen zur Verfügung steht.

Es kann aber auch Konstellationen geben, in denen der Product Owner im Daily Scrum nicht hilfreich ist, z. B. wenn dadurch das Daily Scrum zu einem Reporting in seine oder ihre Richtung verkommt. Darauf sollten Scrum Master und der Rest des Scrum-Teams gemeinsam achten, und sie sollten festlegen, wie sie das Daily Scrum handhaben wollen.

Mehr zum Daily Scrum findet sich in Kapitel 4.

2.3.3 Sprint-Review

Das *Sprint-Review ermöglicht* das Lernen über das Produkt: Entwickeln wir das richtige Produkt? Hat es die richtigen Features? Ist es so benutzbar, wie es für die Anwender passt? Erfüllt das Produkt seinen Zweck? Welche Funktionen fehlen noch, damit es benutzt werden kann?

Nebenbei wollen wir auch eine inhaltliche Rückschau auf den Sprint vornehmen und schauen, welche Sprint Backlog Items erledigt wurden.

Damit diese Ziele mit dem Sprint-Review erreicht werden können, müssen Stakeholder (insbesondere Kunden und Anwender) am Sprint-Review teilnehmen. Sie geben das wertvollste Feedback zum Produkt.

Das Sprint-Review beginnt mit einer Demonstration des Produktinkrements durch die Entwickler. Dann folgt die Akzeptanz bzw. Ablehnung der entwickelten Features durch den Product Owner (wenn das nicht bereits vorher geschehen ist). Danach folgt das Einsammeln von Feedback, und der Product Owner stellt fest, ob das Sprint-Ziel erreicht wurde.

Der Scrum Master moderiert das Sprint-Review. Im Sprint-Review zeigt sich schnell, wie viel Einfluss eine gute Moderation auf die Ergebnisse hat. So neigen eini-

ge Entwickler dazu, sich im Sprint-Review zu rechtfertigen, anstatt das Feedback erst einmal offen anzunehmen. Schließlich wollen wir ja durch das Feedback etwas über die Stakeholder-Bedürfnisse lernen, sodass ein besseres Produkt entstehen kann.

Für das Sprint-Review empfiehlt der Scrum Guide eine Länge von maximal vier Stunden bei einem Monatssprint. Bei kürzeren Sprints sollte es entsprechend kürzer ausfallen.

Genaueres zur Bedeutung des Sprint-Reviews und zu seinem Ablauf findet sich in Kapitel 3.

2.3.4 Sprint-Retrospektive

In der *Sprint-Retrospektive* arbeitet das gesamte Scrum-Team (inkl. Product Owner) daran, seine Zusammenarbeit und den Entwicklungsprozess zu verbessern, um die gemeinsame Effektivität zu erhöhen.

An der Sprint-Retrospektive nimmt das gesamte Scrum-Team teil, weil auch die Zusammenarbeit des Product Owners und der Entwickler thematisiert werden sollte. Der Scrum Master moderiert die Sprint-Retrospektive.

Der Scrum Guide legt nicht exakt fest, wie Retrospektiven durchzuführen sind; wir geben in Kapitel 5 genauere Hinweise.

Wichtig ist, dass der Scrum Master das Scrum-Team so durch die Sprint-Retrospektive führt, dass am Ende konkrete Verbesserungsmaßnahmen herauskommen, die das Scrum-Team möglichst bereits im nächsten Sprint umsetzen kann.

2.4 Der Sprint

In anderen iterativen Ansätzen heißt ein Entwicklungsabschnitt *Iteration*. In Scrum sprechen wir von *Sprints*[3]. In Scrum wird diese spezielle Namensgebung verwendet, weil es spezielle Anforderungen an Sprints gibt, die im allgemeinen Iterationskonzept nicht zwingend sind: eine maximale Länge von 30 Tagen und lieferbare Produktinkremente als Ergebnis.

Die Sprints eines Scrum-Teams haben immer die gleiche Länge. Diese liegt zwischen einer und maximal vier Wochen. Für den Sprint wird zu Beginn im Sprint Planning die Planung vorgenommen. Die vereinbarten Inhalte, die die Entwickler zum Ende des Sprints in ein lieferbares Produktinkrement überführt haben wollen, dürfen während des Sprints nicht geändert werden: Der Sprint ist vor Störungen von außen geschützt. So dürfen z. B. keine Teammitglieder für andere Aufgaben abgezogen werden.

