2 Überblick über den Scrum-Ablauf, die Verantwortungen, Meetings, Artefakte und Prinzipien

»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 Managementrahmenwerk, 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 Buches 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.

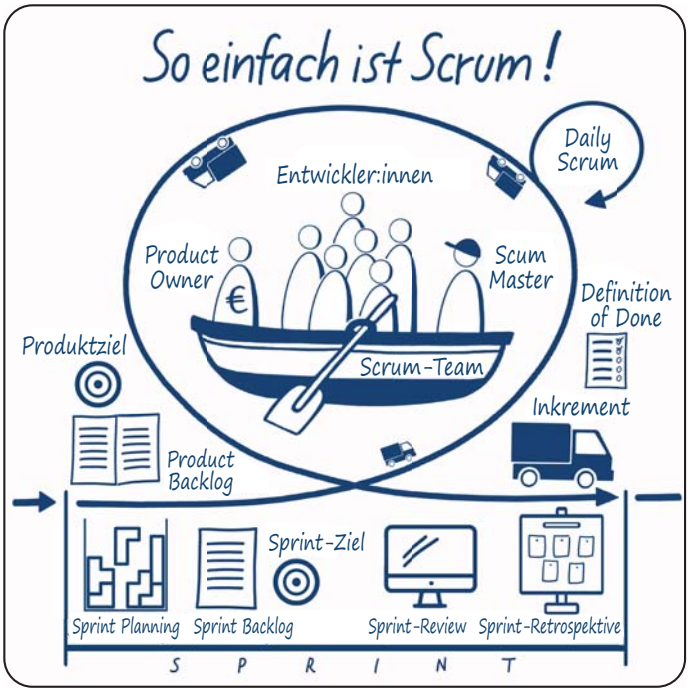

Abb. 2–1 *Der Scrum-Ablauf passt auf einen Bierdeckel.*

Am Anfang steht ein *Produktziel*, was 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 Product Backlog 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:innen* sind für die Umsetzung der Product Backlog Items verantwortlich. Der Product Owner ist damit für das Was und die Entwickler:innen für das Wie der Entwicklung verantwortlich.

Zu Beginn jedes Sprints führen Product Owner und Entwickler:innen 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:innen 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:innen 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:innen fokussiert arbeiten können.

Zur Abstimmung unter den Entwickler:innen findet jeden Werktag während des Sprints das *Daily Scrum* statt. Im Daily Scrum treffen sich die Entwickler:innen 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:innen 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:innen 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 (Kund:innen, Anwender:innen, 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:innen 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 logisch betrachtet 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:innen* 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 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:innen 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:innen

Entwickler:innen 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:innen 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 Programmierer:innen auch andere Expert:innen wie Tester:innen, Designer:innen oder Mitarbeitenden aus dem Betrieb als Entwickler:innen ins Scrum-Team eingebunden.

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

Die Entwickler:innen organisieren sich und ihre Arbeit selbst. Dabei sind alle gleich, es soll keine Titel und auch keine Subteams (die Entwickler:innen, die Tester:innen, die Frontend-Entwickler:innen 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 Spezialist:innen mit besonderen Skills geben. Trotzdem wird die Verantwortung gemeinsam getragen.

Der Scrum Guide sieht die Entwickler:innen 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 Expert:innen zur Verantwortung zu ziehen.

Bei Überlastung des Product Owners übernehmen die Entwickler:innen 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 oder sie 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 indem er oder sie

- die Teammitglieder in Selbstmanagement und interdisziplinärer Zusammenarbeit coacht,
- dem Scrum-Team hilft, fokussiert an der Erstellung hochwertiger Produktinkremente zu arbeiten,
- dabei unterstützt, dass alle Hindernisse (Impediments) aus dem Weg geräumt werden, damit das Scrum-Team Fortschritte erzielen kann, und
- sicherstellt, 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 indem er oder sie

- bei der Suche nach effektiven Techniken zur Definition des Produktziels und dem Management des Product Backlog unterstützt,
- dem ganzen Scrum-Team dabei hilft, die Notwendigkeit klarer und präziser Product Backlog Items zu verstehen,

- für die Etablierung einer empirischen Produktplanung für ein komplexes Umfeld sorgt und
- nach Wunsch oder Bedarf bei der Zusammenarbeit mit Stakeholdern hilft, z. B. durch Moderation.

Der Scrum Master dient der gesamten Organisation unter anderem dadurch, dass er oder sie

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

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 Entwickler:innen 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 oder Projektleiterin in Scrum

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

Die Aufgaben klassischer Projektleitung verteilen sich auf die Verantwortlichkeiten Product Owner, Scrum Master, Entwickler:innen 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:innen 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:innen, wie viele Items in den Sprint gezogen werden. So verhindern die Entwickler:innen 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:innen 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:innen 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 Entwickler:innen gemeinsam vorgenommen wird, erhöht sich seine Qualität, und die Wahrscheinlichkeit steigt, dass die Entwickler:innen 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:innen in einen sinnvolle 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:innen 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 den Kapiteln 3 und 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:innen 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:innen geht, muss der Product Owner nicht zwingend am Daily Scrum teilnehmen. Wir empfehlen dieses jedoch, weil er oder sie 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:innen 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 Kund:innen und Anwender:innen) am Sprint-Review teilnehmen. Sie geben das wertvollste Feedback zum Produkt.

Das Sprint-Review beginnt mit einer Demonstration des Produktinkrements durch die Entwickler:innen. 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 einige Entwickler:innen 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:innen 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:innen 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 abgebrochen 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. Zu ihnen 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.

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

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:innen nicht sichtbar werden, sondern immer einen Durchstich mit einem kleinen zusätzlichen Nutzen (siehe Abb. 2–2).

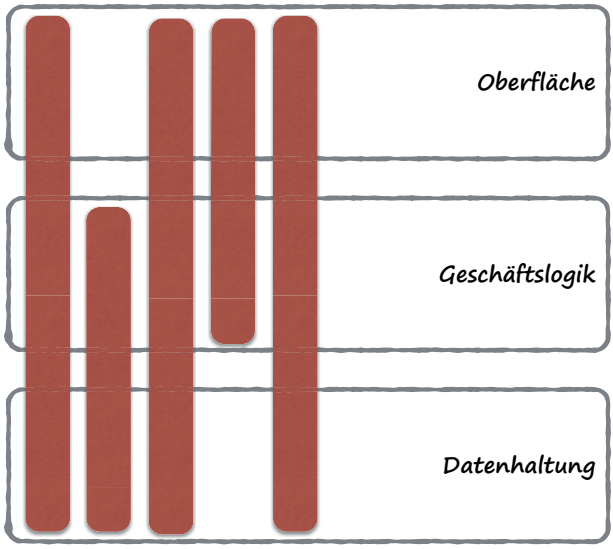

Abb. 2–2 *Produktentwicklung in vertikalen Schnitten*

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 Endkund:innen 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.

