3. Auflage

Robert Wiechmann · Sven Röpstorff

Scrum in der Praxis

Erfahrungen, Problemfelder und Erfolgsfaktoren

→ Mit einem Geleitwort von Ralf Kruse

dpunkt.verlag

Inhalt

4 Die Arbeit im Scrum-Team

Bevor ein Scrum-Team mit der Entwicklung starten kann, sind im Vorfeld neben der Zusammenstellung des Teams noch zahlreiche weitere Fragen zu klären. Vieles spielt sich dabei vor allem im Aktionsbereich des Scrum Masters und Product Owners ab, die einen sehr wichtigen Beitrag für einen vernünftigen Start leisten. Wir haben die Erfahrung in der Praxis gemacht, dass nicht nur der Beginn, sondern auch die weitere Durchführung eines Projekts im Wesentlichen von dieser soliden Vorbereitung abhängen. Für den Start wählen wir daher die Form eines Kick-offs (siehe Abschnitt 4.1), der für die umfängliche Einstimmung auf das Projekt sehr wertvoll ist und in dem auch ein paar Grundlagen für die gemeinsame Arbeit festgelegt werden. Auf diese Vorphase gehen viele agile Methoden und auch der Scrum Guide nicht ein. Sie erhalten in diesem Buchabschnitt einige Ideen und praktische Hinweise, um Scrum-Teams für die bevorstehenden Aufgaben vorzubereiten.

Nachdem wir uns diesem nicht offiziellen Scrum-Bestandteil gewidmet haben, werden wir genauer auf die Arbeit mit dem Product Backlog eingehen (siehe Abschnitt 4.2). Es geht uns hier vor allem um die praktische Arbeit, vom strukturellen Aufbau bis hin zur Arbeit mit Bedürfnissen von Anwendern (User Stories).

Sobald sich das Product Backlog langsam füllt, ist meist eine Einschätzung bezüglich der Komplexität der anstehenden Anforderungen gefragt. Daher tauchen wir in Abschnitt 4.3 in das Themengebiet Schätzungen ein und beleuchten es genauer. In diesem Teil liegt der Fokus auf unterschiedlichen Schätzverfahren, die sowohl zu Beginn als auch während eines laufenden Projekts angewandt werden können. Zudem werden zahlreiche Tipps gegeben, genügend aussagekräftige Schätzungen zu erzielen. Um überhaupt Aussagen über komplexe Anforderungen und Problemstellungen machen zu können, führen wir mit unserem Scrum-Team ein Backlog Refinement durch (siehe Abschnitt 4.4). Mithilfe dieser sich wiederholenden Arbeitstermine werden die

Details von Anforderungen besprochen und das Product Backlog sowie der Releaseplan angepasst.

Ebenso wie das Backlog Refinement kein offizielles Scrum-Event ist, ist auch der Releaseplan kein offizielles Artefakt. Wir stellen Ihnen in Abschnitt 4.5 die Releaseplanung zu Beginn und während des laufenden Projekts vor.

4.1 Vor dem Start

Bitte anschnallen und die Sicherheitsgurte schließen!

In zwei Wochen sollte es endlich losgehen. Casper, Finn, Sergio und Jordi waren bereits vor Ort, Lara war momentan noch in einem anderen Projekt gebunden. Alva und Mina würden erst zwei Tage vor dem eigentlichen Start zum Team stoßen, um an der initialen Schätzung des Product Backlog teilzunehmen.

Um sicherzustellen, dass die Entwickler am ersten Tag des ersten Sprints wirklich mit der Implementierung starten können, trafen sich alle bereits Anwesenden und überlegten sich, welche Voraussetzungen für einen guten Start in den ersten Sprint erfüllt sein müssten. Dabei kamen erstaunlich viele Dinge ans Licht, um die sie sich in den folgenden zwei Wochen kümmern wollten. Das Spektrum reichte vom Zugang zum Gebäude über die Ausstattung des Teamraums und der Arbeitsplätze bis hin zu den Entwicklungsumgebungen und Serverlandschaften.

Die Anwesenden erstellten ein simples →Scrum-Board an der Wand und schrieben für alle zu erledigenden Dinge Taskkarten. Das Team traf sich ab sofort täglich vor dem Scrum-Board zu einem Daily Scrum und arbeitete die Punkte sukzessive ab. Als Alva und Mina später dazustießen, sprachen sie den anderen ein großes Lob aus: Sie seien noch nie in ein so gut vorbereitetes Team gekommen.

Wenn ein neues Projekt beginnt, muss sich ein Scrum-Team die Projektumgebung oft selbst organisieren oder doch zumindest sicherstellen, dass sie den eigenen Ansprüchen genügt. Gerade bei Scrum-Projekten soll das Scrum-Team gleich mit dem ersten Sprint Funktionalität implementieren und sich nicht noch mit organisatorischen Herausforderungen herumschlagen müssen, die vorher hätten erledigt werden können. Trotzdem sind wir immer wieder überrascht, wie schlecht vorbereitet manche Teams in ihren ersten Sprint geschickt werden. Dabei ist es doch ganz einfach – es wird vor dem ersten

Sprint etwas Zeit für diese Tätigkeiten eingeplant. Zum Üben kann diese vorgelagerte Zeit sogar wie ein Sprint organisiert werden.

In manchen Organisationen wird diese Phase auch als »Sprint Zero« bezeichnet. Wir verzichten hier auf diese Bezeichnung, da es sie im Scrum Guide nicht gibt. Der Scrum Guide sieht diese Arbeiten als Bestandteil des Sprints. Wie auch immer man die Phase nennt, wir halten sie in einigen Fällen durchaus für sinnvoll, z.B. wenn es darum geht, die Arbeitsumgebung so einzurichten, dass ein Scrum-Team schnellstmöglich produktiv arbeiten kann.

In dieser Phase wird die Umgebung und der erste Sprint so vorbereitet, dass man mit dem Startschuss zu Sprint 1 mit der Implementierung von Funktionalität beginnen kann. So vermeidet man die üblichen Herausforderungen, beispielsweise dass man keinen Zugang zum Entwicklungsserver hat, dass ein Entwickler gerne einen zusätzlichen Monitor hätte, dass die Stühle im Projektraum Rückenschmerzen verursachen usw. Je nach Unternehmen und technischem Umfeld sieht diese Phase jedes Mal etwas anders aus.

Auch ein Kick-off-Workshop (vgl. Abschnitt 4.1.1) oder Anforderungsworkshops (vgl. Abschnitt 4.2.2) können gemeinsam in diesem Zeitraum organisiert werden, sofern diese Themen nicht schon vom Scrum Master angegangen wurden.

Obwohl es nun strammen Schrittes auf den tatsächlichen Entwicklungsstart zugeht, gilt es noch ein paar Voraussetzungen zu schaffen, damit man am ersten Tag des ersten Sprints auch tatsächlich mit dem Sprint Planning starten kann.

4.1.1 Kick-off

Der Startschuss fällt

Alle waren schon gespannt auf diesen Tag, auf den sie seit Bekanntwerden des Projekts gewartet hatten. Er sollte mit einem Kick-off-Event starten, für das Finn ein gemeinsames

Frühstück mit allen Teilnehmenden organisiert hatte. Neben dem Scrum-Team hatte er auch in Absprache mit Casper eine Reihe weiterer am Projekt beteiligter Personen aus dem Management und aus den Abteilungen eingeladen.

Nachdem alle genügend Zeit hatten, sich zu stärken und sich kennenzulernen, trommelte Finn alle zum Start des Workshops zusammen. Nach der Vorstellung der Agenda durch Finn und der Vereinbarung der Verhaltensregeln während des Kickoffs kam Casper an die Reihe und begann mit der Vorstellung seines Produktziels.

Casper erzählte eine Geschichte einer Konferenzteilnehmerin, die während ihres Konferenzbesuchs auf gewisse Hürden bei der Auswahl und Bewertung von Vorträgen stieß. Casper goss diese Geschichte unter Zuhilfenahme von Nutzerbefragungen, Auswertungen vergangener Konferenzen und tollen Produktideen in ein motivierendes Produktziel. Alle klatschten begeistert nach seinem halbstündigen Vortrag.

Nachdem erste Fragen geklärt worden waren, war Finn an der Reihe. Er sprach im Detail über das Projektumfeld, die Projektbeteiligten und ihre Verantwortlichkeiten. Er informierte über die Arbeit des Teams mit Scrum im Allgemeinen und die Bedeutung für alle Beteiligten im Besonderen. Vor allem war ihm dabei wichtig, Herrn Hold als ihren Auftraggeber zu überzeugen und sich seiner Unterstützung für das Scrum-Team gewiss zu sein. Zudem wollte er mit dem umfassenden Bild auch diejenigen aus der Firma gewinnen, die bisher eher mit Argwohn auf die Entwicklungsabteilung blickten.

Nach ca. drei Stunden, in denen alle Zeit hatten, sich kennenzulernen, die mobile Konferenz-App nun schon vor ihrem geistigen Auge klarer sehen konnten und ein Ausblick auf die kommenden Schritte gegeben wurde, trennte man sich dankend von allen, die nicht zum Scrum-Team gehörten. Nach einer gemeinsamen Mittagspause ging es nun im zweiten Teil des Tages um die Besprechung der teaminternen Aspekte. Finn hatte sich gut vorbereitet, und die Liste der zu besprechenden Themen war lang …

Auch ein Scrum-Team sollte immer mit einem Kick-off-Workshop starten. Vom Scrum Master ist abzuwägen, in welcher Tiefe und Breite der Termin durchgeführt wird. Dies hängt z.B. von den Rahmenbedingungen des Projekts ab: Wie tiefgehend ist das Verständnis von Agilität bei den Beteiligten und im Unternehmen verankert, wie hoch ist der Professionalisierungsgrad des Scrum-Teams in puncto Scrum oder fachlicher Qualifikation, wie prestigeträchtig oder wichtig ist das Projekt für das Management bzw. Unternehmen oder wie zeitlich aufwendig ist das Unterfangen? All diese Punkte sind mit in Betracht zu ziehen, wenn es um die Ausgestaltung eines Kick-offs geht.

Es ist nicht ungewöhnlich, den Workshop durch verschiedene Teil-Workshops vorzubereiten. Denkbar sind eine Scrum-Schulung für die Teammitglieder und Stakeholder oder die gemeinsame Entwicklung des Produktziels. Dafür bietet sich die Projektvorphase (siehe Abschnitt 4.1) an. Ein Kick-off-Termin kann von einem halben Tag bis zu einer Woche oder länger dauern [LarsenNies 2012].

Bereiten Sie den Kick-off gut vor und überlegen Sie sich den Ablauf und gewünschten Kreis der Teilnehmenden frühzeitig. Sammeln Sie zu klärende und festzulegende Punkte mit den Stakeholdern und den Teammitgliedern ein und verschicken Sie diese am besten mit der Agenda, damit sich alle im Vorfeld eine Meinung bilden können.

Produktziel

Essenziell für das Kick-off ist das Vorhandensein des Produktziels. Wie Stephen Covey in seinem Buch [Covey 2004] beschreibt: »Begin with the end in mind«, sollte man immer mit einer klaren Vorstellung von dem starten, was am Ende erreicht werden soll. Roman Pichler schlägt als Einstieg in ein Projekt die Erstellung eines Produktkonzepts vor, in dem die Bedürfnisse des Kunden, die Herausstellungsmerkmale und Ziele beschrieben werden [Pichler 2008]. Aus einem solchen Konzept ist die Ableitung eines Produktziels leicht möglich, da die wichtigsten Komponenten genau unter die Lupe genommen werden. Es gibt diverse Ansätze, ein Produktziel zu entwickeln, wie Roman Pichler oder Boris Gloger in ihren Büchern berichten [Pichler 2008; Pichler 2014; Gloger 2008].

Abb. 4–1 *Das Produktziel immer im Fokus*

Das Produktziel bildet für alle Beteiligten eine Orientierung und fokussiert das Team, um sich auf das Wesentliche zu besinnen. Das Produktziel gibt den Teammitgliedern einen Grund, ihr Commitment ganz dem Erreichen des Ziels zu widmen und gemeinschaftlich zusammenzuwirken.

Unterstützen Sie als Scrum Master den Product Owner so, dass er gut vorbereitet in das Kick-off geht und alle Anwesenden von seinem Produktziel zu überzeugen weiß. Sie verpassen viel, wenn Sie die Menschen in diesem Moment nicht für das Projekt begeistern können.

Erarbeiten Sie zusammen mit dem Scrum-Team das Produktziel oder lassen Sie den Product Owner es im Vorfeld teamintern präsentieren und mit den Entwicklern diskutieren bzw. verfeinern.

Impact Mapping

Ein hilfreiches Mittel, um zu einem Produktziel zu gelangen, ist die Erstellung einer →Impact Map. Diese Mindmap, die gemeinsam durch das Scrum-Team und die Stakeholder erstellt wird, ist ein erster guter Schritt, um ein gemeinsames Verständnis vom Endergebnis zu erhalten.

Scrum Master und Product Owner können so mit einem gut vorbereiteten Workshop die folgenden Fragen klären:

Warum führen wir das Projekt durch? (Produktziel)

Wer wird durch das Projekt beeinflusst? (Akteure)

Wie wirkt sich das Produktziel auf die Verhaltensweisen der Akteure aus? (Auswirkung, engl. Impact)

Was müssen wir entwickeln, um die gewünschten Verhaltensweisen zu erreichen? (Maßnahmen)

In einer Impact Map werden die einzelnen ermittelten Funktionen mit dem Produktziel verknüpft. Die Übersicht hilft dabei, ein gemeinsames fachliches Verständnis aller Beteiligten zu erreichen und einen Überblick über das zu entwickelnde Produkt zu bekommen.

Impact Mapping dient bei neuen Projekten zum Sammeln von Ideen, die dann vom Product Owner in eine Reihenfolge gebracht und in Backlog Items überführt werden. Normalerweise beginnt dieser Prozess mit dem *Warum?* und endet beim *Was?*, den einzelnen Features. Sollte jedoch bereits eine Ansammlung an Ideen oder Feature-Wünschen vorhanden sein, wird der Prozess der Erarbeitung umgedreht. Für jedes Feature wird dann die Frage beantwortet: »*Wenn dieses Feature umgesetzt wird, welches Verhalten wird dann bei den Akteuren (Wer?) verändert (Wie?)?*« und: »*Warum ist diese Verhaltensänderung wichtig?« (Warum?)*.

Das Ergebnis und die gewonnenen Schlussfolgerungen einer Impact Map können am Ende in eine →Story Map übertragen werden (vgl. Abschnitt 4.2.4).

Wenn Sie mehr zu Impact Mapping erfahren möchten, empfehlen wir Ihnen das Video vom Erfinder Gojko Adzic zum Thema [URL:Adzic].

Ziele

Egal wie das Projekt-Setup aussieht, Ziel des Workshops ist es, dass am Ende alle Beteiligten die gleiche Sprache sprechen und Regeln sowie

Erwartungshaltungen klar kommuniziert sind. An dem Termin sollten alle teilnehmen, die etwas mit der Ausführung zu tun haben oder in einer anderen Art und Weise involviert sind. Im Vordergrund stehen dabei für:

- **das Scrum-Team**
  - das Verständnis des Produktziels, der Anforderungen, Risiken, Teamregeln und der Rahmenbedingungen und für
- **die Stakeholder**
  - alle notwendigen Informationen, um daraus Entscheidungen zum Projekt ableiten und treffen zu können.

Hinzu kommt, dass sich alle Projektbeteiligten kennenlernen und der Termin einen Impuls für den Start des Projekts setzt.

Neben diesem informellen Teil ergänzen wir das Kick-off um einen praktischen Teil, in dem wir beispielsweise Artefakte erarbeiten und Arbeitsvereinbarungen treffen.

Wenn Sie als Scrum Master ein Scrum-Team komplett neu übernehmen, lernen Sie Ihr Team und die Anforderungen kennen, die es an ein Scrum-Projekt hat. Setzen Sie sich mit den Teammitgliedern zusammen und klären Sie, was für sie ein erfolgreiches Projekt ausmacht. Durch den gemeinsamen Austausch erlangen Sie nicht nur ein gutes Gespür dafür, wo Sie auf dem gleichen Nenner liegen, sondern auch, wo es noch Klärungsbedarf gibt.

Vorbereitung

Um richtig in die Vorbereitung einsteigen zu können, gibt es viele Fragen für die Planung zu klären. Voraussetzung, dass überhaupt darüber nachgedacht wird, ein Kick-off durchzuführen, ist laut Diana Larsen und Ainsley Nies [LarsenNies 2012] die Klärung, dass es

- einen Projektsponsor,
- einen Product Owner,
- ein Geschäftsszenario, das benannt werden kann,
- ein Projektbudget sowie
- eine klare Absicht gibt, was man mit dem Projekt erreichen möchte.

Um den Workshop inhaltlich gut vorbereiten und ausgestalten zu können, sollte sich ein Product Owner auf folgende Schwerpunkte konzentrieren:

- das Produktziel,
- das initiale Product Backlog,
- die Erstellung eines ersten groben Plans,
- Markt- und Konkurrenzkennzahlen,
- Nutzerstatistiken, Umfrage- oder Testergebnisse und
- Informationen von Kunden- oder Auftraggeberseite.

Diese Informationen und die Auseinandersetzung damit sollten vor dem ersten Handschlag das Hauptaugenmerk erhalten und können schon tiefer gehend vorbereitet werden. Ein Product Owner wird in den meisten Fällen mit dem Projektsponsor oder anderen Entscheidungsträgern offene Fragen klären und alle essenziellen Fragen zum Produkt vorbereitend bearbeiten.

Wir möchten bei dieser Gelegenheit auf einige Erkenntnisse hinweisen, die oftmals zu Beginn von Scrum-Projekten noch im Raum stehen und die Jonathan Rasmussen auf den Punkt gebracht hat [Rasmussen 2010]:

> *Es ist nicht* möglich, *alle Anforderungen zu Beginn eines Projekts zu sammeln*.
>
> *Alle Anforderungen, die gesammelt werden, werden sich garantiert ändern*.
>
> *Es wird immer mehr zu tun sein, als es Zeit und Geld erlauben werden*.

Die Akzeptanz dieser drei Punkte, die zudem eine direkte Ableitung aus dem Agilen Manifest sind (vgl. Abschnitt 2.2), fällt vielen in der Praxis schwer. Gerade in der Vereinbarung mit dem Management, Vertrieb, Marketing oder der Kundschaft dreht es sich oftmals darum, vorab alles wissen zu wollen und terminlich zu vereinbaren. Dies ist jedoch illusorisch. Daher zielt der zweite Punkt auf die laufende Anpassung des Plans ab, während die Anforderungen Stück für Stück ermittelt werden, um so schrittweise ein klareres Bild zu erhalten (siehe Abschnitt 4.5.1). Scrum hilft uns auch beim dritten Punkt, indem zuerst das Wichtigste gemäß Wertschöpfung umgesetzt wird. Dafür gibt es eine laufende Sortierung, Überprüfung und Ergänzung der Produktanforderungen. Neben dem Product Owner hat auch ein Scrum Master in diesem Stadium des Projekts alle Hände voll zu tun. Seine Aufgabe ist es, sich um jegliche organisatorischen und teamrelevanten Aspekte zu kümmern. Darunter fallen u.a. Aufgaben wie:

- Unterstützung des Product Owners bei der Vorbereitung,
- Schulung des Scrum-Teams und ggf. der Stakeholder,
- Planung und Vorbereitung des Kick-off-Workshops,
- Klärung bekannter organisatorischer Hürden (z.B. Räumlichkeiten, Lizenzen, Arbeitsmaterialien),
- Führen von Bewerbungsgesprächen.

Rückt der Workshop näher, sollte frühzeitig mit der inhaltlichen Ausgestaltung des Termins durch den Scrum Master begonnen werden.

Inhalt

Wie oben erwähnt ist der Umfang der zu behandelnden Themen groß und kann sich über mehrere Tage oder Einzeltermine erstrecken. In unserer Praxis hat sich in den meisten Fällen eine Dreiteilung eines eintägigen Workshops bewährt:

- **Projektspezifisches**
  Scrum-Team und Stakeholder erfahren alles Notwendige zum Projekt.
- **Organisatorisches**
  Offene Fragen werden festgehalten und es folgt ein Ausblick auf die nächsten Schritte mit Stakeholdern und Scrum-Team.
- **Teamspezifisches**
  Das Scrum-Team klärt und definiert unter sich wichtige Aspekte der Zusammenarbeit.

In einem Kick-off könnten dementsprechend die projektspezifischen und organisatorischen Themen am Vormittag besprochen werden und am Nachmittag könnten sich die Teammitglieder dediziert den teamspezifischen Themen widmen. Somit lernen sich am Vormittag alle direkt und indirekt Beteiligten kennen und erhalten die wichtigsten Informationen zum Projekt. Zu beachten ist für die Ausgestaltung des Termins, dass Prioritäten gesetzt werden und ein roter Faden vorhanden ist.

Achten Sie darauf, die Agenda nicht zu detailliert auszuarbeiten und Platz für das Eintauchen in Themenbereiche zu lassen, die das Team für wichtig hält. Die Agenda ist somit nur ein Vorschlag eines roten Fadens, der innerhalb des Kick-offs im offenen Austausch gefüllt wird. Planen Sie großzügig unter Berücksichtigung zeitlicher Freiräume.

Projektspezifische Themen

In diesem ersten Teil des Kick-offs werden alle relevanten Personen, die mit dem Projekt zu tun haben, über das Projekt und die vorhandenen Rahmenbedingungen informiert. Zudem dient dieser Teil dazu, offene Fragen zu klären oder zumindest zu formulieren, um diese im Nachgang anzugehen.

Jonathan Rasmussen stellt in seinem Buch zehn Fragen für den Einstieg in ein agiles Projekt vor, die innerhalb eines Kick-offs geklärt werden sollten [Rasmussen 2010]. Diese Fragen sollten innerhalb des Termins nach aktuellem Kenntnisstand bestmöglich beantwortet werden:

- **Warum sind wir hier?**
  Diese Frage bündelt die Informationen zum Projekt, klärt die Notwendigkeit des Projekts und geht auf das Team und die Stakeholder ein.
- **Wie würden wir in 30 Sekunden das Projekt beschreiben?**
  Im sogenannten →»Elevator Pitch« wird Klarheit durch den Fokus auf die wesentlichen Informationen geschaffen.
- **Wie sieht unser Produkt in einer Werbeanzeige aus?**
  Hier lässt Rasmussen anhand einer »Product Box« das zukünftige Produkt durch die Beteiligten kreativ entwickeln.
- **Was werden wir nicht tun?**
  Um herauszufinden, was Bestandteil des Projekts ist, ist es nützlich, sich anzusehen, was kein Bestandteil des Projekts ist.
- **Wer sind die Projektbeteiligten?**
  Offensichtlich gehört das Scrum-Team dazu, doch wer sind die weiteren Stakeholder?
- **Wie sieht die technische Lösung aus?**
  Wenn schon darüber nachgedacht wurde, wie die Architektur und die technischen Rahmenbedingungen aussehen, wird dieses Wissen schematisch vorgestellt und es werden Fragen geklärt.
- **Welche Risiken sollen vermieden werden?**
  Es geht hier vorrangig um die Beantwortung der Frage, welche Risiken nach Meinung des Scrum-Teams vermieden werden sollten. Das Ergebnis mündet in einer Aktionsliste, die bearbeitet werden sollte, um ein erfolgreiches Projekt durchzuführen.
- **Wie lange werden wir benötigen?**
  Auch wenn es in dieser frühen Phase unrealistisch erscheint, wird hier über

die geschätzte Projektdauer gesprochen (siehe Abschnitt 4.5.2).

- **Was ist uns wichtig, was ist nicht so wichtig?**
  Welche Kompromisse werden zuungunsten von etwas anderem eingegangen, wenn es im Verlauf des Projekts einer Entscheidung bedarf? Dass z.B. der Projektumfang flexibel gehalten werden sollte, steht vielleicht fest, aber wie sieht es mit der Qualität, den Kosten, der Zeit oder anderen für das Projekt wichtigen Komponenten aus?
- **Mit welchen Kosten ist für die Laufzeit zu rechnen?**
  In diesem letzten Schritt wird die Frage beantwortet, welche geschätzten Kosten mit den neugewonnenen Erkenntnissen zu erwarten sind.

Eine Präsentation des Produktziels oder des »Big Picture« ist nicht ausreichend, um Entwickler und Stakeholder abzuholen. Wichtig ist, dass alle die ersten Schritte gemeinsam machen und zusammen kurz- und langfristige Ziele definieren. Das beste Ergebnis eines Kick-offs ist es, wenn sich alle verantwortlich für den Erfolg und das Ergebnis fühlen.

Neben diesen sinnvollen Elementen, die Rasmussen in seinem Buch detailliert beschreibt, halten wir in der Praxis die Erarbeitung weiterer Artefakte wie z.B. die Definition of Done innerhalb des Kick-off-Workshops für sinnvoll (siehe Abschnitt 4.1.2).

Organisatorische Themen

Neben diesem ersten informellen Teil gibt es für alle Anwesenden auch organisatorische Aspekte zu klären. Diese Punkte können im Anschluss formuliert und besprochen werden. Folgende Inhalte könnten u.a. aus dem projektspezifischen Teil des Workshops abgeleitet werden:

- **Zeitpunkte**
  Gibt es wichtige Schlüsseltermine auf dem Weg der Umsetzung (z.B. Messen, Ferienzeit, Marketingkampagnen)?
- **Rollen**
  Welche Rollen gibt es neben dem Scrum-Team und wie sind diese benannt? Wer übt welche Funktion in den Rollen aus (z.B. die Projektsponsorin finanziert das Projekt, das Management stellt das Scrum-Team zur Verfügung)?

- **Teamname**
  Wie soll das Scrum-Team heißen? Gibt es ein Teamlogo, um die Identität zu schärfen?
- **Zusammenarbeit**
  Welche Kommunikationswege werden vorrangig genutzt (z.B. E-Mail, Sprint-Review, Zoom, ein eigener Chatkanal)? Wie werden direkte und indirekte Stakeholder mit einbezogen (z.B. Sprint-Review, Releasemeeting)?
- **Entscheidungen**
  Wer trifft die Entscheidungen (z.B. über die Bereitstellung eines zusätzlichen Budgets)?
- **Reporting**
  Was soll neben dem Sprint-Review noch, an wen und in welcher Form berichtet werden (z.B. Umfrage- oder Testergebnisse)? Wie erhalten die Stakeholder Einblick in den Fortschritt (z.B. digitale Abbildung des Fortschritts per Software, Bereitstellung von Testversionen)?
- **Risiken**
  Welche Risiken sollten noch beseitigt werden (z.B. Installation neuer Büroräume im Ausland)?
- **Offene Fragen und nächste Schritte**
  Welche Fragen konnten nicht geklärt werden? Wann wird es eine Zusammenfassung der Ergebnisse des Kick-offs geben? Welche nächsten kurzfristigen Schritte gibt es und wer unternimmt diese bis wann?

Fügen Sie in den Workshop Elemente ein, die ein Kennenlernen und den Austausch zwischen den Projektbeteiligten ermöglichen. Welche Erwartungen haben die Anwesenden? Welches Projekt haben sie davor betreut und was ist dort abgelaufen? Lassen Sie solche oder ähnliche Fragen in Einzel- oder Gruppengesprächen klären und kombinieren Sie diese mit aktiven Kennenlern-Elementen.

Im dritten Teil des Kick-offs wenden wir uns den teamspezifischen Fragen zu.

Teamspezifische Themen

Im letzten Schritt werden vom Scrum Master zu klärende Themenbereiche zur Sprache gebracht. Die Vereinbarung sämtlicher teamrelevanter Themen ist kein einmaliger Schritt, sondern gemäß dem aktuellen Wissensstand der Beteiligten eine erste Festlegung. Zur Sprache können u.a. die in Tabelle 4–1 aufgeführten Themenkreise kommen.

Tab. 4–1 *Teamspezifische Fragen für ein Kick-off*

Thema	Beispielfragen
Sprint	■ Wie lange dauert ein Sprint (z.B. min. ein bis max. vier Wochen)? ■ Wann beginnt und endet der Sprint (z.B. Wochenmitte, Uhrzeit, Releasezyklus)? ■ Wie gestaltet sich der Sprint-Zyklus (z.B. Definition der Bereitstellung für benötigte Designs)?
Scrum-Events	■ Wann finden die Scrum-Events im Sprint statt (z.B. Wochentag, Uhrzeit)? ■ Wie werden Sprint-Ergebnisse gezeigt (z.B. gemeinsame Sprint-Reviews mit anderen Teams)? ■ Wer ist verantwortlich für das jeweilige Event (z.B. Product Owner für das Backlog Refinement)?
Arbeitszeiten	■ Gibt es Arbeitszeitregelungen des Unternehmens (z.B. flexible Arbeitszeiten, Kernarbeitszeiten)? ■ Wann ist die Hauptarbeitszeit des Teams (z.B. von 9–17 Uhr)? ■ Wann ist das Arbeiten von zu Hause aus gestattet (z.B. Regelung des Unternehmens, Voraussetzungen)? ■ Wann sind dedizierte Arbeitszeiten ohne Unterbrechung von außerhalb und innerhalb des Scrum-Teams gewünscht (z.B. am Nachmittag von 14–17 Uhr)?
Sprache	■ Was ist die Hauptsprache (z.B. Englisch bei mehrsprachigen Teams)? ■ Was ist die Dokumentations- oder E-Mail-Sprache? ■ Wann sind Abweichungen erlaubt?
Werte	■ Welche Werte gelten für das Team besonders (z.B. Vertrauen, Wertschätzung, Mut)? ■ Welche Ziele setzt sich das Team (z.B. höchstmögliche Testabdeckung, häufiges Pair Programming)? ■ Wie steht es um das Thema »Technische Exzellenz« (z.B. Hinweis auf technische Schulden)?

Werkzeuge	▪ Wo werden Teamentscheidungen und -informationen dokumentiert (z.B. Wiki, Backlog Items)? ▪ Welche Software kommt für welche Belange zum Einsatz (z.B. Teamkommunikation)? ▪ Wo wird das Product Backlog gepflegt (z.B. welche Software wird eingesetzt)?
Arbeitsmittel	▪ Welche Infrastruktur oder Hardware wird eingesetzt (z.B. Klärung vorhandener und notwendiger Server)? ▪ Was benötigt das Team noch, um vernünftig arbeiten zu können (z.B. Monitoring-Computer, große Bildschirme)?
Dokumentation	▪ Wie wird dokumentiert (z.B. in der Softwareentwicklung soll Dokumentation im Code für sich selbst sprechen, Coding Guidelines)? ▪ Was sollte alles dokumentiert werden (z.B. Hinweise für den Support)?
Scrum-Board	▪ Aufbau und Physis des Scrum-Boards (z.B. physisches und/oder digitales Scrum-Board, Farbcodierung) ▪ Arbeit mit dem Scrum-Board (z.B. Kriterien für die einzelnen Spalten, Eintrittsmerkmale für Fehler (»Bugs«))
Backlog Items	▪ Welchen Aufbau und welchen Inhalt haben Backlog Items (z.B. User Story, Anwendungsfälle)? ▪ Wie ist mit Backlog Items umzugehen, die Entwickler verfassen (z.B. Besprechung vor dem Backlog Refinement mit dem Product Owner)? ▪ Welche Benutzerrollen gibt es (z.B. Personas)?
Informationen	▪ Welche Informationen möchte das Team regelmäßig erhalten (z.B. Reports, Statistiken)? ▪ Wie meldet sich ein Teammitglied bei den anderen ab, wenn es krank ist (z.B. E-Mail oder Chatnachricht)?
Arbeitsweise	▪ Wie geht das Team mit Echtzeitfehlern und weniger wichtigen Fehlern um (z.B. Abstimmung mit dem Product Owner, Abwesenheit des Product Owners)? ▪ Welche Eintrittskriterien sind für die Aufnahme von Backlog Items in den Sprint zu beachten (z.B. Definition of Ready)? ▪ Welche Fertigstellungskriterien sind zur Abnahme neuer Funktionalitäten zu beachten (z.B. Definition of Done)?
Offene Fragen und nächste Schritte	▪ Wie werden die festgelegten Informationen und Vereinbarungen festgehalten (z.B. Aufhängen im Teamraum)? ▪ Welche offenen Fragen konnten nicht geklärt werden? ▪ Wer kümmert sich neben dem Scrum Master um die Klärung?

Während des Workshops sollten alle wichtigen Eckpfeiler des Projekts festgehalten werden. Wenn Sie innerhalb des Termins keinen eindeutigen Konsens herstellen können, da z.B. wichtige Informationen fehlen, ist dies nicht kritisch. Die Ergebnisse aus einem Kick-off sind lebendige Artefakte, die über die

Projektlaufzeit gepflegt und aktualisiert werden müssen. Diese Arbeitsvereinbarungen drücken aus: »So möchten wir als Team zusammenarbeiten.«

Die inhaltliche Bündelung der drei vorgestellten Themenkreise führt zu einem allumfassenden Startschuss für das Projekt. Verwenden Sie in Absprache mit dem Product Owner eine abgeschwächte Version des Workshops, um mit den Entwicklern in Abständen den Zielfortschritt und getroffene Vereinbarungen auf Aktualität zu prüfen.

Die oben genannten Themenkreise beinhalten schon einige der wichtigsten Kriterien, die für die Arbeit am Produkt relevant sind. Daher legen wir diese zu diesem frühen Zeitpunkt schon fest, da diese Relevanz für die Vorbereitung des ersten Sprints haben können.

Wir nutzen zu Beginn von Projekten häufig ein »Team-Canvas«, das alle wesentlichen Aspekte des Scrum-Teams zusammenfasst. Mithilfe eines vorgegebenen Rahmens sind die Teammitglieder Schritt für Schritt eingebunden, wenn es um die Erstellung ihrer Teamidentität und -kultur geht (*theteamcanvas.com*).

Zu diesem Schwerpunkt finden Sie die Checkliste »Themenkreise eines Kickoffs« unter *link.scrum-in-der-praxis.de/checklisten*.

4.1.2 Definition of Ready/Definition of Done

Für ein einheitliches Verständnis der notwendigen Vorbereitung eines Backlog Item für das Sprint Planning empfiehlt sich die Definition of Ready. Sie ist zwar kein offizielles Scrum-Artefakt, hat sich aber in der Praxis außerordentlich bewährt.

Definition of Ready (DoR)

Die Definition of Ready wird zwischen den Entwicklern und dem Product Owner vereinbart. Sie beschreibt, welche Erwartungen die Entwickler an ein Backlog Item haben, bevor es im Sprint Planning für den Sprint ausgewählt werden kann. Roman Pichler fordert in [Pichler 2014], dass Backlog Items klar (im Sinne eines gemeinsamen Verständnisses), realisierbar und testbar sein sollten. Diese Begriffe sind zunächst sehr weit gefasst, und es ist Aufgabe der Entwickler und des Product Owners, diese gemeinsam zu konkretisieren. Haben die Visual Designs zu Sprint-Beginn final zu sein oder reicht ein Entwurf oder klickbarer Prototyp zunächst aus? Benötigen wir finale Texte vorab oder werden sie während des Sprints im Dialog mit dem Product Owner festgelegt?

Achten Sie als Scrum Master darauf, dass die Definition of Ready nicht als Mittel für eine bis auf das letzte i-Tüpfelchen ausformulierte Spezifikation missbraucht wird. Sie soll keineswegs den Dialog zwischen Product Owner und Team ersetzen, sondern lediglich dafür sorgen, dass Backlog Items im Sprint Planning so vorliegen, dass keine für die Umsetzung notwendige Information fehlt.

Eine einfache Definition of Ready könnte z.B. folgendermaßen aussehen:

- Das Backlog Item ist geschätzt.
- Das Backlog Item wird von allen verstanden.
- Das Backlog Item hat einen klaren Geschäftswert.
- Das Backlog Item ist nicht größer als … Story Points.
- Das Backlog Item kann ohne größeren Forschungsaufwand in einem Sprint umgesetzt werden.
- Das Backlog Item beschreibt nur eine Anforderung.
- Das Backlog Item enthält gut definierte Akzeptanzkriterien.
- Die technische Machbarkeit wurde geprüft.

