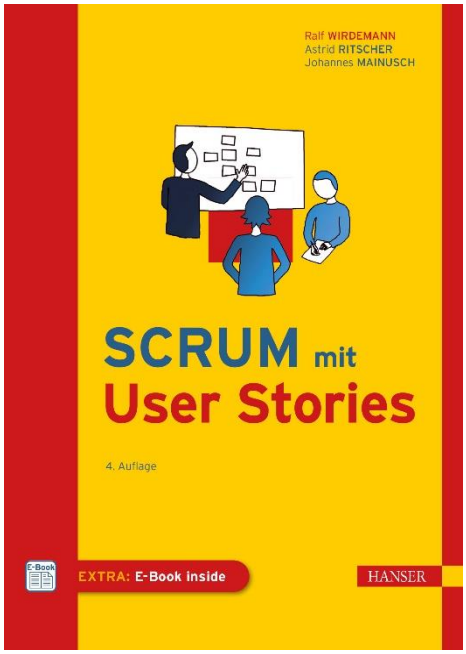
HANSER




# Leseprobe

zu

# Scrum mit User Stories

von Ralf Wirdemann, Astrid Ritscher und Johannes Mainusch

Print-ISBN: 978-3-446-47369-0
E-Book-ISBN: 978-3-446-47438-3
E-Pub-ISBN: 978-3-446-47526-7

Weitere Informationen und Bestellungen unter
https://www.hanser-kundencenter.de/fachbuch/artikel/9783446473690
sowie im Buchhandel

# Inhalt

# Vorwort zur 4. Auflage

Wenn diese 4. Ausgabe von *Scrum mit User Stories* erscheint, hat sich nach mehr als zwei Jahren Pandemie ein Teil der Arbeitswelt stark verändert, insbesondere Arbeit, die sich dank Informationstechnologie vom Büro nach Hause verlagern lässt. Und es ist abzusehen, dass mobiles Arbeiten bleibt und hybride Arbeitsweisen, die es Teammitgliedern ermöglichen, von zu Hause oder aus dem Büro miteinander zu arbeiten, weiterentwickelt werden.

Softwareteams standen Anfang 2020 von heute auf morgen vor der Herausforderung, ihren Arbeitsalltag vom Büro ins Homeoffice zu verlegen. Plötzlich wurden Teams getrennt, für die es zuvor selbstverständlich war, im selben Raum und mit extrem kurzen und effizienten Kommunikationswegen zusammenzuarbeiten. Diese Zäsur der Arbeitswelt ist die Hauptmotivation für die Neuauflage dieses Buches: Nach mehr als zwei Jahren im Homeoffice hatten wir das Bedürfnis aufzuschreiben, was für uns funktioniert hat und worauf es beim mobilen Arbeiten ankommt.

Ich (Ralf) persönlich habe die Umstellung auf das Homeoffice in der Rolle des Entwicklers und Architekten miterlebt. Entsprechend war ich nicht hauptverantwortlich für die Neustrukturierung unserer neuen Arbeitssituation, sondern habe die Umstellung in mehreren Teams aus der zweiten Reihe mehr beobachtend und beratend begleitet. Aus diesem Grund war es mir wichtig, die bei der Mobilisierung agiler Teams entstehenden Herausforderungen aus erster Hand zu bekommen. Es wurde Zeit für eine neue Co-Autorin.

Für die 4. Auflage meines Buches konnte ich dafür Astrid Ritscher gewinnen. Astrid ist Agile-Coach bei der Otto GmbH & Co KG in Hamburg und stand vor gut zwei Jahren vor der Herausforderung, ihre agil arbeitenden Teams auf das mobile Arbeiten vorzubereiten. Astrid hat ihre Erfahrungen mit dem neuen Kapitel „Mobiles Arbeiten“ beigesteuert, in dem sie beschreibt, was sie und ihre Teams in den letzten zwei Jahren gelernt haben.

Das neue Kapitel widmet sich den Auswirkungen mobilen Arbeitens und seinen Anforderungen und Möglichkeiten an die Arbeit eines Teams und seiner Stakeholder mit Scrum. Für mobiles Arbeiten allgemein und die Besonderheiten von Scrum-Meetings und anderen agilen Arbeitsweisen (z. B. Pairing) werden Alternativen, Ideen und Empfehlungen vorgestellt. Der letzte Abschnitt des Kapitels geht auf hybrides Arbeiten ein.

Wir hoffen, Ihnen mit diesem neuen Thema eine weitere wichtige Facette der Agilen Softwareentwicklung näherzubringen, die Ihnen bei dem Neueinstieg oder der Optimierung Ihrer agilen Arbeitswelt behilflich ist.

### Hinweis zur Verwendung geschlechterneutraler Sprache

Wenn bei personellen Bezeichnungen die männliche Form gewählt wurde (z. B. Mitarbeiter, Techniker), so sind damit in gleicher Weise weibliche Mitarbeiter oder Transgender-Mitarbeiter gemeint.

*Hamburg, im Mai 2022* *Ralf Wirdemann, Astrid Ritscher und Dr. Johannes Mainusch*

# 4 User Stories

User Stories sind ein etabliertes und weit verbreitetes Konzept für die Beschreibung und das Management von Anforderungen in agilen Softwareprojekten. Im Vergleich zu traditionellen Mitteln des Anforderungsmanagements sind User Stories sehr viel unschärfer und offener formuliert. Sie lassen viel Raum für Änderungen und helfen dabei, ein System zu entwickeln, das der Kunde wirklich will, und nicht eines, das vor einem Jahr spezifiziert wurde.

User Stories beschreiben Anforderungen aus Sicht des Benutzers. Der sichtbare Teil einer User Story ist ihre Story-Karte, die den Kern der Anforderung in einem Satz beschreibt. Die Story-Karte dient im Wesentlichen als Platzhalter für die spätere Konversation zur User Story. Denn genau hierauf kommt es an: Konversation ist der wichtigste Bestandteil einer User Story. Das bedeutet konkret, dass die Details und konkrete Ausprägung einer User Story erst während ihrer Entwicklung im Dialog zwischen Product Owner und Team geklärt werden.

Der Kommunikationsaspekt einer User Story ist etwas, das viele Entwicklungsteams noch nicht ausreichend verstanden haben. Vielfach werden User Stories als eine Art moderne Use Cases der agilen Softwareentwicklung verstanden. Use Cases liefern zwar, ähnlich wie User Stories, einen konkreten Geschäftswert, sind aber von vornherein wesentlich vollständiger und genauer spezifiziert als User Stories.

Die Unvollständigkeit von Stories ist Teil ihres Konzepts. User Stories akzeptieren den Umstand, dass Benutzer und Kunden erst relativ spät wissen, was sie wollen, nämlich erst dann, wenn sie Teile des Systems kennengelernt und ausprobiert haben. Sowohl dem Team als auch dem Kunden muss klar sein, dass eine User Story zunächst etwas sehr Vages ist, was aber umso konkreter wird, je näher die Story an ihre eigentliche Umsetzung rückt und erst während der Entwicklung sehr konkret wird. Das entscheidende Mittel hierfür ist die Kommunikation, das heißt das Gespräch über die User Story.

Dieses Kapitel führt in das Konzept von User Stories ein, beschreibt deren Aspekte und Eigenschaften und erklärt, was gute Stories sind und wie man sie schreibt. Des Weiteren wird begründet, weshalb sich User Stories so gut für die agile Softwareentwicklung und insbesondere für Scrum eignen.