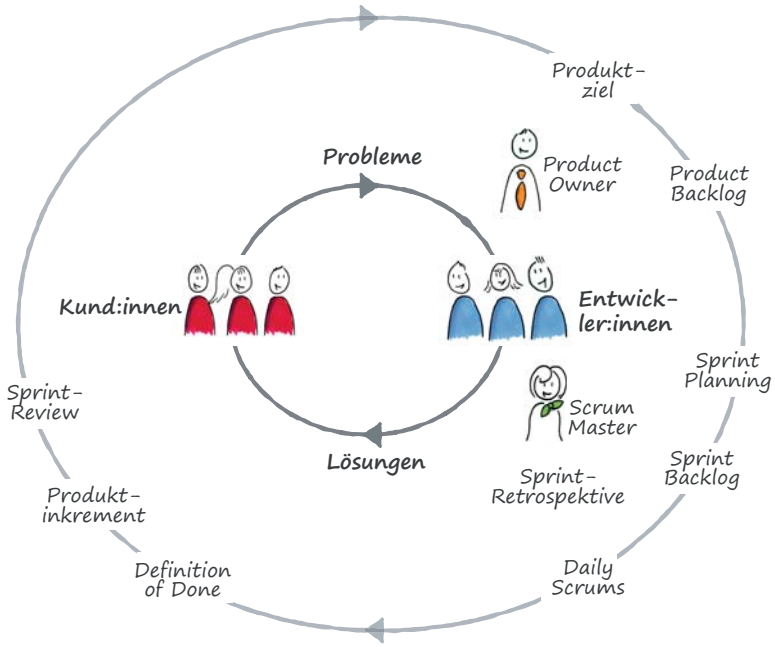

Abb. 2–3 *Scrum-Mechanik als Hilfsmittel zur Installation des agilen Kernzyklus*

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.

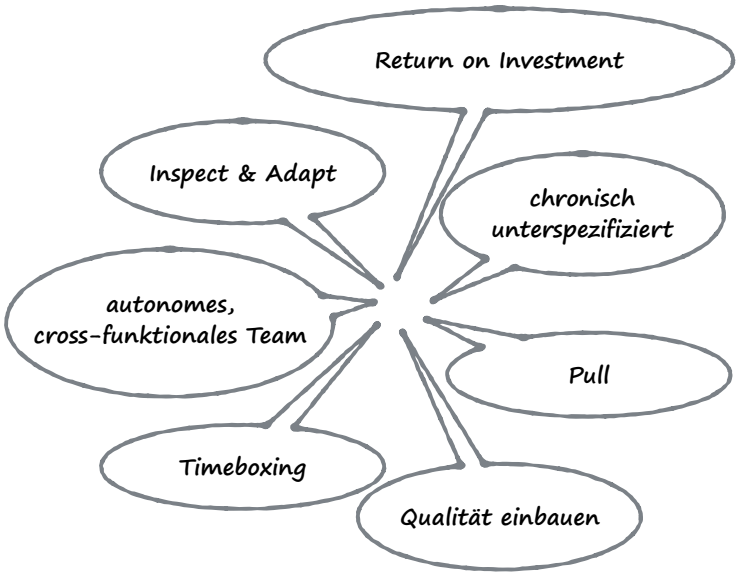

Abb. 2–4 *Scrum-Prinzipien*

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:innen herausholt. Die Wertmaximierung erfolgt aber nicht dadurch, dass durch Druck mehr aus den Entwickler:innen herausgeholt 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:innen sie gut benutzen können. Zum anderen muss sie eine hohe interne Qualität aufweisen; nur so können die Entwickler:innen 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:innen 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-Entwickler:innen verwendet werden (siehe Kap. 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 oder Projektmanagerin plant die Arbeit für die Entwickler:innen 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:innen. Bei Überlastung geraten die Entwickler:innen 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:innen ab.

Einschub: 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 über Releaseplanung sowie [Reinertsen2009]).

Scrum setzt im Sprint Planning auf das Pull-Prinzip. Hier ziehen sich die Entwickler:innen nur so viele Aufgaben in den Sprint, wie sie auch realistisch leisten 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 wollen 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:innen 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 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 Kund:innen, Anwender:innen, 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 Entwickler:innen 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:innen erstellen einen Plan für die Umsetzung.
- Im Daily Scrum stimmen sich die Entwickler:innen 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.

7 Streiflicht auf fortgeschrittenes Scrum

»Equipping organizations to tackle the future would require a management revolution no less momentous than the one that spawned modern industry.«

Gary Hamel[1]

Bisher haben wir in diesem Buch die Scrum-Grundlagen behandelt. Dadurch sollte klar geworden sein, wie Scrum für *ein Team* funktioniert, das an *einem Ort* zusammenarbeitet (Co-Location).

In diesem Kapitel wollen wir einige fortgeschrittene Themen streifen: *Scrum einführen, Scrum skalieren, agile Unternehmen, verteiltes Scrum, die veränderte Rolle von Führungskräften* sowie *Vertragsgestaltung für agile Entwicklung*. Für jedes Thema gibt es eigene Bücher, und dieses Kapitel kann nicht mehr leisten, als ein kurzes Streiflicht zu geben. Das Ziel ist also nicht, das jeweilige Thema vollständig abzuhandeln, sondern eine Vorstellung davon zu vermitteln, wo die jeweiligen Herausforderungen liegen. Insbesondere raten wir dringend davon ab, mit den drei Themen nur auf Basis der spärlichen Informationen dieses Kapitels zu experimentieren. Man sollte sich vorher deutlich intensiver informieren und sich am besten Unterstützung von jemandem holen, der oder die bereits mehrjährige Erfahrung mit dem jeweiligen Thema gesammelt hat.

7.1 Scrum einführen

Die Beschäftigung mit Scrum beginnt mit einem Pilotprojekt. (Ausnahme sind extreme Krisensituationen, in denen das Unternehmen mit dem Rücken zur Wand steht und der Weg über *ein* Pilotprojekt zu langsam wäre.)

Bevor das erste Scrum-Pilotprojekt startet, sollte Klarheit darüber herrschen, was man sich von Scrum erhofft. In dieser Orientierungsphase kann das Unternehmen auf Basis der Ziele bewerten, welcher agile Ansatz am erfolgversprechendsten erscheint. Hier ist es sinnvoll, Beratungsleistung von erfahrenen agilen Expert:innen in Anspruch zu nehmen.

1. Siehe [Hamel2009].

Wenn das Pilotprojekt Erkenntnisse über Scrum geliefert hat, sollte entschieden werden, ob und in welchem Umfang Scrum im Unternehmen eingeführt wird (siehe Abb. 7–1).

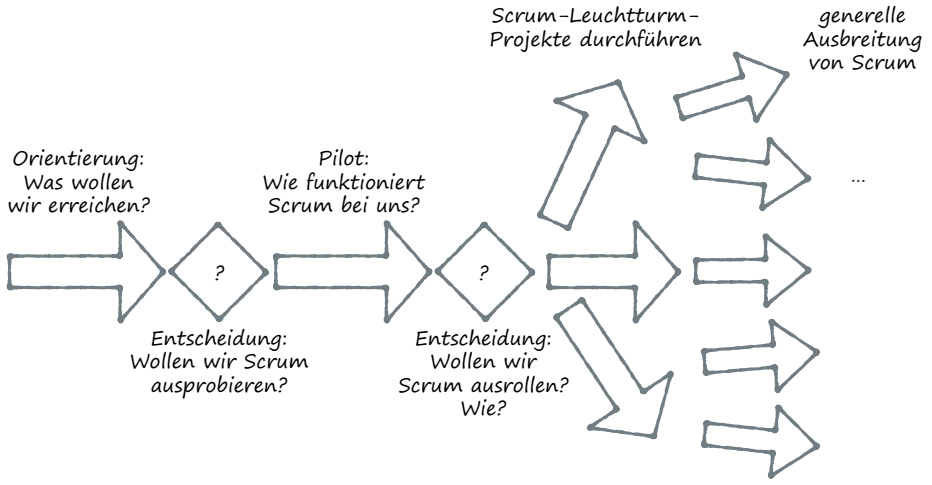

Abb. 7–1 *Typische Phasen der Einführung von agilem Vorgehen/Scrum*

Wenn man sich für das Ausprobieren eines konkreten Verfahrens wie Scrum entschieden hat, startet man das Pilotprojekt, das idealerweise diese Eigenschaften aufweist:

1. Das Projektziel ist mit den herkömmlichen Ansätzen vermutlich nur mit großem Risiko oder gar nicht erreichbar.
2. Man kann es sich erlauben, dass das Projekt scheitert. Es muss aber ausreichend relevant sein, dass ein Scheitern schmerzhaft ist.
3. Das Projekt weist keine oder wenige Abhängigkeiten von anderen Projekten/Teams/Abteilungen auf. Es sollte aber auch nicht vollkommen anders sein als die Projekte, für die in Zukunft Scrum eingesetzt werden soll.