Richtig eingesetzt kann eine Definition of Ready ein wichtiger Baustein auf dem Weg zu einer Effektivitätssteigerung des Teams sein. Sie ist, wie Abbildung 4–2 zeigt, das zu passierende Tor zum Sprint und zwingt den Product Owner dazu, sehr früh sehr sauber zu arbeiten.

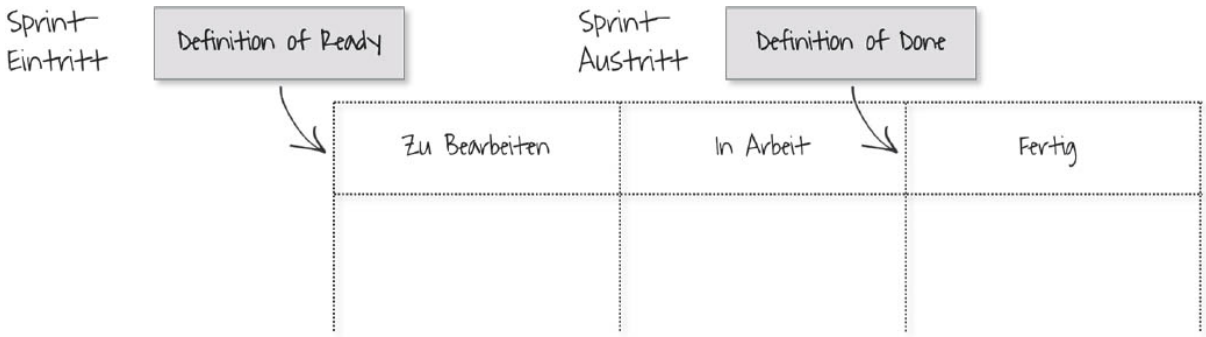

Abb. 4–2 *Definition of Ready (DoR) und Definition of Done (DoD)*

Auch wenn Sie nicht mit einer Definition of Ready arbeiten: Ermutigen Sie die Entwickler, unzureichend vorbereitete Backlog Items im Sprint Planning nicht zu akzeptieren. Bringen Sie Product Owner und Entwickler zusammen, um die Backlog Items zu erstellen (siehe Abschnitt 4.4).

Kein Scrum-Team tut sich einen Gefallen damit, die Lieferung unklarer Backlog Items zuzusagen, denn ein Verfehlen des Sprint-Ziels ist dadurch sehr wahrscheinlich.

Definition of Done (DoD)

> *»In dem Moment, in dem ein Product-Backlog-Eintrag die Definition of Done erfüllt, wird ein Increment geboren.« (scrumguides.org)*

Die Kriterien, die ein Scrum-Team für die Zusammenarbeit und die Bearbeitung von Backlog Items festlegt, sind sehr individuell. Es gibt Scrum-Teams, in denen die Vereinbarungen etliche Seiten füllen, aber auch welche, die mit weniger als einer DIN-A4-Seite auskommen. Die Kriterien, die gemeinschaftlich festgelegt werden, dienen in der täglichen Arbeit als Prüfliste für die Entwickler, um den Fortschritt eines Backlog Item auf dem Weg vom Eingang in den Sprint bis zur Veröffentlichung zu planen.

Die DoD ist Teil des Scrum Guide. Dort wird die DoD als »formale Beschreibung des Zustands des Inkrements, wenn es die für das Produkt erforderlichen Qualitätsmaßnahmen erfüllt«, bezeichnet. Sie zielt vor allem auf die Transparenz, was »fertig« bedeutet, innerhalb des Scrum-Teams ab.

Darüber hinaus definiert der Scrum Guide (*scrumguides.org*):

> *»Die Definition of Done schafft Transparenz, indem sie allen ein gemeinsames Verständnis darüber vermittelt, welche Arbeiten als Teil des Increments abgeschlossen wurden. Wenn ein Product-Backlog-Eintrag nicht der Definition of Done entspricht, kann es weder released noch beim Sprint Review präsentiert werden. Stattdessen wandert es zur zukünftigen Berücksichtigung in das Product Backlog zurück.«*

Dies bedeutet, dass die Definition of Done für ein Scrum-Team so aufgebaut sein sollte, dass Product Owner am Ende eines Sprints das Backlog Item akzeptieren können.

Da jedes neue Inkrement auf bestehende aufbaut, ist im Laufe der Zeit die DoD anzupassen, um bei steigender Komplexität dem Interesse an Qualität gerecht zu werden.

Eine einfache DoD könnte folgendermaßen aussehen:

- Mindestens zwei Entwickler haben an einem Backlog Item gearbeitet oder es wurde alternativ ein →Codereview durch einen zweiten Entwickler durchgeführt.
- Bestehender Code wurde überarbeitet bzw. entfernt.
- Automatisierte Tests existieren und laufen.
- Notwendige (Inline-)Dokumentation wurde ergänzt.
- Der Product Owner hat das Backlog Item akzeptiert.

Manchmal gibt es in Organisationen eine DoD, die als Standard der Organisation vorgegeben ist. In diesem Fall gilt diese als Mindestmaß für alle Teams und kann ggf. teamintern ergänzt werden.

Für den Fall, dass mehrere Teams gemeinsam an einem Produkt arbeiten, schreibt der Scrum Guide vor, dass die Teams sich auf eine gemeinsame Definition of Done einigen und sich daran halten sollten.

Es gilt hier, genau wie bei allen Vereinbarungen und Ergebnissen des Kick-offs: Die Definitionen sind lebendige Dokumente, die laufend überprüft und angepasst werden sollten. Gute Zeitpunkte für die Überprüfung sind das Daily Scrum und die Sprint-Retrospektive.

Wie gelangt man zu den Definitionen? Wir nutzen in der Praxis eine Abwandlung von David Koontz, die zügig ein Ergebnis liefert und alle aktiv in die Erstellung einbindet [URL:Koontz]. Je nach Seniorität des Scrum-Teams ist eine halbe bis eine Stunde notwendig, um die Kriterien zu definieren. Die Methode kann sowohl zur erstmaligen Festlegung als auch zur Aktualisierung der Definitionen genutzt werden.

Neben der bekannten DoD und der DoR sind wir in der Praxis auch schon auf Teams gestoßen, die hier nicht haltmachen. Beispielsweise werden dann auch »Sprint-Definitionen«, »Releasedefinitionen« oder »Vereinbarungen zum Umgang mit Fehlern« erarbeitet, die zusätzliche Kriterien enthalten (siehe auch Abschnitt 6.1).

Ermittlung der DoR und DoD

Nachfolgend beschreiben wir eine einfache Lösung, um die DoR und DoD innerhalb eines Workshops digital oder gemeinsam in einem Meetingraum zu ermitteln. Viel wichtiger als die Erstellung und Einigung auf die Arbeitsvereinbarungen ist jedoch das Befolgen. Für ein Scrum-Team geben sie richtungsweisend die Bedingungen für die Arbeit miteinander und am Produkt vor.

Vorbereitung

Es ist für den Start in den Workshop notwendig, einige Kriterien als Vorschlag auf Karten zu notieren. Die Auswahl der Kriterien ist lediglich ein Vorschlag, um einen schnellen Einstieg zu gewährleisten und um zur Erstellung weiterer Kriterien anzuregen. Es sollten nicht mehr als 10–20 Karten für die jeweilige Definition erstellt werden. Das hilft später im Workshop dabei, den Fokus auf die wesentlichen Punkte zu lenken.

Es empfiehlt sich, auch ein paar leere Karten zur Verfügung zu stellen, damit zusätzliche Kriterien dokumentiert werden oder bestehende Kriterien ergänzt bzw. umformuliert werden können.

Alle vorliegenden Kriterien sind lediglich Vorschläge und können innerhalb des Workshops umgeschrieben, ergänzt oder entfernt werden.

Versuchen Sie gemeinsam mit den Beteiligten, die Kriterien messbar zu machen. Es ist z.B. sehr einfach, zu behaupten, dass automatische Tests existieren und laufen. Aber wie sieht die Testabdeckung aus? Eine prozentuale Angabe mit dem aktuellen Stand (z.B. 60%) ist klarer und hilft, Verbesserungen für die Zukunft anzustreben.

Nachdem die Karten ausgedruckt vorliegen, werden zwei Flipcharts zu Hilfe genommen, auf denen drei Bereiche markiert sind: »Jetzt«, »Demnächst« und »Später«, wie in Abbildung 4–3 dargestellt.

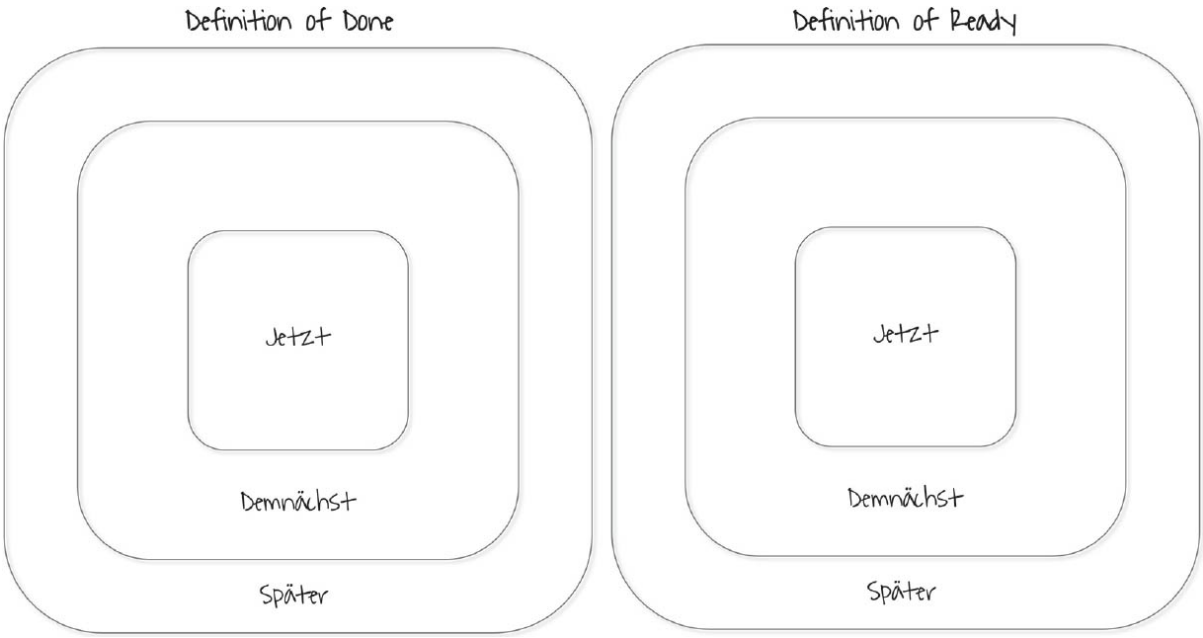

Abb. 4–3 *Leere Arbeitsbereiche*

In diese Bereiche werden die Karten durch die Teammitglieder später platziert. Mit diesen Bereichen soll verdeutlicht werden, dass die Definitionen reifen können und einer ständigen Aktualisierung oder Pflege unterliegen. Die

Kategorisierung zeigt auch Entwicklungspotenzial und macht Aufgaben sichtbar, die das Team als Nächstes angehen sollte. Zum Beispiel kann bei einem Team, das aktuell noch nicht über automatisierte Tests verfügt, dieses Kriterium nicht in »Jetzt« erscheinen, sondern höchstens in den anderen Bereichen. Die Beteiligten nehmen sich also vor, dieses Thema im Auge zu behalten und zeitnah anzugehen, um es in die DoR oder DoD zu integrieren.

Verkomplizieren Sie die Kriterien nicht und achten Sie bei der Erarbeitung des Ergebnisses darauf, dass alle ein einheitliches Verständnis haben. Zu viele Kriterien führen dazu, dass sich das Team unter Umständen überfordert fühlt, alle Regeln einzuhalten. Ergänzen Sie die Kriterien nach und nach, wenn diese sich erst einmal etabliert haben.

Durchführung

Der Scrum Master erläutert den Sinn des Treffens und erklärt kurz das Vorgehen mit Verweis auf die vorbereiteten Flipcharts und die ausgebreiteten Kärtchen mit den Kriterien auf dem Tisch.

Wenn dies geschehen ist, sind alle aufgefordert, sich die Karten anzusehen und diese zu verteilen. Grundbedingung dafür ist, dass niemand sprechen darf und sich bei Unstimmigkeiten anderweitig geeinigt werden muss. Zur Not werden Karten, die nicht verstanden oder als nicht sinnvoll eingestuft werden, erst einmal zur Seite gelegt. Wenn jemand eine Karte ergänzen möchte, notiert er seine Anmerkung darauf. Dieser erste Teil geht sehr zügig und resultiert meistens in einem ersten Einverständnis.

Der Scrum Master sollte nun alle auffordern, sich dieses Ergebnis anzusehen und sich kurz die Zeit zu nehmen, um die damit verbundene Bedeutung zu verinnerlichen. Es hilft in diesem Schritt, einzeln durch die Kriterien zu gehen und diese inhaltlich klarzustellen, sodass alle das gleiche Verständnis erhalten. Dabei werden Fragen zu den Kriterien aufkommen, die in einer Diskussion geklärt werden. Die Kriterien auf den Karten werden umgeschrieben, ergänzt oder in eine andere Kategorie verschoben. Sind neue Karten hinzugekommen oder fehlen noch Kriterien, ist jetzt der richtige Zeitpunkt, diese zu besprechen bzw. hinzuzufügen. Dabei befragt der Scrum Master alle der Reihe nach. Jedes Teammitglied kann etwas verändern und begründet seine Entscheidung oder setzt aus, wenn es keine Veränderung vornehmen möchte. Die Veränderungen werden so lange durchgeführt, bis am Ende eine einvernehmliche Meinung herrscht und alle dem Ergebnis zustimmen. Abbildung 4–4 zeigt beispielhaft ein Ergebnis des Termins.

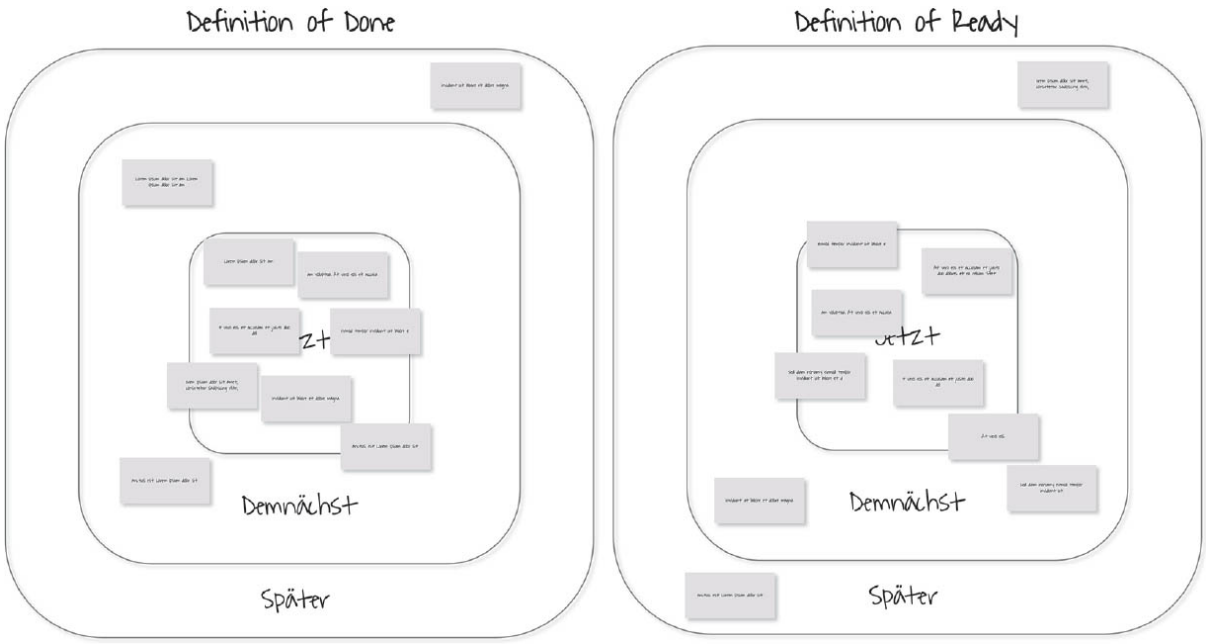

Abb. 4–4 *Kriterien für die DoR und DoD*

Der Vorteil, beide Definitionen zu erarbeiten, liegt darin, dass für alle klar herausgearbeitet wird, welches die Eintrittskriterien sind und welche Kriterien den Austritt definieren. Gleichzeitig ermöglicht dies eine bessere inhaltliche Abstimmung der DoR und DoD aufeinander. Es liegt im Ermessen des Scrum Masters, ob die DoR und DoD in einem gemeinsamen Termin oder in zwei Schritten erarbeitet werden.

Wir lassen die Beteiligten oftmals beide Definitionen erstellen. Dafür mischen wir die mitgebrachten Kriterien und lassen Diskussionen darüber, ob ein Kriterium nun zur DoD oder DoR gehört, bewusst zu. Sie können in diesem Schritt jedoch auch nacheinander die einzelnen Definitionen erarbeiten lassen.

Nachbereitung

Am Ende sollte das fertige Ergebnis unter Umständen noch einmal strukturiert aufbereitet, klarer formuliert oder erläutert werden. Die direkte Platzierung am Scrum-Board ist zudem sinnvoll, da die Kriterien im Hinblick auf die dort täglich stattfindende Abstimmung in die Diskussion mit einbezogen werden können oder als Erinnerung dienen.

Die Kriterien in den »Demnächst«- und »Später«-Bereichen können parallel zum Sprint weiterentwickelt werden. Denkbar ist auch, diese als Backlog Items

zu entwickeln, falls die Komplexität höher ist, um sie in einem der nächsten Sprints zu bearbeiten.

Alle drei bis sechs Monate sollte der Workshop wiederholt werden, um die Definitionen zu aktualisieren.

Überlegen Sie sich, ob sowohl die Definition of Ready als auch die Definition of Done im Anschluss von allen unterschrieben werden sollen, um eine Verbindlichkeit herzustellen, auf die bei Bedarf verwiesen werden kann.

4.1.3 Umgang mit Fehlern

Umgang mit Fehlern

Nachdem das Team über den Aufbau des Scrum-Boards und den Sprint-Zyklus gesprochen hatte, stand das Thema »Fehler und wie gehen wir damit um« auf der Liste, die Finn für den Workshop vorbereitet hatte.

Finn leitete das Thema ein, indem er Casper und den Entwicklern die verschiedenen Typen von Fehlern vorstellte. Dafür hatte er eine Übersicht vorbereitet, zu der er jeweils ein Beispiel vortrug.

»Implementierung neuer Funktionalität« stand als Erstes auf seinem Überblick. »Stellt euch vor«, begann er, »ihr seid mitten im Sprint und Mina findet einen Fehler bei einem der Sprint Backlog Items, was würdet ihr in diesem Fall tun?« Sergio meldete sich zu Wort und antwortete: »Sofort beheben, damit das Backlog Item abgeschlossen werden kann.« Finn nickte und stellte die Frage, wie das Team denn gerne diese Art von Fehlern am Scrum-Board sichtbar machen wolle. Nach einigen Diskussionen folgten sie dem Vorschlag von Finn und einigten sich darauf, eine rote Karte mit kurzer Beschreibung des Fehlers und dem Namen der Person aufzuhängen, die ihn gefunden hatte.

Bei der zweiten Kategorie von Fehlern, nämlich die, die während des Sprints auftreten und die Beendigung eines Sprint Backlog Item und somit das Sprint-Ziel in Gefahr bringen, einigte sich das Team genau wie bei der ersten Kategorie auf eine rote Karte. Finn brachte dazu folgendes Beispiel an: »Stellt euch vor, ihr arbeitet an einem Backlog Item und mitten in der Arbeit stellt ihr fest, dass euch noch eine Übersetzung eines Textes fehlt. Wie gehen wir damit um?«

Die nächste Kategorie betraf Fehler, die während der Implementierung des Sprint Backlog gefunden wurden, jedoch keinen Einfluss auf das Erreichen des Sprint-Ziels haben. »Jordi, stell dir vor, du arbeitest gerade an einem neuen Backlog Item und während der Arbeit fällt dir eine Inkonsistenz auf. Was würdest du in diesem Fall tun?« Jordi überlegte kurz und antwortete dann: »Ich würde den Fehler beheben.« Finn nickte und sagte: »Das ist eine Möglichkeit, fällt euch noch eine andere ein?« Mina meldete sich zu Wort und merkte an: »Müsste Jordi den Fehler nicht an Casper und das Team melden?« Finn gab auch ihr Recht. Am Ende einigten sich alle darauf, dass diese Art von Fehlern, wenn sie innerhalb von wenigen Minuten behoben werden können, einfach umgesetzt und das Team darüber informiert wird. Wenn Fehler nicht in diese Kategorie fallen, ist Casper mit einzubeziehen, damit er entscheidet, wie wichtig die Behebung des Fehlers ist.

Zu dem Beispiel passte auch der nächste Fehlertyp, den Finn ansprach. »Wenn ein Dritter einen Fehler meldet, wie gehen wir damit um?« Hier einigte sich das Team darauf, genau wie bei der letzten Kategorie vorzugehen. Jedoch wurde festgelegt, dass für diese Fälle an die Dritten kommuniziert werden sollte, dass die Fehler über die E-Mail-Adresse »bugs@sidp.de« einzureichen sind und somit direkt an Casper gehen.

Casper hatte am Ende noch einige Fragen, die sie gemeinsam klären konnten. Unter anderem einigten sie sich darauf, dass Casper als Product Owner die Entscheidungen darüber trifft, ob ein Fehler, der nicht den Sprint betrifft, so wichtig ist, dass er während des Sprints bearbeitet wird. Finn gab ihm dabei zu verstehen, dass in diesem Fall das Sprint-Ziel in Gefahr gebracht werden könnte, da das Team sich auf die Bearbeitung dieser Fehler konzentrieren würde, was Casper aber akzeptierte.

Finn war fürs Erste zufrieden und setzte den Workshop mit dem nächsten Punkt auf der Agenda fort.

Wir werden in unserer täglichen Arbeit immer wieder gefragt, wie in Scrum mit Fehlern (»Bugs«) umgegangen wird. Generell gilt es natürlich, die Qualität der ausgelieferten Inkremente dahingehend zu erhöhen, dass kaum noch neue Fehler auftauchen. Es ist die Aufgabe des Scrum Masters, das Scrum-Team dafür zu sensibilisieren und zu entwickeln. Da Produktentwicklung ohne Fehler aber nicht möglich ist, stellen wir hier eine Vorgehensweise vor, die uns in den letzten Jahren erfolgreich dabei geholfen hat, mit neuen Fehlern umzugehen.

Zero Bug Policy

Die Kundschaft und Stakeholder erwarten selbstverständlich ein qualitativ hochwertiges Produkt und auch Scrum-Teams haben in der Regel den Anspruch, möglichst gute Qualität abzuliefern. Dieser Anspruch wird oft durch Druck von außen zerstört, weil neue Funktionalitäten dringend geliefert werden »müssen«, auch wenn es »quick and dirty« ist.

Bringen Sie das Scrum-Team und die Stakeholder an einen Tisch und thematisieren Sie das Thema »Qualität«. Was verstehen die Beteiligten darunter? Sind die unterschiedlichen Perspektiven klar? Verstehen die Stakeholder den (technischen) Qualitätsanspruch des Scrum-Teams? Versteht das Scrum-Team den (fachlichen) Qualitätsanspruch der Stakeholder und den Druck des Marktes? Worauf kann man sich gemeinsam einigen?

In der Regel sind sich alle einig, dass Fehler sowohl aus technischer als auch aus fachlicher Sicht unerwünscht sind. Aus diesem Grund kann der nachfolgende Vorschlag meist sehr einfach und erfolgreich umgesetzt werden.

Aufräumen

Bevor es daran geht, neue Fehler zu bearbeiten, sollte zunächst eine Bestandsaufnahme gemacht werden:

- *Wie viele Fehler sind momentan noch offen?*
- *Wie alt sind diese Fehler?*
- *Sind sie tatsächlich noch aktuell oder wurde die Meldung nur nicht aktualisiert?*

Wir finden in den Verwaltungssystemen unserer Projekte immer wieder Fehler, die schon über ein Jahr alt oder noch älter sind. Allein das spricht schon dafür, dass der Fehler entweder unwichtig ist oder sich niemand ernsthaft dafür interessiert. Oft wurden solche alten Fehler automatisch irgendwo miterledigt und die Fehlermeldung wurden nie geschlossen.

Schlagen Sie vor, einfach alle alten Fehler zu schließen und zu warten, ob sich jemand meldet. Falls das zu radikal ist, sollten zumindest alle Fehler, die älter als sechs Wochen sind, geschlossen werden.

Sobald festgelegt wurde, welche Fehler tatsächlich noch relevant sind, wird über die Priorität und das weitere Vorgehen innerhalb des Scrum-Teams gesprochen. Manche Scrum-Teams einigen sich mit den Stakeholdern auf einen sogenannten »Bug Sprint«, in dem versucht wird, so viele Fehler wie möglich zu schließen, um schnellstmöglich in den neuen Modus der raschen Erledigung neuer Fehler zu gelangen. Andere nehmen sich für jeden Sprint eine feste Anzahl an Fehlern vor.

Welche der obigen Variationen gewählt wird, hängt ganz vom jeweiligen Kontext ab, und auch von der Anzahl der bestehenden Fehler. Wir haben gute Erfahrungen mit einmaligen (!) Bug Sprints gemacht, weil hier ein gewisser »Wir rocken das«-Spirit im Team entsteht. Achten Sie aber darauf, dass Bug Sprints nicht zur Regel werden und Sie sich als Scrum Master auf die Behebung der Ursachen für die Fehleranfälligkeit fokussieren.

Zukünftiger Umgang mit Fehlern

Nachdem aufgeräumt wurde und sich alle einig sind, dass zukünftig nie wieder so einen Berg an Fehlern entstehen soll, können sich alle um aktuell auftretende Fehler kümmern.

Zunächst sollte unterschieden werden, in welchem Kontext die Fehler gefunden wurden und wie schwerwiegend sie sind. Fehler, die im Entwicklungssystem während des Testens gefunden wurden, sind anders zu behandeln als Fehler, die im Livesystem bestehen und ggf. ein Abwandern der Kundschaft bedeuten.

Um der Zero Bug Policy Rechnung zu tragen, gilt folgende Basisregel:

> *Fehler werden idealerweise sofort, spätestens aber im nächsten Sprint behoben*.

Im Zweifelsfall entscheidet der Product Owner über die Dringlichkeit.

Fehler im Entwicklungssystem

- *Fehler, die beim Entwickeln neuer Produktlösungen im aktuellen Sprint entstanden sind*
  Über diese Fehler sollte ein Scrum-Team gar nicht reden, es gehört zum Sprint, am Ende ein fehlerfreies Inkrement zu liefern.
- *Fehler, die nicht im aktuellen Sprint entstanden sind, aber die Entwicklung der Backlog Items im Sprint behindern (und damit das Sprint-Ziel gefährden)*
  Hier gilt das Gleiche: Diese Fehler sollten sofort behoben werden. Wer einen solchen Fehler findet, informiert alle idealerweise sofort, spätestens aber im nächsten Daily Scrum.
- *Fehler, die nicht im aktuellen Sprint entstanden sind, noch nicht live sind und die Entwicklung der Sprint Backlog Items im Sprint nicht behindern* Bei diesen Fehlern sollte ein Entwickler ein wenig Fingerspitzengefühl walten lassen. Wenn es ein einfacher Fehler ist, kann er gleich behoben werden.

Allerdings sollten die Entwickler darüber informiert werden, falls Seiteneffekte bestehen oder möglich sind. Wenn der Fehler sich als komplex herausstellt, sollte der Product Owner informiert werden. Gemeinsam mit dem Product Owner kann gemeinsam entschieden werden, ob der Fehler noch im Sprint behoben oder offiziell in den nächsten Sprint aufgenommen werden soll.

Fehler im Livesystem

Wenn man sich auf eine Zero Bug Policy geeinigt hat, sollten insbesondere Fehler im Livesystem unverzüglich bearbeitet werden. Sollte ein Fehler im Livesystem existieren, der die Kundschaft sehr unglücklich macht, steht die sofortige Behebung natürlich außer Frage. Für alle anderen Fehler hat sich die folgende Vorgehensweise bewährt:

Jeden Tag wird vor dem Daily Scrum geprüft, ob in den letzten 24 Stunden ein neuer Fehler im Livesystem aufgetreten ist. Wer diese Prüfung vornimmt, entscheidet das Scrum-Team. Weiter unten in diesem Abschnitt beschreiben wir ein Bug-Standup, das zur Verteilung von Fehlern in Scrum-Teams genutzt werden kann.

Sollte ein Fehler aufgelaufen sein, wird dieser im Daily Scrum kurz vorgestellt und am Scrum-Board visualisiert.

Bis zum nächsten Daily Scrum sollte mindestens eine Analyse des Fehlers erfolgt sein. Sollte es sich nur um eine Kleinigkeit handeln und die Analyse gleichbedeutend mit der Behebung sein, wird der Fehler sofort behoben.

Sofern sich herausstellt, dass die Behebung etwas komplexer ist, sollte der Product Owner entscheiden, ob der Fehler noch in diesem Sprint behoben oder für den nächsten Sprint eingeplant werden soll. Das Beheben im aktuellen Sprint könnte das Sprint-Ziel gefährden und hat zur Folge, dass der Umfang des Sprints neu verhandelt werden muss.

Wie kommen die Fehler ins Team?

Sofern es nur ein Team gibt, das an einem Produkt arbeitet, ist klar, wo der Fehler behoben werden muss. Im Folgenden beschreiben wir eine gängige Variante:

- Fehler laufen beim Product Owner auf und dieser entscheidet über die Dringlichkeit.
- Wenn nicht eindeutig ist, dass der Fehler warten kann, gibt der Product Owner den Fehler beim nächsten Daily Scrum ins Team hinein mit der Bitte

um Analyse.

- Es gibt eine Vereinbarung zwischen dem Product Owner und dem Team, dass die Analyse bis zum nächsten Daily Scrum erfolgen muss. Ergebnisse können sein:
  - Der Fehler wurde gleich während der Analyse behoben.
  - Der Fehler zieht geringfügige Arbeiten nach sich, er kann behoben werden, ohne das Sprint-Ziel zu gefährden.
  - Der Fehler ist nicht so einfach, die Behebung im aktuellen Sprint könnte das Sprint-Ziel gefährden.
  - Der Fehler ist komplex, die Behebung im aktuellen Sprint führt sicher zum Reißen des Sprint-Ziels.
- Mit diesen Informationen trifft der Product Owner eine Entscheidung.

Schaffen Sie einen Eingangskanal für direkte Anfragen, z.B. einen Slack-Channel. Dieser sollte nur diesem einen Zweck dienen, d. h., wenn dort eine Nachricht auftaucht, wissen alle, dass es wirklich dringend ist. Achten Sie darauf, dass er nicht für andere Themen missbraucht wird.

In der Praxis kommt es häufig vor, dass nicht ein einzelnes Team an einem Produkt arbeitet, sondern mehrere Teams. Oft werden Fehler in irgendeinem Team gemeldet und nicht selten fühlt dieses Team sich nicht zuständig. Dies führt zu einem Fehler-Ping-Pong zwischen den Teams anstelle einer Lösung. Dieser Herausforderung kann mit einem Bug-Standup begegnet werden.

Bug-Standup

Jeden Tag treffen sich noch vor den Daily Scrums der Teams Vertreter aller am Produkt beteiligten Teams zu einem Bug-Standup (vgl. Daily Scrum in Abschnitt 5.2). Dies sind häufig die Product Owner, es können aber auch beliebige Teammitglieder sein. In diesem Bug-Standup werden die neu aufgelaufenen Fehler der letzten 24 Stunden in die Teams verteilt. Auch hier gilt das Pull-Prinzip, d.h., die Teamvertreter nehmen sich die Fehler vor, von denen sie glauben, dass sie in ihrem Team zu beheben sind. Das Bug-Standup ist erst beendet, wenn sich für jeden Fehler ein Team gefunden hat, das ihn analysiert. Natürlich kommt es vor, dass ein Fehler nicht immer sofort eindeutig einem Team zuzuordnen ist. An dieser Stelle erwarten wir einen entsprechenden Pragmatismus von den beteiligten Personen.

4.1.4 Letzte Vorkehrungen

Bevor es nun endgültig mit dem ersten Sprint losgehen kann, gilt es noch ein paar Punkte zu überprüfen.

Teamraum

Die Grundidee von Scrum ist es, dass das Scrum-Team zusammen auf ein gemeinsames Ziel hinarbeitet, daher ist es generell eine gute Idee, das komplette Scrum-Team in einem Raum zu versammeln und dort gemeinsam und fokussiert am Sprint-Ziel und am Produktziel arbeiten zu lassen. Wo immer dies möglich ist, arbeiten wir daran, dass die Teammitglieder zusammensitzen.

In der letzten Zeit wird dies jedoch immer seltener, da Teammitglieder sich oft über die ganze Welt verteilen und auch das Homeoffice immer mehr genutzt wird. Diesem Umstand tragen wir in Kapitel 6 Rechnung, das sich dem Arbeiten in verteilten Szenarien widmet.

Für den Fall, dass Sie das seltene Glück haben, Ihr komplettes Scrum-Team an einem Ort versammeln zu können, geben wir Ihnen unter *link.scrum-in-der-praxis.de/teamraum* viele Tipps und Anregungen.

Scrum-Board

Struktur des Scrum-Boards

Das Scrum-Board ist in erster Linie ein Werkzeug zur Organisation für die Entwickler. Es soll dem Scrum-Team helfen, den Gesamtüberblick über die Aufgaben im Sprint zu behalten, die Aufgaben zu strukturieren und jederzeit eine Fortschrittskontrolle zu haben. Am Scrum-Board wird geplant, koordiniert, diskutiert und entschieden. Es stellt die Informationszentrale des Scrum-Teams während des Sprints dar. Gerade weil die Ansprüche an das Scrum-Board hoch sind, sollte es in einer simplen Struktur gehalten werden. Je komplexer die Struktur, desto pflegeaufwendiger und fehleranfälliger ist das Scrum-Board und dementsprechend auch unzuverlässig. Ein kompliziertes Werkzeug wird weniger benutzt als ein simples.

Virtuelle Scrum-Boards

Für virtuelle Scrum-Boards gibt es viele Onlinetools, von kostenlosen bis hin zu kostenpflichtigen und kommerziellen Tools. Sie alle verbindet ein gemeinsamer Nachteil: die Virtualität. Kein Team kann sich vor ihnen versammeln, um eine

Diskussion zu führen, die Haptik des Anfassens und Verschiebens einer Karte fällt weg, niemand kann auf einen Blick den Fortschritt des Sprints oder einzelner Backlog Items erkennen, wenn er den Teamraum betritt. Außerdem müsste jemand, der das virtuelle Scrum-Board sehen oder bearbeiten möchte, sich zunächst in einem System anmelden und zum Scrum-Board navigieren. Dies alles führt dazu, dass ein Scrum-Board seine Vorteile nicht voll entfalten kann und einige Teammitglieder dazu tendieren, sich andere Wege zu suchen, ihre Aufgaben zu strukturieren.

In der Praxis treffen wir leider kaum noch auf physische Scrum-Boards, da selbst zusammensitzende Scrum-Teams größtenteils auf virtuelle Boards umgestiegen sind (siehe Kap. 6). Aus diesem Grund haben wir Ihnen die Informationen zu physischen Scrum-Boards unter *link.scrum-in-der-praxis.de/scrumboard* zur Verfügung gestellt.

4.1.5 Häufige Herausforderungen

In dieser frühen Phase des Projekts sind viele Fragen noch ungeklärt und mögliche Fehler vielschichtig. Für einige Herausforderungen, die auftreten können, geben wir nachfolgend ein paar Hinweise.

Schlechte Vorbereitung

Die Einstiegsszenarien für Projekte können sehr vielfältig sein und u.a. dazu führen, dass ein Scrum-Team keine Zeit zur Vorbereitung findet oder eine gute Vorbereitung unterschätzt. Wenn z.B. aufgrund einer Markteinführung eines neuen Konkurrenzprodukts ein Scrum-Team zusammengestellt wird, das sofort mit der Implementierung eines Produkts starten soll, oder für ein zu entwickelndes Produkt noch kein Product Owner gefunden wurde, sollte nicht einfach so gestartet werden. Diese oder ähnliche Szenarien machen eine gute Vorbereitung schwierig oder sogar sinnlos.