Manchmal ist es allerdings nicht sinnvoll, den Sprint wie geplant zu Ende zu führen. Das ist z. B. dann der Fall, wenn das Sprint-Ziel nicht mehr erreichbar ist oder das Sprint-Ziel obsolet geworden ist. Dann kann der Sprint auch vorzeitig ab-

3. Der Name soll nicht suggerieren, dass man sich im Sprint total verausgabt und erst einmal eine Erholungspause braucht. Der Name soll auf die Kürze hinweisen.

gebrochen werden kann: Man spricht von einer *Abnormal Sprint Termination*. Mehr zum Sprint-Abbruch findet sich in Kapitel 3.

Die Sprint-Länge wird vom Scrum-Team gemeinsam festgelegt. Es ist nicht immer einfach, die optimale Sprint-Länge zu finden. Manchmal müssen Scrum-Teams ein wenig experimentieren, um ihr Optimum zu finden. Der geeignete Ort für die Reflexion über die Sprint-Länge ist die Sprint-Retrospektive. Auf keinen Fall sollten die Längen der Sprints ständig dem zu erledigenden Umfang angepasst werden.

Für die optimale Sprint-Länge gibt es eine Reihe von Einflussfaktoren. Dazu zählen:

- Der Sprint sollte lang genug sein, sodass es von Sprint-Review zu Sprint-Review ausreichend Fortschritt am Produkt zu sehen gibt.
- Der Sprint sollte so kurz sein, dass Product Owner und Stakeholder mit neuen Wünschen bis zum nächsten Sprint Planning warten können.
- Der Sprint sollte lang genug sein, sodass sich Selbstorganisation im Team entfalten kann.
- Der Sprint sollte kurz genug sein, sodass die Planungsfähigkeit des Teams nicht überschritten wird.

2.5 Artefakte

An dieser Stelle wollen wir uns einen ersten Überblick über die in Scrum verwendeten *Artefakte* verschaffen. Im Rest des Buches gehen wir auf die Verwendung der Artefakte genauer ein.

2.5.1 Product Backlog

Das *Product Backlog* ist das zentrale Artefakt zur Produktdefinition in Scrum. Der Product Owner pflegt und priorisiert das Product Backlog mit den enthaltenen Product Backlog Items. Die Product Backlog Items beschreiben die von außen wahrnehmbaren Produkteigenschaften. Viele Scrum-Teams verwenden User Stories und Epics zur Beschreibung von Product Backlog Items. Vorgeschrieben sind diese in Scrum aber nicht.

Der Product Owner ordnet/priorisiert die Product Backlog Items so, dass der Produktnutzen optimiert wird.

Der Scrum Guide definiert das Produktziel als das Commitment des Product Backlog. Mit dem Produktziel wird also definiert, auf welches Ziel die Product Backlog Items einzahlen sollen.

Details zu User Stories, Epics sowie zur Priorisierung des Product Backlog finden Sie in Kapitel 3.

2.5.2 Sprint Backlog

Das *Sprint Backlog* enthält hoch priorisierte Product Backlog Items, die für den Sprint ausgewählt wurden, sowie den Plan für die Umsetzung.

Der Plan für die Umsetzung besteht bei den meisten Scrum-Teams aus kleineren technischen Aktivitäten (Tasks). Im Gegensatz zu Product Backlog Items müssen einzelne Tasks für sich noch keinen erkennbaren Nutzen im Produkt ergeben. Prinzipiell ist jede andere Form von Plan aber auch möglich.

Das Sprint-Ziel ist das Commitment des Sprint Backlog: Analog zum Produktziel auf Product-Backlog-Ebene gibt es also an, auf welches kleinere Ziel die Sprint Backlog Items einzahlen sollen.

Der Aufbau des Sprint Backlog wird detaillierter in Kapitel 4 beschrieben.

2.5.3 Lieferbares Produktinkrement

Das *lieferbare Produktinkrement* (*Shippable Product Increment*) ist das Ergebnis der Arbeit eines Scrum-Teams im Sprint. Je Sprint entsteht eine Erweiterung und/oder Änderung des bisherigen Produktinkrements, sodass sich der Nutzen und der Wert des Produkts kontinuierlich erhöhen.

Die Inkrement-Idee bezieht sich auf vertikale Schnitte durch das Produkt. Wir erstellen also in einem Sprint nicht nur technische Dinge (wie vielleicht die Datenbank), die für Anwender nicht sichtbar werden, sondern immer einen Durchstich mit einem kleinen zusätzlichen Nutzen (siehe Abbildung 2.2).