Es finden sich selten alle drei Eigenschaften in Perfektion wieder. In der Regel bringt es wenig, auf das perfekte Pilotprojekt zu warten. Im Zweifelsfall ist es besser, von den jetzt anstehenden Projekten das auszuwählen, das die Kriterien am besten erfüllt, und dann loszulegen.

Das Pilotprojekt wird neben dem erstellten Produkt vor allem Erkenntnisse darüber liefern, was Scrum für das Unternehmen bedeutet. Es sollte deutlich werden, welche Vorteile Scrum für das Unternehmen haben kann und welche Auswirkungen potenziell ins Haus stehen, wenn man Scrum im größeren Umfang einführen will.

Vor dem Start des Pilotprojekts sollte definiert werden, wann die Ergebnisse wie ausgewertet werden. Auf Basis dieser Auswertung entscheidet das Management dann, ob und wie Scrum weiter im Unternehmen ausgerollt wird.

7.1.1 Veränderte Verhaltensweisen

Viele Unternehmen haben festgestellt, dass das bloße Kopieren der Scrum-Mechaniken nicht den gewünschten Erfolg bringt:[2] Mechaniken verändern Verhaltensweisen meist nicht nachhaltig und bringen nicht den erwünschten Kulturwandel.

Sehen wir uns ein einfaches Modell zu menschlichem Verhalten an (siehe Abb. 7–2). Wir alle haben ein Wertesystem im Kopf. Ein Glaubenssatz 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 auf unser Wertesystem zurück (»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«).

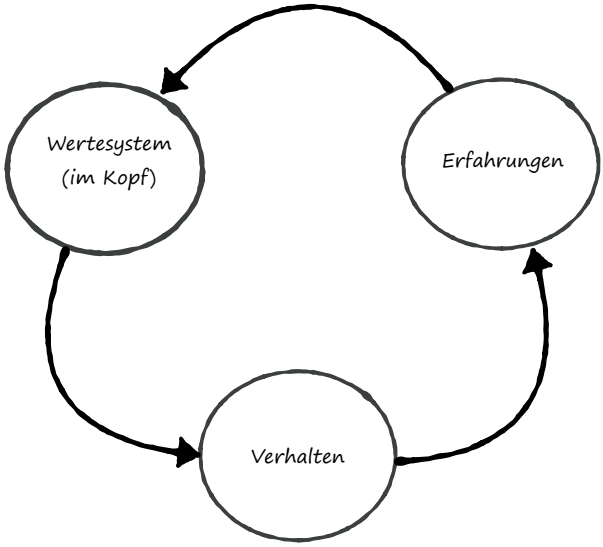

Abb. 7–2 *Selbstverstärkender Zyklus menschlicher Verhaltensweisen*

Es kann sehr anspruchsvoll sein, diese Zyklen zu durchbrechen. So fordert Scrum beispielsweise, dass die Sprints vor Interventionen von außen geschützt sind. Das widerspricht möglicherweise den Glaubenssätzen im Unternehmen, und Scrum wird »pragmatisch« angepasst: Es wird trotzdem im Sprint interveniert (»Meine Erfahrung sagt mir ja, dass es schiefgeht, wenn die Entwickler:innen nicht engma-

2. Davon kann übrigens auch das produzierende Gewerbe ein Lied singen, das seit Jahrzehnten versucht, durch das Kopieren von Toyota-Praktiken so erfolgreich wie Toyota zu werden.

schig kontrolliert werden«). Solche Anpassungen von Scrum an die Glaubenssätze des Unternehmens erkennt das Unternehmen selten als problematisch. Sie erscheinen vollkommen logisch und werden häufig sogar als »alternativlos« angesehen. Hier hilft der Blick von außen durch eine Person mit langjähriger agiler Erfahrung.

Veränderungen an Verhaltensweisen erreichen wir effektiver durch Coaching, also dadurch, dass wir direkt das Verhalten der Beteiligten beeinflussen (siehe Abb. 7–3). Dieses geänderte Verhalten führt zu neuen Erfahrungen, die irgendwann das Wertesystem im Kopf ändern, und dann wird das gewünschte Verhalten automatisch erzeugt; Coaching ist dann für diesen Aspekt nicht mehr notwendig. Scrum hat dafür die Scrum-Master-Verantwortung vorgesehen. Der Scrum Master coacht die Beteiligten in der effektiven Anwendung von Scrum und hält z.B. Führungskräfte von unangemessener Intervention im Sprint ab. Ein intern besetzter Scrum Master ohne vorherige Scrum-Erfahrung und ohne externen Impuls wird Schwierigkeiten haben, die gewünschten Verhaltensweisen zu bewirken: Er ist selbst in den Glaubenssätzen des eigenen Unternehmens »gefangen«. Interne – gerade erst ausgebildete – Scrum Master profitieren selbst stark von externem Coaching.

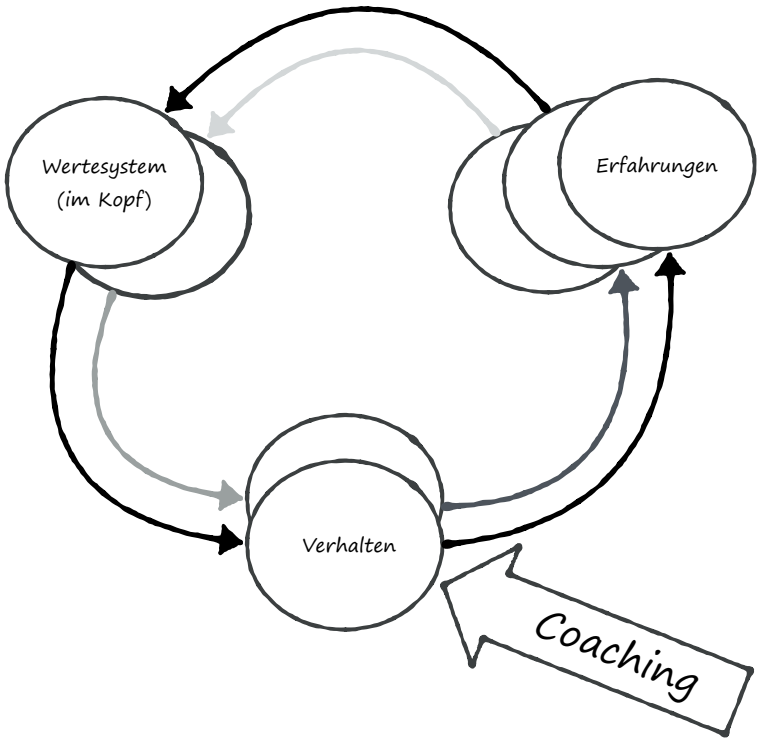

Abb. 7–3 *Verhalten durch Coaching ändern*

7.1.2 Scrum im Unternehmen verankern

Wenn das Pilotprojekt erfolgreich war und man bereit ist, die durch den Piloten aufgedeckten organisatorischen Probleme im Unternehmen anzugehen, kann der Einsatz von Scrum im Unternehmen ausgeweitet werden.