Auch wenn Sie einen schlechten Start erwischt haben, ist immer noch Zeit, ein Kickoff durchzuführen, selbst wenn Sie dies nicht zu Beginn eines Projekts in Angriff nehmen konnten.

Es ist sogar möglich, dass Sie mit einem Scrum-Team, das schon über Monate zusammenarbeitet, einen Workshop durchführen, der den Sinn eines »Neustarts« hat. Dies ist z.B. für Teams sinnvoll, die in einer Produktsparte über lange Zeit diverse Projekte realisieren.

Unklares Verständnis

Wir erleben zu Beginn eines Projekts immer wieder, dass während des Kickoffs ein »Aha-Effekt« entsteht, wenn der Product Owner den wahren Grund für das Projekt erläutert. Viele haben häufig ein ganz anderes Bild von dem, was umgesetzt werden soll, da es auf der einen Seite noch zu vage ist und die Beteiligten auf der anderen Seite verschiedene Interessen vertreten. Im Kickoff ist es möglich, diese Unterschiede durch die Diskussion in einem offenen Forum aufzuzeigen, damit alle den Grund für das Projekt verstehen. In der Diskussion werden viele weitere Produkte oder Ideen auftauchen, die jedoch unter Umständen nichts mit den Anforderungen an das Produkt zu tun haben, an dem gearbeitet werden soll.

Fragen Sie offen, warum die Anwesenden hier sind und welchen Grund es für das entsprechende Projekt gibt. Wenn es überhaupt einer Diskussion bedarf, wird im Nachhinein jeder verstanden haben, warum alle bei dem Termin dabei sind.

Mangelnde Beachtung der eigenen Regeln

Es ist eine gute Idee, Kriterien für die Zusammenarbeit zu definieren. Der wichtigere Schritt ist jedoch, dass diese Kriterien auch Beachtung finden. In der Praxis werden oftmals die Arbeitsvereinbarungen unbewusst ausgehebelt, wenn es eine »ganz besondere« Situation gibt, der man gerecht werden möchte. Zum Beispiel, wenn ein Backlog Item im Sprint Planning akzeptiert wird, obwohl es nicht, wie in der DoR vereinbart, vorab geschätzt und mit dem Team besprochen wurde. Spätestens beim nächsten Daily Scrum fällt dann auf, dass vielleicht das Sprint-Ziel des Teams in Gefahr ist, da Informationen unklar sind und das Backlog Item deshalb nicht abgeschlossen werden kann.

Natürlich kann es vorkommen, dass Regeln als falsch oder hinderlich wahrgenommen werden. In diesem Fall sollte dieser Punkt aber zur Sprache gebracht und besprochen werden und die Regel nicht einfach durch Nichtbeachtung ausgehebelt werden. Wenn dies mit einer Regel anfängt, geht es oft mit anderen Regeln weiter.

Machen Sie den Teammitgliedern klar, dass Scrum die Wertorientierung bei der Entwicklung von Inkrementen fördert und die Einhaltung der eigenen Kriterien für die Zusammenarbeit essenziell ist. Als Letztes sollte an der Qualität gespart werden. Die Einhaltung von Inspect & Adapt inklusive der »Ready«- und »Done«-Kriterien sollte allen im Scrum-Team eine Herzensangelegenheit sein.

Überstrukturierte Scrum-Boards

Wir sind in unserer Praxis mitunter auf Scrum-Teams gestoßen, die bis zu zehn Spalten auf ihrem Scrum-Board hatten. Die Pflege dieser Spalten ist recht aufwendig, was oft dazu führt, dass der Status am Scrum-Board nicht aktuell war. Außerdem wurde eine Menge Platz verschwendet, denn eine Taskkarte kann sich immer nur in genau einer Spalte befinden, sodass viele Spalten sehr dünn besiedelt oder gar leer waren. Auf diese Weise wurde die zur Verfügung stehende Wandfläche nicht optimal genutzt.

Verwenden Sie Taskkarten statt Spalten, um Prozessschritte abzubilden. Aufgaben wie »Codereview durchführen«, »Testfälle schreiben«, »Automatische Tests durchführen« oder »Definition of Done überprüfen« benötigen keine eigene Spalte am Board, sondern können als Task durch die Spalten »Open«, »Work in Progress« und »Done« wandern.

Teams mit großen Scrum-Boards haben fast immer eine stark prozessorientierte Vergangenheit. Sie versuchen, die bekannten Übergabepunkte, z.B. von der Entwicklung zum Testen, so abzubilden, wie sie es aus ihrem bisher benutzten Projektmanagement- oder Ticketverwaltungssystem kennen.

Versuchen Sie, das Team weg von der Prozesssicht hin zu den Grundlagen von Agilität im Allgemeinen und Scrum im Besonderen zu führen. Bemühen Sie das Agile Manifest und die agilen Werte. Weisen Sie auf die Vorteile von vertrauensvoller Zusammenarbeit hin, auf das gemeinsame Ziehen am gleichen Strang. Wenn Sie das Gefühl haben, dass das Team bereit ist, Ihnen zu folgen, schlagen Sie ein Experiment vor, für einen Zeitraum von zwei oder drei Sprints ein vereinfachtes Scrum-Board zu benutzen.

Unzureichende Pflege des Scrum-Boards

Wenn ein Scrum-Board von den Entwicklern nicht gepflegt wird, liegt dies meist daran, dass die Teammitglieder es nicht als »ihr« Werkzeug akzeptieren. Es ist fast immer ein Signal dafür, dass nicht genügend kommuniziert wird. Das Schreiben der Karten wird als Pflicht angesehen, nicht als Werkzeug für die Strukturierung oder zur Kommunikationsunterstützung.

Machen Sie den Entwicklern klar, dass ein Scrum-Board ein Werkzeug des Teams ist, die Informationsachse des Sprints. Schlagen Sie dem Team dazu ein Experiment vor: Ein Teammitglied, das mitten im Sprint aus dem Urlaub zurückkommt oder nach ein paar Tagen Krankheit wieder gesund ist, soll nur anhand des Scrum-Boards einschätzen, was bereits erledigt ist, wie der aktuelle Stand ist und was als Nächstes erledigt werden sollte.

Weiterführende Empfehlungen zum Kapitel und Thema »Vorbereitung« stellen wir unter *link.scrum-in-der-praxis.de/empfehlungen* zur Verfügung.

4.2 Product Backlog

Das dynamische Product Backlog

Casper und Finn gingen nach einem langen Arbeitstag noch gemeinsam ein Bier trinken. Sie wollten noch ein paar Kleinigkeiten besprechen und ansonsten den Abend nutzen, sich etwas besser kennenzulernen. Finn erzählte Casper eine Anekdote aus seinem Leben als Scrum Master:

»Ich habe einmal ein Team vorgefunden, das bereits seit gut zwei Wochen mit einer sehr unstrukturierten Liste von Anforderungen sprintete. Ein paar Tage vor Ende des Sprints legte der Product Owner dem Team ein vierzigseitiges Dokument mit dem Titel ›Product Backlog Sprint 1‹ mit dem Hinweis vor, das Team solle doch bitte dieses Dokument durchlesen und

bestätigen, dass genau der beschriebene Inhalt im Sprint-Review gezeigt würde. Die Argumentation für dieses Dokument bestand darin, dass der Auftraggeber sehr konventionell veranlagt sei und diese Art der Dokumentation für das Sprint-Review (die ›Lenkungsausschusssitzung‹) fordere. Der Inhalt des Dokuments wich dazu noch stark von dem vorher im Sprint Planning gemeinsam vereinbarten Sprint-Ziels des Sprints ab.«

Casper blickte Finn lange und ernst an und sagte schließlich: »Wenn der Auftraggeber das so will, dann ist das eben so. Ich sehe das genauso, das Team muss eben spuren!« Als er Finns zutiefst entsetzten Blick sah, fing er laut an zu lachen, prostete Finn zu und sagte: »Hey, das war ein Scherz! Man merkt, dass wir uns noch nicht so lange kennen.«

Ein Product Backlog ist gewissermaßen der Heilige Gral eines Scrum-Projekts. Es gibt nur einen Hüter des Grals (Product Owner), er verspricht Glückseligkeit (die Erreichung des Produktziels) und ist umgeben von einer Gemeinschaft, die Mangel leidet (die Nutzerinnen des zukünftigen Systems, vertreten durch die Stakeholder).

Wir geben in diesem Abschnitt Hinweise zum Inhalt und zur Struktur eines Product Backlog, stellen User Stories und Hilfsmittel wie Story Maps und Things-that-matter-(TTM-)Matrizen vor und beschreiben Techniken zur Zerlegung von User Stories.

4.2.1 Inhalt und Struktur

Für ein Scrum-Team gibt es nur eine »Single Source of Truth«: das Product Backlog. Es sollte alle Themen und Tätigkeiten enthalten, die für die Produktentwicklung durch das Scrum-Team relevant sind. Dies bedeutet, es enthält z.B. auch Aufgaben zum Training der Benutzer, zum Erstellen von Handbüchern oder zur Vorbereitung von Audits. Dies ist sehr wichtig, denn wenn es zusätzliche gleichwertige Aufgabenquellen gibt, ist eine ordentliche Priorisierung nicht möglich.

Aufgabentypen

Moderne Ticketverwaltungssysteme bieten eine Vielzahl von verschiedenen Typen zur Strukturierung des Product Backlog an. Wir haben jedoch die Erfahrung gemacht, dass ein Product Backlog um so komplexer wird, je mehr Aufgabentypen man hat, daher verwenden wir nur einige wenige, die wir im Folgenden erläutern:

- **Epic**
  Epics sind große Backlog Items, die in kleinere Backlog Items heruntergebrochen werden (siehe Abschnitt 4.2.5). Sie bilden die übergeordnete Struktur für User Stories und Aufgaben und enthalten daher

auch nur eine sehr grobe Beschreibung, die auf tieferer Ebene detailliert wird.

- **User Story**
  Wir werden in Abschnitt 4.2.3 sehen, dass User Stories nichts anderes als auf eine bestimmte Art und Weise beschriebene Backlog Items sind. Da User Stories heutzutage der Quasistandard für die Beschreibung von fachlichen Anforderungen sind, benutzen wir diesen Aufgabentyp regelmäßig.
- **Aufgabe**
  Es gibt manchmal klare Aufgaben, bei denen eine Formulierung als User Story künstlich wirken würde. Diese Aufgaben kommen ohne die Beschreibung aus der Benutzerperspektive ins Product Backlog. Dies können z.B. generelle technische Tätigkeiten wie das Bereitstellen einer Datenbank sein.
- **Unteraufgabe**
  Unteraufgaben sind Unterstrukturen von User Stories und Aufgaben. Sie werden während des Sprint Planning von den Entwicklern erstellt.
- **Bug/Fehler**
  Die Behebung von Fehlern stellt keine Tätigkeit im Sinne einer User Story oder einer Aufgabe dar, sondern es geht um ein Fehlverhalten bereits gelieferter Funktionalität. Die Strukturierung als eigenes Element hilft bei der Beurteilung der Qualität.

Dies sind die Strukturelemente eines Product Backlog, die aus unserer Sicht völlig ausreichend für dessen Strukturierung und Übersichtlichkeit sind.

Nicht funktionale Anforderungen (Non functional requirements, NFRs)

Oft taucht die Frage auf, wie denn mit nicht funktionalen Anforderungen umzugehen sei. Dies sind z.B. Performance, Skalierbarkeit, Portierbarkeit, Kompatibilität, Zuverlässigkeit, Verfügbarkeit, Wartbarkeit, Sicherheit, Mehrsprachigkeit oder Benutzbarkeit.

Diese Anforderungen sind natürlich völlig legitim, aber da wir in Scrum immer den Nutzer und den Wert für den Nutzer in den Fokus stellen, scheint es auf den ersten Blick schwierig, sie als Backlog Items zu formulieren. Der Trick besteht darin, sie nicht als eigenständige Aufgaben zu betrachten, sondern sie im Rahmen der Umsetzung der werthaltigen Funktionalität gleich mit zu berücksichtigen. Wenn z.B. klar ist, dass ein Produkt auch mit älteren iOS-Versionen funktionieren soll (Kompatibilität), wird bei der Entwicklung jeder

einzelnen Funktionalität darauf geachtet, anstatt ein Backlog Item »Unterstützung iOS 9.3.6« zu erstellen.

Neben der inhaltlichen Komponente stellt die Struktur des Product Backlog einen wichtigen Erfolgsfaktor dar. Zu wenige Informationen können zu Missverständnissen führen, wenn zwischen Entwicklern und Product Owner keine intensive Kommunikation stattfindet. Zu viele Informationen werden oft unübersichtlich, widersprechen sich manchmal sogar (z.B. wenn die Beschreibung eines Formulars nicht mehr dem Designentwurf entspricht) und erinnern stark an Fachkonzepte aus dem traditionellen Projektmanagement. In unserer Praxis haben wir sehr verschieden aufgebaute Product Backlogs angetroffen.

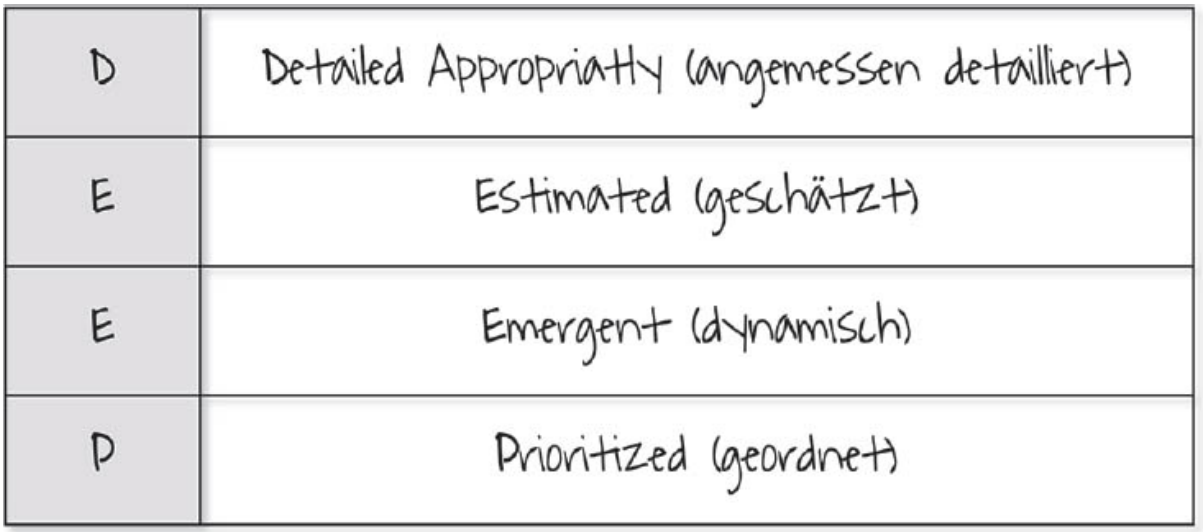

Abb. 4–5 *DEEP-Kriterien*

Prinzipiell gilt, dass ein Product Backlog sich an den DEEP-Kriterien orientieren sollte, die von Mike Cohn und Roman Pichler vorgeschlagen wurden:

- **Angemessen detailliert**
  Backlog Items, die zeitnah umgesetzt werden sollen, sollten so gut verstanden sein, dass sie problemlos in einem der nächsten Sprints umgesetzt werden können. Weiter in der Zukunft liegende Backlog Items müssen jetzt noch nicht detailliert betrachtet werden.
- **Geschätzt**
  Damit das Product Backlog auch zur Planung genutzt werden kann, sollten die schon genauer beschriebenen Backlog Items geschätzt sein.
- **Dynamisch**
  Das Product Backlog ist kein festgeschriebenes Fachkonzept, sondern ein dynamisches Dokument. Anhand des Feedbacks, das man bekommen hat,

und anhand der Dinge, die man zwischenzeitlich gelernt hat, werden Backlog Items geändert, gelöscht, ergänzt oder umpriorisiert.

- **Geordnet**
  Durch die Ordnung des Product Backlog nach Wert wird sichergestellt, dass immer die höchstwertigen Backlog Items oben stehen und als Nächstes für die Umsetzung betrachtet werden.

Achten Sie darauf, dass das Product Backlog nicht überstrukturiert wird. Dies kann dazu führen, dass es unübersichtlich wird, weil wichtige Informationen verstreut liegen. Außerdem erhöht sich der Wartungsaufwand, um alle Felder korrekt zu füllen.

Im Folgenden stellen wir eine Version vor, die unserer Erfahrung nach eine ausreichende generische Grundstruktur darstellt. Es kann natürlich das eine oder andere Element weggelassen oder hinzugefügt werden, falls das Projekt es erfordert. Spätestens zu Beginn eines Sprints sollten diese Informationen für alle relevanten Backlog Items vorliegen.

- **ID**
  Zur eindeutigen Identifizierung wird z.B. eine laufende Nummer (#4711) oder eine Kombination aus ID und Abkürzung des Scrum-Teams (SIDP-2022) verwendet. Wird eine Software für die Pflege des Product Backlog eingesetzt, wird diese Nummer automatisch vergeben. Sie dient nicht nur als eindeutige Identifizierung, sondern gibt auch Auskunft über die Aktualität der Anforderung.

Wenn Sie im Backlog Refinement (siehe Abschnitt 4.4) feststellen, dass ein Backlog Item schon sehr lange im Product Backlog steht, fragen Sie nach, ob es noch relevant ist. Möglicherweise ist es längst überholt und kann geschlossen werden.

- **Zusammenfassung**
  Hier wird kurz und knapp zusammengefasst, worum es bei diesem Backlog Item geht. Idealerweise ist die Zusammenfassung so kurz, dass sie in einer Listendarstellung in eine Zeile passt.

- **Beschreibung**
  Dies ist der eigentliche Inhalt des Backlog Item. Hier wird beschrieben, worum es in dem Backlog Item geht. Sofern mit User Stories (siehe Abschnitt 4.2.3) gearbeitet wird, sollte dieser Abschnitt mit der formulierten User Story beginnen. Darüber hinaus können hier ergänzende Informationen zum besseren Verständnis hinterlegt werden (z.B. Links zu detaillierteren Anforderungen, Besonderheiten für den Test oder wichtige Ansprechpartner).
- **Akzeptanzkriterien**
  Akzeptanzkriterien beschreiben die fachlichen Bedingungen, unter denen ein Backlog Item als fertig gilt. Sie stellen die fachlichen Abnahmekriterien des Product Owners dar. Wir gehen in Abschnitt 4.2.3 auf die Akzeptanzkriterien ein.
- **Schätzung**
  Viele Scrum-Teams schätzen die Komplexität von Backlog Items. Diese Schätzung wird zur Planung der Sprints benötigt (siehe Abschnitt 4.3).
- **Epic**
  Backlog Items gehören meist in den Kontext eines Epics, daher wird das zugehörige Epic im jeweiligen Backlog Item referenziert.
- **Task**
  Als Tasks werden die konkreten Aufgaben bezeichnet, die zur Erledigung eines Backlog Item (vgl. Abschnitt 4.1.2) notwendig sind. Sie werden im Sprint Planning erstellt und bilden eine Unterstruktur der Backlog Items.

In einigen Ticketverwaltungssystemen werden diese Tasks auch als Subtasks oder Unteraufgaben bezeichnet und als Strukturelement auf der Ebene der User Stories eingesetzt.

- **Priorisierung**
  Das Product Backlog kann nach unterschiedlichen Kriterien geordnet sein, z.B. nach Wert, Risiko, Priorität oder Notwendigkeit. Die Priorisierung legt diese Ordnung innerhalb des Product Backlog fest.

Über die oben beschriebene Grundstruktur hinaus sind weitere Spalten für andere, im jeweiligen Projekt benötigte Informationstypen denkbar.

Außerdem kann es Product Backlogs mit unterschiedlichen Schwerpunkten geben. Ein Product Backlog sollte nicht immer ein Produkt in Form einer Anwendung für Nutzerinnen beschreiben, denn Scrum ist nicht auf die Produktentwicklung beschränkt. Für viele unterschiedliche Arten von Projekten bietet sich ein Product Backlog als strukturierte Anforderungsdokumentation an.

4.2.2 Anforderungsworkshops

Nachdem der Product Owner allen Beteiligten im Projekt-Kick-off das Produktziel des Projekts vorgestellt hat, ist dieses in konkrete Backlog Items herunterzubrechen. In traditionell geführten Projekten geschieht dies in der Regel dadurch, dass jemand aus dem Fachbereich sich hinsetzt und ein mehr oder weniger umfangreiches Dokument verfasst, das das zu erstellende Produkt und sein Verhalten bis ins letzte Detail exakt beschreibt. In Scrum-Projekten hingegen wird versucht, frühzeitig und eng mit Kunden, Nutzerinnen und Stakeholdern zusammenzuarbeiten, um so eine möglichst umfassende Sicht auf das zukünftige Produkt zu entwickeln. Außerdem werden die Entwickler in den Prozess eingebunden, um sowohl deren fachliche Expertise zu nutzen als auch die Machbarkeit der Ideen und Wünsche zu beurteilen. Dazu bietet es sich an, gemeinsame Anforderungsworkshops durchzuführen. In der Vorbereitung eines Scrum-Projekts können diese Workshops sich je nach Projektgröße über mehrere Tage hinziehen. Als Visualisierung der Workshop-Ergebnisse schlägt Roman Pichler ein Product Vision Board vor [URL:Pichler b]. Im laufenden Sprint werden neu hinzukommende oder veränderte Anforderungen in der Regel in den meist wöchentlich stattfindenden Backlog Refinements (siehe Abschnitt 4.4) besprochen. Anforderungsworkshops können daher unterschiedliche Schwerpunkte und damit unterschiedliche Strukturen haben, die wir im Folgenden beschreiben.

Produktgestaltungsworkshop

Für einen Produktgestaltungsworkshop ist inhaltlich bis auf das Produktziel und viele gute Ideen keine weitere Vorbereitung notwendig. Neben dem Scrum-Team sollten idealerweise Endkunden bzw. -nutzer dabei sein, zumindest aber ihre Vertreter innerhalb der Organisation, also Stakeholder. Dies können je nach Produkt auch Personen aus dem Vertrieb, dem Marketing, der Personal- oder der Finanzabteilung sein. Alle zusammen überlegen sich »ihr« Produkt. Aus ersten Ideen werden Epics, aus Epics werden Backlog Items bzw. User Stories (siehe Abschnitt 4.2.3).

Diese Workshops sind keine Zwei-Stunden-Workshops, die man neben dem Tagesgeschäft erledigt. Denken Sie in ganzen Tagen, je nach Projektgröße können mehrere Tage notwendig sein. Vielleicht können Sie sich mit Ihrem Scrum-Team außerhalb der Firma zurückziehen, um Ablenkungen zu vermeiden. Berücksichtigen Sie die anfallende Zeit im Sprint Planning.

Besonders wichtig an diesen Workshops ist der psychologische Aspekt. Alle zusammen gestalten gemeinsam »ihr« Produkt, es wird also nicht von einer Einzelperson vorgegeben. Leider wird diese Form eines Anforderungsworkshops viel zu oft übersprungen, weil aktuelle Organisationsformen es immer noch vorsehen, dass Produktmanager Produkte entwerfen, die von den Entwicklern anschließend umgesetzt werden sollen.

Im Folgenden stellen wir Ihnen einen möglichen Ablauf eines Produktgestaltungsworkshops vor:

- Alle Teilnehmenden ermitteln gemeinsam in einem kreativen Brainstorming mögliche Nutzerrollen des Produkts. Dabei geht es nicht um Perfektion, sondern um Kreativität, es dürfen also auch spaßige Vorschläge gemacht werden.
- Aus den ermittelten Rollen werden mittels Dot-Voting diejenigen ermittelt, die nach Meinung der Teilnehmenden die Zielgruppe repräsentieren.
- Nun werden Paare gebildet und jedes Paar wählt sich eine Rolle aus. Anschließend beschreibt jedes Paar aus dieser Perspektive die Anforderungen an das Produkt. Die jeweiligen Anforderungen können z.B. auf große Post-its geschrieben werden. Beenden Sie die Phase, wenn Sie bemerken, dass die Luft raus ist.
- Nun kommen wieder alle zusammen und jedes Paar stellt den anderen seine Vorschläge vor. Die anderen Teilnehmenden geben Feedback.
- Nun kann auf zwei Arten weitergemacht werden: Entweder bearbeiten die Zweiergruppen mit dem Feedback der anderen und den Ideen die bisher bearbeitete Rolle weiter oder die Teilnehmenden verteilen sich neu und bearbeiten in einer zweiten Runde eine andere Rolle und ergänzen die bereits vorliegenden Anforderungen.

Diese Übung kann über mehrere Runden fortgeführt werden. Der Product Owner hat nun durch eine kurze kreative Übung einen großen Pool an Ideen für das Produkt bekommen. Er sollte sie nun konsolidieren und aufbereiten. Sofern

genug Ideen vorhanden sind, könnte der nächste Schritt der unten beschriebene Big-Picture-Workshop sein oder die Überführung in eine →User Story Map (siehe Abschnitt 4.2.4).

Wenn Sie in der Situation sind, ein neues Scrum-Team aufzubauen, um ein neues Produkt zu schaffen, binden Sie die Teammitglieder so früh wie möglich ein. Das Team wird es Ihnen auf lange Sicht durch Loyalität danken.

Big-Picture-Workshop

Um einen gesamten Überblick über Abhängigkeiten und die ausstehenden Aufgaben zu erhalten, empfiehlt es sich, gemeinsam in einem Workshop ein »Big Picture« zu erstellen. Vor diesem Workshop liegt bereits ein Entwurf eines Product Backlog mit Backlog Items vor. Idealerweise ist es im Rahmen eines oben beschriebenen Produktgestaltungsworkshops entstanden, in der Praxis wird es aber meist durch den Product Owner erstellt. Oft werden die Entwickler in diesem Workshop zum ersten Mal mit den Backlog Items konfrontiert. Ziel ist es, die Entwickler zu informieren und das »Big Picture« zu vermitteln. Auf diese Weise soll eine erste Einschätzung der Entwickler hinsichtlich Machbarkeit und technischer Abhängigkeiten erlangt werden. Häufig gibt es als Zugabe noch fachliche Verbesserungsvorschläge.

Eine Agenda für einen Big-Picture-Workshop könnte folgendermaßen aussehen:

- Der Product Owner stellt das Produktziel vor und erläutert die daraus resultierenden Backlog Items. Wir empfehlen für diesen Schritt den Einsatz einer Story Map (siehe Abschnitt 4.2.4). An dieser Stelle sollte sich der Product Owner den kritischen Fragen der Entwickler stellen, indem er z.B. den Wert für Backlog Items erklärt, die keinen klar erkennbar wirtschaftlichen Nutzen für das Produktziel aufweisen.
- Die Backlog Items werden einzeln durchgegangen. Mittels der TTM-Matrix (siehe Abschnitt 4.3.4) nähern sich alle der Komplexität der Backlog Items und dokumentieren die Ergebnisse. Oft werden in diesem Schritt Abhängigkeiten erkannt, die eine veränderte Reihenfolge oder eine Zerlegung der Backlog Items notwendig machen. Die Änderungen werden gemeinsam durchgeführt.
- Mit dem gewonnenen fachlichen Verständnis und dem Gefühl für die Komplexität wird als letzter Schritt eine Schätzung durchgeführt. Da es sich

meist um eine größere Menge von Backlog Items handelt, empfiehlt sich hier der Einsatz des Team Estimation Game (vgl. Abschnitt 4.3.3).

In den folgenden Abschnitten beschreiben wir Hilfsmittel für eine besondere Ausprägung von Backlog Items, die User Stories.

4.2.3 User Stories

User Stories stellen eine Spezialform der allgemeinen Backlog Items dar, die durch Mike Cohn eine weite Verbreitung in Scrum-Projekten gefunden hat. Ralf Wirdemann beschreibt User Stories im Zusammenhang mit Scrum sehr ausführlich in seinem Buch [Wirdemann 2017], sodass wir hier nur kurz ein paar Dinge in Erinnerung rufen möchten, bevor wir konkrete Praxistipps geben. Mike Cohn [Cohn 2004] definiert eine User Story folgendermaßen: »*User Stories sind kurze, einfache Beschreibungen einer Funktion, die aus der Perspektive der Person erzählt werden, die die neue Fähigkeit wünscht, normalerweise ein Nutzer oder Kunde des Systems.*« Sie folgen in der Regel einer einfachen Vorlage:

»***Als*** *[Benutzerrolle]*

möchte ich *[Ziel]*,

um zu *[Grund für das Ziel]*.«

Alistair Cockburn hat einmal gesagt, eine User Story sei ein Versprechen zur Kommunikation über den Inhalt der User Story. Eine unseres Erachtens sehr treffende Definition.

Der wichtige dritte Teil im Muster der User Story, der den Nutzen der Anforderung beschreibt, wird leider oft vernachlässigt, da scheinbar ja schon alles gesagt ist. Um dem entgegenzuwirken, versuchen Sie doch mal, die User Story umzudrehen und den Nutzen voranzustellen:

Um [Grund für das Ziel] möchte ich als [Benutzerrolle] [Ziel].

Bereits 2001 hat Ron Jeffries in [URL:Jeffries b] mit dem prägnanten Kürzel »CCC« (vgl. Abb. 4–6) auf die drei wichtigsten Elemente einer User Story aufmerksam gemacht.

Abb. 4–6 *3C-Kürzel*

- **Karte**
  Das erste »C« steht für »Card«, sinnbildlich also für die (Kartei-)Karte, auf die die User Story geschrieben wird und auf deren Vorderseite sie passen sollte. Heutzutage nutzt wohl kaum noch jemand Karten, sondern meist Post-its oder elektronische Tools.
- **Konversation**
  Das zweite »C« steht für »Conversation«, also ein Gespräch. Es handelt sich um den Austausch zwischen Entwicklern und Product Owner und findet in der Regel während des Backlog Refinement statt. Inhalt des Gesprächs ist das gemeinsame Verständnis aller Beteiligten, was genau unter der mit einem Satz auf der Karte sehr vage formulierten Anforderung zu verstehen ist.
- **Akzeptanzkriterien (auch: Conditions of Acceptance)**
  Mit der stattgefundenen Konversation wurden hoffentlich alle Fragen geklärt, und am Ende haben alle ein gemeinsames Verständnis über den Inhalt der Anforderung. Damit dies auch so bleibt, wird das Ergebnis festgehalten. Formal betrachtet steht das dritte »C«, die »Confirmation«, also für die Dokumentation einer vorher stattgefundenen Konversation. In der Praxis sprechen wir hier von Akzeptanzkriterien.

Akzeptanzkriterien

Akzeptanzkriterien sind ein untrennbarer Bestandteil einer User Story, ohne sie wäre eine User Story nicht vollständig. Sie sind Bestandteil der Umsetzungsvereinbarung, die Product Owner und Entwickler während des Sprint Planning treffen. Product Owner sollten sich also genau überlegen, welche Akzeptanzkriterien für die Erledigung eines Backlog Item wichtig sind. Für die Entwickler sind Akzeptanzkriterien nicht nur zu erreichende Ziele, sondern aus guten Akzeptanzkriterien lassen sich zudem einfach Tests ableiten.

Selbst wenn Sie in Ihrem Projekt nicht mit User Stories arbeiten, lohnt es sich, Anforderungen mit Akzeptanzkriterien zu versehen, da sie ein gemeinsames Verständnis des erwarteten Ergebnisses herstellen.

Akzeptanzkriterien zu schreiben, bedeutet allerdings nicht, das bekannte Pflichtenheft oder Fachkonzept aus dem traditionellen Projektmanagement auf Scrum zu übertragen. Sie geben zunächst nur den Rahmen vor, in dem sich die Entwickler während des Sprints bewegen können. Details werden laufend im Dialog zwischen Product Owner und Entwicklern geklärt und ggf. in den Akzeptanzkriterien ergänzt.

Ermutigen Sie die Entwickler dazu, User Stories ohne Akzeptanzkriterien abzulehnen. Unterstützen Sie den Product Owner, indem Sie gemeinsam mit ihm vor den relevanten Treffen (Backlog Refinement, Sprint Planning) durch die User Stories gehen und fehlende Akzeptanzkriterien aufzeigen.

Auch wenn die Details noch ergänzbar und verhandelbar sind, steht der Rahmen, also die Akzeptanzkriterien, spätestens mit dem Ende des Sprint Planning fest, meist schon am Ende des Backlog Refinement (vgl. Abschnitt 4.1.2, Definition of Ready). Vergessene Kriterien einer bereits in der Umsetzung befindlichen User Story dürfen nicht einfach nachträglich durch den Product Owner hinzugefügt werden, sondern sollten in eine neue User Story in einem neuen Sprint einfließen.

Manchmal handelt der Product Owner mit den Entwicklern aus, dass ein zusätzliches Akzeptanzkriterium noch berücksichtigt wird, dafür aber eine andere User Story wegfällt. Dies sollte jedoch die Ausnahme sein, denn andernfalls wird der Gedanke von Scrum korrumpiert, dass der Sprint ein geschützter Raum ist, in dem keine Änderungen mehr vorgenommen werden dürfen. Früher oder später wird es sonst zum Normalfall, kurzfristig noch etwas zu ändern.

Besser ist es, Product Owner dahingehend zu unterstützen, dass genügend Zeit und Domänenwissen vorliegt, um hinreichend gute User Stories und Akzeptanzkriterien zu schreiben.

INVEST-Kriterien

Gute User Stories berücksichtigen das sogenannte INVEST-Prinzip, sie erfüllen somit die folgenden Anforderungen in Abbildung 4–7.

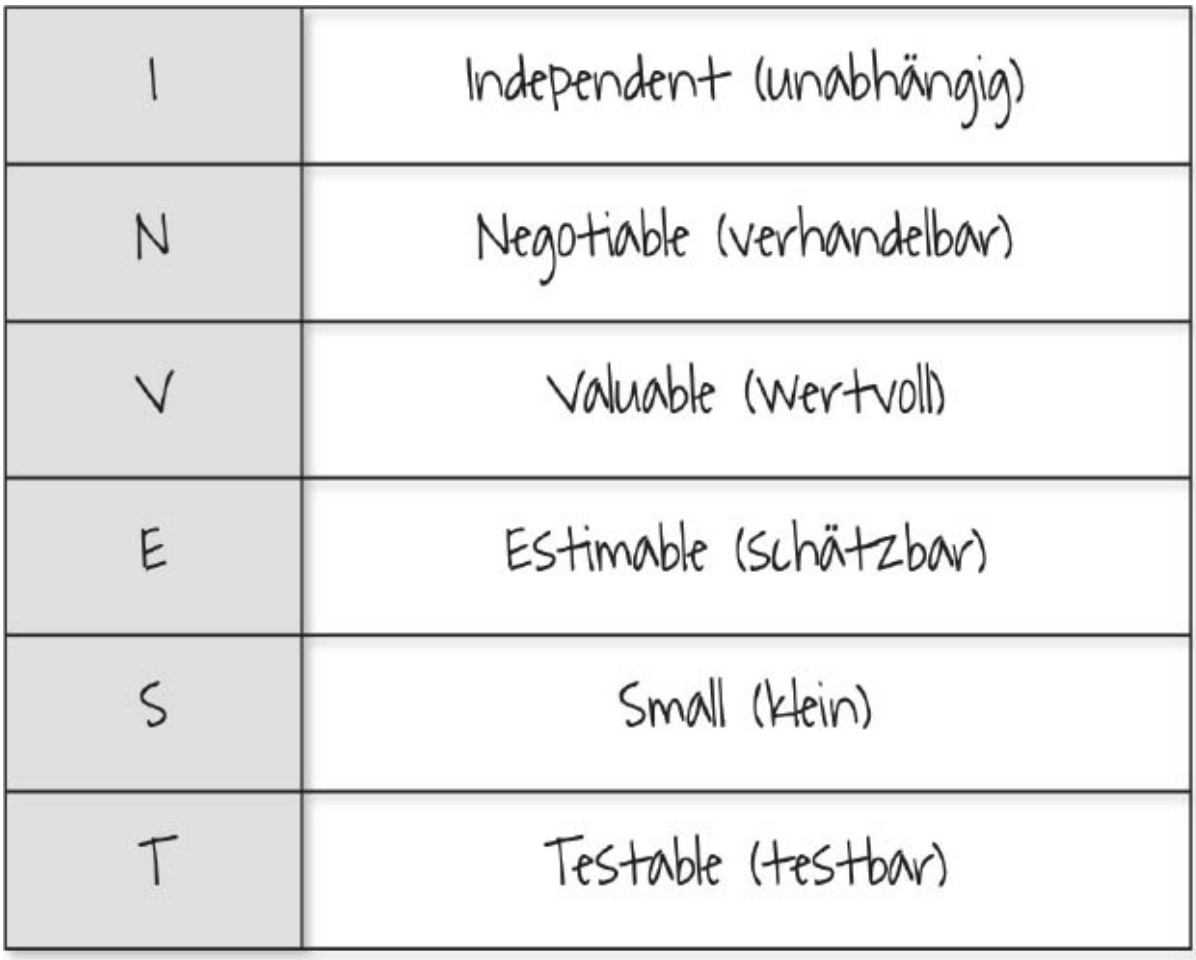

Abb. 4–7 *INVEST-Kriterien*

Beim Beschreiben von Anforderungen ist es nützlich, sich diese Kriterien immer wieder vor Augen zu führen, da sie dabei helfen, sich auf das Wesentliche zu fokussieren und zu hinterfragen, ob ein konkretes Akzeptanzkriterium wirklich wichtig ist, um schnell Wert zu schaffen. Oft wird dabei erkannt, dass einzelne Akzeptanzkriterien gut in eine nachfolgende User Story ausgelagert werden können. Im Folgenden geben wir einen Überblick über die INVEST-Kriterien:

- **Unabhängig**
  Um möglichst flexibel in der Umpriorisierung von User Stories zu sein, sollten sie keine Abhängigkeiten untereinander haben. Zugegeben, diese Anforderung ist nicht einfach zu erfüllen, aber im Zusammenspiel mit den

Zerlegungstechniken, die wir in Abschnitt 4.2.5 vorstellen, ist es fast immer möglich.