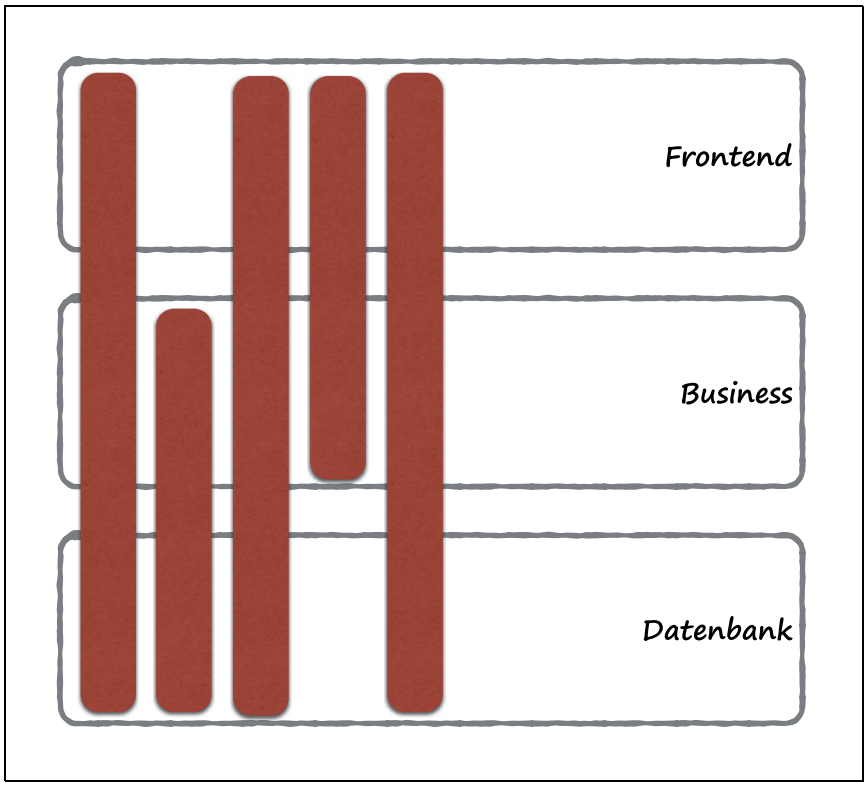

Abb. 2.2 *Produktentwicklung in vertikalen Schnitten*

Mit vertikalen Schnitten können wir früher Feedback über das Produkt erhalten. Wir können schneller feststellen, ob wir auf dem richtigen Weg sind und welche Produkteigenschaften noch fehlen (siehe auch Abschnitt 2.3.3). Außerdem können wir früher zu Produktversionen kommen, die tatsächlich produktiv genutzt werden.

Vertikale Schnitte bringen besondere technische Herausforderungen mit sich, die in Kapitel 4 genauer beschrieben sind.

Der Scrum Guide versteht die Definition of Done als das Commitment des Produktinkrements: Jedes Produktinkrement entspricht der Definition of Done. Umgekehrt kann man sagen, dass immer dann, wenn ein Sprint Backlog Item der Definition of Done entspricht, ein Produktinkrement entstanden ist.

2.6 Prinzipien

Bisher haben wir nur über die Mechanik von Scrum gesprochen: Welche Verantwortungen gibt es? Welche Meetings? Welche Artefakte? Das ist wichtig, um Scrum verstehen und anwenden zu können. Wir dürfen allerdings nicht aus den Augen verlieren, wozu wir die Scrum-Mechanik verwenden. Wir haben in Kapitel 1 den agilen Kernzyklus vorgestellt, in dem das agile Team Probleme der Endkunden in kurzen Zyklen löst. Wir begreifen die Scrum-Mechanik als Hilfsmittel, diesen Kernzyklus gut zum Laufen zu bekommen. Wie in Abbildung 2.3 dargestellt, steht der agile Kernzyklus im Vordergrund und die Scrum-Mechanik im Hintergrund.