Wir haben bereits gesehen, dass es um Kulturwandel geht. Wird diese neue Kultur zunächst nur in Teile des Unternehmens getragen, kommt es zu Spannungen mit der existierenden Unternehmenskultur. Stephen Denning spricht davon, dass das Unternehmen mit sich selbst im Krieg liegt (siehe [Denning2010]). Während bei kleineren Unternehmen die Kultur in »einem Rutsch« gewandelt werden kann, müssen größere Organisationen schrittweise vorgehen.

Um letztlich nutzlose Spannungen zu minimieren, muss die neue Kultur klar von der existierenden Kultur abgegrenzt werden. Im Pilotprojekt ist diese Abgrenzung über den Piloten erfolgt. Es war allen klar, dass das Pilotprojekt eine Sonderrolle einnimmt, quasi eine »Erlaubnis zum Anderssein« hat. Der Scrum Master hat das Pilotprojekt vor unangemessenen Eingriffen der alten Unternehmenskultur geschützt. Wenn beschlossen wird, Scrum im Unternehmen weiter auszubreiten, verschwindet diese Sonderrolle.

Für die Ausbreitung von Scrum benötigen wir daher andere Mechanismen. Idealtypisch kann man hierbei das *Scrum Studio* und *autonome Business Units* unterscheiden. Wir beschreiben beide Ansätze weiter unten.

7.1.3 Kulturwandel im Unternehmen

Kulturwandel kann man nicht verordnen. Die Unternehmenskultur wird im Wesentlichen durch das Verhalten der Menschen im Unternehmen – insbesondere das der Führungskräfte – geprägt: Wie gehen sie mit Fehlern um? Wofür wird Anerkennung gezollt? Wie ist der Umgangston? Welche gemeinsamen Rituale gibt es? Wie wird mit Konflikten umgegangen? Also findet ein Kulturwandel über Verhaltensänderungen der Menschen im Unternehmen statt.

In einem größeren Kontext das Verhalten von Menschen »mit der Gießkanne« zu ändern, ist extrem aufwendig und risikoreich. Es ist daher sinnvoll, die neue Kultur schrittweise im Unternehmen auszubreiten – ausgehend von einer oder mehreren agilen Keimzellen (siehe Abb. 7–4).

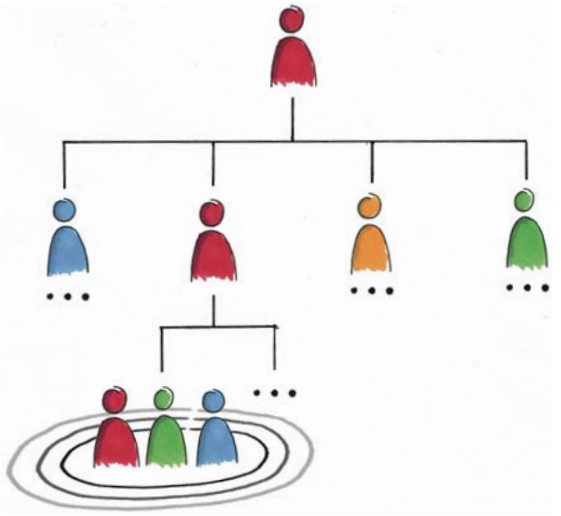

Abb. 7–4 *Die neue Kultur breitet sich schrittweise von Keimzellen her aus.*

Für dieses organische Ausbreiten von Keimzellen aus hat es sich bewährt, das Scrum-Pilotteam aufzuteilen und die entstandenen zwei oder drei Teams mit neuen Teammitgliedern aufzufüllen. Wenn diese neuen Teams erfolgreich agil gearbeitet und die neue Kultur verinnerlicht haben, werden sie nach dem gleichen Schema wieder aufgeteilt usw. (siehe Abb. 7–5). So wird die neue Arbeits- und Denkweise im persönlichen Arbeitskontakt der Teammitglieder transportiert.

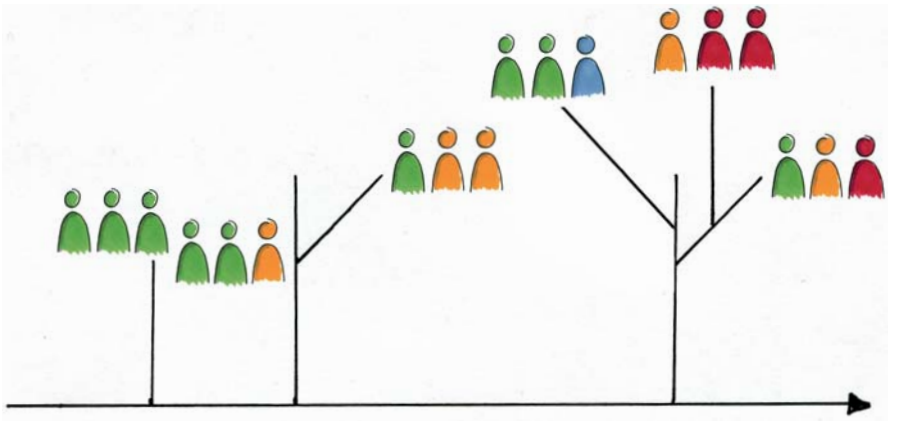

Abb. 7–5 *Organische Ausbreitung von Verhaltensweisen und Erfahrungen*

Für die Verbreitung der agilen Kultur eignen sich außerdem regelmäßige *Open-Space-Veranstaltungen* (siehe [Owen2008]) und *Communities of Practice* (siehe [LaveWenger1991]). Diese beiden Instrumente werden umso wichtiger, je stärker von der dargestellten organischen Ausbreitung durch personelle Rotation abgewichen wird.

Die Herausforderung der Transition besteht also nicht in den konkreten Praktiken für die Koordination mehrerer Teams etc., sondern in der Ausbreitung der agilen Kultur im Unternehmen.

7.1.3.1 Scrum Studio

Das *Scrum Studio* bietet dem Rest des Unternehmens an, Entwicklung nach Scrum durchzuführen (siehe [SchwaberSutherland2012]). Es stellt dazu Expertise in Form von Coaches und je nach Ausbaustufe auch Räumlichkeiten und agile Teams zur Verfügung.

Das Studio ist eine eigene Organisationseinheit mit eigenen Regeln und einer eigenen Kultur. Wer die Angebote des Studios nutzen will, schließt einen »Nutzungsvertrag« ab, mit dem die gegenseitigen Erwartungen und Verpflichtungen vereinbart werden. Hier wird z.B. festgelegt, welche Rechte und Pflichten ein Product Owner hat, der oder die über das Studio ein Projekt durchführen möchte.

Wer im Studio arbeitet, ist während dieser Zeit den Regelungen des Studios unterworfen und von den Regelungen des Unternehmens befreit. Mit diesen Vereinbarungen wird ein Adapter zwischen der neuen agilen Welt und der klassischen Welt im Unternehmen geschaffen.

7.1.3.2 Autonome Business Units

Der Ansatz der *autonomen Business Units* geht noch einen Schritt weiter als das Scrum Studio. Die autonomen Business Units sind direkt end-to-end für einen bestimmten Geschäftsbereich verantwortlich. Die Business Units sind dabei autonom gegenüber den anderen Business Units und der restlichen Hierarchie im Unternehmen. Die Business Units können selbst entscheiden, ob und welche zentralen Dienstleistungen sie vom Rest des Unternehmens in Anspruch nehmen. Dieser Ansatz mag radikal erscheinen, aber eine Firmenstruktur aus autonomen Business Units ist nicht so unüblich. Sie findet sich in mehreren Unternehmen (siehe z.B. [Semler2001], [Laloux2014]).