## 4.1 Was sind User Stories?

Eine User Story beschreibt eine Anforderung an ein Softwaresystem. Die Anforderung besitzt einen konkreten und sichtbaren Mehrwert für den Kunden. Die User Story „Als Coach will ich mein Profil einstellen“ besitzt einen solchen Mehrwert. Wenn die Story implementiert wird, verfügt das System über eine neue Funktionalität, die den Geschäftswert der Anwendung erhöht. Ein Gegenbeispiel ist die Anforderung „Die Software soll in Java programmiert werden“. Dies ist keine User Story, da es dem Kunden keinen Mehrwert im Sinne seines Geschäftsmodells liefert, wenn die Anwendung in Java programmiert wird.

Ein weiteres wichtiges Merkmal einer User Story ist ihre Bedeutung für den Kunden. Die Story muss für den Kunden eine Bedeutung im Kontext seines Geschäftsumfelds besitzen. Beispielsweise hat die Anforderung „Die Anwendung muss auf zwei Applikationsservern laufen“ keinerlei Bedeutung für den Kunden im Sinne seines Kerngeschäfts. Natürlich ist es wichtig, dass die Anwendung ausfallsicher läuft. Aber eben nicht als Geschäftswert an sich und deshalb ist die Anforderung keine User Story.

Den Begriff User Story kennen viele Entwickler. Einer weit verbreiteten Meinung nach ist eine User Story eine handgeschriebene Karte mit Informationen zur umzusetzenden Anforderung. Das stimmt zwar, ist aber nur die halbe Wahrheit.

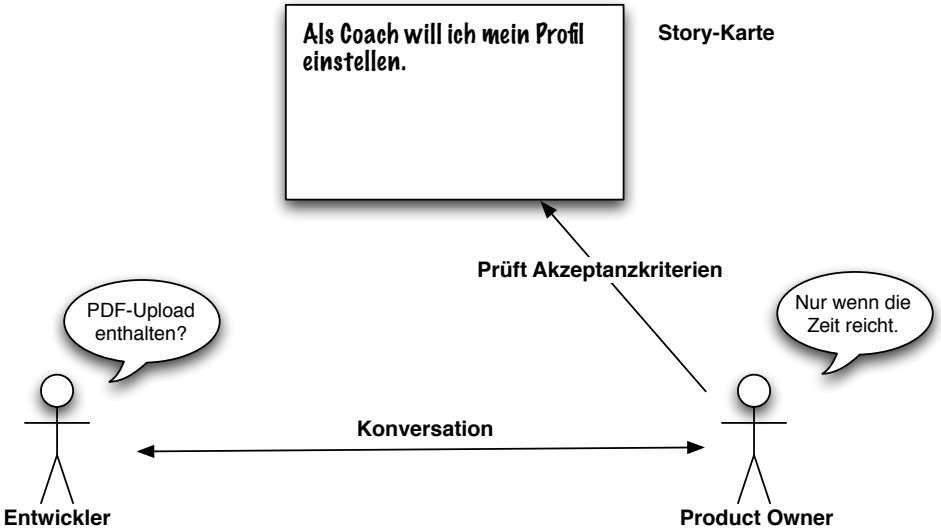



**Abbildung 4.1** Karte, Konversation, Akzeptanzkriterien

Abbildung 4.1 zeigt, dass eine User Story aus drei Teilen besteht: Karte, Konversation und Akzeptanzkriterien. Der wichtigste Teil der Story ist die ihr zugrunde liegende Konversation. Und genau daran soll die Karte erinnern. Sie ist ein Versprechen des Teams an den Product Owner, sich im Detail über die Story zu unterhalten, sobald sie konkret wird. Sobald es an die konkrete Entwicklung der Story geht, beginnen die Entwickler und der Product Owner einen Dialog, der bis zur Fertigstellung der Story andauert und in vielen kleinen Feedback-Schleifen die Details der Story herausarbeitet.

Der dritte Teil einer User Story sind neben Karte und Konversation ihre Akzeptanzkriterien, die bestimmen, wann eine Story vollständig umgesetzt und im Sinne der geltenden „Definition of Done“ fertig ist (siehe dazu Abschnitt 10.3.3).

User Stories basieren auf der von Ron Jeffries (einem der Begründer des Extreme Programmings) 2001 eingeführten Alliteration *CCC* [Jeffries 2001]. Das erste C steht für die Karte (Card), das zweite C für die Konversation (Conversation) und das dritte C für die Akzeptanzkriterien (Confirmation) der Story.

### 4.1.1 Story-Karte

Die Story-Karte ist der sichtbare Teil einer User Story und beschreibt in einem Satz den Kern der umzusetzenden Anforderung. Eine Story-Karte repräsentiert eine Anforderung, beschreibt sie aber nicht im Detail. Die Story-Karte bringt den Kern der Anforderung mit einer zielorientierten Aussage auf den Punkt:

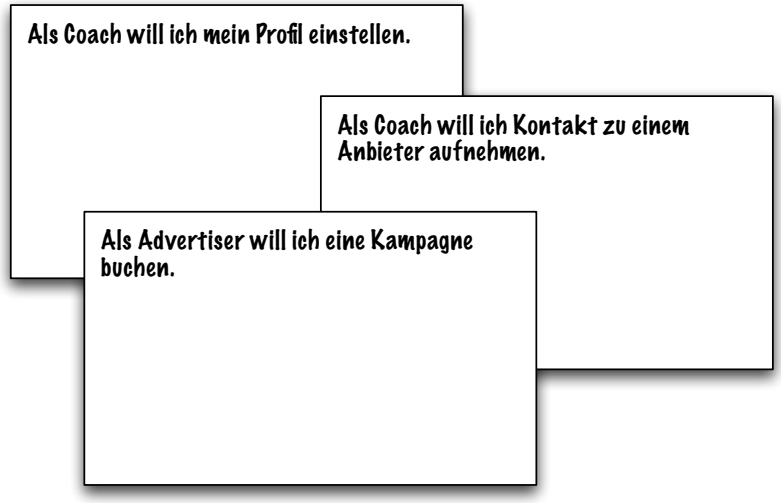



**Abbildung 4.2** Story-Karten

Die Story-Karte drückt das Ziel des Benutzers möglichst klar aus. In einem Marktplatz für Internetwerbung will ein Advertiser Kampagnen buchen. Dies ist sein Ziel. Umfasst die Software eine Funktion, mit der er dieses Ziel erreichen kann, dann besitzt die Software einen Wert für ihn.

Story-Karten werden üblicherweise auf A5-Karteikarten geschrieben. Karteikarten kann man für das gesamte Team gut sichtbar an ein Whiteboard hängen, lassen sich einfach umsortieren, neu schreiben oder auch durchreißen, wenn die Anforderung keine Gültigkeit mehr hat oder fertig entwickelt wurde. Darüber hinaus haben Karteikarten nur begrenzt Platz zum Schreiben. Selbst wenn man wollte, kann man nicht beliebig viele Details auf einer Karte notieren. Indem man bewusst kleine Karteikarten wählt, verzögert man das Festlegen von Details.

Karteikarten untermauern auch die veränderliche Natur von User Stories. Wenn ein Detail einer Story nicht mehr gilt, wird es durchgestrichen. Ergibt sich ein neues Detail, wird es einfach auf die Karte geschrieben. Das geht viel schneller, als die entsprechende Stelle in einem elektronischen Anforderungsdokument zu suchen, zu ändern und neu zu verteilen.