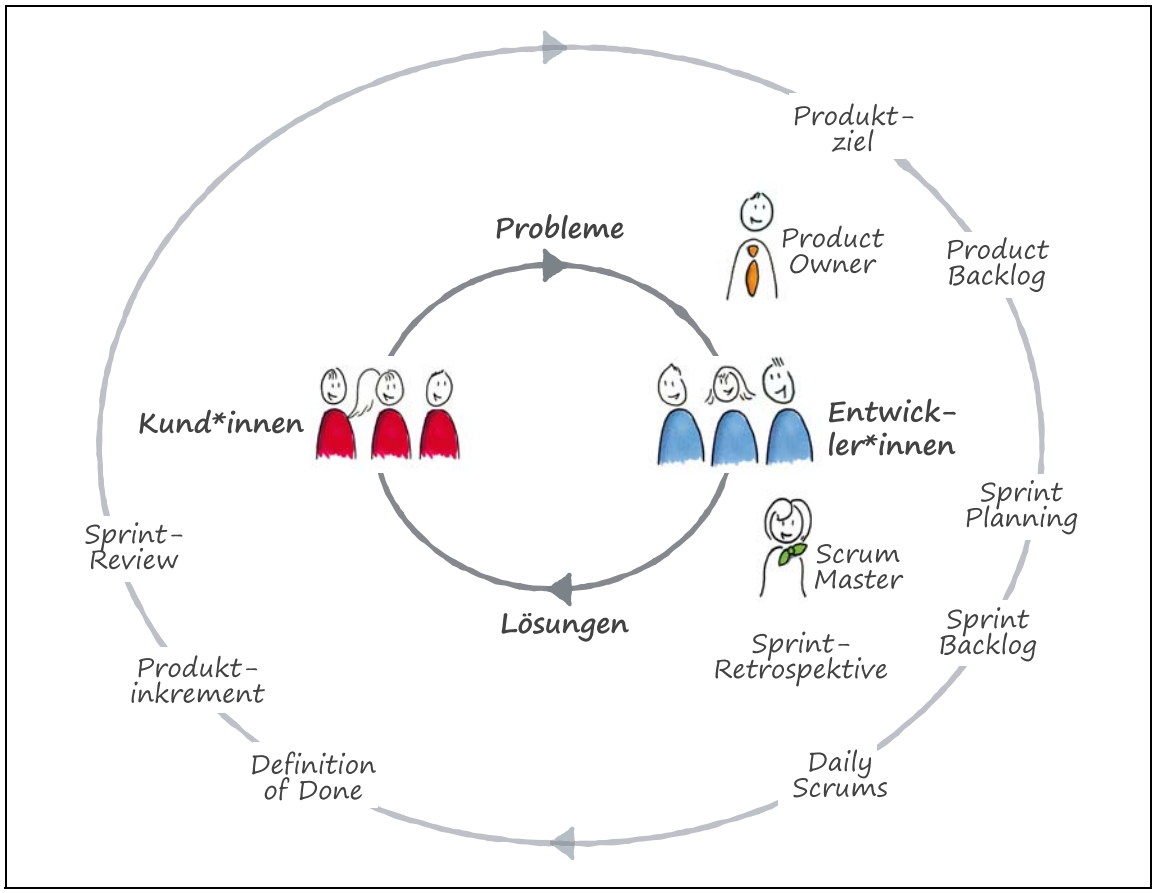

Abb. 2.3 *Scrum-Mechanik als Hilfsmittel zur Installation des agilen Kernzyklus*

Der agile Kernzyklus spiegelt sich auch in den *Scrum-Prinzipien* wider. Unserer Erfahrung nach sind der agile Kernzyklus und die Scrum-Prinzipien wichtiger als die Scrum-Mechanik. Wenn die Scrum-Prinzipien gelebt werden, aber die Scrum-Mechanik nicht vollständig umgesetzt ist, ist das besser als der umgekehrte Fall. Wenn Anpassungen an Scrum also helfen, den Kernzyklus und die Prinzipien besser umzusetzen, spricht wenig gegen die Anpassungen. Viel häufiger wird allerdings das Scrum-Framework angepasst, weil man die Schmerzen scheut, die mit einer Änderung der Arbeitsweise und Hierarchien einhergehen. Leidtragende sind der agile Kernzyklus und die Scrum-Prinzipien.