7.1.4 Agilität mit agilen Verfahren ausbreiten

Mit dem Wissen um den schrittweisen Kulturwandel müssen wir davon ausgehen, dass die Ausbreitung von Agilität immer wieder überraschende Effekte zeigen wird; schließlich geht es darum, die Verhaltensweisen und Denkmodelle von Menschen zu ändern. Wir können also nicht am Anfang den Endzustand festlegen, einen detaillierten Plan aufstellen und diesen dann umsetzen. Stattdessen benötigen wir ein Verfahren, das auf schrittweises Vorgehen setzt. Das Verfahren muss es erlauben, auf dem Weg zu lernen und das Gelernte in das weitere Vorgehen zu integrieren.

Unser Transitionsvorgehen (siehe Abb. 7–6) basiert auf unseren Erfahrungen mit unzähligen agilen Transitionen seit 2005. Wir haben gelernt, dass Organisationsveränderungen komplex sind und ein iteratives Vorgehen mit häufigem Inspect & Adapt brauchen.

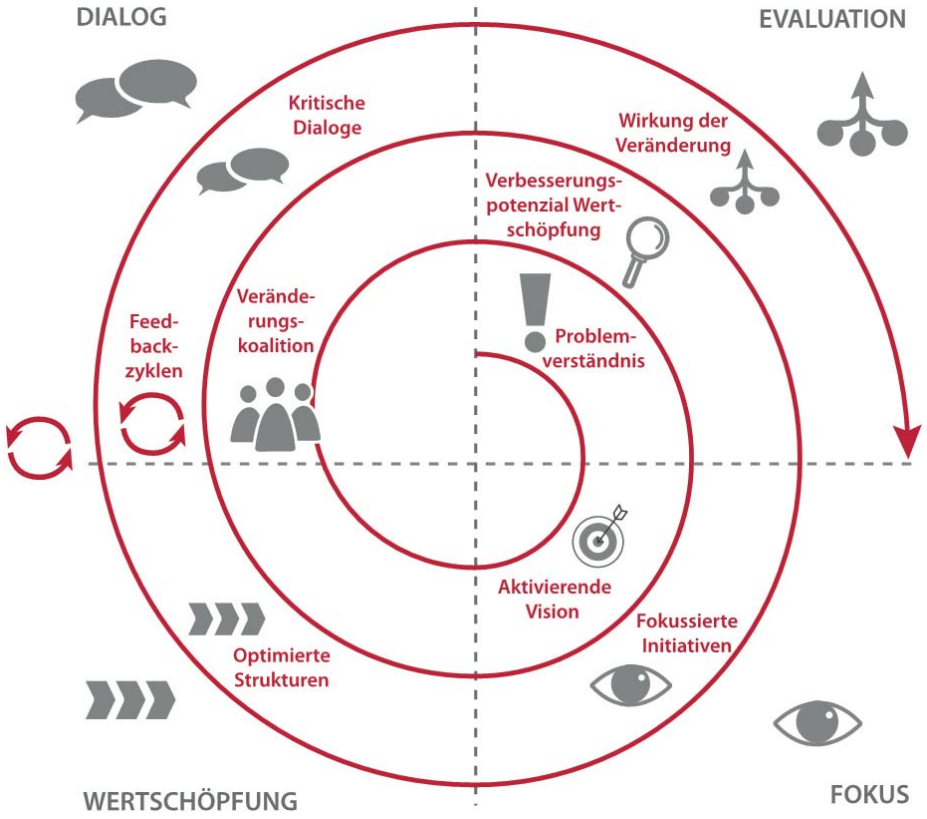

Abb. 7–6 *Transitionsvorgehen*

Jede Transition beginnt mit einem klaren und geteilten *Problemverständnis*. Nachdem klar gemacht ist, dass Veränderung notwendig ist, muss die Veränderungsenergie in eine gemeinsame Richtung gelenkt werden. Das passiert über eine *aktivierende Vision* der Zukunft. Ein gemeinsames Verständnis über das Problem zu schaffen und die Beteiligten und Betroffenen in Richtung der aktivierenden Vision zu führen, ist eine essenzielle Leadership-Aufgabe.

Eine *gestaltende Veränderungskoalition* sorgt dann dafür, dass sich auch tatsächlich etwas ändert. Sie besteht aus einer Gruppe von Menschen, die die Veränderung wollen und den Einfluss besitzen, sie Wirklichkeit werden zu lassen.

Und dann beginnt das eigentliche iterative Arbeiten. Wir suchen nach Chancen, wie Agilität *Wertschöpfung verbessern* kann. Passend zu den identifizierten Verbesserungspotenzialen werden *fokussierte Initiativen* gestartet, die daran arbeiten,

das Verbesserungspotenzial zu heben. Für jede Initiative ist eine kleine Gruppe von Menschen verantwortlich.

Im Rahmen der Initiativen werden *Strukturen für die Wertschöpfung optimiert* (z.B. werden neue agile Teams gebildet oder die Führungsstruktur angepasst) und *Feedbackzyklen* installiert. Die Feedbackzyklen sorgen dafür, dass die veränderten Strukturen kontinuierlich Rückmeldung über ihre Effektivität erhalten. Damit dieses Feedback sinnvoll in Aktionen überführt werden kann, braucht es Räume für *kritische Dialoge*. Wir sorgen dafür, dass die Menschen miteinander sprechen, die miteinander sprechen müssen, um Wertschöpfung für Kund:innen zu erzielen. Und natürlich wollen wir für jede Veränderungsinitiative wissen, ob sie erfolgreich war. Also evaluieren wir die *Wirkung der Veränderung* und passen auf dieser Basis ggf. die Initiative an.

Bei diesem Vorgehen wenden wir Wirkungs- und Veränderungsprinzipien an (siehe Abb. 7–7). Sie definieren, welche Haltung wir bei agilen Transitionen an den Tag legen.

Abb. 7–7 *Wirkungs- und Veränderungsprinzipien*

Die *Wirkungsprinzipien* geben uns Orientierung bei der Wirkung, die wir mit Agilität erreichen wollen: Agilität ist kein Selbstzweck und das Ziel einer agilen Transition besteht nicht darin, eine Reihe agiler Teams zu installieren. Stattdessen geht es darum, *Wirkung für Kund:innen und das eigene Unternehmen* zu schaffen: bessere Produkte, zufriedenere Kund:innen, höhere Umsätze etc. Dazu ist es in der heutigen Zeit zunehmender Dynamik notwendig, dass *unternehmerisches Verständnis, Denken und Handeln auf allen Ebenen* im Unternehmen gestärkt wird. Und nicht zuletzt erkennen wir an, dass Agilität nicht überall im Unternehmen die beste Wahl ist. Agile Arbeitsweisen spielen ihre Stärken dort aus, wo unter großer Unsicherheit Neues geschaffen wird. In anderen Bereichen sind Lean-Ansätze oder klassische plangetriebene Arbeitsweisen besser geeignet.