### 4.1.2 Konversation

Der Schlüssel zu einer guten User Story ist Konversation. Eine Story wird konkret, sobald sie im Rahmen eines anstehenden Sprint umgesetzt werden soll. Dies ist der Zeitpunkt, zu dem das Team sein Versprechen einlöst und die Details der Story mit dem Product Owner bespricht.

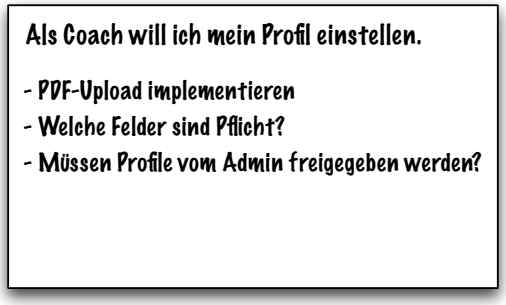



**Abbildung 4.3** Details einer User Story

Details oder Fragen zu einer User Story werden auf der Vorderseite der Story-Karte notiert. Handgeschriebene Story-Karten haben den Vorteil, dass Entwickler oder Product Owner mal schnell zum Whiteboard gehen und ein Detail oder eine Frage auf die Karte schreiben können.

Basierend auf intensiver Kommunikation zwischen Team und Product Owner werden mehr und mehr Details einer Story herausgearbeitet und die Story iterativ vervollständigt. Die Kommunikation kann dabei so intensiv werden, dass so gut wie gar keine Notizen mehr gemacht werden. Der Entwickler setzt die besprochenen Details im Sprint unmittelbar um und präsentiert dem Product Owner das Ergebnis direkt am laufenden System.

### 4.1.3 Akzeptanzkriterien

Ein wichtiges Scrum-Prinzip ist, dass nur fertige, das heißt abgeschlossene Dinge einen Wert haben. Die Akzeptanzkriterien einer Story geben vor, wann die Story fertig ist und einen Mehrwert für den Kunden liefert. Akzeptanzkriterien werden vom Product Owner gemeinsam mit dem Team erarbeitet und auf die Rückseite der Story-Karte geschrieben (Abbildung 4.4).

Akzeptanzkriterien sind genauso wenig fest wie die Details einer User Story. Zwar überlegt sich der Product Owner, welchen Kriterien die Story genügen soll, muss aber keinesfalls daran festhalten. Die Details der Story ergeben sich aus dem Dialog während ihrer Umsetzung.

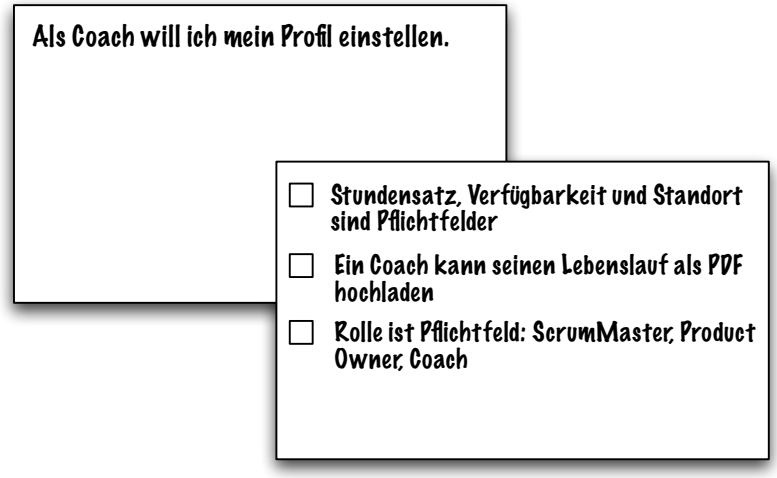



**Abbildung 4.4** Akzeptanzkriterien auf der Rückseite der Story-Karte

Nimmt die Story dabei eine andere Richtung als ursprünglich geplant, ändern sich auch ihre Akzeptanzkriterien.

Für die Entwickler sind die Akzeptanzkriterien wichtige Anhaltspunkte, was denn eigentlich zu tun ist. Sie konkretisieren das Ziel der Story und geben vor, wann das Team mit seiner Arbeit fertig ist. Für Tester und Product Owner sind Akzeptanzkriterien wichtige Hinweise für die Durchführung von Akzeptanztests, nachdem die Story fertig entwickelt und integriert wurde.

Ausführliche Informationen über das Schreiben von Akzeptanzkriterien und die Durchführung von Akzeptanztests finden Sie in Kapitel 11, *User Stories Akzeptanztesten*.

## 4.2 Warum User Stories?

Jeder Softwareentwickler weiß, wie schwierig es ist, Anforderungen so genau zu spezifizieren, dass sich eine schriftliche Spezifikation als Basis für die zu entwickelnde Software eignet. Kunden wissen in der Regel erst dann, was sie wollen, wenn sie eine erste Version der Software gesehen und ausprobiert haben. Selbst wenn eine sehr genaue Beschreibung der Anforderungen gelingt, bekommt der Kunde im besten Fall das, was zu Anfang des Projekts aufgeschrieben wurde. Und das ist mit großer Wahrscheinlichkeit nicht das, was er sich vorgestellt hat. Schlimmer noch: Die Erwartungen des Kunden haben sich bis zum Projektende auch noch verschoben.

Überspitzt formuliert bedeutet obige Beobachtung: Möchte man etwas entwickeln, was der Kunde wirklich will, dann sollte man es nicht aufschreiben. User Stories verlagern den Schwerpunkt des Anforderungsmanagements vom Schreiben aufs Sprechen. Sie forcieren die verbale statt der geschriebenen Kommunikation und fördern den Dialog zwischen

Entwickler und Kunden. Stories sind frei von technischem Jargon und werden vom Kunden und Entwickler gleichermaßen verstanden. Eine enge Zusammenarbeit wird möglich und der Kunde wird Teil des Teams.

In Scrum wird der Kunde vom Product Owner repräsentiert. Idealerweise steht der Product Owner dem Team in Vollzeit zur Verfügung und kann so Anforderungen „just in time“ diskutieren und unmittelbar Feedback liefern. Es ist sichergestellt, dass die Entwickler – wenn überhaupt – nur für sehr kurze Zeit in die falsche Richtung laufen. Der Product Owner beeinflusst die Details einer Story zum Zeitpunkt ihrer Entstehung und bekommt das, was er haben will. Kurze Feedback-Schleifen helfen dem Entwickler, sich in den Kunden hineinzuversetzen und zu verstehen, was er will.

User Stories eignen sich sehr gut für die iterative Entwicklung und damit für die Verwendung in Scrum. Es müssen nicht alle Stories geschrieben werden, bevor mit der Entwicklung begonnen wird. Ideen oder Stories, die noch nicht für den nächsten Sprint bestimmt sind, können als Epic[1] notiert und zur Seite gelegt werden. Nur die konkret anstehenden Stories müssen detaillierter behandelt werden. Details werden verzögert, bis man ein besseres Verständnis vom Problem hat. Es wird keine Zeit damit verschwendet, Dinge zu detaillieren, die man später vielleicht gar nicht benötigt. Es ist nicht ungewöhnlich, dass ein Teil der ursprünglichen Funktionalität zu einem späteren Zeitpunkt verworfen wird, weil der Kunde ständig dazulernt und erst im Laufe des Projekts erkennt, was er wirklich will.