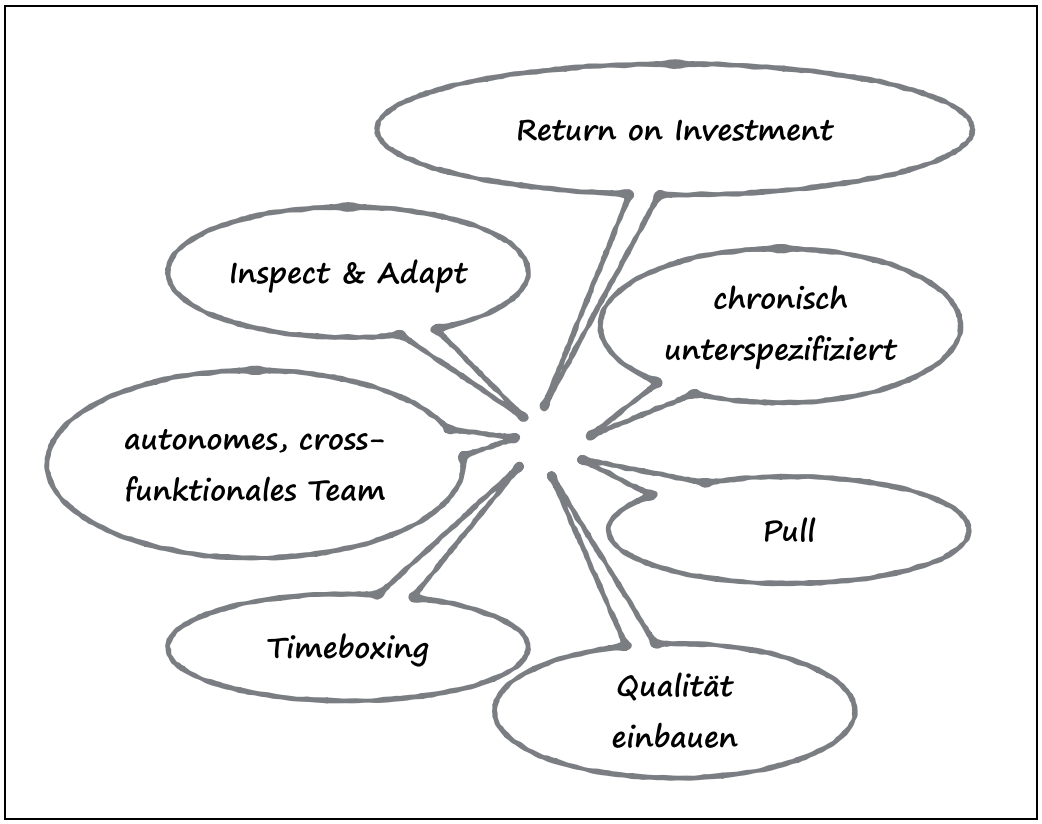

Abb. 2.4 *Scrum-Prinzipien*

Abbildung 2.4 zeigt die Scrum-Prinzipien, die wir im Folgenden genauer beschreiben.

2.6.1 Autonomes und cross-funktionales Team

Bereits in Kapitel 1 haben wir gesagt, dass Scrum-Teams autonom und cross-funktional sein sollen. Sie sollen möglichst wenig Abhängigkeiten nach außen aufweisen, um selbstgesteuert schnell agieren zu können.

2.6.2 Inspect & Adapt (auch: empirisches Management)

Ebenfalls in Kapitel 1 haben wir beschrieben, dass das Scrum-Framework dazu dient, *Inspect & Adapt* auf das Produkt und den Prozess anzuwenden.

Inspect & Adapt ist auch die Grundlage empirischen Managements, also die Grundidee, aus Erfahrungen und Beobachtungen zu lernen und daraus Vorhersagen über die Zukunft abzuleiten. Mehr zu empirischem Management findet sich in Kapitel 3.

2.6.3 Timeboxing

Timeboxes sind Zeiträume fester Länge, die mit Arbeit gefüllt werden. Ist die geplante Timebox abgelaufen, halten wir auf jeden Fall inne und reflektieren (auch wenn die geplante Arbeit nicht vollständig erledigt ist).

Das Prinzip des Timeboxing findet sich in Scrum sowohl für die Meetings als auch für den Sprint wieder. Timeboxes werden hier verwendet, um einen Fokus herzustellen: Wenn uns nicht beliebig viel Zeit zur Verfügung steht, dann müssen wir sehr genau entscheiden, was wir mit dieser begrenzten Zeit tun. Insofern führt auch die Sprint-Timebox dazu, dass der Product Owner priorisieren muss: Was ist jetzt das Wertvollste, das wir tun können?

Eine zweite wichtige Funktion von Timeboxes ist *das Erzwingen schwieriger Entscheidungen*. Wenn der Sprint zu Ende ist, müssen wir z. B. im Sprint-Review eine Reihe schwieriger Entscheidungen fällen:

- Hat der Sprint ausreichend Wert geschaffen bezogen auf die Investition?
- Können wir mit dem Produktinkrement produktiv gehen?
- Müssen wir etwas am weiteren Plan anpassen (und können wir damit vielleicht nicht der Erwartungshaltung aller Stakeholder genügen)?
- Müssen wir die Teamzusammensetzung ändern?
- Müssen wir das Projekt abbrechen?

Mit dieser Sichtweise (Erzwingen schwieriger Entscheidungen) kann man auch einfach darüber befinden, was zu tun ist, wenn eine Meeting-Timebox (z. B. für das Sprint Planning) abgelaufen ist. Wir sollten uns dann diese schwierigen Fragen stellen:

- Reicht das, was wir erreicht haben, aus, um weiterzumachen (also z. B., um in den Sprint zu starten)?
- Wenn nicht: Ist das bisher gewählte Vorgehen geeignet, um das Meetingziel absehbar zu erreichen?
- Wie lang müsste eine weitere Timebox sein, damit wir das Meetingziel noch erreichen können?