Die *Veränderungsprinzipien* geben uns Orientierung bei der Durchführung der Veränderung. Zunächst ist es nicht sinnvoll, das aktuelle Unternehmen mit allem, was daran erfolgreich war und ist, abzuwerten. Stattdessen wollen wir dort *starten, wo das Unternehmen steht*. Ausgehend davon arbeiten wir an *Strukturen, an der Kultur und an dem persönlichen Führungsstil* der Beteiligten. Eine agilere Organisation ist immer auch eine partizipativere Organisation. Das muss bereits in der Transition angelegt sein: Das *System muss angemessen involviert* werden. Wer wie stark an der Transition partizipiert, hängt wesentlich davon ab, wo das Unternehmen bezüglich Partizipation steht. Es ist ineffektiv, eine Transition mit größtmöglicher Partizipation aller Mitarbeitenden durchzuführen, wenn bisher wenig Partizipation vorhanden war. Und nicht zuletzt *verändern wir iterativ und inkrementell*. Unternehmen sind komplexe soziotechnische Systeme, bei denen man bei Veränderungen nicht sicher vorhersagen kann, welche Effekte sich einstellen. Daher brauchen wir kontinuierliches Inspizieren und Adaptieren auch bei der agilen Transition.

7.1.5 Globale Optimierung

Bei der umfassenden Einführung von Scrum kommt man letztlich nicht umhin, das Gesamtproblem in mehrere kleine Probleme zu zerlegen. Damit geht immer die Gefahr lokaler Optimierungen einher: Es wird auf ein lokales Kriterium hin optimiert, wobei das große Ganze Schaden nimmt.

Um lokale Optimierungen zu verhindern, ist *Transparenz in alle Richtungen* notwendig. Alle Beteiligten sollten Zugang zu allen Informationen haben, die helfen, bessere Entscheidungen zu treffen. Den so bereitgestellten Informationen muss Bedeutung verliehen werden. Die dazu notwendige Interpretationsarbeit sollte im *häufigen Face-to-Face-Kontakt* der Beteiligten stattfinden.

Führungskräfte müssen dafür sorgen, dass die notwendigen Informationen und Interpretationsfähigkeiten bei den Beteiligten vorhanden sind, und den Rahmen setzen (z. B. durch die Unternehmensidentität), den die Mitarbeitenden brauchen, um die Informationen sinnvoll interpretieren zu können.

7.1.6 Coaching

Wie oben beschrieben, senken erfahrene externe Coaches die Gefahr von Fehlschlägen. Viele Unternehmen fragen sich allerdings, ob sich die Investitionen in Coaching wirklich lohnen und wie man einen geeigneten Coach findet.

7.1.6.1 Ökonomie des Coachings

Zur Ökonomie des Coachings kann man grob folgende Überlegung anstellen: Ein mittelgroßes Team kostet das Unternehmen 25.000 Euro monatlich. Ein initiales Coaching könnte so aussehen, dass der externe Coach das Team drei Monate lang begleitet. Er oder sie betreut das Team zunächst vier Tage pro Woche, reduziert sein Engagement dann schrittweise über die drei Monate. Dadurch entstehen Kosten von ca. 35.000 Euro. Wenn der Coach eine Performance-Verbesserung des Teams von nur 20 % erreicht, hat sich die Investition nach sieben Monaten rentiert. In der Regel wird der Coach eine deutlich höhere Verbesserung für das Team erzielen.

Am besten definiert man vor dem Coaching zusammen mit dem Coach, welche Ziele erreicht werden sollen, wie die Zielerreichung gemessen wird und was man bereit ist, dafür auszugeben. Dann kann man kontinuierlich die Effektivität des Coachings messen und ggf. intervenieren.

7.1.7 Externe Coaches auswählen

Man sollte sich für eine externe Unterstützung nach Coaches umsehen, die nachweislich mehrere Jahre in unterschiedlichen Firmen agile Einführungen begleitet haben und die Coachingerfahrung (und vielleicht sogar eine Coachingausbildung) mitbringen. Natürlich muss der Coach sich beim Sponsor oder der Sponsorin der Scrum-Einführung sowie dem Team vorher vorstellen – es spielt eine wichtige Rolle, dass die Chemie stimmt.

In diesem Gespräch darf man ruhig auch nach den Fehlschlägen des Coaches fragen. Kein erfahrener Coach hat nur Erfolge gehabt. Wer das behauptet, hat entweder nur wenig Erfahrung oder er oder sie lügt. Jetzt ist es lehrreich zu beobachten, wie der Coach auf die Fehlschläge reagiert. War immer nur der Kunde oder die Kundin schuld, oder sieht diese Person sich auch in der Verantwortung?

7.1.8 Interne Coaches ausbilden

Wer eine größere Transition vor sich hat, sollte sich nicht dauerhaft von Externen abhängig machen. Es hat sich bewährt, interne Coaches aufzubauen. Das funktioniert am effektivsten dadurch, dass die externen Coaches einen internen Coach als »Schatten« zugeteilt bekommen und dieser »on the job« ausgebildet wird.

7.2 Scrum skalieren

»Scaling agile is the last thing you want to do.«

Martin Fowler

Eine umfangreiche Einführung von Scrum muss mit Teams beginnen, die *verlässlich lieferbare Produktinkremente* entwickeln. Solange diese Fähigkeit im Team nicht vorhanden ist, würde eine skalierte Einführung zu dem führen, was Jerry Weinberg als das erste Gesetz schlechten Managements charakterisiert (siehe [Weinberg1986]):

> »Wenn etwas nicht funktioniert, mach' mehr davon!«

Mit dem Agilen Manifest läuteten die Autoren nicht nur einen Wandel der Mechanik der Entwicklung ein (kurze Iterationen, die lauffähige Software erzeugen), sondern forderten auch grundsätzlich veränderte Verhaltensweisen bei den Beteiligten und damit einen *Kulturwandel* in den Unternehmen.

> Die primäre Herausforderung der agilen Skalierung liegt nicht in den konkreten Praktiken für die Koordination mehrerer Teams, sondern in der Ausbreitung der agilen Kultur.

Daher ist es wichtig, auch bei der Skalierung die agilen Werte und Prinzipien zu beachten. Die Skalierungsprinzipien (siehe [Skalierungsprinzipien2014]) geben spezielle Hinweise dazu, was bei der Skalierung beachtet werden sollte.

7.2.1 Der Agile Scaling Cycle

Neben der Kulturentwicklung muss die Frage geklärt werden, wie die Teams in einem Großprojekt organisiert und koordiniert werden. Dabei muss man darauf achten, dass die Agilität, die man auf Ebene eines Teams erreicht hat, nicht durch klassische hierarchische Steuerungsmechanismen wieder verloren geht. Die Antwort kann auch nicht darin bestehen, eine Blaupause zu kopieren: Schließlich liegt der agilen Entwicklung die Annahme zugrunde, dass sich die optimale Struktur schrittweise entwickeln muss, um den optimalen Nutzen zu gewährleisten.

Der *Agile Scaling Cycle* (siehe Abb. 7–8) setzt diese Denkweise in ein zyklisches Vorgehen mit drei Schritten um. Im Zentrum stehen die agilen Werte und Prinzipien, die wir bei jedem der drei Schritte als Kompass verwenden. Wir beginnen ein neues skaliertes Projekt damit, dass wir *Abhängigkeiten reduzieren*, soweit es möglich ist. Anschließend arbeiten wir im Projekt und *koordinieren die Teams* bezüglich der verbliebenen fachlichen und technischen Abhängigkeiten. Auf Basis der im Projekt gewonnenen Erkenntnisse *entwickeln wir die Organisa-*

tion weiter, sodass wir im nächsten Zyklus des Agile Scaling Cycle mehr Optionen zur Reduktion von Abhängigkeiten haben. Für die Weiterentwicklung der Organisation zeichnet das oben beschriebene Transitionsteam verantwortlich.