User Stories haben eine gute Planungsgröße, können im Rahmen einzelner Sprints vollständig umgesetzt werden und eignen sich gut für die Sprint- und Releaseplanung innerhalb von Scrum. Die Bedeutung und damit die Wichtigkeit einer User Story kann vom Product Owner verstanden und bewertet werden. Der Product Owner kann beurteilen, welche Story den größten Geschäftswert liefert, und dafür sorgen, dass die wichtigsten Stories in den nächsten Sprints entwickelt werden.

Das Entwicklerteam bekommt durch die Verwendung von User Stories das angenehme Gefühl, etwas Wertvolles zu schaffen. Stories definieren klare und greifbare Ziele. Entwickler sehen das Ziel deutlich vor Augen und wissen am Ende des Tages, was sie geschafft haben.

## 4.3 User Stories schreiben

User Stories werden vom Product Owner geschrieben. Die Ideen dafür kommen allerdings von sehr vielen Seiten: künftige Anwender, Auftraggeber, Entwicklungsteam, Marketingabteilung, ScrumMaster usw. Alle Interessenvertreter haben ein Mitspracherecht, wenn es um die Anforderungen des Produkts geht. Der Product Owner muss daher nicht zwangsläufig ein Experte der Anwendungsdomäne sein. Viel wichtiger ist, dass er ein Experte im Anforderungsmanagement mit User Stories ist. Seine Arbeit besteht darin, die Anforderungen der unterschiedlichen Interessenvertreter zu bündeln und aus ihnen User Stories zu formulieren. Er ist der einzige Projektteilnehmer mit Schreibrechten aufs Product Backlog. Alle anderen dürfen nur lesen und ihren Input über den Product Owner einbringen.

[1] Epics sind große User Stories, die in erster Linie als Platzhalter für weiter zu konkretisierende Ideen oder Themenbereiche dienen. Sie werden in Abschnitt 4.3.4 ausführlich beschrieben.

Die Qualität der User Stories hat maßgeblichen Einfluss auf den Erfolg des Projekts. Gute User Stories schüttelt man nicht mal so eben aus dem Ärmel. Fragen wie „Ist das überhaupt eine Story?“, „Ist die Story nicht viel zu groß?“ oder „Wie viel Detail ist genug?“ sind gerade für neue Product Owner schwer zu beantworten. Dieser Abschnitt fasst einige Hinweise zusammen, die Sie beim Schreiben von User Stories beachten sollten.

### 4.3.1 Die Sprache des Kunden

User Stories werden in der Sprache des Kunden geschrieben, das heißt in der Sprache der jeweiligen Anwendungsdomäne. Die Stories einer Bankanwendung verwenden üblicherweise Begriffe wie *Konto*, *Überweisung* oder *Unterkonto*:

- Als Kunde will ich Geld auf andere Konten überweisen.
- Als Kunde will ich Unterkonten einrichten.

Ein anderes Beispiel sind Marktplätze für Werbung im Internet. In diesem Umfeld findet man Begriffe wie *Advertiser* – jemand, der eine Kampagne bucht – oder *Publisher* – jemand, der Werbeflächen anbietet:

- Als Advertiser will ich eine Kampagne buchen.
- Als Publisher will ich Werbeflächen verkaufen.

Der Product Owner muss sich intensiv in die Anwendungsdomäne des Kunden einarbeiten und lernen, in der Sprache dieser Domäne zu denken und zu schreiben. Das Gleiche gilt für das Team, das häufig mit unterschiedlichen Anwendungsdomänen konfrontiert ist. Die Entwickler sind Experten in der Softwareentwicklung, müssen sich aber von Projekt zu Projekt auf neue Anwendungsumfelder einstellen und das zugrunde liegende Geschäft und dessen Vokabular lernen. User Stories helfen das Wissen über die Anwendungsdomäne im Team zu verbreiten. Sie fordern und fördern die Kommunikation zwischen Team und Product Owner und diese ist nur dann effektiv, wenn alle Beteiligten dieselbe Sprache sprechen und verstanden haben, worum es geht.

### 4.3.2 Benutzerrollen

User Stories repräsentieren Ziele, die Benutzer mit dem System erreichen wollen. Unterschiedliche Benutzer haben unterschiedliche Ziele. Thomas zum Beispiel ist ein Scrum-Master und sucht ein neues Projekt. Die Firma B-Simple ist auf der Suche nach einem qualifizierten Scrum-Coach, der sie bei ihrem ersten Scrum-Projekt unterstützt. Oder ein ehemaliger Kunde von Thomas ist so begeistert von dessen Arbeit, dass er ihm gerne eine Empfehlung ausstellen und ihn so bei der Suche nach neuen Projekten unterstützen möchte.

Thomas, B-Simple und der ehemalige Kunde sind Stellvertreter für Gruppen mit unterschiedlichen Zielen. Um die Ziele der jeweiligen Benutzer zu analysieren, werden Benutzer zu Rollen zusammengefasst und das System aus Sicht dieser verschiedenen Rollen betrachtet. Die offensichtlichen Rollen von Scrumcoaches.com:

- Scrum-Coaches
- Projektanbieter
- Ehemalige Kunden

Die Ziele dieser Rollen sind ähnlich offensichtlich: Scrum-Coaches wollen Aufträge finden, Projektanbieter wollen Coaches finden und ehemalige Kunden wollen Coaches bewerten.

User Stories werden immer aus Sicht einer bestimmten Benutzerrolle geschrieben. Benutzerrollen sind die treibende Kraft beim Schreiben von Stories und machen die Stories sehr viel ausdrucksstärker und konkreter. Die Aussage „Ein Scrum-Coach kann sich anmelden“ ist deutlich konkreter als die Aussage „Ein Anwender kann sich anmelden“. Die erste Version bringt klar auf den Punkt, für wen die Story gedacht ist, und beantwortet dadurch implizit eine Reihe von Fragen. Welche Seite sieht der Anwender zum Beispiel nach seiner erfolgreichen Anmeldung? In der ersten Version ist die Antwort einfach: Der Scrum-Coach sieht die Startseite für Coaches, mit den für Coaches zur Verfügung stehenden Funktionen. Die zweite Formulierung macht die Antwort sehr viel komplizierter: Wenn es sich um einen Projektanbieter handelt, dann zeigen wir ihm die Anbieter-Startseite; wenn sich ein Administrator anmeldet, dann zeigen wir ihm ein Menü mit Verwaltungsfunktionen usw. Da ist es einfacher, von vornherein eine Anmelde-Story für jede Benutzerrolle zu schreiben. Ist nicht klar, aus welcher Rollensicht eine User Story geschrieben werden soll, dann ist das entweder ein Hinweis auf eine zu große Story oder auf eine noch nicht vorhandene Rolle.

Neben zusätzlicher Ausdrucksstärke bieten Benutzerrollen den Vorteil, dass sich aus ihnen sehr viel leichter Ziele ableiten und die passenden User Stories schreiben lassen, als wenn man das gesamte System unabhängig davon betrachtet, wer es gerade benutzt. Benutzermodellierung sollte deshalb Teil des initialen Anforderungsworkshops sein. Bevor das Scrum-Team und alle anderen Interessenvertreter mit der Analyse der Anforderungen beginnt, findet ein Rollen-Brainstorming statt, in dem jeder Teilnehmer die ihm in den Sinn kommenden Rollen auf je eine Karteikarte schreibt. Darüber hinaus ist es hilfreich, die Ziele der jeweiligen Rolle gleich mit auf der Karte zu notieren. Das erleichtert das spätere Aussortieren und das Schreiben der ersten Stories. Sind alle Teilnehmer fertig, legen sie reihum die von ihnen geschriebenen Karten auf den Tisch. In der dabei stattfindenden Diskussion werden Duplikate entfernt, Rollen geteilt oder zusammengefasst sowie zusätzlich benötigte Rollen hinzugefügt. Abbildung 4.5 zeigt das Ergebnis des Scrumcoaches-Rollen-Brainstorming.