- **Verhandelbar**
  User Stories sind keine Fachkonzepte, sie beschreiben also die Anforderung nicht bis ins letzte Detail. Das bedeutet im Umkehrschluss, dass es immer Unschärfen gibt, die der Product Owner als Verhandlungsmasse mit den Stakeholdern, aber auch den Entwicklern nutzen kann.
- **Wertvoll**
  Jede User Story sollte einen erkennbaren Wert für die Nutzerin besitzen. Sollte dieser Wert nicht erkennbar sein, sollte die User Story entsprechend geändert werden.
- **Schätzbar**
  User Stories werden für die Planung herangezogen, daher sollten sie so beschaffen sein, dass die Entwickler sie schätzen können.
- **Klein**
  Um sich eine größtmögliche Flexibilität zu erhalten, sollten User Stories möglichst klein gehalten sein, dabei aber natürlich nicht die anderen INVEST-Kriterien verletzen, insbesondere die Werthaltigkeit. Als Minimalanforderung sollte eine User Story so klein sein, dass sie ohne größeren Forschungsaufwand in einem Sprint umsetzbar ist. 6 bis 10 User Stories in einem Sprint sind in der Praxis keine Seltenheit.
- **Testbar**
  Die Umsetzung der schönsten User Story nützt nichts, wenn anschließend nicht sichergestellt werden kann, dass sie ordnungsgemäß funktioniert.

Auch wenn die INVEST-Kriterien immer im Kontext von User Stories genannt werden, empfehlen wir, sie universell für anders beschriebene Backlog Items anzuwenden.

Nachdem alle User Stories vorliegen, wird es Zeit, sie den Stakeholdern vorzustellen und möglicherweise sogar im ganzen Unternehmen öffentlich zu machen, um auf diese Weise Feedback zum geplanten Produkt zu erhalten. Dafür bieten sich Story Maps an, die wir im folgenden Abschnitt beschreiben.

4.2.4 User Story Mapping

Der Scrum Guide sagt über das Product Backlog (*scrumguides.org*):

> *»Das Product Backlog ist eine emergente, geordnete Liste der Dinge, die zur Produktverbesserung benötigt werden. Es ist die einzige Quelle von Arbeit, die durch das Scrum Team erledigt wird.«*

In der Tat finden sich meist Product Backlogs, die nach dem Geschäftswert sortiert sind, d.h., diejenigen Backlog Items werden am höchsten priorisiert, die den größten Wert für das Unternehmen darstellen. Dies ist sinnvoll, hat jedoch zur Folge, dass zusammengehörige Funktionalitäten in kleinere Backlog Items zerlegt werden und sich verteilt in der Listendarstellung des Product Backlog wiederfinden, sodass der Bezug verloren geht. Wer schon einmal versucht hat, seinen Stakeholdern das Produkt anhand des Product Backlog zu erklären oder es ihnen einfach nur zum Lesen gegeben hat, weiß, dass dies ein schwer nachvollziehbares Unterfangen ist.

Jeff Patton hat sich mit dieser Thematik beschäftigt und das sogenannte *Story Mapping* entwickelt. User Story Maps erlauben es, das gesamte Produkt zu visualisieren und damit übersichtlich, erklärbar und einfacher priorisierbar zu machen. Sinnvoll finden wir daran, dass der Ausgang nicht der Geschäftswert ist, sondern das, was für die Kunden oder Nutzerinnen am Wertvollsten ist. [URL:Patton] sagt dazu:

> *»Wir verbringen viel Zeit damit, mit unseren Kunden zusammenzuarbeiten. Wir arbeiten hart daran, ihre Ziele, ihre Benutzer und die wichtigsten Teile des Systems, das wir aufbauen könnten, zu verstehen. Dann kommen wir endlich zu den Details – den Teilen der Funktionalität, die wir bauen möchten. In meinem Kopf sehe ich einen Baum, dessen Stamm aus den Zielen oder gewünschten Vorteilen besteht, die das System antreiben; die großen Äste sind die Benutzer; die kleinen Äste und Zweige sind die Fähigkeiten, die sie benötigen; und schließlich sind die Blätter die User Stories, die klein genug sind, um sie in die Entwicklungsiterationen einzubringen. Nach all dieser Arbeit, nach der Etablierung eines gemeinsamen Verständnisses, habe ich das Gefühl, dass wir alle Blätter vom Baum nehmen, sie in einen Laubsack packen – und dann den Baum*

fällen. Das ist es, was ein flaches Backlog für mich ist. Ein Sack mit kontextfreiem Mulch.«

Das Konzept der User Story Maps möchten wir Ihnen nachfolgend vorstellen.

Aufbau einer User Story Map

Der Aufbau einer User Story Map ist zu vergleichen mit einer Geschichte, die wir erzählen, die mit »Es war einmal …« beginnt. Wie in Abbildung 4–8 unternimmt eine Hauptfigur (z.B. Nutzerinnen, Persona, Akteure) unterschiedliche Aktivitäten, die in einer sinnvollen Abfolge stehen, sodass eine gelungene Geschichte daraus wird.

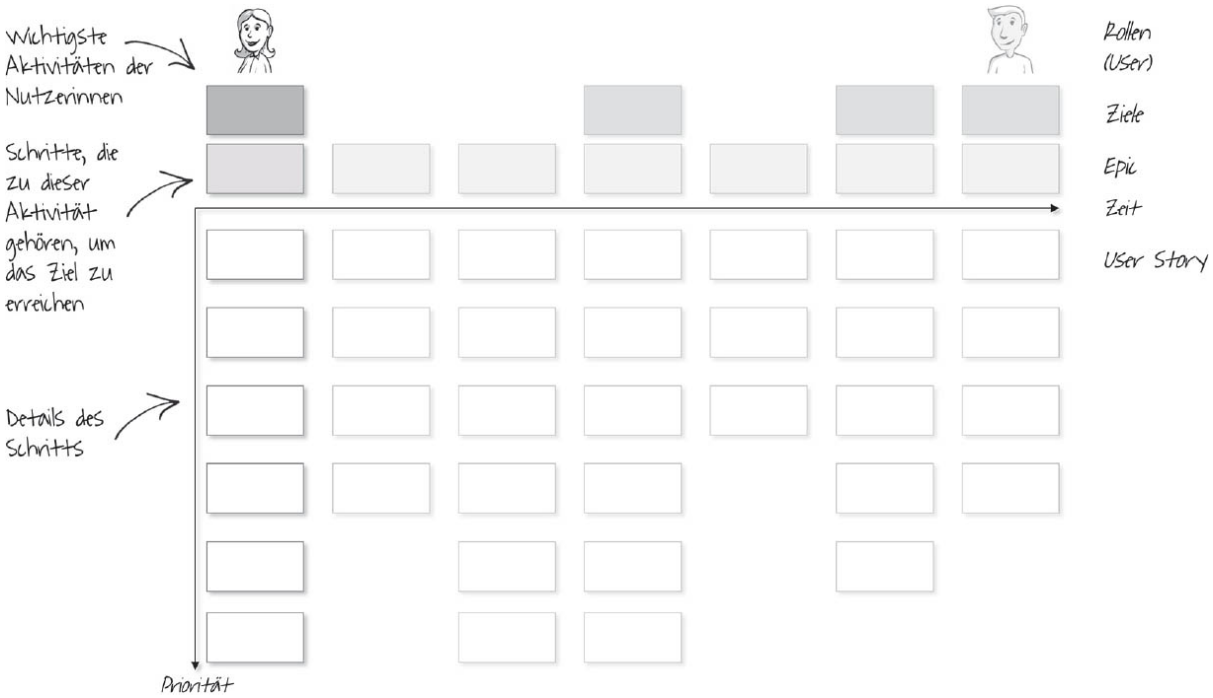

Abb. 4–8 *Aufbau einer User Story Map*

Diese wichtigsten Aktivitäten spiegeln Ziele wider, die eine Benutzerrolle erreichen möchte. Die einzelnen Schritte, die zu diesen Aktivitäten anfallen, werden unter den Aktivitäten in eine zeitliche Abfolge gebracht. Diese werden als Epics bezeichnet und bilden den »Narrative Flow« ab, also den sinnvollen Erzählfluss.

Wir visualisieren gerne die unterschiedlichen Nutzerinnen über den für diese Rolle wichtigsten Aktivitäten, um eine klare Abgrenzung und Übersichtlichkeit zu erzielen. Jede

Information, die uns zu diesem *User* vorliegt, kann dort ausgedruckt (oder hinterlegt) werden.

So wird z. B. sichtbar, dass wir für Nutzerinnen, die Geld für das Produkt oder die Dienstleistung zahlen, jede Menge guter Ideen haben, aber nicht für diejenigen, die nichts bezahlen. Dieser Überblick führt zu sinnvollen Diskussionen und ggf. zur Anpassung des *Flows* der Geschichte.

Nun folgen die einzelnen Details für jede dieser einzelnen Schritte. Diese Details werden durch *User Stories* abgebildet. In einem Story-Mapping-Workshop findet im Grunde hier ein erstes Herunterbrechen der Epics in kleinere User Stories statt.

Wenn also eine User Story Map aufgebaut wird, wird eine Geschichte erzählt, die einen sinnvollen zeitlichen Ablauf der Aktivitäten eines oder mehrerer Nutzerrollen enthält und eine priorisierte Liste an Anforderungen (User Stories) zeigt. Es gibt also zwei Dimensionen in denen eine Priorisierung und Strukturierung stattfindet. Wir erleben es häufig bei der Verwendung von User Story Maps, dass die Aktivitäten (Epics) und Schritte zum Ziel (Features) durchaus konstant bestehen bleiben, während die Details (User Story) sich durch neue Erkenntnisse verändern. Diese Erkenntnisse werden im Backlog Refinement (siehe Abschnitt 4.4) erlangt.

Erstellen einer User Story Map

Nehmen wir einmal folgendes einfache Beispiel an: Eine Konferenzbesucherin möchte einen besuchten Vortrag bewerten. Dies erfolgt in zwei Schritten, die als Aktivitäten bezeichnet werden:

Auswählen des zu bewertenden Vortrags

Bewerten des Vortrags

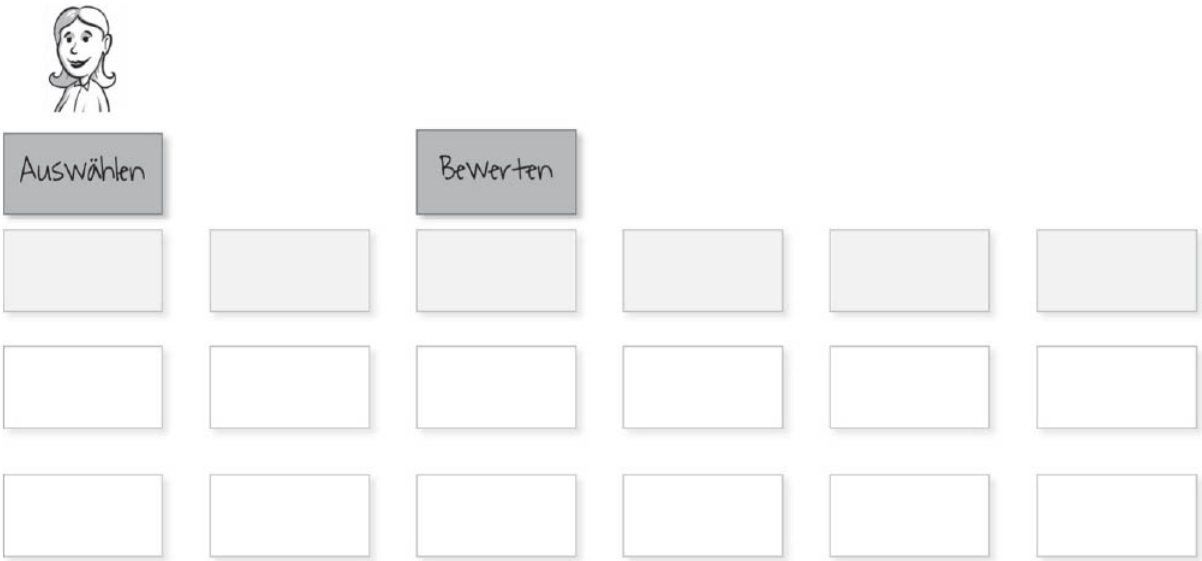

Abb. 4–9 *Aktivitäten einer Nutzerin*

Im nächsten Schritt wird überlegt, welche Schritte innerhalb dieser Aktivitäten ausgeführt werden, um zum Ziel zu gelangen (vgl. Abb. 4–10). Dabei geht es nicht um Details, sondern um eine sehr grobe Beschreibung. Diese Schritte werden unterhalb der Aktivitäten in einer Zeile dargestellt. Sie zeigen, wie die Reise der Benutzer durch das Produkt aussieht. Da sie im nächsten Schritt in User Stories heruntergebrochen werden, stellen diese Schritte die Epics dar.

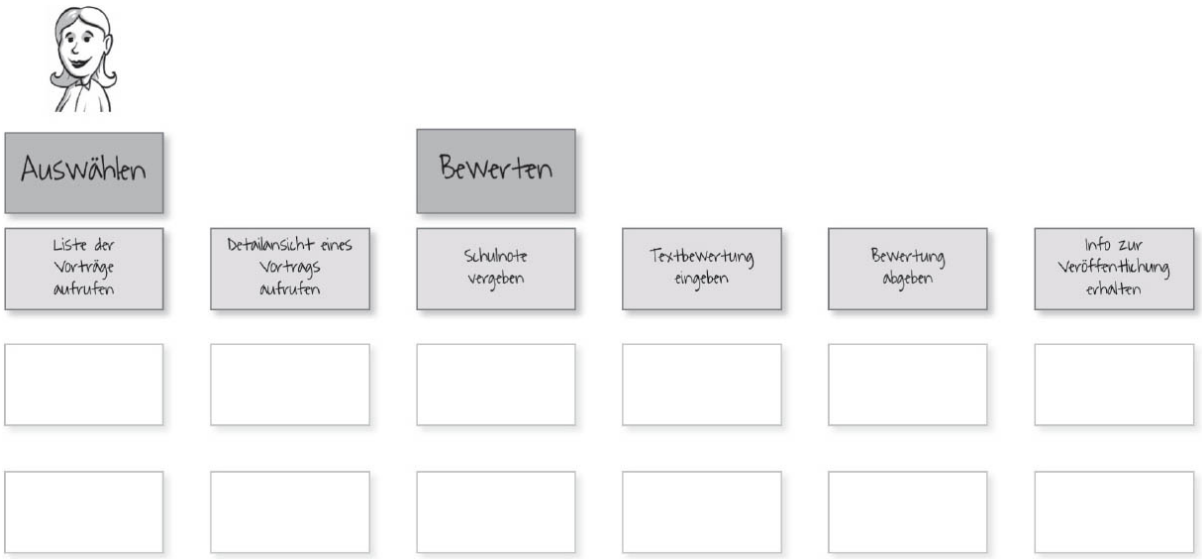

Abb. 4–10 *Schritte einer Nutzerin*

Im Anschluss werden die Epics in einzelne User Stories zerlegt. Die User Stories werden unterhalb der Epics angeordnet, sodass Spalten mit Epics als Spaltenüberschriften entstehen. Das Ergebnis sieht etwa so aus wie in Abbildung 4–11 gezeigt.

Jeff Patton vergleicht dieses Bild mit einer Wirbelsäule und davon abgehenden Rippen, daher hat sich für die Epic-Reihe der Begriff »Backbone« (Wirbelsäule) durchgesetzt.

Nun können die einzelnen User Stories priorisiert werden. Wir verwenden hierfür gerne das →MuSCoW-Prinzip, wobei die wichtigsten unverzichtbaren User Stories weiter oben angeordnet werden und die »Nice to have«-Stories weiter unten. Aus der Perspektive der Konferenzbesucherin konnten so die Aktivitäten und einzelnen Schritte in User Stories heruntergebrochen und priorisiert werden.

Durch die Visualisierung mithilfe einer User Story Map entsteht eine hervorragende Gesprächsgrundlage sowohl für den Dialog mit den Stakeholdern als auch innerhalb des Scrum-Teams. Zusammenhänge können auf einen Blick erkannt werden.

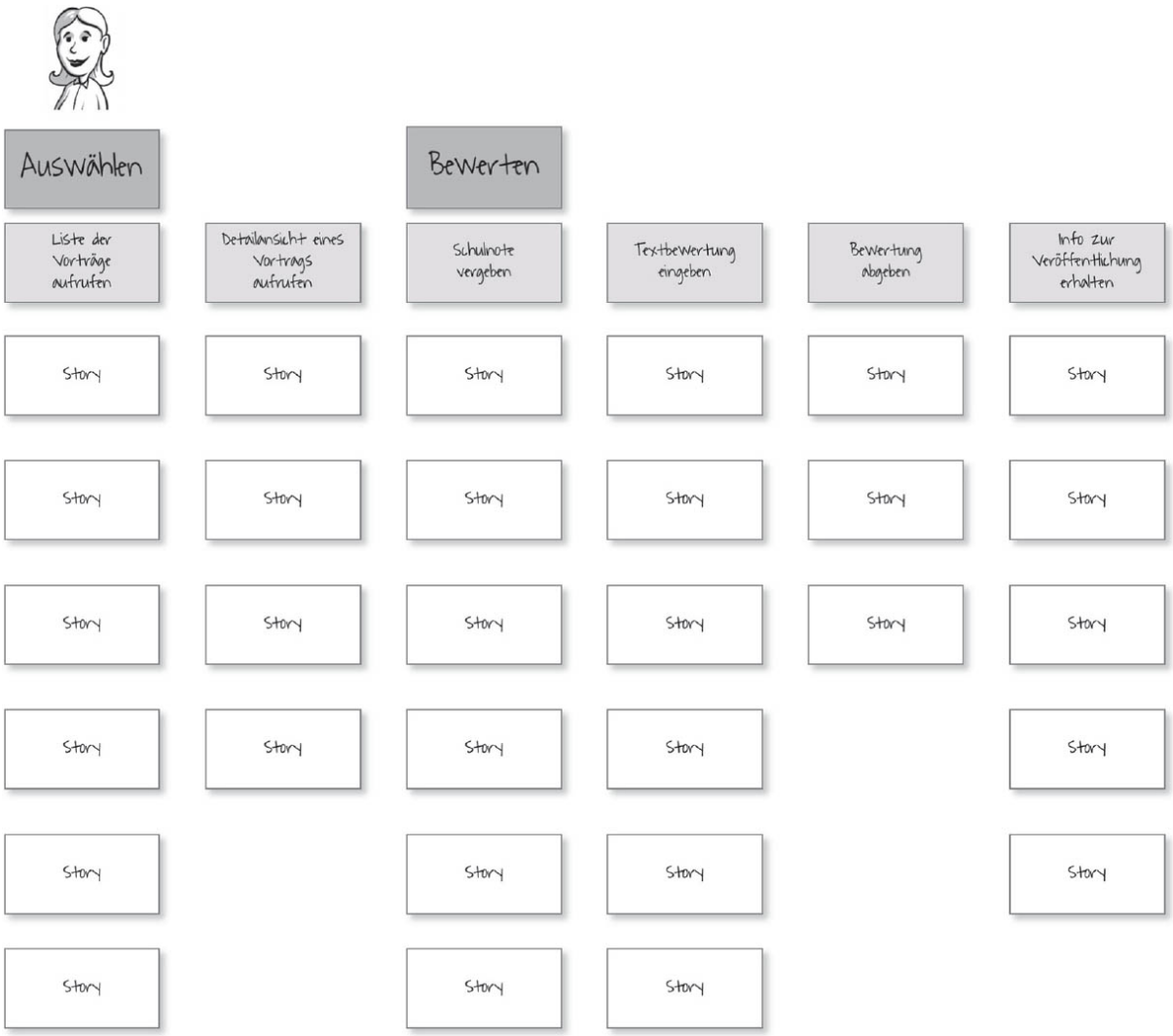

Abb. 4–11 *Details der Schritte*

In einem Fall aus unserer Praxis wurde vom Product Owner und vom Scrum Master gemeinsam ein erster Entwurf einer User Story Map erstellt und dem Auftraggeber vorgestellt. Hier zeigte sich der Wert dieser Form der Visualisierung: Da alle Themen übersichtlich aufbereitet waren, konnte der Auftraggeber Missverständnisse bzw. falsche Annahmen zurechtrücken. Einige User Stories, die für das erste Release vorgesehen waren, waren ihm gar nicht so wichtig, während andere, deren Dringlichkeit dem Product Owner nicht bewusst war, höher priorisiert werden sollten. Ohne die User Story Map wäre es sehr viel schwieriger gewesen, dieses wertvolle Feedback zu diesem frühen Zeitpunkt zu bekommen.

Wir erstellen die User Story Map oft gemeinsam zum Start eines Projekts mit internen und externen Stakeholdern, sodass alle beteiligt sind, alle Interessen gehört und das Gesamtbild und die nächsten Schritte verstanden werden. Dazu laden wir die Stakeholder zu einem gemeinsamen Workshop mit dem Scrum-Team ein und entwerfen die »Geschichte«, indem alle die User Story Map wie oben beschrieben Schritt für Schritt entwickeln. Alle Beteiligten sind aufgefordert, aktiv Vorschläge zu machen und Ideen zu äußern, sodass am Ende eine gemeinsam entwickelte User Story Map steht, mit der sich alle identifizieren können. Alternativ ist es auch möglich, dass der Product Owner mit einer bereits erstellten User Story Map die Anwesenden Schritt für Schritt durch die Geschichte führt und dann die Anwesenden mit einbezogen werden.

Product Owner können nach der Erstellung die User Story Map im Product Backlog abbilden. Die Story Map sollte öffentlich sichtbar oder elektronisch zugänglich gemacht werden. Auf diese Weise wird sie immer wieder für Feedback und Diskussionen sorgen und kann weiterentwickelt werden.

Nach dem Story Mapping kann mit der Ausarbeitung der Details der User Stories begonnen werden. Dabei stellt sich oftmals heraus, dass einige User Stories noch zu groß sind. Im folgenden Abschnitt stellen wir ein paar Techniken zur Zerlegung von User Stories vor, die bei der Einhaltung der INVEST-Kriterien helfen.

Eine öffentlich zugängliche Story Map, die gepflegt wird, kann ein sehr starkes Marketinginstrument für das Projekt darstellen und über den Fortschritt berichten, wenn nach jedem Sprint die erledigten User Stories in der Story Map als »Fertig« markiert werden.

4.2.5 Zerlegung von User Stories

Schneiden macht das Team glücklich

Im letzten Backlog Refinement hat Casper ein paar User Stories vorgestellt, die einfach zu groß waren, um vom Team geschätzt zu werden. Der Auftrag an Casper war klar: »Zerlege die Stories so in kleinere sinnvolle Einheiten, dass wir sie schätzen können und auch eine Chance haben, die Stories in einem Sprint umzusetzen.« Casper war zunächst völlig perplex, handelte es sich doch aus seiner Sicht um zusammenhängende Funktionalitäten, die man nicht weiter trennen kann. Auf Hinweis von Finn hat er sich jedoch ein paar Techniken zur Zerlegung von User Stories angeeignet und bittet nun das Team zu einem erneuten Backlog Refinement, damit alle gemeinsam an den Stories und den Akzeptanzkriterien arbeiten können.

Auch wenn es anfangs nicht so aussieht, können (zu) große User Stories fast immer in kleinere User Stories zerteilt werden. Dadurch erarbeitet sich ein Scrum-Team eine gewisse Flexibilität in der Umsetzung der Anforderungen und gewinnt an Geschwindigkeit bei der Lieferung von Wert.

Nachfolgend stellen wir verschiedene Techniken zur Zerlegung von User Stories vor, die wir an konkreten Beispielen erläutern. Sehr hilfreich ist auch die Übersicht in einem Artikel von Richard Lawrence [URL: Lawrence], die strukturiert verschiedene Möglichkeiten aufführt.

Vertikales Schneiden

Das vertikale Schneiden stellt eine übergeordnete Regel für das Schneiden von User Stories dar. Sie besagt, dass User Stories immer nach fachlichen Gesichtspunkten geschnitten werden sollten, um jederzeit eine nutzbare Funktionalität zur Verfügung zu haben.

Stellen Sie sich vor, Sie haben ein paar Freunde zum Grillen eingeladen. Erfahrungsgemäß ist der Grill zu klein, um alle eingekauften Köstlichkeiten gleichzeitig auf den Tisch zu bringen. Wenn Sie warten, bis alle Filetstücke, Spieße und Tofu-Würstchen fertig gegrillt sind, bevor Sie Ihren Gästen etwas zu essen anbieten, bekommen die meisten kaltes Grillgut. Was machen Sie also? Sie versuchen, allen Gästen mehr oder weniger gleichzeitig etwas Warmes zu essen anzubieten, indem Sie den Grill so bestücken, dass nach der ersten Iteration alle erst einmal etwas zu essen haben. Einige Gäste sind danach schon

satt, andere nehmen gern noch ein zweites oder drittes Stück. Anstatt zu warten, bis alles fertig ist, und mit einem großen Knall auszuliefern, stellen Sie sicher, dass mit jeder Iteration verzehrfertiges Grillgut zur Verfügung steht und die Gäste (Stakeholder/Kunden) zufrieden sind.

SPIDR-Kriterien

In den letzten Jahren hat sich das Akronym SPIDR als Sammlung von Techniken zum Zerlegen von User Stories etabliert. Die einzelnen Buchstaben stehen für unterschiedliche Perspektiven, unter denen die User Stories betrachtet und ggf. heruntergebrochen werden. Diese Perspektiven stehen nicht für sich allein, sondern können oftmals miteinander kombiniert werden.

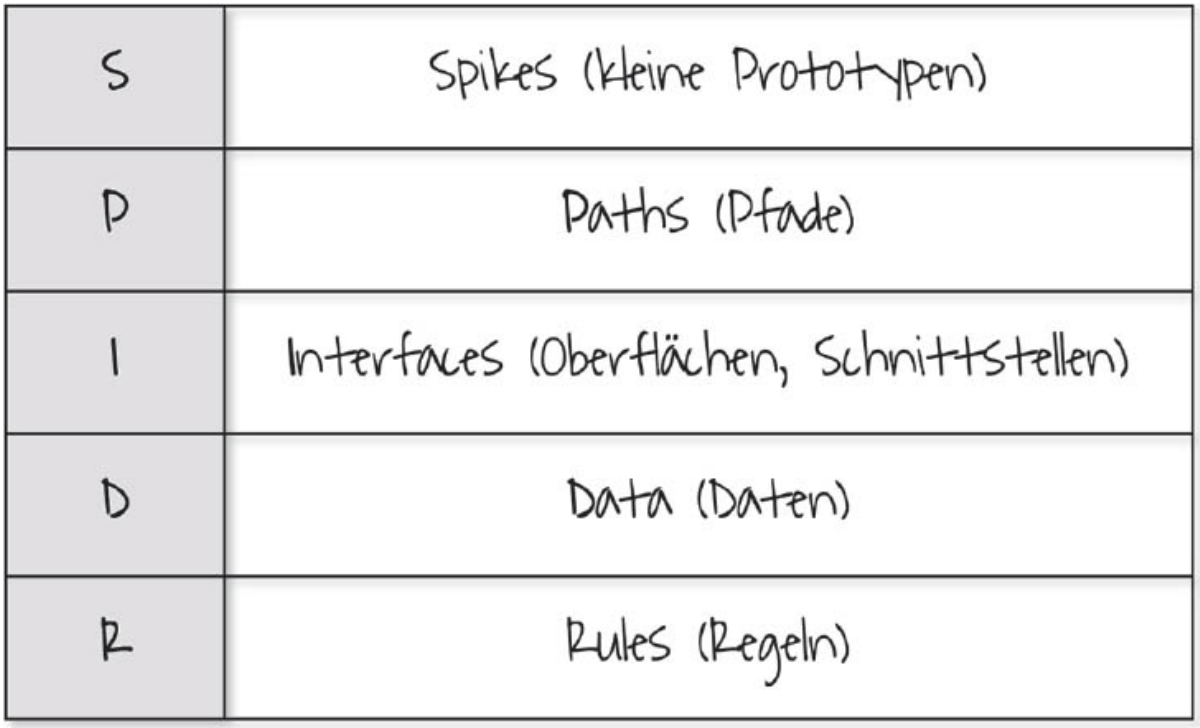

Abb. 4–12 *SPIDR-Kriterien*

Prototypen

Bei Prototypen im agilen Kontext handelt es sich um eine Maßnahme, die durchgeführt wird, um Unsicherheiten und Unklarheiten genauer zu beleuchten und zu verstehen. Ein Spike besteht meist aus einem Prototyp oder einem Forschungsauftrag und liefert den Entwicklern entweder Informationen, wie die User Story in kleinere Stories aufgeteilt werden kann, oder reduziert das technische Risiko. Ein Spike liefert keinen Wert für Nutzerinnen und wird häufig als Aufgabe mit einer Timebox versehen.

Obwohl Spikes im Akronym an erster Stelle aufgeführt sind, sollten sie erst in Betracht gezogen werden, wenn die anderen Möglichkeiten nicht greifen. Ziel sollte es nach wie vor sein, eine User Story in einem Sprint zu erledigen.

Pfade

Manchmal ist eine User Story sehr umfangreich, weil es für die Nutzerinnen mehrere Wege zum Ziel geben kann. In diesem Fall empfiehlt es sich, diese »Pfade« einzeln zu betrachten und dafür eigenständige User Stories zu schreiben.

> *»Als Konferenzbesucherin möchte ich einen von mir besuchten Vortrag bewerten, um den Vortragenden Feedback zu geben.«*

Der Aufruf einer Bewertungsmöglichkeit eines Vortrags könnte auf mehreren Wegen erfolgen:

- durch Antippen einer Schaltfläche in der Detailansicht
- durch Antippen eines Icons in der Übersicht
- durch Wischen der Vortragsbeschreibung nach rechts

Jeder dieser Wege, die zum Ziel führen, könnte als eigenständige User Story abgebildet und als Pfad ausgewählt werden. Damit wäre die Funktionalität vorhanden und die anderen Pfade können später ergänzt werden.

Schnittstellen

Die Zerlegung nach Schnittstelle ist eine eher technische Betrachtung. Hier bietet es sich an, die User Stories nach technischen Gesichtspunkten zu zerlegen, wie z. B.:

- Desktop vs. Mobile
- Browserversion vs. Client
- Desktop-Betriebssysteme (Mac OS, Linux, Windows, …)
- Mobile Betriebssysteme (iOS, Android, Windows Mobile, …)
- Browserversionen (Safari, Chrome, Firefox, Opera, Edge, …)

Bei all diesen Fragestellungen geht es darum, welche Entscheidung den größten Wert für die Nutzerinnen liefert. Haben unsere Kunden eher Android- als iOS-Geräte? Dann sollten wir unseren Fokus zunächst auf die Android-Variante legen.

An den fünf Beispielen zu den Schnittstellen ist deutlich sichtbar, dass die Entscheidung nicht für genau eine Variante getroffen werden muss, sondern dass diese kombiniert werden können und sollten. Nach der Entscheidung »Desktop vs. Mobile« kann z. B. zusätzlich

entschieden werden, ob ein nativer Client benötigt wird oder eine Browserversion ausreicht. Und selbst bei den Browsern kann später noch eine Einschränkung auf die meistgenutzten Browser vorgenommen werden. Auf diese Weise wird aus einer sehr komplexen User Story eine sehr konkrete.

Eine andere Variante aus der Pfad-Perspektive könnte es sein, zunächst eine einfache Benutzeroberfläche zu erstellen, die dann in weiteren User Stories sukzessive verfeinert und ergänzt wird. Die erste Version unterstützt beispielsweise kein Drag & Drop, diese Funktionalität wird durch eine nachfolgende Anforderung hinzugefügt.

Daten

Oft kann eine einfachere Version einer User Story mit einer Teilmenge von Daten implementiert werden. Nehmen wir folgende User Story als Beispiel:

> *»Als Vortragende möchte ich interessierten Konferenzbesuchern im Vorfeld Zusatzinformationen zu meinem Vortrag zur Verfügung stellen, sodass sie besser über eine Teilnahme an dem Vortrag entscheiden können.«*

Um welche Arten von Informationen geht es in diesem Fall? Es sollen ein paar PDF-Dateien, ein Videomitschnitt eines anderen Vortrags, die Präsentation an sich und eine sehr interessante Podcast-Folge zum Thema sein. Wie leicht zu sehen ist, wird diese User Story schnell sehr groß. Hier bietet es sich an, eine Trennung nach Datentypen vorzunehmen, z.B. Textdokumente, Video- oder Audiodateien.

Selbst hier könnte noch ein weiterer Schritt folgen und z.B. nach verschiedenen Dateitypen oder Videoformaten unterschieden werden. Oft ist dies aber gar nicht nötig, und die verschiedenen Formate werden Bestandteil der Akzeptanzkriterien.

Falls es eine User Story gibt, die das Exportieren aller Bewertungen zur Weiterverarbeitung in anderen Programmen zum Inhalt hat, könnten daraus mehrere einzelne User Stories erstellt werden, die jede für sich ein eigenes Ausgabeformat bereitstellt, z.B. Word, Excel oder XML.

Regeln

Viele User Stories enthalten eine Vielzahl von Regeln, um die dazugehörige Funktionalität robust zu machen. Eine Option besteht darin, die Regeln für die erste User Story zu lockern und sie in einer nachfolgenden Story zu behandeln. Stellen wir uns einmal die folgende User Story vor:

> *»Als Konferenzbesucherin möchte ich besuchte Vorträge bewerten, um den Vortragenden und den Veranstaltern Feedback zu geben.«*

Zusätzlich wurde folgender Ablauf in den Akzeptanzkriterien dokumentiert:

»Die Benutzerin wählt einen zu bewertenden Vortrag aus und vergibt eine Schulnote von 1 bis 6. Wenn eine Benotung zwischen 3 und 6 eingegeben wird, ist zusätzliches Feedback als Freitext Pflicht, bei einer 1 oder 2 ist es optional. Um Missbrauch oder Beleidigungen vorzubeugen, werden die Bewertungen nicht sofort freigegeben, sondern vorher durch einen Administrator geprüft.«

Mit diesem Kontext ist schnell klar, dass es sich nicht um eine kleine User Story handelt, sondern vermutlich sogar um ein Epic.