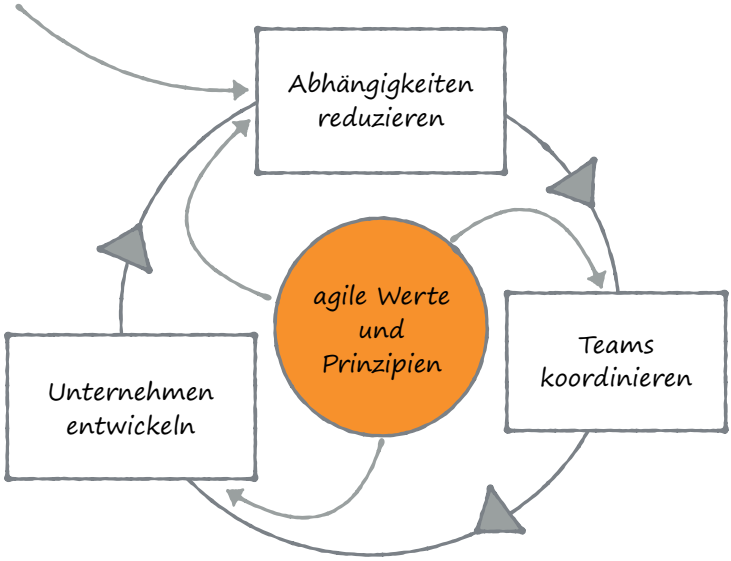

Abb. 7–8 *Agile Scaling Cycle*

Sowohl für die Reduktion von Abhängigkeiten wie auch für die Koordination der Teams stehen Dutzende von Praktiken zur Verfügung, aus denen sich die Teams anfänglich bedienen können. Und je häufiger sie den Agile Scaling Cycle durchlaufen, desto mehr eigene Praktiken werden sich entwickeln, die immer besser an den eigenen Kontext angepasst sind.

Ein Durchlauf durch den Agile Scaling Cycle kann beispielsweise so aussehen:

Beispiel: Abhängigkeiten reduzieren

Die Sprints der Teams laufen synchronisiert ab, sodass sie gleichzeitig beginnen und enden. Dadurch wird es einfacher, regelmäßig lauffähige Produktversionen sicherzustellen.

Wir wählen eine auf *Verticals* basierende Softwarearchitektur, die den Teams maximale Autonomie gewährt und dafür ein Stück weit Code- und Datenredundanz in Kauf nimmt. Die Teams selbst setzen wir cross-funktional zusammen und geben ihnen End-to-End-Verantwortung für die Entwicklung businessrelevanter Features. Allerdings erlaubt der Kontext im Moment vielleicht noch keine Integrationstests im Rahmen der Sprints, sodass im ersten Sprint mit einer suboptimalen Definition of Done gearbeitet wird.

Beispiel: Teams koordinieren

Für die fachliche Koordination der Teams definieren wir einen Product Owner für das Gesamtprodukt, der oder die durch Priorisierung der Features den Produktnutzen optimiert und das Product Backlog verantwortet. Die einzelnen Teams entwickeln Features aus dem Product Backlog heraus.

Die Sprint Plannings der einzelnen Teams finden gleichzeitig statt, sodass die Teams selbstorganisiert technische Abhängigkeiten entdecken und geeignete Maßnahmen vereinbaren können. Dazu kann z.B. ein *Scrum of Scrums* gehören, in dem sich die Teams zu den Features austauschen können, die sie gemeinsam entwickeln.

Das Sprint-Review findet ebenfalls wieder gemeinsam statt – schließlich ist das Ergebnis ein *gemeinsames Produktinkrement*. Im Sprint-Review wird das gemeinsame Ergebnis gezeigt und darauf hingewiesen, dass dieses nicht lieferbar ist – schließlich fehlen die Integrationstests. Es wird sichtbar, dass hier ein relevantes Risiko liegt, weil niemand eine belastbare Abschätzung geben kann, wie viel Zeit der abschließende Integrationstest inkl. Bugfixing benötigen wird. Also wird das Fehlen von Integrationstests als organisatorisches Hindernis vermerkt.

Beispiel: Organisation entwickeln

Das organisatorische Hindernis, dass die aufgeschobenen Integrationstests ein großes Risiko darstellen, geht an das Transitionsteam. Es arbeitet daran, dieses Hindernis zu beseitigen. Dazu verschafft es sich Klarheit über das Problem, indem es mit verschiedenen Personen zu dem Thema spricht. Dabei wird klar, dass die Integrationstests in den Teams nicht möglich sind, weil Testsysteme der notwendigen Drittsysteme nur einmal im Unternehmen verfügbar sind und die verschiedenen Projekte sich in die Quere kämen, wenn alle ständig auf die Testumgebung zugreifen würden. Dass die Testdrittsysteme nur einmal vorhanden sind, liegt daran, dass es die benötigte Hardware nur einmal gibt und dass der Betrieb keinen Freiraum hat, weitere Testsysteme zu betreuen.

Das Transitionsteam veranlasst daraufhin die Beschaffung zusätzlicher Hardware und setzt ein Ausbildungsprogramm auf, um zu erreichen, dass die Teams ihre Testdrittsysteme selbst installieren und betreiben können. Dazu wird ein Pairing mit den Leuten aus dem Betrieb eingeführt. Die dazu notwendige Zeit der Leute aus dem Betrieb geht zulasten des Service für den Rest des Unternehmens. Das Transitionsteam hilft dabei, diese Maßnahme trotzdem umzusetzen, und macht im Unternehmen deutlich, dass es sich um eine Investition handelt, die jetzt getätigt wird und sich in Zukunft auszahlen wird.

Mit der Umsetzung dieser Maßnahme geht es in den nächsten Zyklus des Agile Scaling Cycle. Wichtige Abhängigkeiten von beschränkter Hardware und von der Betriebsabteilung konnten reduziert werden, sodass die Definition of Done erweitert und die Projektplanung für den Product Owner einfacher und verlässlicher wird.

7.2.2 Praktiken zur Reduktion von Abhängigkeiten

Die Praktiken zur Reduktion von Abhängigkeiten lassen sich entlang der zwei Dimensionen »Autonomie« und »Abhängigkeitsebene« differenzieren (siehe Abb. 7–9). So reduzieren synchronisierte Sprints Abhängigkeiten, weil alle Teams gleichzeitig starten und enden. Die Teams brauchen keinen klassischen Projektplan, um zu planen, wer wann fertig sein wird. Continuous Delivery könnte eine Alternative darstellen, weil jedes Team ständig fertig ist und nicht mehr alle Teams auf denselben Endtermin hinarbeiten müssen.

Die mächtigste Praktik zur Reduktion von Abhängigkeiten ist allerdings die Arbeit mit cross-funktionalen Featureteams, die businessrelevante Features end-to-end autonom umsetzen.