2.6.4 Return on Investment (ROI)

Wenn uns also die Sprints zu harten Priorisierungsentscheidungen und zum Fokussieren zwingen, dann stellt sich natürlich die Frage, wonach diese Priorisierung erfolgen sollte. Hier sieht Scrum vor, dass man sich am *Return on Investment (ROI)* orientiert, also den meisten Wert aus der verfügbaren Kapazität der Entwickler her-

ausholt. Die Wertmaximierung erfolgt aber nicht dadurch, dass mehr Druck auf die Entwickler ausgeübt wird, sondern dass das Scrum-Team auf die Product Backlog Items fokussiert, die besonders wertvoll sind.

2.6.5 Qualität einbauen

Die inkrementelle Entwicklung in Scrum erfordert besonderen Fokus auf die interne und externe Softwarequalität. Die Software muss zum einen qualitativ gut genug sein, sodass die Anwender sie gut benutzen können. Zum anderen muss sie eine hohe interne Qualität aufweisen; nur so können die Entwickler auf Dauer mit gleichbleibender Geschwindigkeit Erweiterungen und Änderungen am Produkt vornehmen.

Diese Qualitäten lassen sich nicht mit nachgelagerter Qualitätssicherung in das Produkt »reintesten«. Die Entwickler müssen sich sehr früh darüber Gedanken machen, wie sie die Qualität von Anfang an auf einem hohen Niveau halten. Scrum macht keine Vorgaben dazu, durch welche Techniken eine hohe Qualität geschaffen werden soll. Es steht aber eine ganze Reihe etablierter agiler Entwicklungspraktiken zur Verfügung, die häufig von Scrum-Entwicklern verwendet werden (siehe Kapitel 4).

2.6.6 Pull

Das *Pull-Prinzip* ist ein Kernprinzip aus der Lean Production und lässt sich (im Gegensatz zu einigen anderen Lean-Production-Ansätzen) sehr gut auf die Entwicklung übertragen. Das Pull-Prinzip fordert, dass Aufgaben aktiv angenommen, gepullt (also gezogen) werden, und zwar dann, wenn für die Erledigung wieder Kapazität verfügbar ist.

Im Gegensatz zum Pull-Prinzip steht das *Push-Prinzip*, das häufig in klassischer Softwareentwicklung eingesetzt wird: Ein Projektmanager plant die Arbeit für die Entwickler und weist Aufgaben mit Zeitvorgaben zu. Das Push-Prinzip funktioniert nur in sehr vorhersehbaren Situationen. In Kontexten mit hoher Varianz führt es zu Über- oder Unterlastung der Entwickler. Bei Überlastung geraten die Entwickler in Stress und reduzieren allzu häufig die Qualität, um die Zeitvorgabe einzuhalten.

Das Pull-Prinzip vermeidet Über- und Unterlastung, Warteschlangen verkürzen sich, und es entsteht ein kontinuierlicher Durchfluss (Flow) der Aufgaben durch das Team.

Softwareentwicklung hat fast immer hohe Varianz: Selbst wenn die Anforderungen sehr klar sind, bleibt Softwareentwicklung ein kreativer Prozess (man spricht auch von Wissensarbeit), und bei diesem hängt die Produktivität auch von der individuellen Tagesform der Entwickler ab.

Push und Pull in der Produktion

Die Konzepte von Push und Pull stammen aus der Massenproduktion. In der klassischen Produktion arbeitete man nach dem Push-Prinzip: Es wird versucht, die Auslastung der teuren Maschinen zu optimieren. Man lässt also die teuren Maschinen unter Volllast laufen. In einer perfekten Welt, in der die Maschinen gleichbleibend leistungsfähig sind und die Leistungsfähigkeiten der verschiedenen Maschinen optimal aufeinander abgestimmt sind, optimiert man mit dem Push-Ansatz tatsächlich die Produktion.

In der Praxis ist eine solche ideale Situation nicht zu beobachten. Maschinen haben Varianzen in ihrer Leistungsfähigkeit (z. B. durch geplante Wartung oder ungeplante Reparaturen), irgendwo sind Menschen beteiligt, die Produktion muss für Produktvarianten oder komplett neue Produkte umgestellt werden etc. Dann führt das Push-Prinzip zu einer Lagerhaltung: Maschine A produziert in maximaler Geschwindigkeit. Die nachgelagerte manuelle Montage kann die von Maschine A produzierten Teile nicht in derselben Geschwindigkeit verarbeiten. Also baut man ein Lager von A-Teilen auf, aus dem sich die Montage bedient.