**Abbildung 4.5** Benutzerrollen der Scrumcoaches-Anwendung

Im nächsten Abschnitt erkläre ich, wie die modellierten Benutzerrollen in der Beschreibung von User Stories verwendet werden, indem die Stories nach einem sehr stringenten, dafür aber sehr ausdrucksstarken Muster beschrieben werden.

### 4.3.3 User-Story-Muster

User Stories werden in einem einzigen Satz auf ihrer Story-Karte beschrieben. Für die Beschreibung hat sich das folgende Muster bewährt:

Als <Benutzerrolle> will ich <das Ziel>[, so dass <Grund für das Ziel>].

Das Muster enthält drei Platzhalter: Die *Benutzerrolle* wird durch die entsprechende Rolle ersetzt, aus deren Sicht die Story geschrieben wird. Das *Ziel* drückt den Kern der von der Story-Karte beschriebenen Anforderung aus. Der optionale *Grund für das Ziel* erklärt die Motivation, weshalb die Benutzerrolle überhaupt das genannte Ziel hat. Einige Beispiele verdeutlichen das Prinzip:

**Tabelle 4.1** User Stories nach Muster

| Benutzerrolle | Ziel | Grund |
|---|---|---|
| Als Scrum-Coach | will ich mich anmelden, | so dass ich mein Profil einstellen kann. |
| Als Anbieter | will ich nach Coaches suchen, | so dass ich Kontakt aufnehmen kann. |
| Als ehemaliger Kunde | will ich einen Coach bewerten, | so dass sich seine gute Arbeit herumspricht. |

Das Muster stammt von Mike Cohn. Während er in seinem ersten Buch [Cohn 2004] noch die vereinfachte Form „Der Benutzer kann sich anmelden“ beschreibt, verwendet er in seinem zweiten Buch [Cohn 2006] konsequent die hier beschriebene Form und liefert in seinem Blog die zugehörige Begründung [Cohn 2008: p24]. Mike Cohn hat mit der Beschreibung und Verbreitung dieses Musters etwas genial Einfaches geschaffen, indem er mit ganz wenigen Worten sehr viele Informationen übermittelt. Mit einem einzigen Satz werden drei Fragen beantwortet: Wer will etwas, was will derjenige und warum will er es? Das Muster zwingt den Product Owner geradezu, den Geschäftswert der Story in nur einem einzigen Satz auf den Punkt zu bringen, und das ist ein sehr einfacher, dafür umso wirkungsvollerer Leitfaden für das Schreiben von User Stories.

### 4.3.4 Epics

Nicht alle Stories werden auf derselben Granularitätsebene geschrieben. Je näher eine Story an den nächsten Sprint rückt, desto konkreter sollte sie sein. Um die unterschiedlichen Detailebenen und die jeweilige Nähe zum nächsten Sprint deutlich zu machen, wird zwischen Epics und konkreten User Stories unterschieden. Epics sind große User Stories, die weder vernünftig geschätzt noch innerhalb eines Sprint entwickelt werden können. Abbildung 4.6 zeigt einige Beispiele für Epics.

Ein gutes Beispiel für ein Epic ist die Story „Rechnungsstellung“. Hier wird auf den ersten Blick klar, dass die Story viel zu groß ist und vor ihrer Entwicklung auf eine ganze Reihe konkreterer Einzelstories heruntergebrochen werden muss. Viele der in diesem Epic enthaltenen Stories liegen auf der Hand: „Bezahlung per Kreditkarte“, „Bezahlung per Überweisung“, „Rechnungsversand per Post“ oder „Buchhaltungssystem anbinden“. Einige dieser Sto-

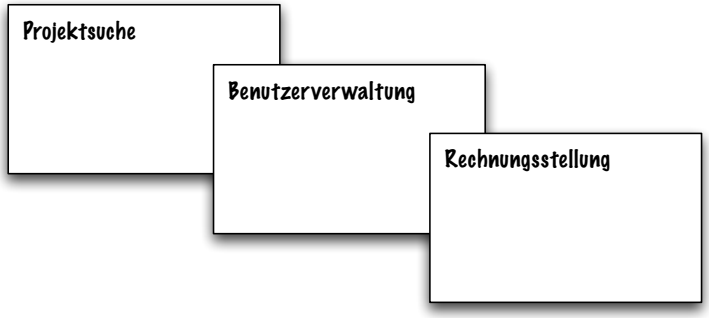



**Abbildung 4.6** Einige Epics

ries, wie die Anbindung des Buchhaltungssystems, sind selber wiederum Epics, die weiter heruntergebrochen werden müssen.

Mit Epics lassen sich große Teile des Systems schnell beschreiben, ohne dabei zu sehr ins Detail zu gehen. Sie sind ein Hilfsmittel, um das Wissen über ein System hierarchisch zu strukturieren und erst dann zu konkretisieren, sobald es wichtig wird. Üblicherweise starten Projekte mit einer Reihe von Epics, die zunächst nur als Platzhalter im Product Backlog stehen und den groben Rahmen des Systems abstecken. Für Scrumcoaches.com könnten dies zum Beispiel die Epics „Projektsuche“, „Profilmanagement“ und „Coach-Suche“ sein. Das Product Backlog wird mit Epics gefüllt und je nach deren Priorität auf konkretere Stories heruntergebrochen. Abbildung 4.7 zeigt das Herunterbrechen des Epics „Profilmanagement“.

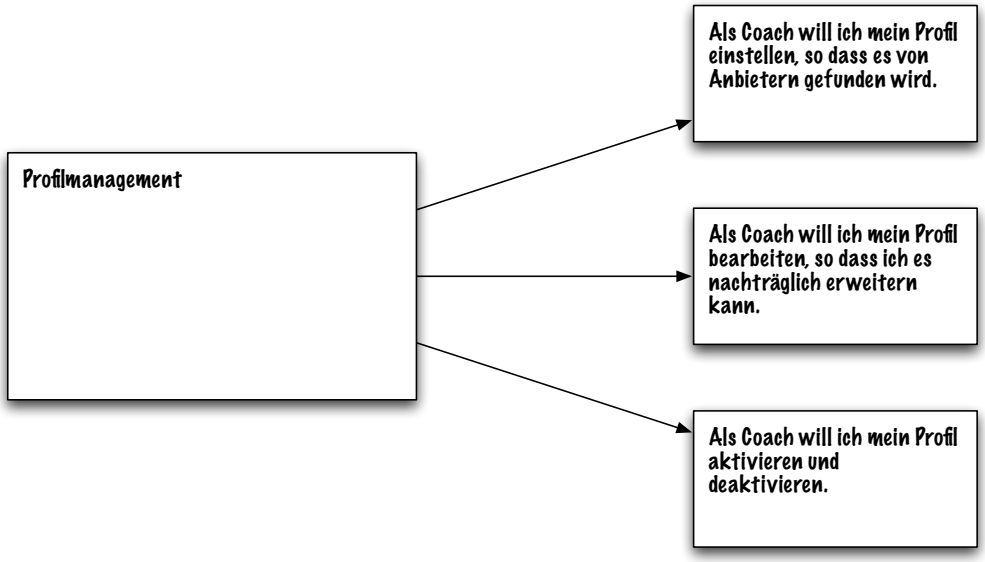



**Abbildung 4.7** Ein Epic wird zu konkreten User Stories

Epics müssen nicht zwingend dem User-Story-Muster *Als <Benutzerrolle> will ich <das Ziel>*[*, so dass <Grund für das Ziel>*] folgen. Häufig entstehen Epics aus einer Idee, die mal eben schnell aufgeschrieben wird. Dabei ist meistens noch nicht klar, welcher Benutzerrolle die Story zugute kommt, so dass es sich nicht lohnt, lange über ihre Formulierung nachzudenken. Ein gutes Beispiel ist erneut das Epic „Rechnungsstellung“, von der eine ganze Reihe Benutzerrollen betroffen sind: Projektanbieter bezahlen für die erfolgreiche Vermittlung, die Zahlungseingänge werden vom Betreiber überprüft oder Coaches zahlen für die Nutzung erweiterter Funktionalität.