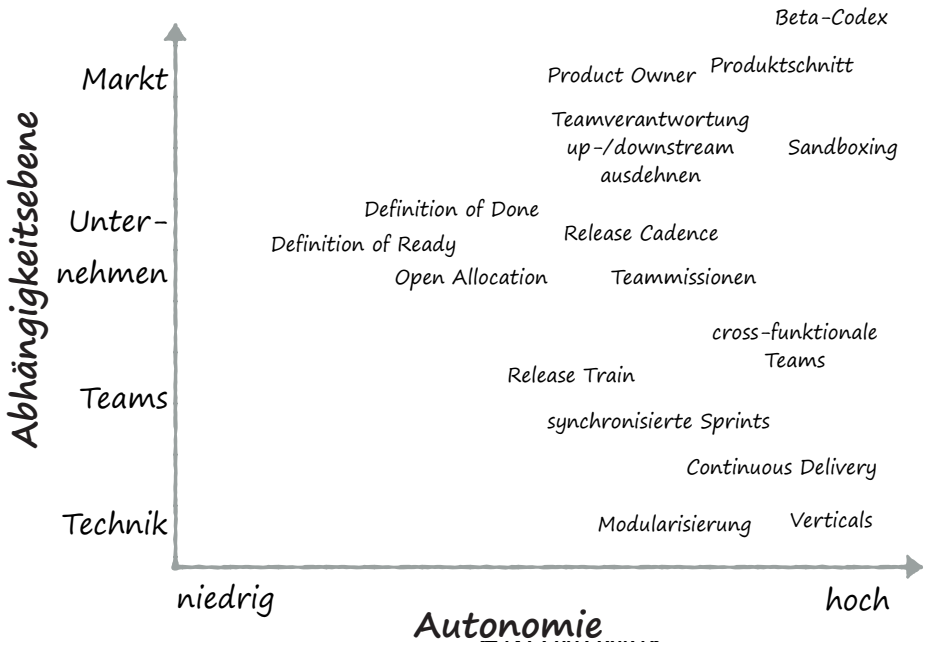

Abb. 7–9 *Beispielhafte Praktiken zur Reduktion von Abhängigkeiten*

7.2.3 Praktiken zur Koordination von Teams

Die Praktiken zur Koordination der Teams lassen sich entlang der zwei Dimensionen »Zeitlicher Horizont« und »Anzahl Teams« verorten (siehe Abb. 7–10). Ein *Scrum of Scrums* funktioniert bis zu einer mittleren Anzahl von Teams, und die damit einhergehende Koordination erstreckt sich in der Regel auf ein bis zwei Tage. *Portfolio-Kanban* adressiert hingegen einen Planungszeitraum von Monaten und funktioniert auch noch für eine größere Anzahl von Teams.

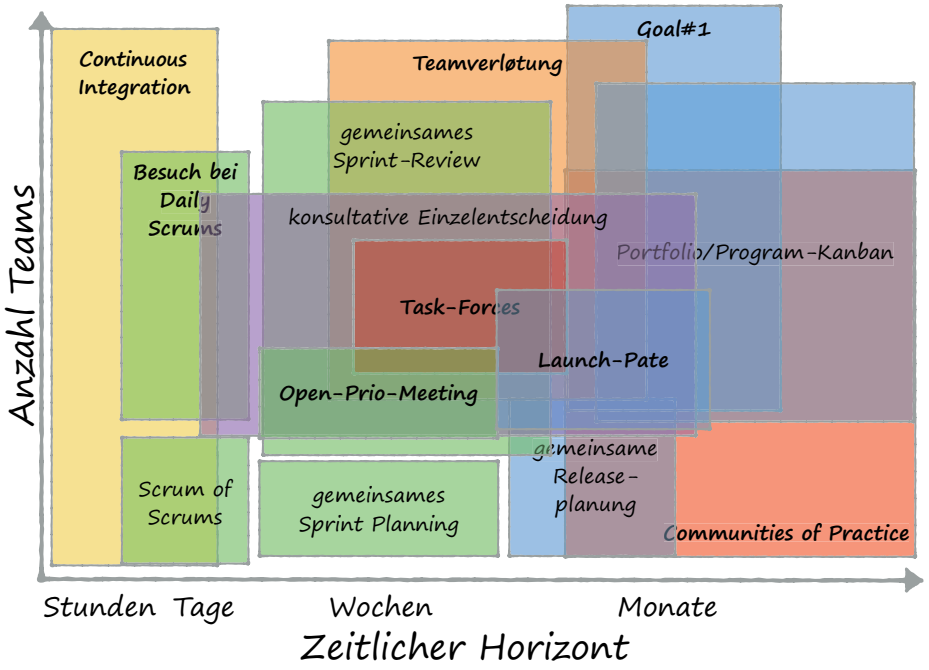

Abb. 7–10 *Beispielhafte Praktiken zur Koordination von Teams*

Generell versuchen wir, die Abhängigkeiten zwischen den Teams zu minimieren. Je erfolgreicher wir damit sind, desto weniger Praktiken zur Koordination benötigen wir. Als Erfolgsmetrik könnte man zählen, wie viele Koordinationspraktiken nicht (mehr) benötigt werden.

7.2.4 Die Organisation entwickeln

Bei der Arbeit der Teams werden organisatorische Hindernisse sichtbar werden. Nicht alle organisatorischen Hindernisse können von den Teams bzw. deren Scrum Mastern beseitigt werden. Diese Hindernisse gehen an das Transitionsteam, das die Aufgabe hat, die Skalierung zu begleiten. (Details zu Transitionen finden sich in Abschnitt 7.1.4)

7.3 Agile Unternehmen

Ein agiles Unternehmen passt sich flexibel an die Bedürfnisse seiner Umwelt (insbesondere des Marktes) an. Eine Menge agiler Teams führt damit nicht automatisch zu einem agilen Unternehmen. Zuschnitt, Anzahl und Interaktion der Teams müssen sich selbst an die Bedürfnisse der Umwelt anpassen, ohne dass eine zentrale Reorganisation notwendig ist.

Diese Flexibilität entsteht, wenn Struktur, Kultur und Führung sich ändern. Wenn ein klassisches Unternehmen sich in ein agiles Unternehmen wandeln möchte, kann über kurz oder lang kein Stein auf dem anderen bleiben. Dabei ist die Einzigartigkeit jedes Unternehmens zu berücksichtigen: Es gibt keine Blaupause für das agile Unternehmen. Jedes Unternehmen muss seinen eigenen Weg finden und sich schrittweise agiler gestalten.

Das Vorgehen gleicht dem oben genannten Agile Scaling Cycle – ausgehend von konkreten Hindernissen wird das Unternehmen weiterentwickelt in Richtung weniger Struktur und mehr Vertrauen (siehe Abb. 7–11).

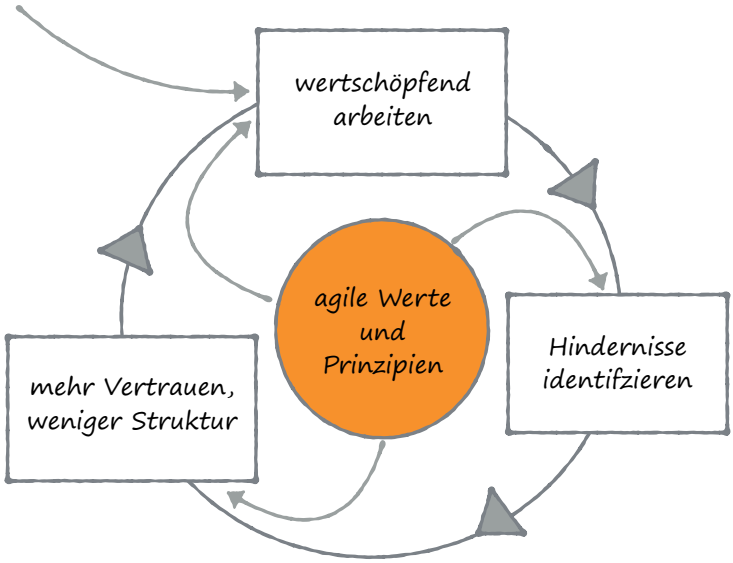

Abb. 7–11 *Iterative Unternehmensentwicklung*

Das Buch »Agile Unternehmen« beschreibt, wie agile Denk- und Arbeitsweisen auf ganze Unternehmen angewendet werden können (siehe [HoffmannRoock2018]).