Das Schneiden einer User Story nach Regeln bedeutet, dass die User Story in Einzelschritte aufgeteilt wird, die den Workflow sukzessive vervollständigen. In diesem Fall wird eine initiale User Story erstellt, die eine Funktionalität ohne Workflowschritte abbildet.

In unserem Beispiel oben lautet diese erste User Story genau wie die originale User Story, aber die Akzeptanzkriterien werden dahingehend geändert, dass nur noch das Bewerten mit einer Schulnote stattfindet. Sobald diese User Story umgesetzt ist, steht eine einfache Funktionalität zur Verfügung, die Nutzern durchaus Mehrwert bietet.

User Story 1

»Als Konferenzbesucherin möchte ich besuchte Vorträge bewerten, um den Vortragenden und den Veranstaltern Feedback zu geben.«

Akzeptanzkriterien:

- *Vorträge können mit einer Schulnote von 1 bis 6 bewertet werden*.
- *Eine einmal erfolgte Bewertung kann nicht mehr geändert werden*.

Anschließend wird diese Funktionalität sukzessive durch weitere User Stories ergänzt, wobei wichtig ist, dass mit jeder User Story wieder eine werthaltige Funktionalität zur Verfügung steht.

User Story 2

»Als Konferenzbesucherin möchte ich meine Bewertung durch ein individuelles Feedback ergänzen, um damit meine Bewertung zu begründen oder besondere Punkte herauszuheben.«

Akzeptanzkriterien:

- *Bei der Eingabe der Bewertung soll es möglich sein, zusätzliche Informationen zur Bewertung hinzuzufügen*.
- *Diese Eingabe von Zusatzinformationen ist optional*.

- *Um die Anzahl der Klicks zu minimieren, soll die optionale Bewertung in der gleichen Ansicht wie die Eingabe der Schulnote erfolgen*.

Ein weiterer Schritt könnte sein, eine durchschnittliche Bewertung aus allen eingegangenen Bewertungen anzuzeigen:

User Story 3

> *»Als Konferenzbesucherin möchte ich durchschnittliche Bewertungen aller bewerteten Vorträge sehen, um zu erkennen, ob andere Teilnehmerinnen auch meiner Meinung sind.«*

Akzeptanzkriterien:

- *In der Vortragsübersicht soll die durchschnittliche Bewertung für alle Vorträge angezeigt werden, die fünf oder mehr Bewertungen erhalten haben*.
- *Außerdem sollen alle Bewertungen zu einem konkreten Vortrag in einer Übersicht angezeigt werden*.

User Story 4

> *»Als Konferenzbesucherin möchte ich eine Übersicht aller Bewertungen zu einem Vortrag sehen, um sie mit meiner vergleichen zu können.«*

Akzeptanzkriterien:

- *Es soll eine Übersicht geben, auf der alle Bewertungen zu einem Vortrag mit ihrer Überschrift und der Schulnote angezeigt werden*.
- *Bei Auswahl einer Bewertung sollen in einer Detailansicht der Bewertungstext und ggf. eingegebene Zusatzinformationen angezeigt werden*.

Wie zu erkennen ist, könnte die Aufteilung der ursprünglichen User Story noch viel weitergetrieben werden. Wichtig dabei ist, dass jede User Story für sich betrachtet ein Bedürfnis der Kunden oder Nutzerinnen erfüllt und damit einen Wert generiert.

Wenn sich ein Product Owner die vier Beispiele oben vergegenwärtigt, stellt er oder sie möglicherweise fest, dass z.B. die vierte User Story für den Anfang gar nicht so relevant ist, und kann diese im Product Backlog weiter unten einordnen. Durch das Aufteilen der ursprünglichen User Story auf vier User Stories verschafft sich ein Product Owner zunächst eine gewisse Flexibilität, da alle User Stories individuell priorisiert werden können. Außerdem stellt er vielleicht fest, dass die eine oder andere User Story gar nicht so relevant ist,

sodass er sie entweder weiter unten einordnen oder sogar ganz streichen kann. So hat er neben der gewonnenen Flexibilität ggf. auch noch die Komplexität reduziert.

Mit dem Schneiden nach Regeln wird möglicherweise die Unabhängigkeit aus den INVEST-Kriterien verletzt. Häufig wird jedoch in den anderen Kriterien so viel Sicherheit hinzugewonnen, dass dies durchaus akzeptabel sein kann. Es bleibt natürlich immer eine Einzelfallentscheidung, die aber leichter fällt, wenn sichtbar wird, was und warum etwas getan wird.

Zusätzlich zu den SPIDR-Kriterien gibt es weitere Perspektiven, unter denen User Stories betrachtet und zerlegt werden können. Einige davon stellen wir im Folgenden vor.

Zerlegen nach Komfort

> *»Als Konferenzbesucherin möchte ich mir eine Liste mit für mich interessanten Vorträgen erstellen, um nicht versehentlich einen davon zu verpassen.«*

Verwöhnten Nutzerinnen erscheint bei dieser User Story möglicherweise gleich ein Touchscreen vor dem geistigen Auge, auf dem Vorträge kategorisiert und sortiert werden können, um somit zu einer Liste von interessanten Vorträgen zu gelangen. Vielleicht ist das auch genau das Szenario, das der Product Owner in vorangegangenen Nutzertests extrahiert hat. Und vielleicht ist es auch mit den heutigen technischen Möglichkeiten so einfach, dass die Entwickler nur müde lächeln.

Aber nehmen wir einmal an, zumindest Letzteres wäre nicht der Fall. Welche Möglichkeiten gäbe es, die Funktionalität trotzdem in einer einfacheren Variante umzusetzen? Hier einige Vorschläge, die in jeder Ausbaustufe wertvoll und nutzbar sind:

- Notizblockfunktion, um interessante Vorträge zu notieren
- Setzen eines Lesezeichens auf einen interessanten Vortrag
- Markieren von Vorträgen in einer Liste als Favorit, Anzeige in der Favoritenliste
- Drag & Drop von Vorträgen aus der Programmübersicht in die Favoritenliste

- Drag & Drop von Vorträgen aus der Programmübersicht in einen Kalender, Warnung bei Terminüberschneidungen

Weitere häufig genutzte Komfortfunktionen sind das Suchen und das Filtern von Informationen. Jedes einzelne Such- oder Filterkriterium könnte eine eigene User Story darstellen, z.B. Vortragende, Vortragstitel, Kategorie, Dauer, Uhrzeit oder Bewertungen.

Zerlegen nach Komplexität

Manchmal gehen Product Owner mit einer vermeintlich simplen User Story in ein Gespräch mit den Entwicklern und sind völlig überrascht, welche Fragen plötzlich auftauchen. »Was ist mit X?«, »Hast du auch an Y gedacht?« und »Was soll passieren, wenn dies oder jenes getan wird …?«, sind Fragen, die beantwortet werden wollen. Die einfache User Story wird plötzlich größer und größer, und ein Product Owner kann möglicherweise auch nicht alle Fragen aus dem Stegreif beantworten.

In diesen Fällen empfiehlt es sich, die ursprüngliche User Story in ihrer einfachsten Form beizubehalten und für die bisher nicht bedachten Variationen und Ergänzungen eigene User Stories zu schreiben.

Es lohnt sich, wenn Sie die INVEST-Regeln zu Beginn eines Refinement wiederholt vorstellen und visualisieren.

Fügen Sie der Darstellung ruhig Hinweise hinzu, um Akzeptanzkriterien richtig zu definieren, z. B. indem Sie Fragewörter »Was ist wenn …?«, »Warum …?«, »Wo …?«, »Wann …?« oder »Wie viel …« ergänzen. Auch Vereinbarungen des Teams, wie z. B. »User Stories sollten nicht größer als 8 Story Points sein« oder »nur eine Anforderung pro User Story aufnehmen«, können dort als Hilfestellung und Erinnerung aufgeführt werden.

Zerlegen nach Performance

> *»Als Konferenzbesucherin möchte ich die Vorträge nach bestimmten Stichworten durchsuchen, um Vorträge für meine persönlichen Interessen zu finden.«*

Um die Komplexität aus dieser User Story herauszunehmen, kann sie unter Performance-Aspekten zerlegt werden:

- **langsam**
  Ein Ergebnis sollte spätestens nach zehn Sekunden erscheinen, in der Zwischenzeit wird eine Animation gezeigt.

- **schnell**
  Das Ergebnis steht nach spätestens zehn Millisekunden zur Verfügung.

Daumenregeln für die Zerlegung von User Stories

Es kann häufig sein, dass eine User Story nach mehreren der oben genannten Methoden geschnitten werden kann. Dann stellt sich die Frage, welche Aufteilung gewählt werden sollte. Wir richten uns nach den folgenden Daumenregeln:

- Wählen Sie die Aufteilung, die es ermöglicht, auf Teile der User Story zu verzichten. Die 80:20-Regel besagt, dass der größte Geschäftswert einer User Story aus einem relativ kleinen Teil der Funktionalität entsteht. Wenn eine Teilungsvariante geringwertige Funktionalität enthüllt und eine andere nicht, liegt der Verdacht nahe, dass Komplexität in den User Stories versteckt ist, die letztendlich den Kunden oder Nutzerinnen keinen Wert mehr bietet. In diesem Fall ist es meistens besser, die 80:20-Lösung zu wählen und auf einen Teil der Anforderungen zu verzichten.
- Wählen Sie die Aufteilung, die mehr ähnlich große User Stories liefert. Eine User Story mit acht Story Points in vier User Stories mit je zwei Story Points zu teilen ist besser als zwei User Stories mit drei und fünf Story Points. Product Owner erhalten so mehr Freiraum beim Priorisieren einzelner Teile der Gesamtfunktionalität.

Die richtige Einstellung

Wie in den oberen Beispielen aufgeführt, gibt es bei der Pflege des Product Backlog und der Product-Backlog-Einträge einiges zu beachten. Vor allem geht es uns in der Arbeit mit Scrum-Teams darum, dass diese Arbeit gemeinsam von allen Teammitgliedern durchgeführt wird. Dazu vermitteln wir in unserer Arbeit die Kriterien eines weiteren Akronyms: SWAT (siehe Abb. 4–13).

S	Shift focus from writing to talking (Verlagern des Fokus vom Schreiben auf das Sprechen)
W	Work with user on a daily basis (tägliche Zusammenarbeit mit Nutzern)
A	Aussume not knowing everything (davon ausgehen, nicht alles zu wissen)
T	Trust developers to find the „how" (Entwicklern vertrauen, das „Wie" zu ermitteln)

Abb. 4–13 *SWAT-Kriterien*

In der Abbildung 4–14 symbolisieren wir mithilfe eines Leuchtturms die bereits vorangegangenen Kriterien zur Pflege eines Product Backlog und von User Stories. Erste Priorität ist dabei, den Wert für Kunden oder Nutzerinnen zu erhöhen bzw. zu liefern. Die SWAT-Kriterien bilden dabei die Basis für alle Interaktionen des Scrum-Teams untereinander und mit den Stakeholdern des Projekts. Die Kriterien spiegeln zudem die Prinzipien und Werte des Agilen Manifests (*agilemanifesto.org*) wider:

- **Sprechen**
  User Stories sind keine leichtgewichtigen Anforderungen, die einfach den Entwicklern über den Zaun geworfen werden. Zuallererst geht es vor allem darum, dass gesprochene Wort zu bevorzugen anstelle der reinen Dokumentation von Anforderungen. User Stories sind dazu da, um kollaborativ Anforderungen zu erstellen. Dies geschieht vor allem durch Involvement und Austausch untereinander.
- **Zusammenarbeit**
  User Stories tragen es schon in ihrem Namen: Sie sind aus der Sicht von Nutzerinnen oder Kunden geschrieben. Von daher sollte es auch dazu gehören, dass diese eine Stimme erhalten und in die Erstellung mit einbezogen werden. Dies kann z.B. durch Nutzertests passieren oder durch eine enge Zusammenarbeit mit dem zuständigen Fachbereich.
- **Offenheit**
  Bei der Arbeit mit komplexen Problemstellungen ist zu beachten, dass nicht alle Bedürfnisse der Kundinnen, Anforderungen oder Lösungswege bekannt sind. Nicht nur Product Owner, sondern auch die Entwickler sollten anerkennen, dass nicht alles vorab bekannt ist und spezifiziert werden kann. Durch die kurzen Sprint-Zyklen erfährt ein Scrum-Team schnell Neues und kann damit weiterarbeiten.
- **Vertrauen**
  Oftmals tendieren Product Owner dazu, nicht nur das »Was?«, sondern auch das »Wie?« zu liefern und die Entwickler damit zu konfrontieren. Zusammengefasst aus den oberen drei Kriterien ist die Verantwortung aller im Scrum-Team, sich in Richtung des Produktziels zu bewegen. Die

Anerkennung der Fähigkeiten jedes einzelnen Teammitglieds bildet die Basis für eine erfolgreiche Zusammenarbeit.

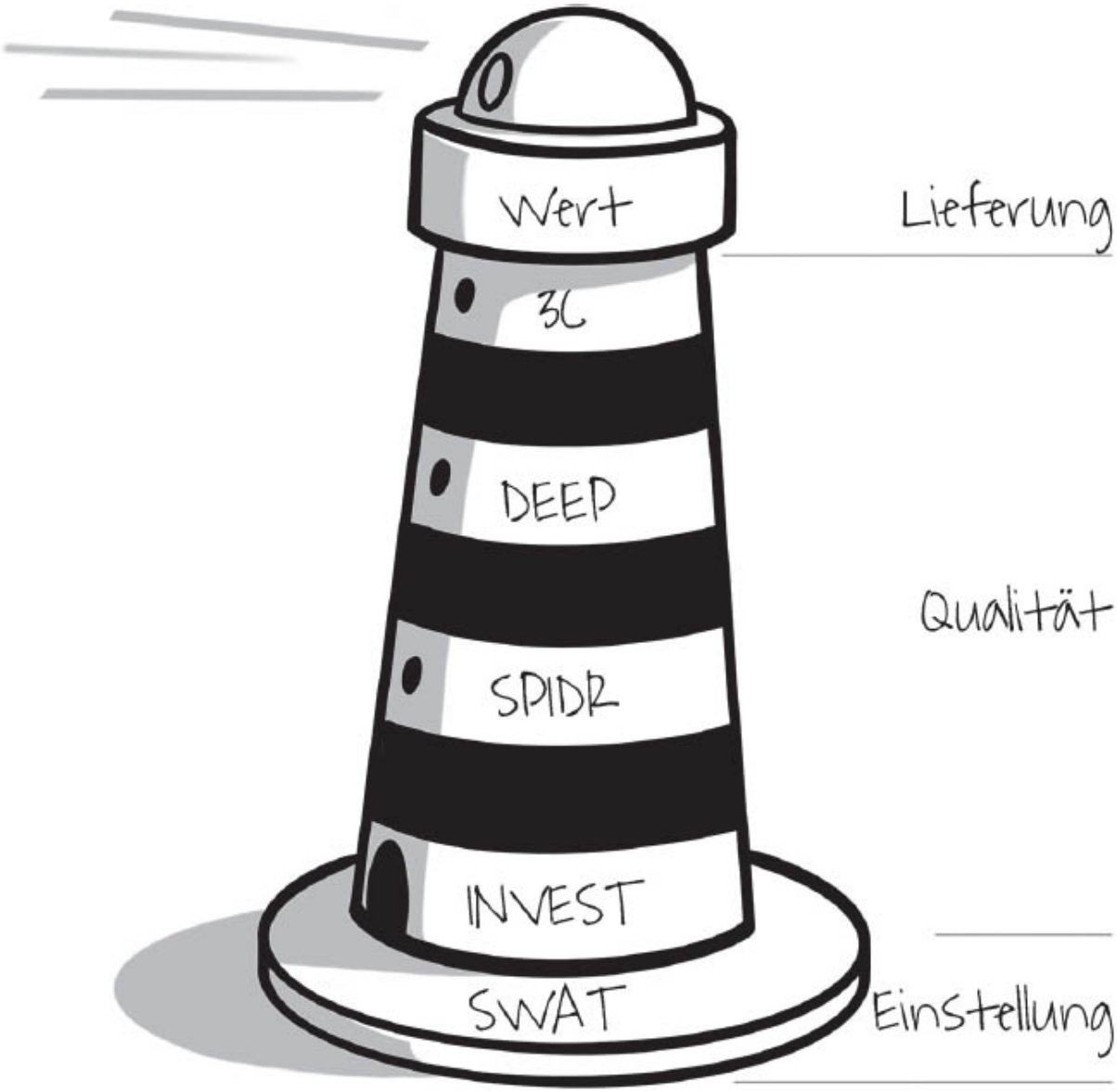

Abb. 4–14 *Merkmale der Backlog-Pflege*

4.2.6 Häufige Herausforderungen

Das Product Backlog ist nicht öffentlich sichtbar

Backlogs werden heutzutage elektronisch gepflegt, meist in einem Verwaltungstool wie z.B. Jira, manchmal aber auch in Excel-Listen. Für Stakeholder ist es allerdings oft schwierig, sich einen Überblick zu verschaffen. Manchmal fehlen ihnen die Berechtigungen, teilweise wissen sie nicht, welche Ansicht sie verwenden sollen, und oft sind sie auch nicht mit der Nutzung des eingesetzten Tools vertraut.

Damit ein Product Backlog seine Kraft entfalten kann, sollte es für alle einsehbar sein. Die Entwickler sollten jederzeit in der Lage sein, das aktuell in Arbeit befindliche Backlog Item im Kontext zu sehen und sich über geplante, möglicherweise beeinflussende Backlog Items zu informieren. Mindestens alle Stakeholder, besser noch alle im Haus, sollten das Product Backlog einsehen können.

Die heutigen Verwaltungssysteme bieten oft eine Schnittstelle, um Daten dynamisch in anderen Systemen anzuzeigen. So ist es beispielsweise möglich, bestimmte Ansichten aus Jira im Schwestertool Confluence anzuzeigen. Nutzen Sie diese Möglichkeit, um z. B. die aktuell im Sprint befindlichen Backlog Items und die für die nächsten Sprints geplanten Backlog Items dort automatisch anzuzeigen.

Kollaboration findet nicht statt

Wir bemerken in nicht wenigen Fällen den Trend dazu, dass Product Backlogs eine hohe Ähnlichkeit zu Anforderungsdokumenten haben, die im Projektmanagement gefordert sind. Die Verwaltungstools werden dann oftmals mit unzähligen Attributen, Pflichtfeldern oder Eingabeanforderungen konfiguriert. Diese Art der Product-Backlog-Behandlung führt oftmals dazu, dass die Diskussionen miteinander in den Hintergrund treten. Es wird darauf beharrt, dass alle Informationen vorliegen müssen, bevor überhaupt miteinander gesprochen wird. Oder es wird dann eben auch nur das getan, was definiert worden ist, bzw. der Austausch an Informationen wird minutiös am Product Backlog Item dokumentiert, sodass als Ergebnis seitenlange Kommentare entstehen.

Eine unserer ersten Handlungen bei der Aufnahme der Arbeit in einem Scrum-Team ist es, dass wir uns den Zustand des Product Backlog ansehen. Sobald wir feststellen, dass es einen enormen Aufwand macht, Einträge im Product Backlog zu erstellen oder deren Inhalte nachzuvollziehen, suchen wir das Gespräch mit dem Scrum-Team und dessen Umfeld.

Es geht uns dabei vor allem um das Infragestellen für dieses Vorgehen und die Aufklärung darüber, dass Backlog Items bzw. User Stories eine Konversation miteinander anregen sollen. Wir stellen dabei den Unterschied zu traditionellen Methoden heraus, die den Schwerpunkt auf die Spezifikation und Dokumentation von Anforderungen legen, anstelle miteinander kollaborativ die Backlog Items zu ergründen. In Ergänzung machen wir deutlich, dass Scrum auf Lean Thinking basiert. Genauso wie das Pflegen eines sehr großen Product

Backlog ist die aufwendige Pflege eines »überkonfigurierten« und »überdokumentierten« Product Backlog Verschwendung (engl. Waste).

Machen Sie den Teammitgliedern deutlich, dass sie noch so viel dokumentieren können, um ein klares und gemeinsames Verständnis zu erzielen. Klarheit und Beteiligung werden sie nur über Gespräche miteinander erreichen.

Die Backlog Items sind zu detailliert beschrieben

Besonders in Organisationen mit einer Historie in traditionellen Projektmanagementmethoden tendieren die Entwickler oft dazu, vom Product Owner perfekt spezifizierte Anforderungen zu erwarten. Unerfahrene Scrum Master und Product Owner steuern oft nicht gegen, sodass an dieser Stelle die Kraft der Kommunikation und des schnellen Feedbacks verloren geht. Darüber hinaus verschanzen sich solche Teams oft hinter dem »geschützten Sprint« und verweigern das durch User Stories geförderte gemeinsame Finetuning mit Hinweis auf die Spezifikation.

Bei den User Stories wird es durch das mittlere »C« im Kürzel »CCC« sehr schön deutlich: Die stattfindende Konversation ist ein wesentlicher Bestandteil eines Backlog Item. Führen Sie Ihr Team durch die Grundlagen von Backlog Items, erläutern Sie, warum sie eben nicht komplett spezifiziert sein müssen. Verweisen Sie auch auf die Schätzungen, die ja per Definition nur relativ grob sind, eben weil in jedem Backlog Item eine gewisse Unsicherheit steckt.

Das Product Backlog ist nicht weit genug im Voraus durchdacht

Durch die iterative Vorgehensweise von Scrum werden neue Erkenntnisse und Feedback so früh wie möglich in den Entwicklungsprozess eingebracht und adaptiert, da zu Beginn eines Projekts noch nicht alle Anforderungen bis zum Projektende sauber durchdacht und formuliert sein können. Nichtsdestotrotz sollte das Backlog ausreichend vorbereitete Backlog Items (vgl. Abschnitt 4.1.2) für die nächsten zwei bis drei Sprints enthalten, um bei unerwarteten Hindernissen handlungsfähig zu sein und andere Backlog Items vorziehen zu können.

Erstaunlicherweise treffen wir oft auf Scrum-Teams, in denen gerade mal die Backlog Items für den aktuellen Sprint hinreichend vorbereitet wurden und das Scrum-Team sich so von Sprint zu Sprint hangelt. Die Gründe dafür sind vielfältig, sie gehen von »nur der Product Owner erstellt Product-Backlog-Einträge« über »Zusatzaufgaben in der Linie«, »unzählige Abstimmungsrunden« bis hin zu Doppelfunktionen (z.B. »Product Owner und Abteilungsleiter«).

Arbeiten Sie gemeinsam mit dem Product Owner, um Zeitfresser aufzudecken. Machen Sie diese transparent, zeigen Sie die Konsequenzen eines für die Entwickler nicht verfügbaren Product Owners auf und machen Sie klar, dass so die Vorteile eines Vorgehens mit Scrum nicht zur Geltung kommen. Verweisen Sie auf das Risiko, dass die Entwickler ohne entsprechende Vorbereitung durch den Product Owner möglicherweise demnächst nicht mehr effizient auf das Projektziel hinarbeiten können, was Auswirkungen auf den geplanten Releasetermin haben kann.

Veraltete Product-Backlog-Einträge

»*Was ist der Wert eines Backlog Item, das seit einem Jahr im Product Backlog steht?*« Mit dieser Frage konfrontieren wir Product Owner gerne, wenn wir feststellen, dass sehr große Product Backlogs mit vielen Einträgen gepflegt werden.

Der Punkt ist, dass das Product Backlog ein lebendiges Artefakt ist und ständig gepflegt wird. Wenn dabei laufend alte Product-Backlog-Einträge verwaltet werden müssen, ist dies ein unnötiger Overhead. Je mehr Einträge ein Product Backlog enthält, desto schneller besteht die Gefahr, dass sie veralten und die investierte Zeit in die Erstellung und das Refinement verschwendet war.

Von daher sollte ein Product Backlog nur das enthalten, wovon ein Product Owner oder das Scrum-Team überzeugt ist, dass es umgesetzt wird. Unnötiges zu beseitigen wird dazu führen, dass über das gesprochen wird, was wirklich relevant ist.

»Warum nicht entfernen?« Oftmals stehen die veralteten Backlog Items am Ende des Product Backlog, da durch Inspect & Adapt in Scrum schnell neue Erkenntnisse und damit neue Backlog Items entstehen. Besprechen Sie mit dem Scrum-Team, was ein sinnvoller Horizont für das Vorhalten von Backlog Items ist.

Wir halten einen Zeitraum von maximal drei Monaten für sinnvoll. Dies bedeutet, wenn ein Backlog Item es in drei Monaten nicht in einen Sprint geschafft hat, sollte es entfernt werden.

Eine Wunschliste für alles

In Gesprächen mit Product Ownern über ihr Product Backlog stellen wir oft fest, dass es alles enthält, was sich die Stakeholder wünschen. Es ist eine Wunschliste an Features, die über die Zeit nicht nur anwächst, sondern auch in den meisten Fällen viel Kapazität der Product Owner erfordert. Diese müssen nicht nur das anwachsende Product Backlog pflegen, sondern auch die Nachfragen der Stakeholder. In einigen Fällen geht es so weit, dass Stakeholder sich gegen die Arbeitsweise des Teams stellen, da ihre Wünsche keine Beachtung finden.

Die Frage, die wir in diesem Fall Product Ownern stellen ist: »*Welchen Beitrag leisten diese Backlog Items im Hinblick auf die Erreichung des Produktziels?*« Es ist also wichtig, dass ein Produktziel vorliegt und Product Owner es für die Entscheidung nutzen, ob sie Anforderungen weiterverfolgen oder nicht.

In dem beschriebenen Szenario könnte sich für einen Scrum Master auch das Gefühl herauskristallisieren, dass der Product Owner eine klare Ansprache gegenüber den Stakeholdern meidet, um diese beispielsweise nicht zu enttäuschen. Ermutigen Sie Product Owner dazu, unter Zuhilfenahme des Produktziels argumentativ auf Stakeholder zuzugehen und deutlich zu machen, dass die angebrachten Wünsche aktuell nicht umgesetzt werden können.

Aufteilung des Product Backlog

Manche Scrum-Teams entscheiden sich dafür, mehrere Product Backlogs zu erstellen. Dann gibt es z.B. ein Release Backlog, ein technisches Backlog oder »Tolle Ideen«-Backlog.

Nicht nur, dass der Pflegeaufwand steigt und die Übersichtlichkeit sinkt, sondern vor allem leidet oftmals die Kollaboration im Scrum-Team darunter. Product Owner sind dann für das »Business« verantwortlich, während sich die Entwickler um die »Technik« kümmern.

Ein Product Backlog ist die einzige Quelle für alles, was mit der Produktentwicklung eines Scrum-Teams zusammenhängt. Es gibt ein einziges Product Backlog, für das der Product Owner verantwortlich ist, dessen Pflege jedoch vom gesamten Scrum-Team übernommen werden sollte. Wie dies geschieht, wird innerhalb des Scrum-Teams definiert.

Das Team ist nicht in die Erstellung der Backlog Items eingebunden

Manche Product Owner interpretieren ihre Verantwortlichkeit so, dass sie allein für das Product Backlog verantwortlich sind. Dies führt dazu, dass sie akribisch über das Product Backlog wachen und sämtliche Backlog Items allein schreiben. Dabei geht natürlich das Wissen der Entwickler nicht in das Produkt ein. Die Entwickler arbeiten täglich am Produkt, testen es, beschäftigen sich inhaltlich damit. Selbst wenn sie nicht zur späteren Zielgruppe gehören, können sie doch gute Hinweise zur Benutzbarkeit geben oder sogar eigene Produktideen vorschlagen.

Weisen Sie in der beschriebenen Situation den Product Owner darauf hin, dass sich seine Verantwortung auf die Existenz und die Priorisierung des Product Backlog beschränkt und dass es durchaus sinnvoll ist, die Entwickler einzubinden. Führen Sie die oben beschriebenen Anforderungsworkshops durch (vgl. Abschnitt 4.2.2) und sorgen Sie für regelmäßig stattfindende Backlog Refinements.

Backlog Items enthalten keinen Wert für die Benutzerin

Wert für Nutzerinnen zu schaffen, ist eines der wichtigsten Ziele von Scrum. Daher sollte aus jeder Anforderung klar erkennbar sein, welchen Wert sie bietet. Oft wird aber auf den Nachweis des Wertes verzichtet, weil er sich vermeintlich von allein erschließt oder weil tatsächlich kein benennbarer Wert vorhanden ist.

Ermutigen Sie die Entwickler, den Wert von Backlog Items für die Nutzerinnen zu hinterfragen. Wenn der Product Owner den Wert nicht darlegen kann, ist es völlig legitim, die Existenz des Backlog Items infrage zu stellen. Oft helfen hier auch die oben beschriebenen Zerlegungstechniken (vgl. Abschnitt 4.2.5).

Zu diesem Schwerpunkt finden Sie die Checkliste »Scrum-Vorbereitungen zum Start« unter *link.scrum-in-der-praxis.de/checklisten*.

4.3 Schätzung von Komplexität

Für den Product Owner sind Schätzungen eine hilfreiche Unterstützung, um seine Planung vorzubereiten und gegenüber Stakeholdern auskunftsfähig zu sein. Für das gesamte Scrum-Team ist das Besprechen der nächsten Schritte wichtig, denn es will auf ein gemeinsames Ziel hinarbeiten und Klarheit erhalten. Auch die Organisation oder der Kunde verlangt Transparenz, da sie daran interessiert sind, was als Nächstes umgesetzt wird und wann eine nächste Lieferung erfolgt. Dafür sind für den Product Owner zwei Werte interessant:

- *Die Komplexität der Backlog Items:* Es wird u.a. deutlich, welche Backlog Items noch unklar sind und in kleinere Einheiten zerlegt werden sollten.
- *Die Anzahl der Story Points*, die in einem Sprint bearbeitet werden können, die sogenannte →Velocity.

In Scrum liegt die Verantwortung für die Schätzungen bei den Entwicklern, die im Mittel ca. 5–10% ihrer Kapazität für diese Planungsaufgaben einsetzen sollten. Da jedoch jedes Teammitglied das Prinzip »Selbstverantwortung« unterschiedlich schnell verinnerlicht, wirkt dieses Planen – mit dem viele Entwickler eigentlich gar nichts zu tun haben wollen – manchmal störend. Es kommt die Frage darüber auf, ob Schätzungen nicht Verschwendung sind, wenn Backlog Items eventuell gar nicht umgesetzt werden. Oder viel Zeit zum Debattieren darüber eingesetzt wird, um welche Nummer es gehen soll.

Wertvoll am Schätzen ist, wenn das Scrum-Team zusammenkommt und ein Gespräch führt. Dadurch erreicht es Folgendes:

- *Wie komplex ist eine Anforderung?*
  Durch die Klärung und das wiederholte Besprechen von Backlog Items erhalten alle ein gemeinsame Verständnis zur Komplexität.
- *Was müssen wir noch besser verstehen?*
  Wenn eine Anforderung zu groß ist, wird sie in kleinere Teile zerlegt, die innerhalb eines Sprints umsetzbar sind.
- *Welche Priorität hat eine Anforderung?*
  Durch die Angabe von Schätzungen kann das Product Backlog sortiert werden.

Stabile Rahmenbedingungen sind die Grundlage für die Einhaltung von Schätzungen. Die Komplexität, die gerade in der Produktentwicklung enorm ist, kann sich durch ein ständig wechselndes und instabiles Umfeld erhöhen. Daher ist es notwendig, für klare Strukturen zu sorgen, die den Entwicklern die Gewissheit geben, dass sie Vergleiche zu bereits umgesetzten Backlog Items

herstellen können. Dieser Punkt ist sehr wichtig für das Funktionieren eines Scrum-Teams, da es sich für die Fertigstellung verpflichtet und für die Umsetzung der Backlog Items einsteht. Diese Rahmenbedingungen zu schaffen ist Aufgabe des Scrum Masters und der Organisation.

Am Anfang heißt es daher oft, Überzeugungsarbeit zu leisten und offene Fragen aus dem Weg zu räumen. Es gilt dann immer wieder in Erinnerung zu rufen, dass alle als verbundene Einheit für die Entwicklung des Produkts verantwortlich sind und damit also auch für die Planung.

Wer sich mit Schätzungen auseinandersetzt, sollte im Blick behalten, dass es sich nur um eine grobe Annahme auf Basis der verfügbaren Informationen handeln kann und niemals um eine konkrete Vorhersage der Zukunft.

4.3.1 Schätzungen

In früheren Versionen von Scrum war »Estimation« ein eigenes Event. In der Praxis wird es inzwischen im Rahmen des Backlog Refinement durchgeführt, dessen Struktur wir in Abschnitt 4.4 genauer beschreiben. Die Entwickler schätzen die Größe der vom Product Owner vorgestellten Backlog Items anhand der Komplexität. Der Termin sollte insbesondere seitens des Product Owners gut vorbereitet sein. Es ist eine Aufgabe des Scrum Masters, den Product Owner darin zu unterstützen, ein sortiertes und öffentlich zugängliches Product Backlog zur Verfügung zu stellen.

Das Schätzen von Backlog Items sollte nach Bedarf stattfinden, jedoch mindestens einmal in jedem Sprint. Hierzu ist eine Regelung mit dem Product Owner zu finden. In der Praxis haben sich zweiwöchige Sprints in vielen Scrum-Teams und Unternehmen durchgesetzt. Ein einstündiges Backlog Refinement pro Sprint-Woche ist eine gute Basis, um einen kontinuierlichen Informationsfluss, aktuelle Schätzungen und hinreichend Feedbackschleifen zu gewährleisten. Sollte Bedarf für weitere Backlog Refinements bestehen, können weitere Treffen eingeplant werden. Gerade in der Anfangsphase eines Projekts ist der Bedarf oft größer.

#NoEstimates

Es gibt seit einigen Jahren auch die #NoEstimates-Bewegung (*noestimatesbook.com*), die auf die Abgabe von Schätzungen in jeglicher Form keinen Wert legt. Die Planung basiert auf der Erfahrung, wie viele Backlog Items innerhalb eines Sprints bearbeitet werden können. Dieser Durchsatz pro Iteration kann dann zur Planung genutzt werden.

Unserer Meinung nach sind Schätzungen weder gut noch schlecht. Es hängt davon ab, wie diese Informationen verwendet und wie viel Vertrauen in die Schätzungen gesetzt wird. Das Maß an Vertrauen, das gerechtfertigt ist, hängt vom Kontext ab, in dem sich ein Scrum-Team bewegt. Wenn komplexe Arbeit vor uns liegt, werden Schätzungen, egal wie viel ein Scrum-Team in den Vorgang des Schätzens investiert, falsch liegen. Dies bedeutet jedoch nicht, dass die Prognosen, die damit erstellt werden können, nicht hilfreich sind.

Wie oben beschrieben, ist letzten Endes nicht so sehr das Ergebnis selbst, sondern vielmehr das Ergebnis der Gespräche mit den richtigen Personen zur richtigen Zeit wichtig.

Sorgen Sie als Scrum Master dafür, dass alle pragmatisch mit Schätzungen umgehen. Verkomplizieren Sie den Prozess nicht. Unterbrechen Sie z. B. eine Diskussion zwischen Entwicklern, wenn es darum geht, ob es drei oder fünf Story Points sind. Nehmen Sie fünf. Oder, wenn Sie sehr viele Backlog Items zu schätzen haben, lassen Sie die Entwickler ganz einfach drei Stapel machen: »Haben wir verstanden, ist klein genug«, »Ist zu groß« und »Ich habe keine Ahnung«.