Epics eignen sich zum Festhalten von User Stories, von denen man noch nicht genau weiß, ob man sie wirklich braucht. Beispielsweise könnte der Product Owner von Scrumcoaches.com im Laufe des Projekts auf die Idee kommen, einige Zusatztools für Scrum-Projekte über die Plattform anzubieten. Das Naheliegendste ist vielleicht ein elektronisches Product Backlog und der Product Owner notiert die Anforderung als Epic im Backlog.

Epics haben eine niedrige Priorität. Sie sind zu wenig konkret, als dass sie direkt ins Selected Backlog eines Sprint übernommen werden könnten. Der Product Owner ist dafür zuständig, die Epics des Product Backlog regelmäßig durchzugehen und zu prüfen, welche Bedeutung das Epic wirklich hat. Wichtige Epics steigen in ihrer Priorität, was zur Folge hat, dass sie in mehrere Stories aufgeteilt werden müssen. Kapitel 7, *User Stories fürs Product Backlog*, beschreibt verschiedene Techniken für das Herunterbrechen von Epics auf konkrete User Stories.

### 4.3.5 Themen

Neben Epics findet man in der Literatur noch den Begriff des *Themas* [Cohn 2004]. Ein Thema ist kein eigenständiger Story-Typ, sondern eine Menge zusammengehöriger User Stories, die sich um ein bestimmtes funktionales Thema herum gruppieren. Themen entstehen beim Aufteilen von Epics in konkrete User Stories. Wie Sie im vorangehenden Abschnitt lesen konnten, enthält das Epic „Profilmanagement“ eine ganze Reihe von User Stories. Das Epic wird durch Aufteilung in User Stories zum Thema, das jetzt die neue logische Klammer um den funktionalen Bereich „Profilmanagement“ ist.

In meiner praktischen Arbeit mit User Stories sind Themen zumeist ein theoretisches Konzept geblieben. Meiner Erfahrung nach reichen Epics und User Stories aus, um ein mit User Stories gefülltes Product Backlog zu erarbeiten und zu pflegen, so dass der Begriff des Themas im weiteren Verlauf des Buches nur noch selten auftaucht. Bereiche, in denen Themen sinnvoll sind, sind Priorisierung und Release Management. Häufig lassen sich ganze Themen einfacher priorisieren als einzelne User Stories. Ein praktisches Beispiel zum themenbasierten Release Management kann ich aber doch bieten: Das Thema „Rechte und Rollen“ ist ein Beispiel aus einem konkreten Projekt. Das Thema bestand aus insgesamt sieben User Stories, wobei sich der Gesamtwert für den Kunden erst offenbart hat, nachdem alle sieben Stories umgesetzt waren. Wir haben das Thema auf zwei Sprints verteilt, die Software aber erst wieder ausgeliefert, nachdem das komplette Thema umgesetzt war.

### 4.3.6 Wie viel Detail?

Product Owner kommen häufig aus dem klassischen Produkt- oder Projektmanagement und sind es gewohnt, Anforderungen von vornherein sehr genau abzustimmen und detailliert zu beschreiben. User Stories erfordern ein Umdenken. Details werden nicht mehr weit im Vorfeld der Story ausgearbeitet und aufgeschrieben, sondern erst, wenn die Story konkret wird. Für den Product Owner wird die Story konkret, sobald er absehen kann, dass die Story Teil eines der nächsten Sprints werden wird. Ab diesem Punkt kann er beginnen, Informationen zu sammeln, Rücksprache mit dem Kunden zu halten und sich zu überlegen, wie er dem Team die Story im Sprint Planning Meeting präsentieren wird. Für das Team wird die Story meistens erst im Sprint Planning Meeting konkret, wo es darum geht, sie genauer zu analysieren und erste Details in Form von Notizen, Fragen und Akzeptanztests zu notieren. Die wirklichen Details werden allerdings erst während der Entwicklung der Story im Dialog zwischen Team und Product Owner geklärt. Solange aber nicht abzusehen ist, ob und wann die Story wirklich entwickelt wird, lohnt sich eine intensive Beschäftigung mit deren Details noch nicht.

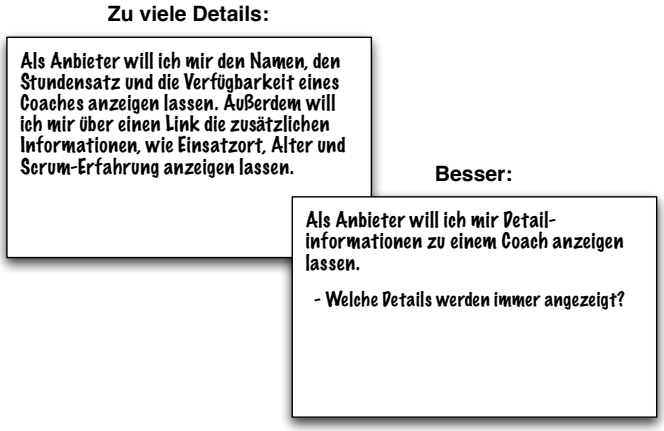



**Abbildung 4.8** Zu viele Details

Abbildung 4.8 zeigt zwei Story-Karten für dieselbe User Story, in der es um die Anzeige der Details eines Suchergebnisses geht. Wenn die Story geschrieben wird, ist vielleicht noch gar nicht klar, welche Attribute ein Coach besitzt und welche davon optional sind. Besser ist es, die Story offen zu lassen und erforderliche Details als Fragen zu formulieren. Die Story wird allgemein gehalten, wodurch Raum für ihre Verhandlung während der Entwicklung entsteht (siehe dazu auch Abschnitt 4.4.2).

Das Gegenteil von zu detaillierten Stories sind zu weit gefasste Stories, die keine abgeschlossene Funktionalität repräsentieren. „Als Projektanbieter will ich das von mir eingestellte Angebot managen." Diese Story ist nicht konkret genug. Sie ist zu offen formuliert und kann alles Mögliche beinhalten. Besser sind Stories wie: „Als Projektanbieter will ich ein Projektangebot einstellen" oder „Als Projektanbieter will ich ein Projektangebot löschen". Jede Story sollte eine abschließbare Anforderung repräsentieren. Zu offene Stories sind nur schwer schätz- und planbar, da sie keine Kriterien für ihren Umfang besitzen.