Diese Lagerhaltung bringt eine Reihe von Problemen mit sich. Es entsteht unnötige Arbeit für Einlagerung, Verwaltung und Entnahme der Teile aus dem Lager sowie für den Transport zwischen Produktion und Lager. Außerdem werden Probleme in der Produktion verschleiert: Wenn Maschine A einen Defekt hat und minderwertige Teile produziert, entdeckt man den Fehler erst in der Montage und hat dann bereits ein Lager voll minderwertiger Teile aufgebaut.

Mit dem Pull-Prinzip werden diese Probleme vermieden. Ein Produktionsschritt holt (pullt) sich Teile vom vorgelagerten Produktionsschritt. Maschine A produziert Teile also nur, wenn die Montage welche braucht. Auch in der Pull-Produktion werden Lagerbestände aufgebaut, damit die Montage Teile direkt dann zur Verfügung hat, wenn sie benötigt werden, und nicht auf Maschine A warten muss. Im Gegensatz zu Lagern bei Push-Systemen sind diese Lagerbestände kleiner und wachsen nicht unkontrolliert.

Pull-Systeme führen also dazu, dass Lagerhaltung reduziert wird. Dadurch werden auch Aufwand und Zeit für Einlagerung, Verwaltung und Entnahme reduziert. Durch die kleineren Lagerbestände fallen außerdem Probleme in der Produktion schneller auf. Natürlich führt das Pull-Prinzip dazu, dass die Maschinen in der Regel nicht voll ausgelastet sind. Überraschenderweise führt die Produktion nach Pull-Prinzip trotzdem zu kürzeren Produktionszeiten und höherer Produktivität. Toyota hat das Pull-Prinzip für die Produktion über Jahrzehnte entwickelt und optimiert, und heute wird es bei den meisten Autoherstellern verwendet.

Näheres findet sich bei [Ohno1988].

Die Parallelen zur schlanken Produktion (Lean Production) sollten nicht missverstanden werden: Es handelt sich bei Scrum keineswegs um eine Übertragung der Lean Production auf die Softwareentwicklung. Es haben sich parallel ähnliche Strukturen entwickelt, es gibt aber auch drastische Unterschiede. So sind Varianzen in der Produktion auf jeden Fall zu vermeiden, während einige Varianzen in Scrum gewollt sind; siehe dazu auch Abschnitt 6.5 zur Planung großer Vorhaben sowie [Reinertsen2009].

Scrum setzt im Sprint Planning auf das Pull-Prinzip. Hier ziehen sich die Entwickler nur so viele Aufgaben in den Sprint, wie sie auch realistisch erfüllen können.

2.6.7 Bewusst unterspezifiziert

Man kann Scrum vorwerfen, dass es unterspezifiziert sei. Es lässt viele Fragen offen (z. B. wie genau geschätzt werden soll, welche Entwicklungspraktiken im Sprint anzuwenden sind). Aber gerade darin liegt auch eine große Stärke von Scrum: Es ist sehr universell einsetzbar für die verschiedensten Arten der Softwareentwicklung, für die Entwicklung von Hardware und für viele softe Themen wie Marketing, Vertrieb, Reorganisationen in Unternehmen etc.

Scrum bietet mit seinen Verantwortungen, Meetings und wenigen Spielregeln einen Rahmen, mit dem man nach Inspect & Adapt innovative Produkte entwickeln kann. Auch für den Prozess selbst gilt dabei Inspect & Adapt.

Zusammengefasst kann man sagen: Ein Vorteil von Scrum ist seine breite und flexible Anwendbarkeit. Die Herausforderung besteht darin, dass man es wagen muss, althergebrachte Arbeitsprozesse zu hinterfragen und ggf. zu ändern. Der Nachteil ist, dass man an der einen oder anderen Stelle selbst denken muss und eine spezifische Lösung für sein eigenes Vorgehen finden muss.

2.7 Scrum-Werte

Der Scrum Guide definiert die folgenden *Scrum-Werte:*

- Commitment
- Fokus
- Offenheit
- Respekt
- Mut

Ähnlich wie die Prinzipien dienen auch die Werte dazu, die Entwicklung nach Scrum konkret mit Leben zu füllen. Sie sind auch dann nützlich, wenn man über Änderungen am Prozess nachdenkt. Man kann anhand der Werte oft gut diskutieren, ob eine vorgeschlagene Prozessänderung näher an die Scrum-Werte heranführt oder eher davon wegführt.