4.3.2 Schätzeinheiten

In der Literatur gibt es unterschiedliche Meinungen darüber, ob Aufwand oder Komplexität zum Schätzen verwendet werden sollte. Der Scrum Guide erwähnt hierzu (*scrumguides.org*):

»Das Refinement des Product Backlogs ist der Vorgang, durch den Product-Backlog-Einträge in kleinere, präzisere Elemente zerlegt und weiter definiert werden. Dies ist eine kontinuierliche Aktivität, wodurch weitere Details wie Beschreibung, Reihenfolge und Größe ergänzt werden. Die Attribute variieren oft je nach Arbeitsumfeld. Developer:innen, die die Arbeit erledigen werden, sind für die Größenbestimmung umsetzungsverantwortlich.«

Wir bevorzugen das Schätzen von Komplexität anhand von Story Points aus folgenden Gründen:

- **Die Grundannahme für Schätzungen sind keine Aufwände, sondern die relative Komplexität von Backlog Items zueinander.**
  Es geht um die Aussage, wie komplex ein Backlog Item im Verhältnis zu einem anderen Backlog Item ist, also um das relative Verhältnis von Anforderungen zueinander. Für die Ermittlung der Velocity des Teams ist es letztendlich egal, in welcher Einheit man schätzt, solange sie in sich konsistent ist und von den Entwicklern gleichermaßen interpretiert wird und nicht mit der Velocity anderer Teams verglichen wird.
- **Menschen schätzen besser relativ als absolut.**
  Wenn beispielsweise eine Person am Strand steht und aufs Meer schaut, kann diese ziemlich leicht feststellen, ob eine Boje sich vor, hinter oder neben einem Schiff befindet. Jedoch kann diese kaum mit hinreichender Gewissheit sagen, ob sich Schiff oder Boje 500 Meter oder 700 Meter vom Strand entfernt befindet. Schätzungen in Story Points bieten einen impliziten Puffer, wogegen eine stundenbasierte Schätzung von 4 Stunden nach 4,5 Stunden oft bereits kritisch hinterfragt wird. Ergebnisse relativer Schätzungen, wie beispielsweise in Story Points, sind genauer als absolute Schätzungen in Personentagen.
- **Die Dauer der Umsetzung eines Backlog Item hängt stark davon ab, wer daran arbeitet.**
  In einem Scrum-Team verfügen in der Regel nicht alle Teammitglieder über die gleiche Erfahrung, sodass für eine Aufwandsschätzung bereits während der Schätzung festgelegt werden müsste, wer das Backlog Item umsetzen soll. Das allerdings widerspricht dem Gedanken der Selbstorganisation, kurzfristig in der Lage zu sein, sich anders als geplant aufzustellen. Eine Schätzung in Komplexität ist immer eine Schätzung des gesamten Scrum-Teams, nicht einer Einzelperson.
- **Die Komplexität eines Backlog Item ändert sich nicht.**
  Schätzungen nach Aufwand müssen über die Laufzeit angepasst werden, um abzubilden, wer an einem Backlog Item arbeitet. Ein Junior wird vermutlich mehr Zeit benötigen als ein Senior. Bei einer Schätzung der Komplexität bleibt diese konstant und zieht keine aufwendige Planungskorrektur nach sich.

Das wirklich Wichtige an Schätzungen im Kontext von Scrum-Projekten ist, dass Prognosen durch das Schätzen erstellt werden können, diese jedoch immer vage bleiben.

»*Wir können doch nicht 500.000 € investieren, ohne eine Schätzung zu haben, wie lange es dauert!*«, lautet oftmals das Argument von Kunden oder Organisationen. Wie ist es jedoch, wenn wir mit einem Projekt beginnen, das drei Monate dauert? Als Auftraggeberin fällt dabei erst einmal nur ein Investment von 50.000 € an. Gleichzeitig steht am Ende der Zeit ein gewünschtes Ergebnis zur Verfügung. Wenn dies nicht der Fall ist und es andere Erwartungen gab, dann kann eine neue Entscheidung getroffen werden.

Story Points

Story Points sind eine Bezeichnung für eine abstrakte Größe, die die Komplexität eines Backlog Item beschreibt. Man könnte einen beliebigen Begriff benutzen, solange er nichts Messbares beschreibt.

Die 2005 von Mike Cohn [Cohn 2005] eingeführte modifizierte Fibonacci-Reihe ist nach wie vor der Klassiker der Schätzskalen für Story Points. Sie lautet 0, 1, 2, 3, 5, 8, 13, 20, 40, 100. Es ist leicht zu sehen, dass die Abstände mit der Höhe der Werte immer weiter auseinanderliegen. Dies liegt daran, dass jede Zahl keinen absoluten Wert darstellt, sondern einen Bereich und dass mit steigender Komplexität auch das Risiko und die Unsicherheit größer werden.

Wenn ein Backlog Item auf acht Story Points geschätzt wird, bedeutet dies einfach, dass ihm die Zahl 8 zugeordnet wird. Alle Backlog Items, die ähnlich komplex sind, erhalten auch eine 8 als Schätzung. Eine 8 bedeutet dabei »irgendwo zwischen 5 und 13«, während eine 3 »irgendwo zwischen 2 und 5« bedeutet. Ein Vorteil dieser numerischen Skala ist der, dass damit eine Velocity berechnet werden kann.

Die Fibonacci-Reihe ist unser Favorit, allerdings lassen wir die 0 komplett weg und überzeugen unsere Teams, sich auf Werte bis maximal 13, manchmal sogar nur bis 8 zu beschränken. Der Grund dafür liegt darin, dass eine 20 »irgendwo zwischen 13 und 40« bedeutet. Mit anderen Worten: Die Unsicherheit ist so groß, dass das Backlog Item möglicherweise doppelt so groß sein könnte. Das lässt letzten Endes keine sinnvolle Planung zu, daher bitten wir den Product Owner bereits an dieser Stelle um eine Zerlegung des Backlog Item.

Leider werden wir immer noch oft gefragt, wie viele Personentage oder -stunden denn eine 8 bedeutet. Dies kann nicht für ein konkretes Team beantwortet werden und schon gar nicht generell, da eine 8 für jedes Team etwas anderes bedeutet und auch innerhalb eines Teams eine gewisse

Bandbreite vorhanden ist. Nach einigen Sprints kann oft eine Tendenz erkannt werden, dass ein Team beispielsweise zwischen 25 und 30 Story Points je Sprint umsetzt, aber selbst dies ist in Bezug auf Tage oder Stunden eine nicht verwertbare Information, da sich diese mit jeder Veränderung im Umfeld eines Scrum-Teams und der Anzahl der Netto-Sprint-Arbeitsstunden ändern kann.

Fairerweise wollen wir nicht unerwähnt lassen, dass Ron Jeffries, der Erfinder von Story Points, schon seit einigen Jahren von der Verwendung abrät: »*Story Points wurden erfunden, um die Dauer zu verschleiern, damit bestimmte Manager das Team nicht aufgrund von Schätzungen unter Druck setzen*« [URL:Jeffries c].

Personentage oder -stunden

Aus dem traditionellen Projektmanagement sind noch die Schätzungen in Tagen oder sogar Stunden bekannt. Dabei wird für ein bestimmtes Arbeitspaket (in unserem Fall: Backlog Item) geschätzt, wie lange die Umsetzung dauern wird. Aus der Summe der Schätzungen wird anschließend der Gesamtaufwand berechnet.

Das Problem der Schätzung in Tagen oder Stunden ist, dass sie den Anschein erwecken, es könne die Zukunft vorausgesagt werden. Hinzu kommt, dass Schätzungen oftmals zu einem Zeitpunkt verlangt werden, bei dem die Beteiligten am wenigsten Know-how besitzen: zu Beginn eines Projekts. Auf Basis dieser Zahlen werden Scrum-Teams dann oftmals »beim Wort« genommen. Es werden dann je nach Wichtigkeit und Kontext des Projekts alte Verhaltensweisen, wie z.B. Mikromanagement und Eingriffe ins Team, vorgenommen.

T-Shirt-Größen

Einige Teams ordnen den Backlog Items sogenannte T-Shirt-Größen zu, um sich komplett von Zahlen zu lösen. Dazu werden meist die folgenden Werte benutzt: XS, S, M, L, XL, XXL.

Dies ist eine schöne Variante, um zu verhindern, dass Zahlenwerte implizit doch wieder auf Personentage umgerechnet werden. Allerdings werden die Abstände zwischen den Schätzwerten nicht deutlich, und eine Velocity lässt sich auch nur berechnen, wenn man die T-Shirt-Größen wieder auf Zahlenwerte umrechnet.

Die sechs Kategorien der T-Shirt-Größen lassen sich hervorragend auf die ersten sechs Elemente der Scrum-Fibonacci-Reihe projizieren. So wird ein XS zu einer 1, ein S zu einer 2 usw. Auf diese Weise können Sie ohne Zahlen schätzen lassen und trotzdem anschließend eine Velocity berechnen.

Es gibt viele Möglichkeiten, Schätzwerte für Backlog Items zu ermitteln. Eine sehr bekannte und weitverbreitete Methode ist Planning Poker, das von James Grenning als Erstem beschrieben und später von Mike Cohn [Cohn 2005] populär gemacht wurde. Planning Poker werden wir aufgrund seines hohen Bekanntheitsgrades nicht weiter vertiefen, es kann bei Henning Wolf und Stefan Roock [WolfRoock 2021] nachgelesen werden.

Welche Skala Sie auch wählen, schränken Sie die Anzahl der verfügbaren Kategorien ein. Die Scrum-Fibonacci-Reihe beispielsweise kennt 10 unterschiedliche Schätzwerte, die unserer Erfahrung nach keinen Sinn ergeben, da insbesondere die großen Werte so sehr mit Unsicherheit und Risiken behaftet sind, dass sie wertlos sind. Wir haben gute Erfahrungen mit der Beschränkung auf vier bis maximal sechs Kategorien gemacht.

Beim Schätzen von Backlog Items geht es um das Erzielen eines Konsenses zwischen den Entwicklern. Schätzverfahren lassen sich im Wesentlichen in folgende Gruppen einordnen:

- **Zuordnung von Schätzwerten zu Backlog Items**
  Hier wird ein Backlog Item einzeln besprochen und von den Entwicklern mit einem Schätzwert versehen. Zu dieser Gruppe gehört z.B. *Planning Poker*.
- **Zuordnung von Backlog Items zu Schätzwerten**
  Bei dieser Gruppe liegen die Schätzwerte offen aus, und die Backlog Items werden ihnen zugeordnet. So funktioniert z.B. *Magic Estimation* [URL: Overeem].
- **Zuordnung von Backlog Items zu Backlog Items**
  Schätzwerte werden anfangs nicht genutzt, sondern die Backlog Items werden direkt zueinander in Relation gesetzt und die Gruppen erst am Ende mit Schätzwerten versehen. Das *Team Estimation Game* (siehe

Abschnitt 4.3.3) ist ein Beispiel für diese Gruppe. Diese Gruppe liefert unseres Erachtens die beste Kombination aus Geschwindigkeit und Genauigkeit.

4.3.3 Initiale Schätzung

Schock im Kick-off

Gleich zu Beginn des Kick-offs präsentierte Casper sein Produktziel und schaffte es, dass das ganze Team Feuer und Flamme für die neue App war. Auch Casper selbst hatte sich in den letzten Tagen schon viele Gedanken gemacht und informierte alle, dass er bereits ein Product Backlog mit 60 Backlog Items vorbereitet hatte, die er möglichst bald mit dem Team verfeinern und schätzen wollte.

Schlagartig änderte sich die Stimmung im Raum. Die Euphorie war dahin. Finn bemerkte dies und hakte nach. Dabei kam heraus, dass die Teammitglieder bisher nur Planning Poker als Schätzverfahren kannten und schlechte Erfahrungen damit gemacht hatten. Schätzrunden arteten in Diskussionen aus, Teammitglieder hielten sich zurück, weil sie nicht mehr wussten, worum es ging, oder weil sie den Termin mit ihrer Beteiligung nicht unnötig verlängern wollten. Viele Schätzungen wurden nach zehn Minuten abgebrochen, weil die Teammitglieder sich nicht einigen konnten. Selbst mit einem gut strukturiert durchgeführten Backlog Refinement würden sie mit Planning Poker zwei bis drei Tage nur für das Schätzen benötigen. Und im Endeffekt wären die Schätzungen sowieso nur sehr grob und müssten später verifiziert werden, also warum sollten sie sich das antun?

Als erfahrener Scrum Master konnte Finn die Teammitglieder glücklicherweise beruhigen. Er versprach, dass sie für die initiale Schätzung kein Planning Poker nutzen würden und dass sie die 60 Backlog Items im Handumdrehen geschätzt haben würden. Die Teammitglieder vertrauten auf Finns Erfahrung, und das Kick-off konnte gut gelaunt fortgesetzt werden.

Achten Sie als Scrum Master darauf, wie sich Teammitglieder während des Schätzens verhalten. Festzustellen ist häufig, dass einige eher eine passive Rolle einnehmen und andere den aktiven Part. Schätzungen sollten im Konsens entstehen und nicht als aufgezwängt wahrgenommen werden. Suchen Sie ggf. das Gespräch mit passiven Teammitgliedern, um die Gründe für die Zurückhaltung zu erfahren.

Zu Beginn eines neuen Projekts kommt es oft vor, dass der Product Owner vorgearbeitet hat und ein umfangreiches Product Backlog präsentiert. Da es sich in den meisten Fällen um Backlog Items handelt, bei deren Erstellung die Entwickler nicht eingebunden waren, sind diese Anforderungen in der Regel noch recht oberflächlich und auch noch nicht zu Ende durchdacht. Aus diesem Grund empfiehlt es sich, Schätzverfahren einzusetzen, die zügig zu einer ersten, gemessen an der aktuellen Qualität der Backlog Items guten Einschätzung der Komplexität führen.

Viele Scrum-Teams verwenden Planning Poker als universelle Schätzmethode. Dabei ordnen die Teammitglieder den vom Product Owner vorgestellten Backlog Items so lange Komplexitätspunkte (Story Points) zu, bis sich die Entwickler auf einen gemeinsamen Wert geeinigt haben. Zwischenzeitlich werden die Anforderungen immer wieder diskutiert. Dieses Verfahren ist per se gar nicht schlecht, weil es auf ein gemeinsames Verständnis des Backlog Item für alle Beteiligten abzielt.

Unter den oben beschriebenen Umständen halten wir Planning Poker nicht für eine geeignete Methode. Zum einen dauert das Schätzen eines umfangreichen Backlog sehr lange (4–7 Backlog Items pro Stunde sind ein erreichbarer Erfahrungswert), zum anderen haben die Anforderungen meistens noch nicht die Qualität, um sich detailliert mit ihnen auseinanderzusetzen. Warum auch sollte Zeit damit verschwendet werden, sich Gedanken über Anforderungen zu machen, die möglicherweise gar nicht oder durch zwischenzeitliches Feedback bzw. neue Erfahrungen stark verändert umgesetzt werden?

Schätzungen sollten immer in einer →Timebox durchgeführt werden, damit die Konzentration erhalten bleibt und alle fokussiert mitarbeiten.

Die Vorsortierung des Product Backlog durch den Product Owner sorgt dafür, dass stets die aktuell wichtigsten Backlog Items zuerst besprochen und geschätzt werden.

Wir stellen im Folgenden ein Verfahren vor, mit dem umfangreiche Product Backlogs sehr zügig und effektiv geschätzt werden können.

Team Estimation Game

Das Team Estimation Game wurde im Jahr 2008 von Steve Bockman [URL:Duan a] vorgestellt. Wir haben das Team Estimation Game mit verschiedenen Scrum-Teams durchgeführt und kommen im Schnitt auf eine Zeit von ungefähr einer

Minute pro Backlog Item; oft geht es sogar deutlich schneller. Teammitglieder, die es gewohnt waren, Planning Poker zu spielen, reagieren sehr positiv auf dieses schnelle und kurzweilige Verfahren. Neulinge heben oft den Spaßfaktor hervor, denn es wird oft und viel gelacht, insbesondere wenn bei einem Backlog Item Uneinigkeit herrscht und es mehrfach hin- und zurückbewegt wird.

Vorbereitungen

Alle Backlog Items, die geschätzt werden sollen, werden auf separate Karten geschrieben und liegen verdeckt als Stapel auf dem Tisch. Es wird eine freie Fläche (z.B. ein (digitales) Whiteboard) benötigt.

Das Scrum-Team ist komplett vertreten. Der Scrum Master in seiner moderierenden Funktion informiert zu Beginn alle über die Regeln und legt eine Timebox für die Durchführung fest. Die Entwickler stellen sich in einer Reihe hintereinander auf, der Product Owner steht vor den Entwicklern mit dem Gesicht zum ersten Entwickler in der Reihe.

Neben den unten beschriebenen Regeln für die Einsortierung der Backlog Items gilt als wichtigste Vereinbarung, dass während der Durchführung alle Entwickler, die nicht gerade an der Reihe sind, still sind und aufmerksam zuhören.

Bereiten Sie den Termin eng mit dem Product Owner vor und weisen Sie bereits im Vorfeld auf die Rolle während des Team Estimation Game hin.

Falls das Verfahren noch nicht bekannt ist, schreiben Sie die Regeln auf und erläutern Sie diese den Beteiligten.

Wenn möglich, bitten Sie die Entwickler, sich bereits vor dem Schätzen zumindest grob mit den zu schätzenden Backlog Items vertraut zu machen. Gegebenenfalls ergeben sich schon im Vorfeld grundsätzliche Fragen, die geklärt werden können.

Durchführung

Ein Entwickler fängt an und nimmt sich das oberste Backlog Item, liest die Anforderung laut vor und legt die Karte auf die Arbeitsfläche. Der Product Owner kann bei Bedarf um eine kurze Erläuterung des Backlog Items gebeten werden. An dieser Stelle ist das stille und aufmerksame Zuhören der Unbeteiligten besonders wichtig, um die Information zu nutzen, wenn sie selbst an der Reihe sind.

Es hat sich bewährt, die Karte durch den Product Owner vorlesen zu lassen, weil die Stimme und die Lautstärke der Vortragenden gleich bleiben und die Beteiligten sich daran gewöhnen können. Die Teammitglieder sprechen oft unterschiedlich laut oder vergessen sogar manchmal, die Anforderung überhaupt vorzulesen. Zudem kann der Product Owner gleich einige Erläuterungen zu dem jeweiligen Backlog Item geben und die Akzeptanzkriterien nennen.

Das nächste Teammitglied nimmt die nächste Karte, liest sie laut vor und legt sie ebenfalls auf die Arbeitsfläche. Es gibt dabei drei Möglichkeiten:

- Das Backlog Item ist *weniger komplex* als das bereits vorliegende, die neue Karte wird über die bereits auf der Fläche liegende Karte gelegt.
- Das Backlog Item ist ähnlich komplex wie das erste. Dann wird die Karte neben die bereits auf der Fläche liegende Karte gelegt.
- Das Backlog Item ist *komplexer* als das bereits vorliegende, die neue Karte wird unter die bereits auf der Fläche liegende Karte gelegt.

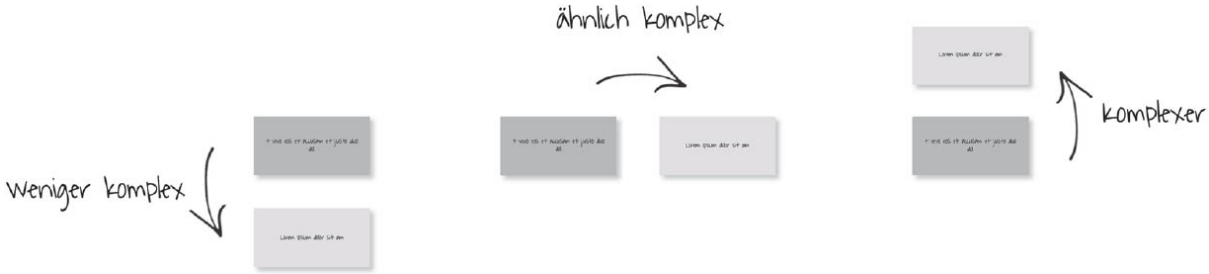

Abb. 4–15 *Möglichkeiten der Komplexitätsschätzung*

Ab der dritten Karte kann ein Teammitglied aus den folgenden Optionen wählen:

- **Schätzen** des nächsten Backlog Item wie oben beschrieben
- **Umlegen** einer bereits liegenden Karte, verbunden mit einer kurzen Erklärung – dabei sind keine Diskussionen erlaubt
- **Aussetzen**, weil das Teammitglied die Anforderung z.B. überhaupt nicht einschätzen kann

Umlegen: Markieren Sie die Karte beim Umlegen (z.B. mit einem farbigen Punkt). Wenn eine Karte mehrfach umgelegt wird, wird sie schließlich aus dem Spiel genommen, da sie offensichtlich noch nicht hinreichend verstanden wurde.

Aussetzen: Die Karte eines aussetzenden Teammitglieds sollte nicht einfach zu einem späteren Zeitpunkt neu verlesen werden, sondern sie sollte als »frei verfügbar« auf der Arbeitsfläche platziert werden. So kann die nächste Person, die an der Reihe ist, diese anstelle einer neuen Karte vom Stapel wählen.

Ende

Diese Runde ist beendet, wenn keine Backlog Items mehr verfügbar sind oder wenn die vorgesehene Timebox überschritten ist. Alle versammeln sich nun vor der Arbeitsfläche mit den Karten und begutachten das Ergebnis. Gemeinsam können nun noch Karten verschoben werden.

Auch für diesen Schritt sollte eine Timebox vorgesehen und mitgeteilt werden, damit die Teammitglieder fokussiert bleiben und sich nicht in Diskussionen verlieren. Die Timebox kann je nach Größe des Backlog variieren, maximal 10 Minuten sind ein guter Erfahrungswert.

Nun ist sehr schnell ein Überblick über die Komplexität des gesamten Product Backlog entstanden. Am Ende des Spiels wird die Blockbildung der Karten sichtbar, die sich hervorragend auf die Fibonacci-Reihe projizieren lässt. Sollten es zu viele Blöcke sein, kann gemeinsam mit allen überlegt werden, ob und wie Blöcke zusammengefasst werden können, z.B. indem die beiden »kleinsten« zu einem Block zusammengefügt werden. Abbildung 4–16 zeigt ein Beispiel eines Ergebnisses des Team Estimation Game.

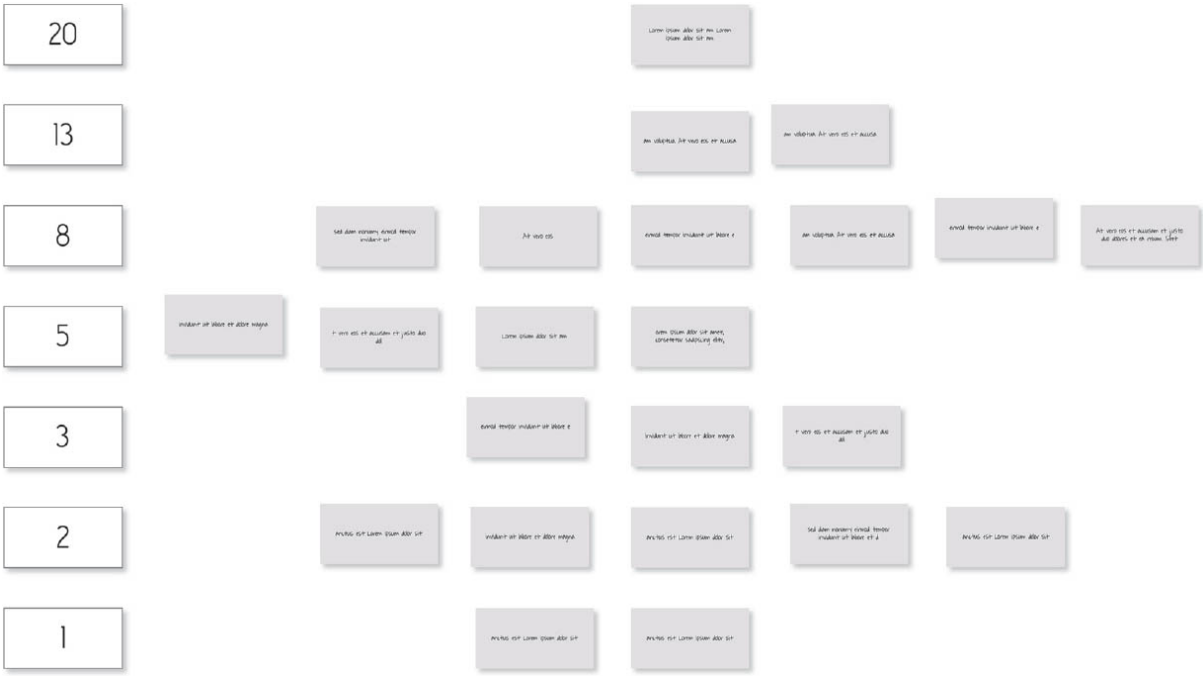

Abb. 4–16 *Team Estimation Game – Ergebnis*

Falls Sie schon vor dem Schätzen die Kategorien der Scrum-Fibonacci-Reihe einschränken möchten, können Sie vorher mit Klebeband Markierungen anbringen. Erfahrungsgemäß sind (je nach Projekt) Backlog Items ab einer bestimmten Komplexität zu groß, um sinnvoll mit ihnen zu arbeiten. Die Größe kann nicht generell festgelegt werden, sie unterscheidet sich von Projekt zu Projekt.

Vorteile

Das Verfahren kommt extrem schnell zu einem Ergebnis, weil die Backlog Items nicht diskutiert werden, sondern die Einschätzung einer einzelnen Person erfragt wird. Der Fokus liegt auf der gemeinsamen Konzentration auf das Wesentliche (dem Schätzen) und nicht auf einer zu diesem Zeitpunkt nicht erforderlichen Detaildiskussion. Die bei einigen Backlog Items notwendige Perspektive anderer Teammitglieder ergibt sich durch die Möglichkeit, Karten umzuhängen oder sie im letzten Schritt zu diskutieren. Des Weiteren ist sichergestellt, dass sich alle Teammitglieder aktiv an der Schätzung beteiligen.

Nachteile

Das Verfahren funktioniert sequenziell, d.h., es ist immer nur ein Teammitglied aktiv. Obwohl das Verfahren recht zügig abläuft, kann es bei größeren Scrum-Teams etwas dauern, bis eine Person wieder an der Reihe ist. Manchmal führt

dies dazu, dass Teammitglieder unkonzentriert sind und nicht zuhören, während andere an der Reihe sind, sodass sie ggf. wichtige Informationen verpassen.

Wir hören manchmal, dass durch das Weglassen der Diskussion bei diesem Verfahren die »Weisheit der Vielen« (engl. »Wisdom of the crowd«) nicht genutzt wird. Durch das abschließende gemeinsame Begutachten des Ergebnisses wird dies jedoch weitestgehend egalisiert.

Die Ergebnisse des Team Estimation Game sind nach unserer Erfahrung trotz der sehr viel schnelleren Vorgehensweise mindestens so gut wie die des sehr viel länger dauernden Planning Poker, sodass die Nachteile durch die Vorteile mehr als aufgewogen werden.

Wir setzen das Team Estimation Game nicht nur für initiale Schätzungen zu Beginn eines Projekts oder für große Product Backlogs ein, sondern auch in einer abgespeckten Variante in unseren wöchentlichen Backlog Refinements. Dazu wählen wir einmalig gemeinsam mit den Entwicklern für jede Kategorie von Story Points ein Referenz-Backlog-Item. Diese → Referenz-Items legen wir auf dem Tisch aus. Wenn der Product Owner z.B. fünf neue Backlog Items schätzen lassen möchte, bitten wir die Entwickler, jedes neue Backlog Item in Relation zu den auf dem Tisch liegenden Backlog Items zu schätzen.

4.3.4 Things-that-matter-Matrix

Wie war das gleich noch mal?

Im Sprint Planning besprach das Team alle relevanten Backlog Items für den Sprint. Am Ende fragte Finn, welche Backlog Items das Team in den Sprint übernehmen will, und plötzlich ging es los: »Kannst du das Item XY noch mal kurz zeigen?«, »Gehörten da eigentlich die Designs dazu?«, »Für Item YZ war jetzt nur dies und das notwendig, richtig?«.

Glücklicherweise hatte Finn im Rahmen eines Big-Picture-Workshops zu Beginn des Projekts gemeinsam mit Casper und den Entwicklern alle Backlog Items besprochen und die für das Team wichtigen Informationen in einer Matrix festgehalten. Vor dem Sprint Planning hatte er die damals dokumentierten Informationen für die vorgesehenen Backlog Items auf einem Flipchart notiert. Auf einen Blick kann das Team nun erkennen, welche Backlog Items

voraussichtlich einen besonders hohen Testaufwand haben, ob es externe Abhängigkeiten gibt und vieles mehr.

Während des Backlog Refinement fällt uns häufig auf, dass ein Team Backlog Items zum Vergleich heranzieht, über die wir gerade vor ein paar Minuten gesprochen haben. Das ist einerseits sehr gut, da die Backlog Items zueinander in Beziehung gesetzt werden, andererseits führt der permanente Wechsel zwischen den Backlog Items auch dazu, dass der Fokus verloren geht. Als visuelles Hilfsmittel haben wir die »Things-that-matter-Matrix« (TTM-Matrix) von James King schätzen gelernt [URL:King]. Sie strukturiert die Backlog Items in einer übersichtlichen Tabelle und hilft dabei, ein besseres Gefühl für die Backlog Items zu entwickeln und sie auf einer selbst definierten Ebene miteinander zu vergleichen. Wir benutzen die TTM-Matrix im Backlog Refinement zum Schätzen und Vergleichen neuer Backlog Items.

Vorbereitungen

Vor der ersten Nutzung der TTM-Matrix wird eine Tabelle auf ein Flipchart gezeichnet, und in die erste Spalte werden die IDs der Backlog Items eingetragen. Alle weiteren Spalten enthalten zunächst noch keine Überschriften. Der Scrum Master erklärt den Aufbau und das Ziel der Matrix und bittet das Team, sich für einige Kategorien zu entscheiden, die als Spaltenbeschriftungen eingetragen werden. Dies können z.B. Technologien, andere Teams oder fachliche oder technische Abstraktionsebenen sein, die dem Team helfen, die Komplexität der Items einzuschätzen. Abbildung 4–17 zeigt ein Beispiel einer leeren TTM-Matrix.

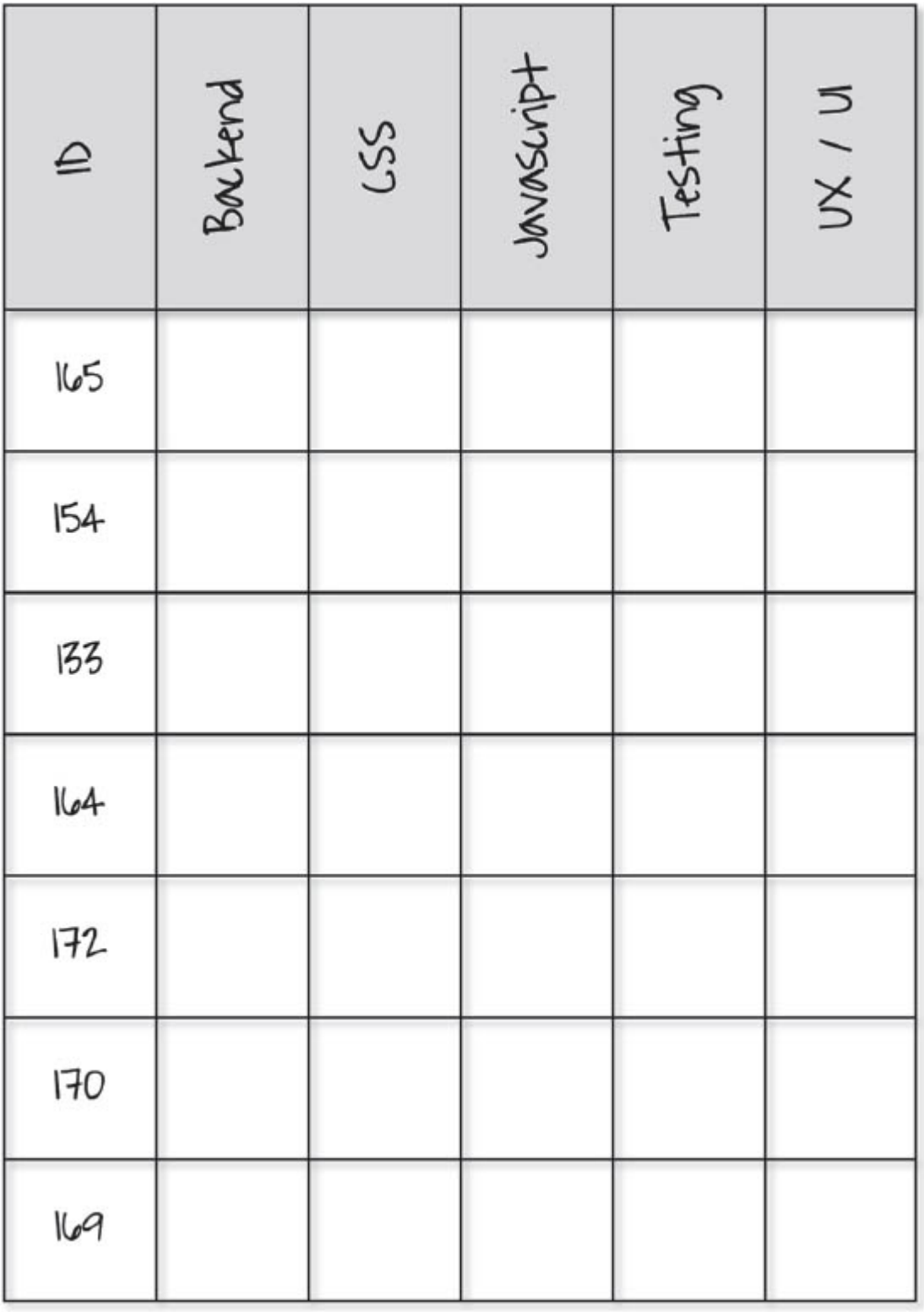

Abb. 4–17 *Leere TTM-Matrix*

Durchführung

Die Backlog Items werden nun nacheinander vom Product Owner vorgestellt, und die Entwickler geben eine grobe Einschätzung der Komplexität der einzelnen Kategorien. Wir benutzen in der Regel die T-Shirt-Größen S, M, L (vgl.

Abschnitt 4.3.2). Diese reichen häufig aus, da es sich nur um eine grobe Schätzung und Orientierung handelt.

ID	Backend	CSS	Javascript	Testing	UX / UI
165	S			S	
154	S	S		S	S
133	M	L		L	
164	L	M		M	
172	M			M	M
170		S	S	M	L
169	S			S	

Abb. 4–18 *TTM-Matrix mit T-Shirt-Größen*

Wir benutzen die TTM-Matrix je nach Kontext unterschiedlich. Geht es um eine initiale Schätzung (vgl. Abschnitt 4.3.3), lassen wir die Entwickler zunächst alle

Backlog Items nach den Kategorien bewerten. Wenn alle Backlog Items mit T-Shirt-Größen versehen sind, fangen wir wieder oben an und bitten das Team um eine Schätzung des ersten Backlog Item mit Story Points. Diesen Schritt wiederholen wir mehrmals, bis wir an einen Punkt kommen, an dem die Bewertung der Kategorien identisch oder sehr ähnlich mit einem bereits geschätzten Backlog Item ist. Nun stellen wir die folgende Frage:

> *Das Backlog Item oben habt ihr mit einer 5 geschätzt, das legt nahe, dass dieses Backlog Item auch mit einer 5 geschätzt werden kann. Ist das korrekt?*

In den meisten Fällen trifft das zu. Indem sich die Entwickler inhaltlich mit den Themen beschäftigen, fällt ihnen die Schätzung quasi in den Schoß, da ähnliche Konstellationen bereits geschätzt wurden.

Wir fügen oft noch eine zusätzliche Spalte am Anfang mit der Überschrift »Geschäftswert« ein. Nachdem der Product Owner die User Story vorgestellt hat, fragen wir die Entwickler als Erstes, ob alle den Wert der User Story verstanden haben. Falls nicht, bekommt der Product Owner eine weitere Chance, den Geschäftswert herauszustellen. Ist er danach immer noch nicht klar, kann man sich das Betrachten der weiteren Spalten sparen und mit der nächsten User Story fortfahren.

Auch wenn in einem Backlog Refinement nur ein paar wenige neue Backlog Items geschätzt werden, kommt die TTM-Matrix zum Einsatz. Wir hängen dazu die Flipcharts mit den bereits bewerteten und geschätzten Backlog Items gut sichtbar an die Wand, sodass die Entwickler sofort erkennen können, ob die gerade für das aktuelle Backlog Item besprochene Bewertung schon einmal vorkam und die Schätzung übernommen werden kann.

Mit der Zeit kommt eine Menge Flipcharts zusammen, die nicht alle in jedem Backlog Refinement benötigt werden. In diesem Fall wählen wir einige repräsentative Backlog Items aus, sammeln sie auf einem Flipchart und benutzen nur noch dieses als Referenz.