### 4.3.7 Keine Technik

User Stories sind frei von technischem Jargon. Den Product Owner oder Kunden interessiert es nicht, ob die Anwendung auf zwei Applikationsservern läuft oder in Ruby programmiert wurde. Diese Informationen liefern keinen sichtbaren Mehrwert im Sinne des Kerngeschäfts des Kunden und haben deshalb in User Stories nichts zu suchen. Statt technische Details und nicht-funktionale Anforderungen mit auf die Story-Karte zu schreiben, werden sie als sogenannte *Constraints*[2] formuliert und auf separate Karteikarten geschrieben (siehe dazu auch Abschnitt 7.5.2).

### 4.3.8 Keine Benutzeroberfläche

User Stories treffen keinerlei Annahmen über die Benutzeroberfläche. Eine Story beschreibt ausschließlich das Ziel einer Rolle, wie „Als Kunde will ich eine Mail versenden“. Teil der Story ist nicht, über was für eine Art Dialogfenster der Mailtext eingegeben wird. Die Benutzeroberfläche ergibt sich aus der Kommunikation mit dem Product Owner und wird im Rahmen der Entwicklung iterativ konkretisiert.

## 4.4 Eigenschaften guter User Stories

Mike Cohn hat in [Cohn 2004] die Eigenschaften guter User Stories mit Hilfe des Akronyms INVEST beschrieben:

**Tabelle 4.2** Eigenschaften guter User Stories

| | | |
|---|---|---|
| I | Independent | User Stories sollen unabhängig voneinander sein. |
| N | Negotiable | User Stories sollen verhandelbar sein. |
| V | Valuable | User Stories sollen einen Wert für den Kunden haben. |
| E | Estimatable | User Stories sollen schätzbar sein. |
| S | Small | User Stories sollen klein sein. |
| T | Testable | User Stories sollen testbar sein. |

Es lohnt sich, die Einzelbedeutungen des Akronyms zu verinnerlichen und sich beim Schreiben von User Stories immer wieder vor Augen zu führen.

### 4.4.1 Independent – unabhängige User Stories

User Stories sollen unabhängig voneinander sein. Abhängige Stories erzeugen ein Reihenfolgeproblem in Bezug auf ihre Umsetzung. Ist zum Beispiel Story B wichtiger als Story A im Sinne eines höheren Geschäftswerts für den Kunden, dann sollte Story B auch vor

[2] Auf Deutsch „Neben-“ oder „Randbedingung“.

Story A umgesetzt werden. Ist Story B jedoch von Story A abhängig, da Story A notwendige Voraussetzung für B schafft, dann sind die Stories nicht mehr unabhängig voneinander priorisierbar und wir haben ein Reihenfolgeproblem.

Beispiele für abhängige Stories sind die Stories „Als Coach will ich mich registrieren“ und „Als Coach will ich mich anmelden“. Die Anmelde-Story macht ohne Registrierung keinen Sinn und muss entsprechend nach der Registrierung umgesetzt werden. Eine Lösung für das Auflösen von Abhängigkeiten ist das Zusammenfassen der abhängigen Stories. Die Registrierungs-Story könnte beispielsweise um das Akzeptanzkriterium „Ein Coach kann sich nach erfolgreicher Registrierung einloggen“ erweitert werden.

Beim Zusammenfassen von User Stories stellt sich allerdings schnell das Problem von zu großen Stories, die es genauso zu vermeiden gilt wie abhängige Stories (siehe Abschnitt 4.4.5). Führt das Zusammenfassen von User Stories zu einer zu großen Story, ist es besser, die abhängige Story niedriger zu priorisieren. Prioritäten geben die Abarbeitungsreihenfolge von User Stories vor und stellen so sicher, dass voneinander abhängige Stories in der richtigen Reihenfolge bearbeitet werden.

### 4.4.2 Negotiable – verhandelbare User Stories

User Stories sollen verhandelbar sein. Sie sind kein festgezurrter Vertrag, der jede Einzelheit bis ins kleinste Detail beschreibt. Stattdessen werden die Details einer Story während ihrer Entwicklung zwischen Product Owner und Entwickler verhandelt. Ein Beispiel: Der Product Owner will unbedingt, dass Feature X noch mit in die Story kommt. Dem Entwickler fehlt die Zeit und er bietet an, Feature Y wegzulassen und dafür X zu realisieren. Nun ist es am Product Owner zu entscheiden, welches der beiden Features wichtiger ist.

Die Verhandelbarkeit wird zusätzlich interessant, wenn es um das Erreichen des Sprint-Ziels geht. Sprint-Ziele korrespondieren in der Regel mit einer Reihe von User Stories, die gemeinsam das festgelegte Ziel realisieren. Die beiden Stories „Als Coach will ich mein Profil erfassen“ und „Als Anbieter will ich nach Coaches suchen“ könnten zum Beispiel Teil des Sprint-Ziels sein, den Anbietern eine erste Version der Suchfunktionalität zur Verfügung zu stellen. Angenommen, die erste Story des Sprint gerät ins Stocken und das Team läuft Gefahr, die Such-Story nicht mehr fertigzubekommen. Um das Sprint-Ziel nicht zu gefährden, könnten Product Owner und Team den Funktionsumfang der ersten Story verhandeln und so weit reduzieren, dass ausreichend Luft für die Such-Story bleibt. Eine Möglichkeit wäre zum Beispiel, die relativ aufwendige Funktion zum Hochladen von Dateien nachträglich aus der Profilerfassungs-Story zu nehmen und als neue Story ins Product Backlog zu schreiben.

### 4.4.3 Valuable – wertvolle User Stories

User Stories sollen einen erkennbaren Mehrwert für den Anwender des Systems liefern. Er ist es, der etwas davon haben muss, dass die Story realisiert wird. Für Stories wie „Als Anbieter will ich nach Coaches suchen“ liegt der Mehrwert auf der Hand. Der Anbieter erhält als konkreten Mehrwert, dass er über diese Funktion einen qualifizierten Coach für sein Projekt finden kann.

Nicht für jede neue Anforderung ist der Mehrwert so offensichtlich. Typische Beispiele sind nicht-funktionale Anforderungen wie Sicherheit oder Logging. Diese Anforderungen sind zwar wichtig und müssen erledigt werden, aber eher als Teil einer User Story und nicht als eigenständige Story. Sicherheits- und Logging-Funktionalität sollten dann eingebaut werden, wenn man sie im Rahmen einer User Story benötigt. Beispielsweise muss bei der Anmelde-Story protokolliert werden, wann sich welcher Benutzer am System anmeldet, das heißt: Logging ist ein konkreter Task der Story.

Aber auch rein technische Stories können einen Mehrwert für den Benutzer bieten, wenn sie entsprechend anwendergerecht aufbereitet werden. Statt „Einführung eines Connection Pools“ kann man auch eine Story der Art „Die Software soll von 50 Benutzern gleichzeitig genutzt werden können“ schreiben. In diesem Fall wird der Mehrwert wieder offensichtlich, denn der Anwender kann die Anwendung immer noch nutzen, wenn neben ihm noch 49 andere Anwender mit dem System arbeiten. Allerdings gelingt es nicht für jede technische Anforderung, sie in eine für den Anwender werthaltige Story zu gießen. Die Automatisierung des Deployments ist ein solches Beispiel, von dem erst mal nur das Team etwas hat. Kapitel 7, *User Stories fürs Product Backlog*, unterbreitet einige Vorschläge, wie sich auch diese Art von Funktionen als User Stories formulieren lassen.

Wertvolle User Stories entstehen am ehesten, wenn, wie in diesem Buch gefordert, der Product Owner als Vertreter des Kunden die Stories schreibt. Die zugehörigen technischen Anforderungen der User Story werden später von den Entwicklern gefunden, wenn es an die Aufteilung der Story auf konkrete Einzeltasks geht.