- *Commitment* meint, dass die Beteiligten aus freien Stücken Verantwortung für das gemeinsame Ziel übernehmen und dieses ihnen wichtiger ist als ihre persönlichen Ziele.
- *Fokus* besagt, dass sich das Scrum-Team auf seine Aufgabe konzentriert (und nicht durch zusätzliche Aufgaben abgelenkt wird). Es bedeutet auch, dass fokussiert im Sprint gearbeitet wird und man sich auf die wertvollen Product Backlog Items konzentriert. Für Stakeholder ist es mitunter schwer zu ertragen, wenn ihre Interessen aktuell nicht im Fokus sind.
- *Offenheit* ist eine klare Absage an politische Spielchen oder das Verstecken von Problemen oder Fakten. Alles soll auf den Tisch kommen und für alle sichtbar gemacht werden. Offenheit bedeutet aber mehr als nur Transparenz. Wir wol-

len auch offen dafür sein, Dinge zu entdecken, die nicht zu unseren Plänen oder den eigenen Annahmen passen, und offen bleiben für neue alternative Lösungswege.

- *Respekt* den Beteiligten gegenüber ist ein hoher Wert, der es uns erlaubt, anders miteinander umzugehen und gemeinsame Lösungen zu finden, die die Interessen und Bedürfnisse vieler Beteiligter berücksichtigen oder zumindest betrachtet haben. Respekt ist auch eine wichtige Voraussetzung für Offenheit.
- *Mut* ist erforderlich, wenn man neue Wege geht. Und neue Wege muss man gehen, wenn man auf der Suche nach Innovation ist – im Produkt wie im Prozess.

2.8 Das Kapitel in Stichpunkten

- Scrum ist kein Prozess und keine Methode. Scrum ist ein Rahmenwerk (Framework) für das Management von Entwicklungsvorhaben.
- Scrum definiert das Scrum-Team, darin drei Verantwortlichkeiten, drei Artefakte und vier Meetings/Events, die durch Sprints zusammengehalten werden. Alles andere muss situationsspezifisch ausgefüllt werden.
- Der Product Owner verantwortet den Produkterfolg. Er optimiert den Produktnutzen durch Priorisierung der Produkteigenschaften (Features).
- Die Entwickler sind cross-funktional (interdisziplinär) besetzt, organisieren sich selbst und entscheiden autonom über das »Wie« der Entwicklung.
- Der Scrum Master sorgt dafür, dass Scrum effektiv angewendet wird. Er ist auch darauf bedacht, dass alle Beteiligten (auch außerhalb des Scrum-Teams) Scrum richtig verstehen, und er coacht die Beteiligten in der effektiven Anwendung. Der Scrum Master sorgt für die Beseitigung von Hindernissen, die der Entwicklung im Weg stehen. Er führt durch Dienen.
- Stakeholder ist der Sammelbegriff für alle Personen, die nicht zum Scrum-Team gehören, aber trotzdem ein Interesse an dem Produkt bzw. der Entwicklung haben. Das können Kunden, Anwender, Management, der Betriebsrat etc. sein.
- Das Product Backlog ist die Menge der Produkteigenschaften, von denen wir heute glauben, dass sie im Produkt vorhanden sein müssen. Der Product Owner sorgt für das Product Backlog und priorisiert die Einträge.
- Das Produktziel beschreibt den Zweck des Product Backlog.
- Das Sprint Backlog enthält die hoch priorisierten Product Backlog Items, die für den Sprint selektiert wurden, sowie einen Plan zur Umsetzung dieser Einträge.
- Das Sprint-Ziel beschreibt den Zweck eines jeden Sprints.
- Das Produktinkrement wird von Entwicklern erzeugt und ist das Ergebnis des Sprints. Es muss lieferbar sein (es dürfen also keine technischen Arbeiten offengeblieben sein, die die sofortige Auslieferung verhindern würden). Ob das Produktinkrement ausgeliefert wird oder nicht, entscheidet der Product Owner.

- Die Definition of Done legt die Qualität des Produktinkrements fest.
- Im Sprint Planning werden hoch priorisierte Einträge aus dem Product Backlog selektiert, und die Entwickler erstellen einen Plan für die Umsetzung.
- Im Daily Scrum stimmen sich die Entwickler werktäglich darüber ab, wie sie sich so organisieren können, dass sie bis zum nächsten Daily Scrum den optimalen Fortschritt bezogen auf das Sprint-Ziel erreichen.
- Im Sprint-Review präsentiert das Scrum-Team das Produktinkrement relevanten Stakeholdern, um Feedback für die weitere Entwicklung des Produkts zu erhalten.
- In der Sprint-Retrospektive reflektiert das Scrum-Team über den vergangenen Sprint und definiert Verbesserungsmaßnahmen, mit denen es im nächsten Sprint noch effektiver arbeiten kann.
- Das Scrum-Framework soll helfen, die Scrum-Werte und -Prinzipien zu etablieren. Die Scrum-Mechanik ohne die Werte und Prinzipien ist am Ende nicht viel wert.