4.3.5 Häufige Herausforderungen

Das Team möchte vorab eine Research Story

Vor allem Scrum-Teams, die vorher in einem traditionellen Kontext gearbeitet haben, tun sich oft mit dem Schätzen schwer. Sie sind es gewohnt, dass Schätzungen möglichst genau sein sollen, und kommen nicht gleich mit den Unsicherheitsbereichen der Story Points zurecht. Solche Scrum-Teams fragen oft nach sogenannten Research oder Forschungs-Stories, in denen der technische Weg erarbeitet und die anschließende Schätzung scheinbar genauer wird. In Ausnahmefällen ist solch ein Vorgehen legitim (vgl. Abschnitt 4.2.5, Prototypen), aber im Alltag sollten Research Stories keine Rolle spielen.

Erläutern Sie den Entwicklern ggf. erneut das Prinzip von Story Points und der darin enthaltenen Unsicherheit bei höheren Schätzwerten. Vervollständigen Sie die Erläuterung mit der Herleitung der Velocity und der damit verbundenen Releaseplanung. Zeigen Sie so, dass es meist keinen Wert hat, möglichst exakt zu schätzen, weil die Unsicherheit schon eingeplant ist.

Schätzungen sind immer sehr hoch

In Abschnitt 4.3.2 haben wir uns die Scrum-Fibonacci-Reihe angesehen und dabei festgehalten, dass jedes Scrum-Team seinen eigenen Schätzmaßstab hat. Trotzdem treffen wir immer wieder auf Teams, die ausschließlich hohe Schätzungen haben, oft mindestens eine 8 oder mehr. Es kann natürlich mal vorkommen, aber generell sollte zumindest die Spanne zwischen 1 und 8 ausgenutzt werden.

Oft liegt der Grund darin, dass die Entwickler sich nicht trauen, zu niedrig zu schätzen, und daher immer einen großzügigen Puffer einplanen. Finden Sie heraus, was der Grund dafür ist. Es könnte z.B. sein, dass die Entwickler schon mal Ärger bekommen haben, weil sie ihre Schätzung nicht eingehalten haben.

Story Points werden in Personentage umgerechnet

Selbst wenn mit Story Points geschätzt und damit die Velocity berechnet wird, gibt es immer wieder Versuche, die Story Points in Personentage umzurechnen: *»Ihr seid 8 Entwickler, ihr sprintet 10 Tage, macht 80 Tage. Eure Velocity ist 20 Story Points, also entspricht 1 Story Point 4 Tagen. Eine User Story mit 5 Story Points benötigt also 20 Tage Aufwand. Bingo!«*

Was für eine fatale Fehlannahme! Wer bei der Erklärung der Story Points aufgepasst hat, erinnert sich, dass Komplexität geschätzt wird und dass eine 5 »irgendwo zwischen 3 und 8« bedeutet. Wenn die obige Rechnung sinnvoll wäre, würde dies bedeuten, dass die User Story einen Aufwand zwischen 12 und 32 hat. Damit wird sich kein Projektcontroller zufrieden geben, der seinen Job ernst nimmt.

Zeigen Sie die Fehleranfälligkeit der Umrechnung wie im obigen Beispiel auf und versuchen Sie herauszufinden, wozu diese Umrechnung dienen soll. Was ist das Ziel? Geht es um Planung, um Kosten oder etwas ganz anderes? Zeigen Sie dann, wie das Ziel auf andere Art erreicht werden kann, z.B. durch Nutzung der Velocity zur Releaseplanung (siehe Abschnitt 4.5).

Schätzungen werden als Vorhersage der Zukunft interpretiert

Im traditionellen Projektmanagement wird sehr viel Wert auf genaue Schätzungen gelegt und auch eine Menge Zeit dafür investiert. Dieses Denken wird häufig auf agile Projekte übertragen und daher herrscht oft die gleiche Erwartungshaltung an die Genauigkeit von Schätzungen. In agilen Projekten schätzen wir allerdings nur grob und verwenden die Zeit lieber dafür, schnell erste Ergebnisse zu liefern und daraus zu lernen.

Arbeiten Sie auch mit dem Umfeld des Scrum-Teams und machen Sie immer wieder klar, wie geschätzt wird, warum auf diese Weise geschätzt wird und welche Vorteile diese Vorgehensweise hat.

Oftmals ist die einzige Möglichkeit, sich dem Drang nach genauen Schätzungen zu erwehren, das laufende Liefern von Inkrementen. Dies bedingt dann eine Umkehr der Haltung von denjenigen, die Schätzung verlangen. Sie sind aufgefordert, mit Sorge dafür zu tragen, dass sich ein Scrum-Team auf seine Arbeit konzentrieren kann.

Zu diesem Schwerpunkt finden Sie unter *scrum-in-der-praxis.de/das-buch/checklisten/* die Checklisten »Estimation-Praxistipps« und »Schätzungen

durchführen«.

4.4 Backlog Refinement

Durchkämmt die Wüste!

Regelmäßig traf sich das komplette Scrum-Team, um gemeinsam am Product Backlog zu arbeiten und die nächsten Schritte zu planen. Zunächst trugen alle das Feedback zusammen, das sie während des letzten Sprint-Reviews aufgenommen hatten. Einige Gäste des Sprint-Reviews hatten Casper nach dem Event auch persönlich angesprochen und Vorschläge gemacht. Alle gemeinsam besprachen die Punkte und prüften, ob sie einen Wert für das Produktziel besaßen. Einige Anregungen wurden in die Akzeptanzkriterien bestehender Backlog Items aufgenommen, einige wurden zu neuen Backlog Items entwickelt und einige wurden verworfen.

Nachdem das Feedback verarbeitet war, betrachtete das Scrum-Team gemeinsam die Backlog Items, die im Product Backlog weit oben standen. Ein sehr umfangreiches Backlog Item war noch nicht weiter unterteilt worden. Casper war sich noch nicht sicher, ob es überhaupt jemals umgesetzt werden sollte. Daher wurde es zunächst ignoriert. Alle erstellten gemeinsam die Akzeptanzkriterien für vier zukünftige Backlog Items, die in einem der nächsten Sprints erledigt werden sollten.

Im nächsten Schritt wurden die neuen und geänderten Backlog Items geschätzt bzw. erneut geschätzt. Finn hatte dazu ein paar bereits erledigte Backlog Items unterschiedlicher Größe aus den letzten Sprints mitgebracht, damit die Entwickler die neuen Backlog Items in Relation zu den erledigten schätzen konnten.

Zwei der neuen, eben erstellten Backlog Items waren auch dabei, weil sie sehr einfach umzusetzen waren. Mina hatte Einwände gegen ein Backlog Item, weil es nicht so einfach zu testen sei, wenn nicht ein anderes, für später vorgesehenes Backlog Item vorher erledigt würde. Casper überlegte kurz und entschied dann, dass das andere Backlog Item höher priorisiert wird und ein Kandidat für den nächsten Sprint sein würde. Lara wies darauf hin, dass die vorgesehenen Backlog Items noch ihre Zuarbeit als Designerin benötigen, und schlug vor, zunächst zwei andere Backlog Items vorzuziehen. Alva bot an, ihr bei den Designs zu helfen.

Zum Abschluss des Backlog Refinement aktualisierten alle gemeinsam den Releaseplan. Damit traf das Team gewissermaßen schon eine Vorauswahl für den nächsten Sprint. Auf Basis der Velocity wurden die nächsten Sprints grob mit Backlog Items befüllt.

Stellen Sie sich doch einmal folgendes Szenario vor. Seit einigen Wochen liegt Ihnen die Familie mit dem Wunsch in den Ohren, endlich mal wieder das

Lieblingsessen der Familie zu kochen. Nun ist der Tag gekommen und Sie haben sich für den Abend vorgenommen, das Essen zuzubereiten. Sie wollen die Arbeit beginnen und stellen erschreckt fest, dass die Küche durch ein Kunstprojekt Ihrer Tochter belegt ist und diverse Zutaten fehlen. Ihr Plan löst sich in wenigen Minuten in Luft auf.

Wenn wir dieses Bild jetzt einmal für ein Scrum-Team verwenden, dann könnte es sein, dass erst im Sprint Planning deutlich wird, dass es nicht losgehen kann. Geholfen hätte ein Gespräch, das die Familie informiert sowie bei der Vorbereitung einbezieht, sodass der Zubereitung nichts im Wege steht. Letztendlich sorgt ein Backlog Refinement genau für diesen Zustand – ein Gefühl für die anstehende Arbeit zu erhalten und die Arbeit in kleine Einzelschritte zu zerlegen.

Wir Autoren fragen uns seit Langem, warum das Backlog Refinement immer noch keinen Einzug in den Scrum Guide als weiteres Scrum-Event gehalten hat. Er beschreibt jedoch (*scrumguides.org*):

> *»Das Refinement des Product Backlogs ist der Vorgang, durch den Product-Backlog-Einträge in kleinere, präzisere Elemente zerlegt und weiter definiert werden. Dies ist eine kontinuierliche Aktivität, wodurch weitere Details wie Beschreibung, Reihenfolge und Größe ergänzt werden. Die Attribute variieren oft je nach Arbeitsumfeld.«*

Der Formulierung des Scrum Guide folgend, empfehlen die meisten Scrum Master einem Scrum-Team, ein Backlog Refinement in die Sprints mit aufzunehmen.

4.4.1 Ziele

Der Zweck des Backlog Refinement ist es also, große Arbeitspakete in kleinere zu zerlegen. Daraus ergibt sich für die Bearbeitung eine schnellere Fertigstellung und somit frühzeitiges Feedback durch die Stakeholder.

Das Backlog Refinement gibt einem Scrum-Team die Möglichkeit, gemeinschaftlich am Produkt zu arbeiten und es aktiv mitzugestalten. Dabei erzeugt das Backlog Refinement ein gemeinsames Verständnis über die

kommenden Schritte, die zu klärenden Fragen oder technischen wie organisatorischen Herausforderungen. Neben den Entwicklern profitieren vor allem Product Owner vom Backlog Refinement, da einerseits Feedback zu den Backlog Items durch die Entwickler erfolgt und somit andererseits die Entscheidungsfindung, was als Nächstes zu tun ist, unterstützt wird. Die Sortierung und die dadurch entstehende Priorisierung des Product Backlog bleibt weiterhin in der Verantwortung des Product Owners.

Backlog Refinements sind eine gute Gelegenheit, um auch die Meinung und Unterstützung von Dritten mit einzubeziehen. Frühzeitig jemanden aus einem anderen Team einzuladen, damit diese Person die geplanten Maßnahmen mitnehmen und geeignete Maßnahmen planen kann, macht genauso Sinn, wie partiell Stakeholder einzuladen, die ihre fachlichen Anforderungen direkt einbringen.

In einem Artikel vergleicht John Sonmez sehr anschaulich das Product Backlog mit einer unaufgeräumten Wohnung, deren Bewohner umziehen möchte [URL:Sonmez]. Wenn nun die Umzugshelfer (Entwickler) am Tag des Umzugs (Sprint-Beginn) dieses Chaos (ungepflegtes Product Backlog) vorfinden, verschwenden sie wertvolle Zeit, weil sie warten müssen, bis der Bewohner (Product Owner) die Dinge so sortiert, dass sie umgezogen (entwickelt) werden können. Wäre dies vorher passiert (Backlog Refinement), würde der Umzug (Sprint) viel effizienter vonstattengehen.

Das Ergebnis des Backlog Refinement ist ein Product Backlog, das inhaltlich auf dem aktuellen Stand ist, soweit möglich sinnvoll geschätzte Backlog Items enthält und vom kompletten Scrum-Team mitgetragen wird. Darüber hinaus ergibt sich ein aktualisierter Releaseplan, der dem Scrum-Team und der Organisation einen Überblick über die voraussichtliche Verfügbarkeit von Funktionalitäten und über die nach heutigem Stand aktuelle Restlaufzeit des Projekts gibt.

Als Scrum Master sollten Sie Product Owner dabei unterstützen, dass das Product Backlog nicht ein Sammelbecken für Ideen und Anforderungen wird. Dies führt zu Unübersichtlichkeit und einem unnötigen Pflegeaufwand.

Product Owner sollten den nächsten Sprint gut mit den Entwicklern vorbereitet haben und gemeinsam in die darauffolgenden zwei Sprints blicken, ohne hier schon die Detailplanung fertig zu haben.

Bis vor einiger Zeit gab es anstatt des Backlog Refinement ein regelmäßiges Estimation. Man hat jedoch mehr und mehr festgestellt, dass es über das Estimation hinaus noch eine Menge zu besprechen und zu planen gibt und dass die Einbindung der Entwickler in die Entscheidungsfindung des Product Owners einen enormen Mehrwert für das Projekt darstellt. So ist aus dem ehemaligen Estimation das heutige Backlog Refinement geworden, das in der Regel ein Estimation als Teil der Agenda enthält.

Auch unterscheidet es sich deutlich von einem Sprint Planning, da das Scrum-Team im Backlog Refinement das Produkt oder Projekt gemeinsam mittelfristig auf einer strategischen Ebene plant, wohingegen es im Sprint Planning lediglich operativ die nächsten Schritte plant.

Ein Backlog Refinement umfasst schwerpunktmäßig folgende Aktivitäten:

- **Klärung von Backlog Items**
  Es werden diejenigen Backlog Items durch den Product Owner zur Klärung gebracht, die in naher Zukunft umgesetzt werden. Dazu gehört auch, dass Backlog Items in kleinere Backlog Items zerlegt und auftretende Abhängigkeiten oder offene Punkte angegangen werden.
- **Schätzen von Backlog Items**
  Auf Basis der vorherigen Klärung und wenn ausreichend Details zur Verfügung stehen, schätzen alle Entwickler die Komplexität.
- **Pflege des Product Backlog**
  Auf Basis der ausgetauschten Informationen können neue Backlog Items hinzugefügt, entfernt sowie neu angeordnet werden. Daraus resultiert eine Anpassung des Sprint- bzw. Releaseplans.

Ermutigen Sie alle Beteiligten, sich über die regelmäßigen Backlog Refinements hinaus Gedanken zu den Backlog Items, zur Architektur und zum Design zu machen. Diese Gedanken und Gespräche finden quasi immer und überall statt.

4.4.2 Ablauf

Für das Backlog Refinement hat sich ein wöchentliches Treffen des kompletten Scrum-Teams bewährt, in dem die nächsten Backlog Items durchgegangen werden. Dies fördert eine Kultur des »selbstverwaltenden Mitentwickelns«. Da die Entwickler sich bereits hier intensiv mit den Backlog Items auseinandersetzen, gibt es den angenehmen Nebeneffekt, dass die Sprint Plannings sehr viel schlanker durchgeführt werden können.

Das Event fällt in den Verantwortungsbereich der Product Owner, die für das Stattfinden zuständig sind und das Arbeitstreffen einberufen. Die Dauer und Häufigkeit des Events hängen u.a. vom jeweiligen Projektkontext ab, insbesondere auch von der Sprint-Länge. Für zweiwöchige Sprints hat sich bei uns ein wöchentlicher Termin von einer Stunde bewährt, der im Einzelfall abgesagt werden kann, wenn sich nicht genügend Themen finden, die eine Unterbrechung der Entwicklung rechtfertigen würden. Das komplette Scrum-Team nimmt teil, ggf. bei Bedarf ergänzt um Personen mit Spezialwissen, die bei konkreten Fragestellungen behilflich sein können.

Product Owner können das Arbeitstreffen vorbereiten, indem sie:

- das Product Backlog anhand der letzten Erkenntnisse ordnen,
- die zu besprechenden Backlog Items vorbereiten auf Basis von Feedback, z.B. aus dem Sprint-Review, Gesprächen mit Stakeholdern, Umfragen oder Statistiken,
- Personen benennen bzw. einladen, die neben dem Kernteam anwesend sein sollen und etwas beitragen können.

Folgender Ablauf kann als Orientierung für die Durchführung dienen:

Check-in

Gesamtvorstellung der Product Backlog Items, die besprochen werden sollen
Wiederholung des Produktziels
Erzählen von Anekdoten, Feedback oder Kennzahlen

Vorstellung der Backlog Items (Timebox z.B. 7 Minuten) und Klärung der folgenden Fragen mit allen Entwicklern:

Ist die Beschreibung vollständig und von allen verstanden?
Ist die Größe der Anforderung sinnvoll?
Sind Abhängigkeiten zu Dritten berücksichtigt?
Sind die Akzeptanzkriterien vollständig?

Sind Testfälle formuliert?
Sind benötigte Zuarbeiten oder notwendige Klärungen vorab bereits vorhanden oder angestoßen?
Ist die Definition of Ready erfüllt?
Wer übernimmt welche offenen Aufgaben bis zum nächsten Refinement oder Sprint Planning?

Schätzung aller Entwickler nach gewählter Schätzmethode

Check-out

Wiederholung der offenen Aufgaben
Feedback zum Refinement

Es kann natürlich auch sein, dass komplett neue Anforderungen durch den Product Owner präsentiert werden. In diesem Fall können ein Einsammeln eines ersten Feedbacks oder das gemeinsamen Schreiben der User Story oder Akzeptanzkriterien anfallen.

Hierzu kann der Ablauf des Refinement angepasst und das Team in Kleingruppen aufgeteilt werden. Diese gehen dann ins Gespräch und verfeinern die Backlog Items. Nach Ablauf der Zeit werden die Ergebnisse und Fragen diskutiert und gemeinsam geklärt, was bis zum nächsten Refinement zu tun ist.

Schritt 2 und 3 wird für jedes Backlog Item wiederholt. Sobald die vereinbarte Timebox nicht ausreicht, wird gemeinsam entschieden, ob eine weitere Timebox genutzt wird.

Im Anschluss an das Refinement wird der Releaseplan (siehe Abschnitt 4.5) durch den Product Owner aktualisiert.

Versehen Sie die einzelnen Agendapunkte vor jedem Refinement mit einer Timebox. Die Dauer hängt natürlich vom aktuellen Stand des Projekts ab und variiert damit jedes Mal. Halten Sie diese Zeitbegrenzungen ein, sonst laufen Sie Gefahr, aus jedem Backlog Refinement ohne aktualisierten Releaseplan herauszugehen.

Für ein gezieltes Voranschreiten lohnt es sich auch, eine Timebox für das Besprechen und Schätzen von einzelnen Backlog Items festzulegen. Der Scrum Master hat diese Timebox im Blick und klärt nach Ablauf mit den Beteiligten, ob die auftretenden Fragen ausreichend

beantwortet wurden oder das Backlog Item bis zum nächsten Refinement auf Nachbesserung warten muss.

Wie ein Backlog Refinement letztendlich durchgeführt wird, liegt im Ermessen eines Product Owners. Wir haben die Erfahrung gemacht, dass ein gleichbleibender Ablauf für Rhythmus und inhaltliche Orientierung sorgt.

Auch wenn wir bis hierher zentral Product Owner in der Pflicht gesehen haben, das Backlog Refinement vorzubereiten und einzuberufen, so gilt dies auch für Entwickler. Entwickler sollten Product Owner darüber informieren, wenn auch sie Backlog Items im Rahmen des Backlog Refinement besprechen möchten. Diese Anforderungen sollten schon im Vorfeld im Product Backlog vorhanden sein und von den Entwicklern gepflegt werden. Im Backlog Refinement erhalten dann alle, vor allem auch Product Owner, die notwendigen Informationen, um Entscheidungen für die Sortierung des Backlog Item treffen zu können.

Story Owner

Oftmals wird im Refinement deutlich, dass noch Aufgaben oder Fragen zu Backlog Items offen geblieben sind. Unsere Empfehlung lautet dann häufig, dass sich ein Entwickler pro Backlog Item findet, der sich um die Erledigung dieser offenen Punkte federführend kümmert. Diese Personen fungieren dann für Product Owner als direkte Ansprechpartnerinnen.

Wir gehen oft noch einen Schritt weiter und lassen neben dem Story Owner ein zweites Teammitglied aussuchen, das für den »Support« des Story Owners zuständig ist. Diese Person kommt z.B. im Falle der Abwesenheit des Story Owners als Kontaktperson infrage.

Oftmals ergibt sich aus dieser Konstellation ein Dreiergespann aus Product Owner, Story Owner und Supporter. Diese Art des Vorgehens macht auch Sinn, um z.B. die Entwickler zu mehr Zusammenarbeit zu bringen, Verantwortung schon zu einem frühen Zeitpunkt zu übernehmen und Wissen zwischen Story Owner und Supporter zu verteilen.

Die konkrete Ausprägung der Aufgaben eines Story Owners ist abhängig von der jeweiligen Definition innerhalb des Scrum-Teams. Folgende Tätigkeiten könnten z.B. dazugehören:

- *Erfüllung der DoR*
  Story Owner unterstützen Product Owner dabei, die offenen Fragen und Aufgaben zu klären, sodass ein Backlog Item »Sprint-ready« wird.

- *Höchste Priorität sicherstellen während des Sprints*
  Ein Story Owner, der am obersten Sprint Backlog Item arbeitet und bemerkt, dass jemand aus dem Team an weniger hoch priorisierten Sprint Backlog Items arbeitet, sollte diese Person um Mithilfe bitten, zunächst die Arbeit am höher priorisierten Sprint Backlog Item zu erledigen.
- *Überprüfung der DoD*
  Bevor das Sprint Backlog Item abgenommen werden kann, prüft der Story Owner, ob alle vereinbarten Rahmenbedingungen eingehalten wurden.
- *Einholen der Akzeptanz des Product Owners*

Achten Sie darauf, dass diese Aufgabe nicht zu viel Gewicht bekommt. Es handelt sich eher um eine »Patenschaft«. Erinnern Sie daran, alle im Scrum-Team sind ergebnisverantwortlich. Insbesondere bei Scrum-Teams mit wenig Erfahrung in der Selbstorganisation ist dies ein gutes Hilfsmittel, um den Fokus auf die notwendigen Aufgaben zu lenken und das Verantwortungsbewusstsein zu stärken.

4.4.3 Häufige Herausforderungen

Es wird nur geschätzt

Ein Scrum-Team ist nicht in der Lage, seine Commitments einzuhalten, und reißt einen Sprint nach dem anderen. Der Kunde der Agentur wird zunehmend unruhig und verlangt laufend ein erneutes Schätzen des gesamten Product Backlog. Da das Scrum-Team mittlerweile sehr verunsichert ist, da nicht nur der Druck vom Kunden wächst, sondern der Kunde auch immer Recht hat, folgen alle dieser Aufforderung.

Wie in diesem Beispiel dargestellt, wird von einigen Scrum-Teams verlangt, dass sie alles schätzen, was im Product Backlog steht. Vorrangig um Sicherheit zu erlangen. Das Problem dabei ist jedoch, dass komplexe Arbeit sich nicht durch Schätzung in Luft auflöst und besser beurteilt werden kann. Notwendig sind die direkte Arbeit an Lösungen und das Durchführen von kleinen Experimenten, die Entwicklung von Prototypen oder Fertigstellung von Produktinkrementen, die Inspect & Adapt ermöglichen. Das Refinement nur dafür zu verwenden, um Schätzungen durchzuführen, ist eine Beschäftigungsaufgabe und somit Verschwendung. Ganz davon abgesehen, dass ein Product Backlog lebt und sich ständig verändert.

Wenn Sie früh im Projekt eine grobe Schätzung abgeben sollen, dann nutzen Sie die initialen Schätzverfahren aus Abschnitt 4.3.3 und »refinen« Sie danach das Product Backlog. Durch diese Arbeit erlangen die Beteiligten Klarheit für die nächsten zwei bis drei Sprints. Alles andere bleibt vorerst unberührt und rückt erst dann in den Fokus, wenn es relevant ist.

Darüber hinaus hilft die Pflege eines Releaseplans (siehe Abschnitt 4.5.1), um Unsicherheiten zu minimieren.

Unvorbereitete Backlog Refinements

Backlog Refinement ist eine laufende Aktivität des gesamten Scrum-Teams. Product Owner sind diejenigen, die zum Arbeitstreffen einladen. Demnach ist es auch ihre Verantwortung, respektvoll mit der Zeit der Teilnehmenden umzugehen.

Scrum Master sollten den Mut haben, eine fehlende Vorbereitung eines Product Owners anzusprechen, und auf eine Verschiebung des Refinements drängen. Product Owner möchten das Wissen der Entwickler anzapfen, sodass sie gegenüber Dritten aussagekräftig sind und ihre Planung durchführen können. Daneben gilt es, die Entwickler so wenig wie möglich aus dem Arbeitsfluss während eines Sprints herauszuzerren und für Unterbrechung zu sorgen.

Scrum Master sollten genau hinschauen, wenn Product Owner keine Zeit haben, am Product Backlog zu arbeiten. Prüfen Sie Indikatoren, ob es Grund zum Handeln gibt, wie z.B.: *Ist der nächste Sprint vorbereitet? Sind die User Stories vollständig? Ist das Product Backlog gepflegt? Sind Sprint- und Releaseplanung im Einklang?*

Erinnern Sie vor dem Backlog Refinement die Teammitglieder an das Arbeitstreffen, sodass diese Zeit einplanen, um sich darauf vorzubereiten. Beziehen Sie Product Owner ein, sodass diese schon auf die Backlog Items hinweisen, die es zu besprechen gilt.

Nicht alle nehmen teil

Einige tendieren dazu, nicht alle Entwickler zum Backlog Refinement einzuladen, sondern nur einen Teil. Wir hingegen sind der Meinung, dass die Einberufung des Arbeitstreffens alle angeht, da alle an den Aktivitäten beteiligt sein sollten.

Gleichzeitig gehen so relevante Informationen oder (Vor-)Gespräche nicht verloren. Zudem könnte es die Tendenz geben, vorrangig besonders erfahrene Entwickler einzuladen. Dies könnte dazu führen, dass ein stärkeres Wissens-Ungleichgewicht entsteht, unerfahrene Entwickler sich zurückhalten oder den erfahrenen Entwicklern weitaus mehr Zeit von der Arbeit am Sprint-Ziel genommen wird.

Auch hierbei ist Pragmatismus gefragt, wenn z.B. einige Entwickler in der zweiten Sprint-Woche anderes vorhaben, als sich in ein Arbeitstreffen zu setzen. Grundsätzlich sehen wir es jedoch als gemeinschaftliche Aufgabe, deren Sinn alle verstehen sollten, die auf ein gemeinsames Ziel hinarbeiten.

Komplexität wird nicht anerkannt

Es entsteht oftmals der Eindruck, dass Backlog Refinement etwas ist, was weggelassen oder in einem einstündigen Termin während eines Sprints abgehandelt werden kann. Diese Annahme ist jedoch falsch und es ist ein Bewusstsein bei allen notwendig, dass für komplexe Herausforderungen nicht eben mal ein Meeting einberufen werden kann. Oftmals wird aus den Diskussionen erkenntlich, dass diese sich im Kreis drehen, sich wiederholen oder die Teammitglieder sich in Details verlieren.

Der Entwicklungsprozess einer Anforderung ist in den meisten Fällen mehrstufig und es wird eine wiederholte Auseinandersetzung mit einem Backlog Item notwendig. Ein Scrum-Team versucht auf diese Komplexität mit dem Know-how aus unterschiedlichen Disziplinen zu reagieren. Eventuell ist der oben beschriebene Ablauf und die Durchführung in einem beengten Meetingraum nicht der Ort, wo Kreativität auf Knopfdruck entstehen kann. Scrum Master sollten im Blick behalten, dass eventuell andere Wege und Formate genutzt werden sollten als ein Arbeitstreffen, das einmal in der Woche in einem Meetingraum stattfindet.

Unterstützen Sie Product Owner und Entwickler dabei, indem Sie den Raum für Kreativität öffnen. Sei es z.B. durch das Angebot eines begleiteten Ideenworkshops unter Verwendung von Design Thinking, die Unterstützung beim Zusammentragen und Visualisieren von Gedanken oder das Zusammenbringen mit Stakeholdern, wie Kunden oder Nutzerinnen.

Planen Sie während des Sprint Planning Zeit für die Auseinandersetzung mit neuen, unbekannten Anforderungen ein und sorgen Sie so dafür, dass das Refinement von Backlog Items laufend stattfindet.

Diskussionen ufern aus

Häufig ist es so, dass eine Anforderung noch so unspezifisch ist, dass die Diskussionen dazu ausschweifen. Oder die Entwickler starten einen Schlagabtausch, wie die Anforderung am besten umzusetzen sei, und halten ihre Variante für die jeweils beste.

Als Scrum Master können Sie hier moderierend eingreifen oder schon im Vorfeld Regeln definieren. Nutzen Sie eine Timebox für jedes Backlog Item und führen Sie die Entwickler mit Pragmatismus zur Verteilung der Aufgaben, die bis zum nächsten Backlog Refinement zu erledigen sind. Beispielsweise könnten sich zwei Entwickler mithilfe des Product Owners tiefer in das Thema einarbeiten und beim nächsten Refinement mit einem Vorschlag aufwarten.

Product Owner sollten klar formulieren, welche Art von Feedback sie sich von den Entwicklern wünschen. Wenn eine grobe Idee im Raum steht, dann ist vielleicht nur eine erste Meinung gefragt und nicht gleich die damit verbundene Aufgabe, eine Lösung zu finden. Mit der Vorstellung eines Backlog Item sollte also auch immer das gewünschte Ziel ausgesprochen werden.

Der Releaseplan wird nicht aktualisiert

Der Releaseplan enthält eine in Sprints gruppierte Liste aller Backlog Items bis zu einem Release. Aus ihm kann man grob ablesen, für welchen Sprint bestimmte Funktionalitäten geplant sind und wann mit einem Release zu rechnen ist. Er stellt also ein wichtiges Instrument für die Transparenz des Projekts im Unternehmen dar. Leider kommt es oft vor, dass er nur einmal zu Beginn eines Projekts erstellt wird, um die Frage des Managements nach der Dauer des Projekts zu beantworten.

Unterstützen Sie den Product Owner dabei, den Releaseplan als Kommunikationsinstrument zu nutzen, um die Erwartungshaltungen zu managen. Er sollte immer aktuell sein und regelmäßig als Grundlage für Diskussionen über die Laufzeit und über Veränderungen hinweg verwendet werden. Auch Scrum-Projekte verlaufen niemals exakt so, wie sie zu

Beginn geplant wurden. Der Releaseplan ist ein gutes Medium, um Veränderungen transparent zu machen.

Alle sind zu beschäftigt

Manche Teams schaffen es, Backlog Refinements als unnütze Zusatzmeetings wegzudiskutieren, die ja nicht einmal zu den im Scrum Guide aufgeführten Scrum-Events gehören. Die Pflege des Product Backlog liegt doch sowieso beim Product Owner, und schließlich gibt es ja das Sprint Planning, in dem sich das Team mit den Backlog Items auseinandersetzt. Und das bisschen Schätzen kriegt man da auch noch unter.

Machen Sie den Unterschied zwischen einem Backlog Refinement und einem Sprint Planning klar, am besten an einem anschaulichen Beispiel wie dem oben beschriebenen Umzug. So begegnen Sie wirkungsvoll Fragen des Teams, warum dieses Event denn unbedingt stattfinden muss.

Zu diesem Schwerpunkt finden Sie die Checkliste »Backlog Refinement« unter *link.scrum-in-der-praxis.de/checklisten*.

4.5 Releaseplanung

Die Konferenz rückt näher

In den Anforderungsworkshops hatte Casper gemeinsam mit den Entwicklern Backlog Items erstellt und geschätzt. Diese ergaben in Summe 350 Story Points. In einem Gespräch mit Herrn Hold erwähnte Casper den aktuellen Stand. Herrn Hold interessierte aber nur, dass die Applikation bis zur Konferenz fertig wird: »Wann kann ich denn die App ausprobieren?« Auf diese Frage war Casper vorbereitet. Gemeinsam hatte das Scrum-Team eine initiale Velocity ermittelt, die zeigt, dass vermutlich 25 bis 35 Story Points pro Sprint umgesetzt werden

können. Bis zur Konferenz waren noch acht Sprints vorgesehen, was bedeutete, dass von den insgesamt 350 Story Points zwischen 200 und 280 bis zum geplanten Konferenztermin umgesetzt werden konnten.

Herr Hold war zunächst entsetzt, hatte er doch damit gerechnet, alle Funktionalitäten bis zur Konferenz geliefert zu bekommen. Casper und Finn erklärten ihm das Zustandekommen dieser Werte; sie versicherten Herrn Hold, dass das Product Backlog immer wieder überprüft und optimiert wird und dass die wichtigsten Backlog Items immer als Nächstes umgesetzt würden, sodass man bis zur Konferenz die nach dem aktuellen Stand bestmögliche Applikation zur Verfügung hätte.

Trotzdem wollte Herr Hold noch wissen, wie lange es denn dauern würde, wirklich alle Backlog Items umzusetzen, denn die nächste Konferenz würde nicht lange auf sich warten lassen. Auch auf diese Frage war Casper vorbereitet. Mit der angenommenen Velocity würde es vermutlich zwischen zehn und 14 Sprints dauern, bis der komplette Umfang geliefert werden könnte, natürlich immer unter der Voraussetzung, dass er sich zwischenzeitlich nicht ändert.

4.5.1 Releaseplan

Ein Releaseplan ist kein offizielles Scrum-Artefakt, es gibt daher auch keine eindeutige Definition. Allgemein gesprochen ist der Releaseplan eine nach Sprints gruppierte Liste von relevanten Backlog Items.

Aus dem Releaseplan geht hervor, in welchem Sprint welche Backlog Items nach aktuellem Wissensstand voraussichtlich umgesetzt werden und zur Verfügung stehen.

Stellen Sie sich den Releaseplan als Product Backlog vor, in dem in regelmäßigen Abständen Trennstriche eingefügt wurden. Jeder Trennstrich stellt eine Abgrenzung zwischen zwei Sprints dar. Zwischen zwei Trennstrichen befinden sich die Backlog Items, die nach heutigem Stand für den jeweiligen Sprint vorgesehen sind.

Für die Erstellung eines Releaseplans werden folgende Informationen benötigt:

- Schätzungen für die zu planenden Backlog Items und
- die Velocity des Teams.

Für den Releaseplan werden die bereits geschätzten Backlog Items durch den Product Owner gemeinsam mit den Entwicklern so auf die folgenden Sprints verteilt, dass die aktuelle Velocity möglichst nicht überschritten wird. Der Product Owner versucht dabei, zusammenhängende User Stories nach

Geschäftswert zu gruppieren, und die Entwickler weisen auf mögliche Abhängigkeiten hin.

Wenn Sie nicht die Velocity des Scrum-Teams anhand von fertiggestellten Story Points berechnen, dann können Sie die Anzahl an fertiggestellten Aufgaben oder Sprint Backlog Items pro Sprint zur Ermittlung des Releaseplans annehmen.

Ein Scrum-Projekt läuft in der Theorie so lange, bis der bestmögliche Wert für die Kundschaft geschaffen wurde. In der Praxis ist es jedoch häufig so, dass entweder der Liefertermin oder aber der Umfang fix ist. Daher hilft der Releaseplan bei der Beantwortung der folgenden Fragen:

- **Fester Umfang**
  »Dies ist das Produkt mit den gewünschten Funktionalitäten. Wie lange wird es dauern?« (vgl. Abb. 4–19)
- **Fester Termin**
  »Wir müssen am … pünktlich zur Messe fertig sein. Was kann bis dahin fertiggestellt werden?« (vgl. Abb. 4–20)

Die beiden Abbildungen zeigen das Szenario aus dem obigen Eingangstext:

Sprint	Story ID	Beschreibung	Größe
1	165	Eingabe persönlicher Daten	8
	154	Profilbild hochladen	8
2	133	Captcha integrieren	5
	164	Rechnung als PDF generieren	8
	172	Prototyp für Landingpage erstellen	3
3	169	Sicheres Bezahlen ermöglichen	8
	44	A-B-Test Landingpage einrichten	3
4	200	Chatfunktion implementieren	5
	201	Chat-Notifications einbauen	5
	203	Support Chat-Bot testen	5
...	...	...	...
		Gesamt	350

Abb. 4–19 *Releaseplan mit festem Umfang*

	Story ID	Beschreibung	GröBe
Release 1	165	Eingabe persönlicher Daten	8
	154	Profilbild hochladen	8
	133	Captcha integrieren	5
	164	Rechnung als PDF generieren	8
	172	Prototyp für Landingpage erstellen	3
	...	...	...
		Release 1	200
Release 2	169	Sicheres Bezahlen ermöglichen	8
	44	A-B-Test Landingpage einrichten	3
	200	Chatfunktion implementieren	5
	201	Chat-Notifications einbauen	5
	203	Support Chat-Bot testen	5
	...	...	...
		Release 2	150
		Gesamt	350

Abb. 4–20 *Releaseplan mit festem Termin*

Wenn die Backlog Items geschätzt sind und die Velocity bekannt ist, kann der Product Owner diese beiden Fragen mit einem Releaseplan sehr schnell beantworten.