### 4.4.4 Estimatable – schätzbare User Stories

Der für die Umsetzung einer User Story notwendige Aufwand soll schätzbar sein. Agiles Schätzen von User Stories basiert auf dem Prinzip der relativen Größe. User Stories werden nicht in Dauer, sondern ihre Größe wird in Relation zu anderen Stories geschätzt: User Story A ist halb so groß wie User Story B, aber drei Mal so groß wie User Story C. Ausführliche Informationen zum relativen Schätzen von User Stories finden Sie in Kapitel 5, *Agiles Schätzen*.

Wichtig für eine schätzbare User Story sind weniger ihre Details, als ihre Abgrenzung nach oben sowie nach unten hin: Was muss die Story minimal liefern und was liegt klar außerhalb ihres Funktionsumfangs. Gründe dafür, dass Stories nicht oder nur schwer schätzbar sind, sind zu große Stories, aber auch fehlendes technisches oder Domain-Wissen beim Entwickler.

### 4.4.5 Small – kleine User Stories

User Stories sollen klein sein. Zu große Stories sind zu wenig konkret und deshalb nur schwer schätzbar und können außerdem nicht innerhalb eines Sprints entwickelt werden. Ein Beispiel für eine zu große Story ist die Story „Benutzerverwaltung“, hinter der sich eine ganze Menge an Teilanforderungen verbergen, die sich als jeweils eigenständige User Stories formulieren lassen: unterschiedliche Rechte für verschiedene Benutzerklassen, Freischalten

und Deaktivieren von Benutzern oder auch die Bereitstellung mehrerer Konten für einen einzigen Projektanbieter.

Im Zusammenhang mit Story-Größe spricht man auch von der zeitlichen Dimension einer User Story. Stories müssen nicht zwangsläufig klein sein, sondern nur dann, wenn sie in unmittelbare Sprint-Nähe rücken. Der Detaillierungsgrad, mit dem eine Story beschrieben oder über sie diskutiert wird, hängt davon ab, wie nahe die Story an den nächsten Sprints dran ist. Weit in der Zukunft liegende Stories können wesentlich größer und weniger detailreich sein als Stories, die für den nächsten Sprint anstehen.

Die richtige Story-Größe hängt natürlich auch von der Länge der Sprints ab. Klar ist, dass jede Story innerhalb eines Sprint umsetzbar sein muss. Allerdings gibt es auch eine Grenze nach unten. Bugfixes oder das Ändern einer Hintergrundfarbe sind zu klein, um als User Stories durchgehen zu können. Ein guter Richtwert für die Größe einer Story ist eine Entwicklungsdauer von einem Tag bis zu maximal zwei Wochen, vorausgesetzt, die Länge des Sprint lässt dies zu.

### 4.4.6 Testable – testbare User Stories

User Stories sollen testbar sein. Testbare User Stories haben klar definierte Akzeptanzkriterien, die bestimmen, wann die Story fertig ist. Je geringer die Testbarkeit einer Story, desto größer ihr Testaufwand und desto schwieriger ist zu bestimmen, wann die Story fertig ist.

Testbarkeit findet auf verschiedenen Ebenen statt. Für das Team beginnt das Testen der Story bereits während ihrer Entwicklung mit dem Schreiben von Unit-Tests. Der Code ist erst dann fertig, wenn die Unit-Tests der Story sowie alle anderen Unit-Tests der Anwendung laufen. Darüber hinaus muss das Team entwickelte Stories auch innerhalb einer Integrationsumgebung testen können. Unit- und Entwicklertests sind selbstverständlich und sollten von jedem eigenverantwortlich und selbstorganisiert arbeitenden Team durchgeführt werden.

Vor allem kommt es darauf an, dem Product Owner das Testen zu ermöglichen. Letztendlich ist er es, der jede User Story abnehmen und für fertig erklären muss, und dafür benötigt er eindeutige und messbare Akzeptanzkriterien. Diese sind Teil jeder User Story und je klarer sie formuliert sind, desto einfacher sind das Testen und die Abnahme der Story.

Während die Akzeptanzkriterien für den Product Owner zum Ende der Story hin wichtig werden, sind sie es für das Team bereits während der Entwicklung. Akzeptanzkriterien stecken den Rahmen der Story ab und definieren klar, worauf es ankommt. Aus diesem Grund ist es wichtig, die Akzeptanzkriterien nicht erst am Ende zu schreiben, sondern bereits vor dem Sprint oder spätestens im Sprint Planning Meeting. Ein Beispiel für eine testbare Story mit guten Akzeptanzkriterien ist die Story „Als Coach will ich mich anmelden“. Der Product Owner hat die folgenden Akzeptanzkriterien auf der Story-Karte notiert:

- Die Anmeldung erfolgt über E-Mail und Passwort.
- Das Passwort wird verschlüsselt in der Datenbank gespeichert.
- Das Benutzerkonto wird nach drei aufeinanderfolgenden Fehlversuchen deaktiviert.
- Nach einer erfolgreichen Anmeldung erscheint ein Menü mit Coach-spezifischen Funktionen.

Das Team entwickelt die Story entlang dieser Akzeptanzkriterien und stellt die Story nach Fertigstellung auf dem Integrationsserver bereit. Der Product Owner testet die Story gegen die spezifizierten Akzeptanzkriterien und lässt sich von einem der Entwickler zeigen, dass das Passwort verschlüsselt in der Datenbank steht. Ausführliche Informationen zu Akzeptanzkriterien, Akzeptanztests und der Akzeptanztest-getriebenen Entwicklung von User Stories finden Sie in Kapitel 11, *User Stories Akzeptanztesten*.

## 4.5 Zusammenfassung

- Eine User Story beschreibt eine Anforderung aus Sicht des Benutzers. Sie besteht aus:
  - einer Story-Karte mit einer Beschreibung der Anforderung;
  - der Konversation zwischen Product Owner und Team;
  - den Akzeptanzkriterien, die bestimmen, wann die Story fertig ist.
- Der Schwerpunkt einer User Story ist die Konversation zwischen Product Owner und Team, während die Story entwickelt wird.
- Benutzerrollen repräsentieren unterschiedliche Gruppen von Anwendern und helfen beim Schreiben zielgerichteter und werthaltiger User Stories.
- Die Beschreibung einer User Story sollte dem Muster „Als <Benutzerrolle> will ich <das Ziel>[, so dass <Grund für das Ziel>]." folgen.
- Epics sind sehr große User Stories, die gute Platzhalter für Ideen oder zukünftige Themenbereiche sind, vor ihrer Umsetzung aber auf konkretere User Stories heruntergebrochen werden müssen.
- Gute User Stories genügen den INVEST-Kriterien. Sie sind unabhängig, verhandelbar, wertvoll, schätzbar, klein und testbar.

## 4.6 Wie geht es weiter?

Jetzt kennen Sie die beiden Grundbausteine und damit das Fundament dieses Buches: Scrum und User Stories. Mit den nächsten beiden Kapiteln *Agiles Schätzen* und *Agiles Planen* beginnt der Schwenk hin zum Kern des Buches, der Kombination von Scrum und User Stories. Ziel ist es, User Stories im Product Backlog zu verwalten und so zu planen, dass sie sich für die Entwicklung im Rahmen von Sprints eignen. Um zu planen, wie viele Stories in einen Sprint passen, ist es wichtig zu wissen, wie groß die einzelnen Stories sind, und genau darum geht es im nächsten Kapitel.

# Stichwortverzeichnis