User Story Map

Die in Abschnitt 4.2.4 vorgestellte User Story Map eignet sich auch hervorragend, um die geplanten Releases zu visualisieren. Das Schöne bei dieser Übersicht ist, dass der Bezug zum Kunden oder den Nutzerinnen und deren Interaktion mit dem System sichtbar wird (vgl. Abb. 4–21).

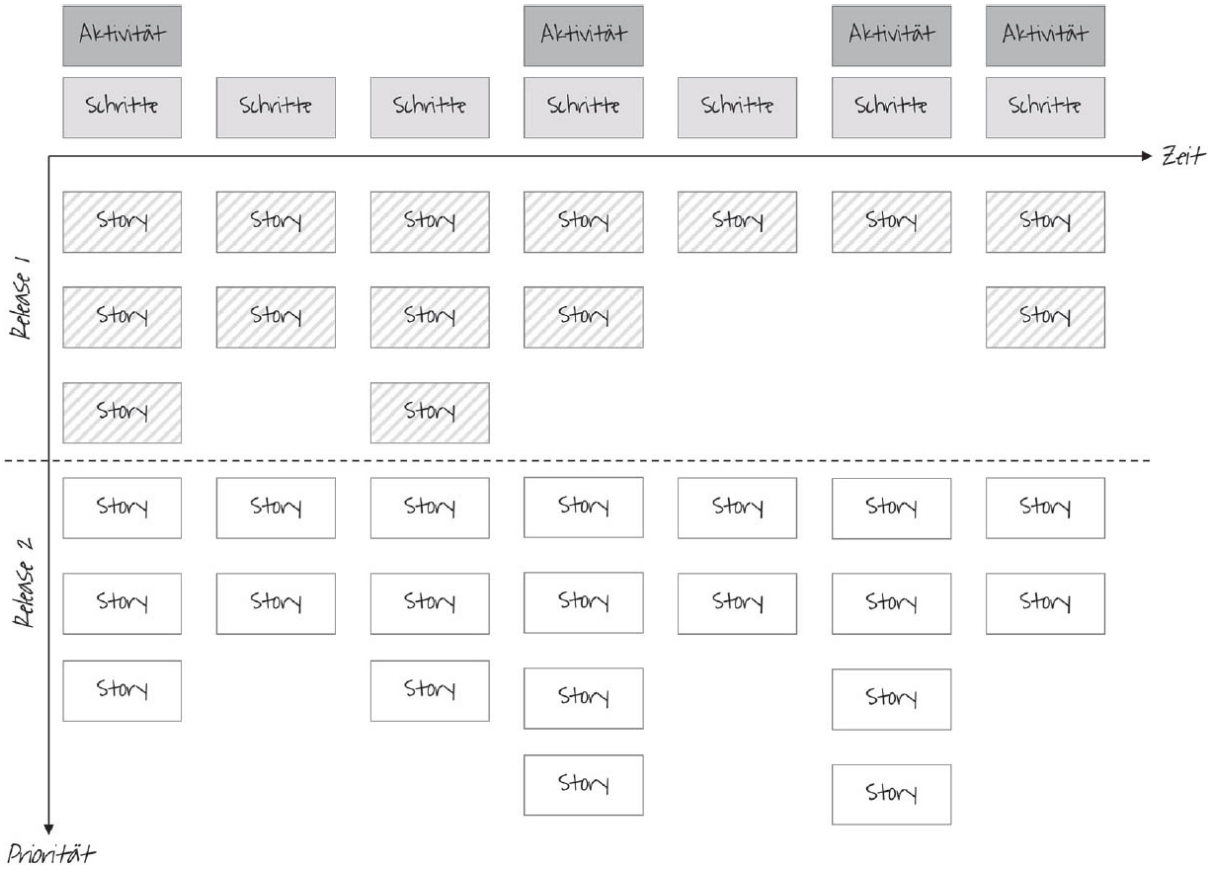

Abb. 4–21 *User Story Map mit Releases*

Neben der Übersicht der Releases kann eine User Story Map noch weitere hilfreiche Informationen enthalten, die zur Visualisierung des Fortschritts genutzt werden können. In Abbildung 4–22 zeigen wir beispielhaft die Erledigung von User Stories und die Vorausschau der geplanten Arbeit (die »2« steht symbolisch für den zweiten Sprint).

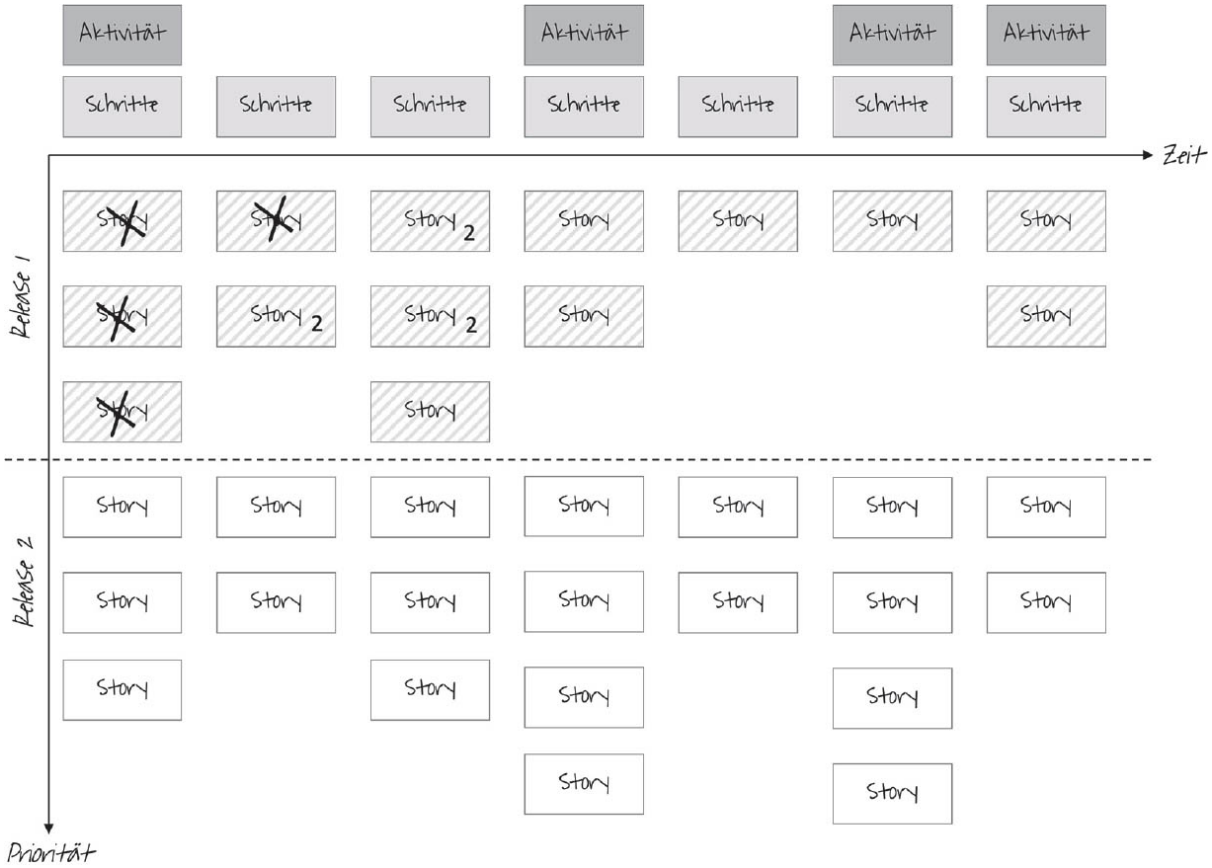

Abb. 4–22 *User Story Map mit Releaseübersicht und Sprint-Fortschritt*

Machen Sie bei jeder Veröffentlichung des Releaseplans deutlich, dass es sich dabei um eine Vorhersage handelt, die nach aktuellem Wissensstand abgegeben wurde. Es handelt sich um ein dynamisches Dokument mit einer Momentaufnahme und ausdrücklich nicht um einen Projektplan im herkömmlichen Sinne. Es ist durchaus möglich bzw. sogar sehr wahrscheinlich, dass sich Änderungen ergeben, weil neue Erkenntnisse gewonnen wurden. Dies spiegelt sich auch in einer schwankenden Velocity des Scrum-Teams wider. Faktoren für Schwankungen können sein:

- Ab- und Zugang von Teammitgliedern
- Unterschiedliche Level an Fähigkeiten
- Unbekannte Werkzeuge oder Aufgaben
- Arbeit an Altlasten
- Unerwartete technische Probleme
- Urlaube, Krankheitsausfälle
- Instabile Rahmenbedingungen der Organisation
- Verteilte Arbeit über Ländergrenzen hinaus

Von daher empfehlen wir, die Velocity als Durchschnitt der letzten vier bis acht Sprints zu berechnen.

Wir empfehlen, den Releaseplan gemeinsam im Scrum-Team im Backlog Refinement zu prüfen und ggf. anzupassen und ihn im Sprint-Review Stakeholdern vorzustellen. Dies erzeugt maximale Transparenz und kurze Feedbackschleifen. Dazu möchten wir auf folgendes Zitat aus dem Scrum Guide hinweisen (*scrumguides.org*), das die Verantwortlichkeit des gesamten Scrum-Teams (vgl. Kap. 3) hervorhebt:

> *»Das Scrum Team ist umsetzungsverantwortlich (responsible) für alle produktbezogenen Aktivitäten …«*

Implizit bedeutet dies, dass der Product Owner nicht allein über Releases entscheidet, sondern das gesamte Scrum-Team.

Wie leicht zu erkennen ist, stellt ein Releaseplan ein sehr mächtiges Werkzeug dar, das auf sehr einfache Weise den Wunsch nach Planung und Vorausschau erfüllt. Leider wird der Releaseplan in der Praxis oft vernachlässigt. Probieren Sie den Releaseplan einmal für ein Quartal aus.

Releaseplanung zu Projektbeginn

Besonders in größeren Firmen wird zunächst eine Kosten-Nutzen-Analyse sowie eine voraussichtliche Projektlaufzeit erwartet, bevor ein Projekt freigegeben wird. Wie aber kann ein Plan erstellt werden, um damit u.a. Kosten zu ermitteln, wenn keine Erfahrungswerte vorhanden sind bzw. keine stabile Velocity vorliegt?

Überlegen wir doch mal, wie ein traditionelles Projekt bis zur Erstellung eines Projektplans (der ja die Zeitschätzung und damit implizit auch die Kostenschätzung enthält) abläuft: In der Anfangsphase hat jemand eine Idee für ein neues Feature oder Produkt. Es werden erste Dokumente (Grobkonzept) geschrieben und viele Diskussionen geführt. All das kostet Zeit und Geld. Es werden erste Schätzungen auf Managementebene eingeholt, die letztendlich mit der Realität sehr wenig zu tun haben. Das Konzept wird verfeinert, ein

Projektleiter wird benannt, der entweder mit sehr vagen oder aber mit sehr detaillierten Anforderungen und Schätzungen einen Projektplan erstellt. Im ersten Fall ist der Projektplan aber wertlos, da zu wenig Informationen vorlagen, im zweiten Fall fließt eine Menge Zeit und Geld in die Erstellung des Projektplans.

Fordern Sie die gleiche Zeit ein, die ein traditionelles Projekt für eine Planungsphase bekommt, wenn Sie mit dieser Frage konfrontiert werden. Verwenden Sie diese Zeit darauf, um erste nutzbare Inkremente zu entwickeln, das Product Backlog voranzutreiben und zu schätzen und letztendlich einen ersten Releaseplan zu erstellen.

Scrum-Projekten wird oft vorgeworfen, aufgrund ihrer Dynamik nicht planbar zu sein. Dieser Argumentation setzen wir entgegen, dass ein Scrum-Projekt, das die gleichen zeitlichen und finanziellen Mittel bekommt wie das oben beispielhaft beschriebene traditionelle Projekt, zum Zeitpunkt der Fertigstellung des ersten Projektplans schon viel mehr liefert als das traditionelle Vorgehen. Wenn dort gerade mal eine Planung erstellt wurde, stehen im Scrum-Projekt bereits erste Inkremente zur Verfügung und die Machbarkeit wurde bewiesen oder widerlegt. Das Scrum-Team hat sich z.B. schon gefunden, eingerichtet und kann nach den ersten drei bis vier Sprints, also nach circa sechs bis acht Wochen, die Entwicklungsgeschwindigkeit ableiten. Erste Unwägbarkeiten und Unsicherheiten wurden überwunden und das Scrum-Team ist nun in der Lage, mithilfe einer groben Einschätzung des Product Backlog und der Velocity eine auf aktuellen Erfahrungen und Fakten basierte Planung abzugeben. Somit kann aus den Kosten für das Scrum-Team mithilfe des Releaseplans eine Einschätzung der Gesamtkosten vorgenommen werden.

Wenn möglich zeichnen Sie anhand einer Beispielrechnung die Unterschiede zwischen den Projekten auf. Vielleicht ist es sogar möglich, Projekte über die gesamte Laufzeit gegenüberzustellen. So wird am Ende ggf. deutlich, dass die »genaue« Planung bei traditionellen Projekten keine besseren Ergebnisse liefert.

Und noch einmal zur Erinnerung: Scrum-Teams arbeiten häufig an komplexen Aufgabenstellungen in einem hochdynamischen Projektumfeld. Das Investment in eine lineare Gantt-Chart-Planung wird niemanden weiterhelfen bzw. ist schon bei der Erstellung falsch.

4.5.2 Release-Burndown-Chart

Ein Release-Burndown-Chart (→Burndown-Chart) ist ähnlich wie ein Sprint-Burndown-Chart (vgl. Abschnitt 5.1.2) aufgebaut. Nachdem ein Releaseplan erstellt und die voraussichtliche Anzahl Sprints bis zum geplanten Termin oder Umfang ermittelt wurde, kann ein einfaches Release-Burndown-Chart wie in Abbildung 4–23 erstellt werden.

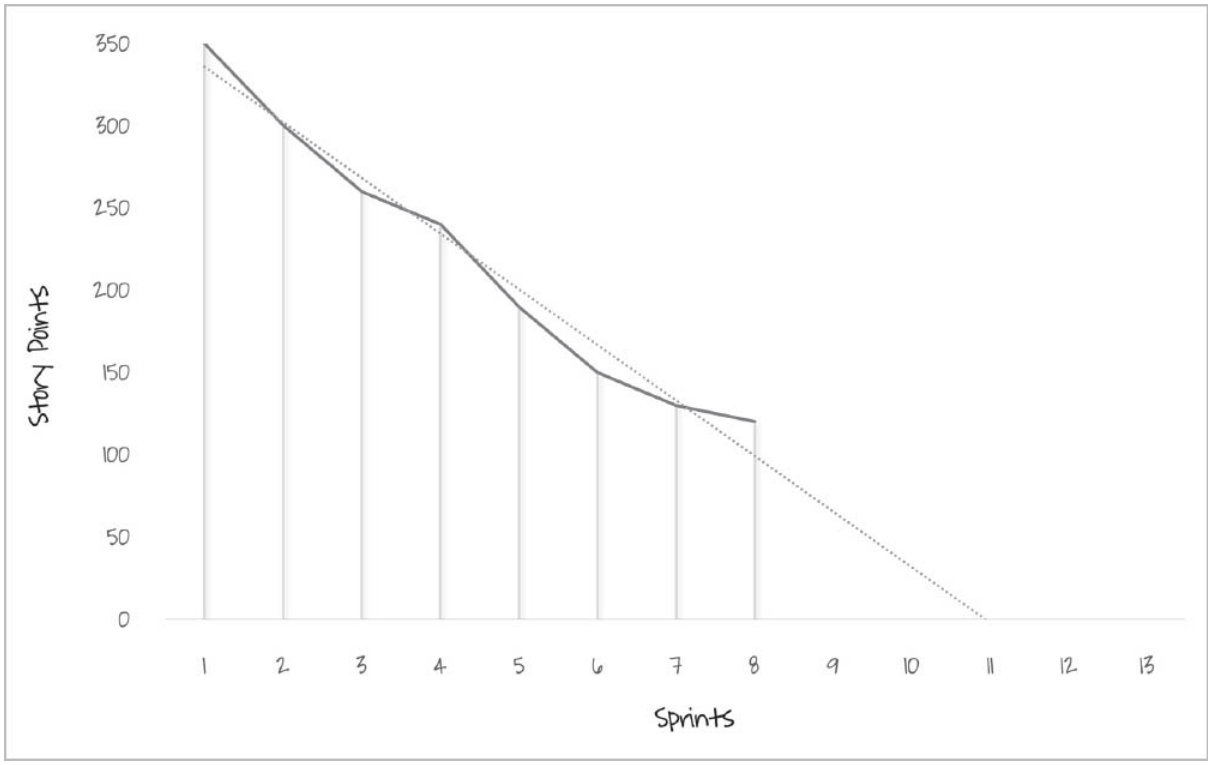

Abb. 4–23 *Einfaches Release-Burndown-Chart*

Die gestrichelte Gerade zeigt den linearen Verlauf, die durchgezogene Linie die nach jedem Sprint noch offenen Story Points. Anhand der Abweichung kann festgestellt werden, ob das Produktziel nach aktueller Planung erreicht werden kann.

Korridor

Wenn, wie im Eingangsbeispiel des Abschnitts erwähnt, ein Scrum-Team zwischen 25 und 35 Story Points pro Sprint liefert, ergibt sich ein Korridor zwischen dem besten und dem schlechtesten Fall. Mit den tatsächlich gelieferten Story Points pro Sprint kann der Verlauf mit den aktuellen Daten ergänzt und somit früh erkannt werden, ob das Release aus dem Ruder zu laufen droht (vgl. Abb. 4–24).

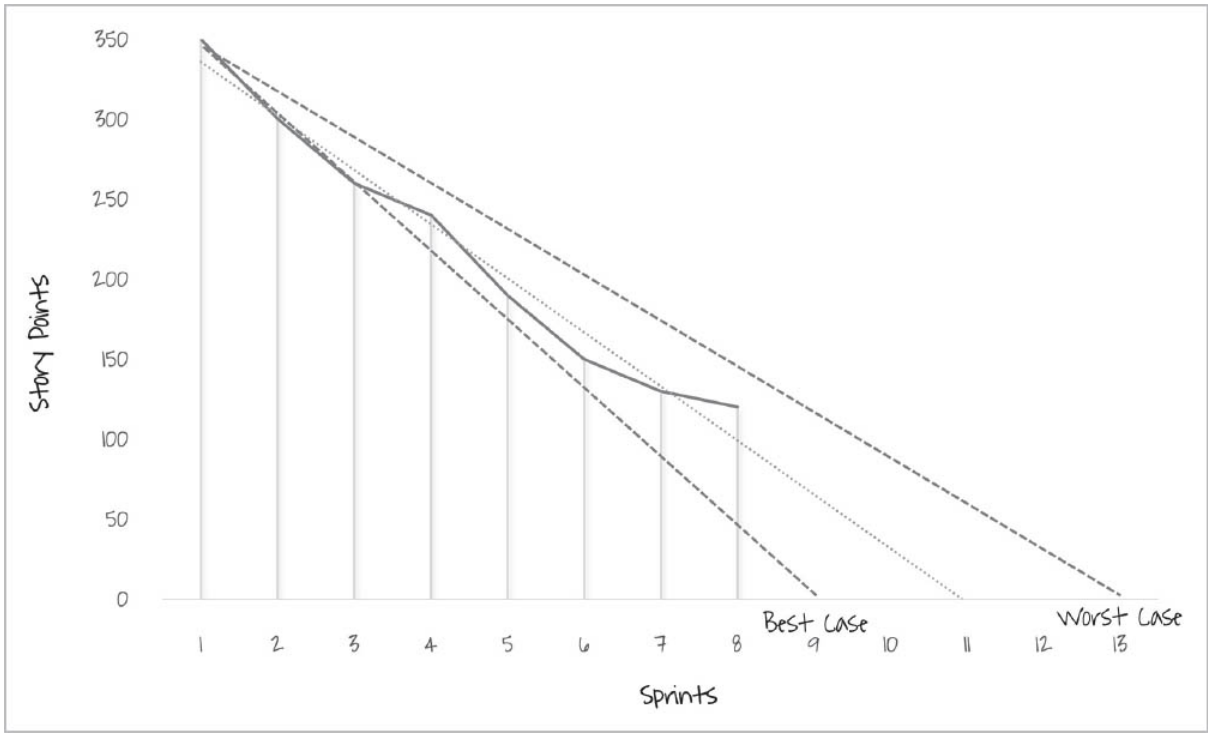

Abb. 4–24 *Release-Burndown-Chart mit Korridor*

Heutige Verwaltungssoftware bietet neben dem Sprint- oder Release-Burndown-Chart viele verschiedenartige grafische Darstellungsformen an. Häufig werden anstatt des Liniendiagramms auch Balkendiagramme verwendet. Hilfreich ist es, wenn diese Diagramme die Veränderung des Umfangs im zeitlichen Verlauf anzeigen können und beispielsweise sichtbar wird, welche Abweichungen es zur initialen Planung gibt und ob diese Aufgaben etwas mit dem Produktziel zu tun haben.

Die oben angesprochenen Werkzeuge sollen dabei helfen, Entwicklungen früh zu erkennen und ggf. gegenzusteuern. Sie unterstützen dabei, Transparenz zu schaffen, sind als Blick nach vorn gedacht und sollten auch so genutzt werden. Vermeiden Sie daher, sie als Rechtfertigung heranzuziehen:

> *»Wir konnten die Funktionalität nicht rechtzeitig liefern, weil ja so viel anderes hinzugekommen ist.« Besser wäre: »Wenn wir die zusätzlichen, ungeplanten Anforderungen betrachten, ist es sehr unwahrscheinlich, dass wir die Funktionalität zum gewünschten Zeitpunkt liefern können. Lasst uns gemeinsam sehen, was wir ändern müssen, damit wir es hinbekommen.«*

4.5.3 Release-Sprint

Sofern ein Scrum-Team ein nicht bereits veröffentlichtes Produkt betreut und regelmäßig nach jedem Sprint neue Funktionalitäten zur Verfügung stellt, kommt es erst nach einigen Sprints zur Veröffentlichung der ersten Produktversion. Im Idealfall sollen alle dafür notwendigen Arbeiten wie Trainings, Marketingaktionen etc. bereits in den Sprints zuvor stattgefunden haben. Allerdings sieht auch hier die Praxis häufig anders aus als die Theorie, sodass wir in diesem Kapitel einen *Release-Sprint* beschreiben, der diese Tätigkeiten bündelt.

Glanzpolitur

Alle wichtigen Funktionalitäten wurden zielgerichtet umgesetzt. Es war noch eine Woche Zeit bis zur Konferenz. Die App wurde als Betaversion zum Download freigegeben und bereits einige Male heruntergeladen. Das Scrum-Team hatte sich für die kommende Woche vorgenommen, das Feedback der Benutzer zu analysieren und auftretende Fehler zu beheben. Außerdem sollte Werbung für die App gemacht werden, damit sie zum Start der Konferenz von möglichst vielen Konferenzteilnehmerinnen genutzt würde.

Als Herr Hold hörte, dass in der letzten Woche vor der Konferenz keine neue Funktionalität mehr eingebaut werden sollte, rief er sofort Casper zu sich. Herr Hold forderte in einem Vieraugengespräch noch weitere Features, die ihm in Aussicht gestellt worden waren. Casper konnte ihn jedoch überzeugen, dass diese Woche für den letzten Feinschliff und die Einweisung aller Personen benötigt wurde. Außerdem würden neue Features nicht mehr hinreichend getestet werden können, und die Gefahr bestünde, dass sich Fehler einschleichen, die dazu führen können, dass die App gar nicht erst benutzt wird oder schlechte Bewertungen erhält. Herr Hold kannte Casper inzwischen gut genug, um zu erkennen, dass die Argumentation durchaus Hand und Fuß hatte. Er verließ sich auf Caspers Urteil.

Im Laufe der Woche legten alle Hand an, um die App noch einmal auf Hochglanz zu polieren und nichts dem Zufall zu überlassen. Zum Start der Konferenz stand eine sauber getestete, mit den wichtigsten Funktionen versehene und mit interessanten Inhalten gefüllte Applikation zur Verfügung.

In vielen Projekten wird der Fehler gemacht, bis zur letzten Minute noch irgendwelche Funktionen zu implementieren. Dabei ist allgemein bekannt, dass Produktentwicklung unter Zeitdruck deutlich fehleranfälliger ist. Außerdem fehlt dadurch meist die Zeit für einen ausreichenden Test. Im schlimmsten Fall wird in

den letzten Tagen etwas kaputt gemacht, was Auswirkungen auf die gesamte Anwendung haben und ggf. sogar den Veröffentlichungstermin gefährden kann.

Ein Release-Sprint ist kein Sprint für Aufräumarbeiten, in dem angehäufte technische Schulden oder Fehler im Produktionssystem bereinigt werden sollen, sondern er dient einzig und allein der Vorbereitung des Release. Achten Sie darauf, dass er nicht missbraucht wird, und machen Sie Sinn und Zweck des Release-Sprints transparent.

Ein Release-Sprint kann vor jedem größeren Release stattfinden. Er dient dazu, alle losen Enden zu identifizieren und zu befestigen. Auch wenn der Name es vermuten lässt, sollte die Dauer eines Release-Sprints nicht zwingend der Dauer eines regulären Sprints entsprechen. Er kann je nach Projekt und Produkt länger oder kürzer als ein regulärer Sprint sein. Es ist in der Regel aber unerheblich, ob vorher zwölf Sprints oder nur drei Sprints stattgefunden haben, der Release-Sprint sollte in der Dauer kaum variieren.

Die Durchführung eines Release-Sprints wird in der Community unterschiedlich gesehen. Gegenstimmen führen an, dass dann am Ende der Sprints kein »Potentially Shippable Code« entstanden sein kann. Sogar Ken Schwaber spricht sich gegen Release-Sprints aus und begründet es damit, dass funktionierende Teams über ein automatisiertes Releasemanagement verfügen. Unserer Meinung nach ist dies ein absolut erstrebenswertes Ziel, das wir in der Praxis jedoch in den seltensten Fällen umgesetzt vorgefunden haben. Wir betrachten Release-Sprints daher als einen Schritt, der unseren Teams auf diesem Weg weiterhilft.

Je konsequenter in den vorangegangenen Sprints funktionsfähige Inkremente geliefert und immer wieder integriert und getestet wurden, desto geringer ist die Notwendigkeit eines Release-Sprints. Sobald ein Produkt erstmalig veröffentlicht wurde, sollte sowieso nach jedem Sprint ein Release möglich sein, sodass spätestens von diesem Zeitpunkt an kein Release-Sprint mehr benötigt werden sollte.

Die Notwendigkeit eines Release-Sprints bedeutet keineswegs, dass während der regulären Sprints unsauber gearbeitet wurde. Im Gegenteil, man sollte sich vor Augen führen, dass das von Scrum geforderte »potenziell auslieferbare Inkrement« nicht immer eine komplette fachliche Produktlösung darstellt, die

Kunden oder Nutzerinnen zur Verfügung gestellt wird bzw. werden kann. Vielmehr kommt es darauf an, in sich abgeschlossene Schritte hin zu einer Produktlösung bereitzustellen, sodass aus fachlicher Sicht jederzeit entschieden werden könnte, dass das Produkt veröffentlicht wird. Zu diesem Zeitpunkt sollten keine weiteren Arbeiten mehr nötig sein.

Wenn es sich um ein technisches Produkt handelt, arbeiten Sie mit sogenannten »Feature Flags«. Dies sind Schalter, mit denen eine Funktionalität ein- oder ausgeschaltet werden kann. So können Sie z. B. die App oder die Website nach jedem Sprint technisch ausliefern, die Funktionalität aber erst dann freischalten, wenn alle Voraussetzungen stimmen.

Folgende Tätigkeiten sind typischerweise Bestandteil eines Release-Sprints, sofern sie für das jeweilige Projekt zutreffen:

- Finales Bereitstellen des Codes in der Produktivumgebung
- Finales Testen in der Produktivumgebung oder einer produktionsnahen Umgebung
- Datenmigration aus Altsystemen
- Aufsetzen und Konfigurieren von administrativen Systemen
- Vorbereitung der Veröffentlichung
- Aktualisierung der Systemdokumentation
- Übergabe des Systems an den operativen Betrieb
- Implementierung von Prozessänderungen in der Organisation
- Training der Anwender und des Supports
- Anstoß von Kommunikations- und Marketingmaßnahmen

Darüber hinaus kann es je nach Projekt weitere Themen geben, die während des Release-Sprints erledigt werden sollten.

In seiner Struktur ist ein Release-Sprint ein Sprint wie jeder andere auch. Führen Sie ein Sprint Planning durch, halten Sie Daily Scrums ab, benutzen Sie ein Scrum-Board. Je nach Art der Aufgaben kann es sein, dass das Team für den Release-Sprint um Personen aus anderen Fachbereichen erweitert werden sollte.

4.5.4 Häufige Herausforderungen

Hardening Sprint

Viele Scrum-Teams verwenden in der Praxis den Release-Sprint als sogenannten »Hardening Sprint«. Dieser ist lediglich dafür da, z.B. aufgeschobene Arbeiten durchzuführen oder technische Schulden zu beseitigen. Schon in dieser Formulierung wird klar, dass es einen Hardening Sprint in diesem Sinne nicht geben kann, denn das Ziel eines jeden Sprints ist es, ein wertvolles prinzipiell auslieferbares Inkrement zu liefern, das die Kriterien der Definition of Done erfüllt.

Wenn ein Hardening Sprint also notwendig wird, wurde konsequent gegen dieses Prinzip verstoßen und nicht auf die erforderliche Qualität geachtet. Dies kann vielerlei Gründe haben. Am häufigsten erleben wir das Nichtleben der agilen Prinzipien und Scrum-Werte (vgl. Kap. 2) und damit eine fehlende Autonomie und Selbstverwaltung des Scrum-Teams (vgl. Kap. 3). Dies wirkt sich dann beispielsweise so aus, dass ein Scrum-Team unter Druck gesetzt wird, einen bestimmten Umfang zu einem (nicht selten willkürlichen) Termin einzuhalten. Die Konsequenz daraus ist nicht nur eine mangelnde Qualität, sondern vor allem enttäuschte Kunden oder Nutzerinnen sowie als Resultat, dass nach dem Release erneut Zeit in Aufräumarbeiten und Fehlerbehebung gesteckt werden muss.

Ein solches Vorgehen missachtet konsequent die Prinzipien des →Lean Thinking. Nicht nur der Hardening Sprint selbst ist reine Verschwendung (engl. Waste), sondern auch die angehäuften oder daraus resultierenden Fehler.

Von daher ist es Ihre Aufgabe als Scrum Master, permanent auf die Lieferung des Scrum-Teams und die Einhaltung der Definition of Done zu achten sowie ein Umfeld zu schaffen, in dem das Scrum-Team in die Lage versetzt wird, die Ergebnis- und Umsetzungsverantwortung zu übernehmen.

Vergleich mit traditionellen Projekten

> *»Wieso könnt ihr nicht sagen, wie lange es dauern wird, andere Projekte können das doch auch?«*

Diese Herausforderung haben wir schon bei der Erstellung des Releaseplans (vgl. Abschnitt 4.5.1) angesprochen. Eine konstruktive Antwort auf die provokante Frage wäre:

»Gebt uns die gleiche Zeit, die ihr einem traditionell geführten Projekt für die Planungsphase geben würdet, dann können wir mit mindestens der gleichen Genauigkeit sagen, wie lange es dauern wird. Außerdem werden wir bis dahin zusätzlich echten Wert generiert haben.«

Wenn der Auftraggeber überzeugt werden kann, den oft immensen Aufwand für die Planungsphase eines traditionellen Projekts lieber in den Projektstart eines Scrum-Projekts zu stecken, wird er erkennen, dass er spätestens nach der gleichen Zeit ebenso belastbare Zahlen für eine Projektentscheidung erhält. Der Vorteil bei Scrum-Projekten besteht darüber hinaus darin, dass

- das Projektteam schon als Team zusammen an dem Produkt gearbeitet hat und sich finden konnte,
- in der Zeit etwas Fertiges, bereits Nutzbares herausgekommen ist und
- man im Zweifelsfall das erarbeitete Ergebnis verwerfen könnte, um mit der gewonnenen Erfahrung von Neuem zu starten.

Fester Umfang und fester Termin

Immer wieder treffen wir auf Projekte, die Umfang und Termin vorgeben, obwohl es doch inzwischen allgemein bekannt ist, dass dies in den seltensten Fällen funktioniert. Wer hier argumentativ nicht weiterkommt, kann versuchen, den Umfang nicht zu detailliert festzulegen, sodass innerhalb eines gewünschten Features noch Potenzial zur Kürzung besteht. Indem zunächst die wichtigsten Eigenschaften eines Features implementiert und veröffentlicht werden, kann durch das frühzeitige Feedback darauf reagiert werden.

In einem solchen Fall ist es hilfreich, mit User Stories zu arbeiten. Die Kommunikation mit Stakeholdern erfolgt in der Regel über den Titel der Story und nicht über die Akzeptanzkriterien. Dies bietet Product Ownern einen gewissen Entscheidungsspielraum bei der konkreten Ausgestaltung der Inhalte.

Um größtmögliche Transparenz zu erzeugen, hat sich →Story Mapping im Zusammenspiel mit der →MuSCoW-Priorisierung bewährt (vgl. Abschnitt 4.2.4).

Releaseplan wird nur nach Story Points erstellt

Der Releaseplan ist nicht nur ein Instrument zur Berechnung eines potenziellen Projektendtermins, sondern soll auch zeigen, wann eine bestimmte Funktionalität voraussichtlich zur Verfügung steht. Dies gilt insbesondere dann, wenn tatsächlich nach jedem Sprint ein Inkrement ausgeliefert werden kann.

Aus diesem Grund ist es wichtig, den Releaseplan nicht unter dem Gesichtspunkt einer optimierten Story-Point-Auslastung zu erstellen, sondern es sollten Abhängigkeiten und sinnvolle Gruppierungen von Backlog Items berücksichtigt werden, und natürlich sollte er der Priorisierung des Product Owners folgen.

Nutzen Sie das Backlog Refinement (siehe Abschnitt 4.4), um den Releaseplan gemeinsam zu aktualisieren.

Release-Sprint nicht geplant

Die wenigsten Teams sind aber so perfekt, dass sie ein Produkt tatsächlich ohne einen Release-Sprint veröffentlichen können, sodass im letzten Sprint plötzlich Hektik ausbricht. Kleinere Änderungen müssen noch eingebaut, plötzlich auftretende Fehler noch behoben werden, und es tauchen Fragen auf, wie »Hat mal jemand an die Performance gedacht?« oder »Wissen die im Betrieb überhaupt, dass sie für das Release eine Nachtschicht einlegen müssen?«.

Zu Beginn eines Projekts macht sich oft niemand Gedanken über die Veröffentlichung des Produkts, sondern berechnet die reine Entwicklungszeit. Da jedoch während des Projekts immer wieder unvorhergesehene Dinge passieren, entsteht unter Umständen Konfliktpotenzial mit dem Auftraggeber.

Planen Sie von Anfang an mit einem Release-Sprint. Planen Sie ihn selbst dann ein, wenn Sie zu Beginn noch nicht exakt sagen können, was genau in diesem Sprint zu tun ist. Sie werden im Laufe des Projekts so viel dazulernen, dass Sie den Sprint später problemlos mit notwendigen Aufgaben füllen können.

Zu diesem Schwerpunkt finden Sie die Checkliste »Releaseplanung durchführen« unter *link.scrum-in-der-praxis.de/checklisten*.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Sonderzeichen

Ziffern