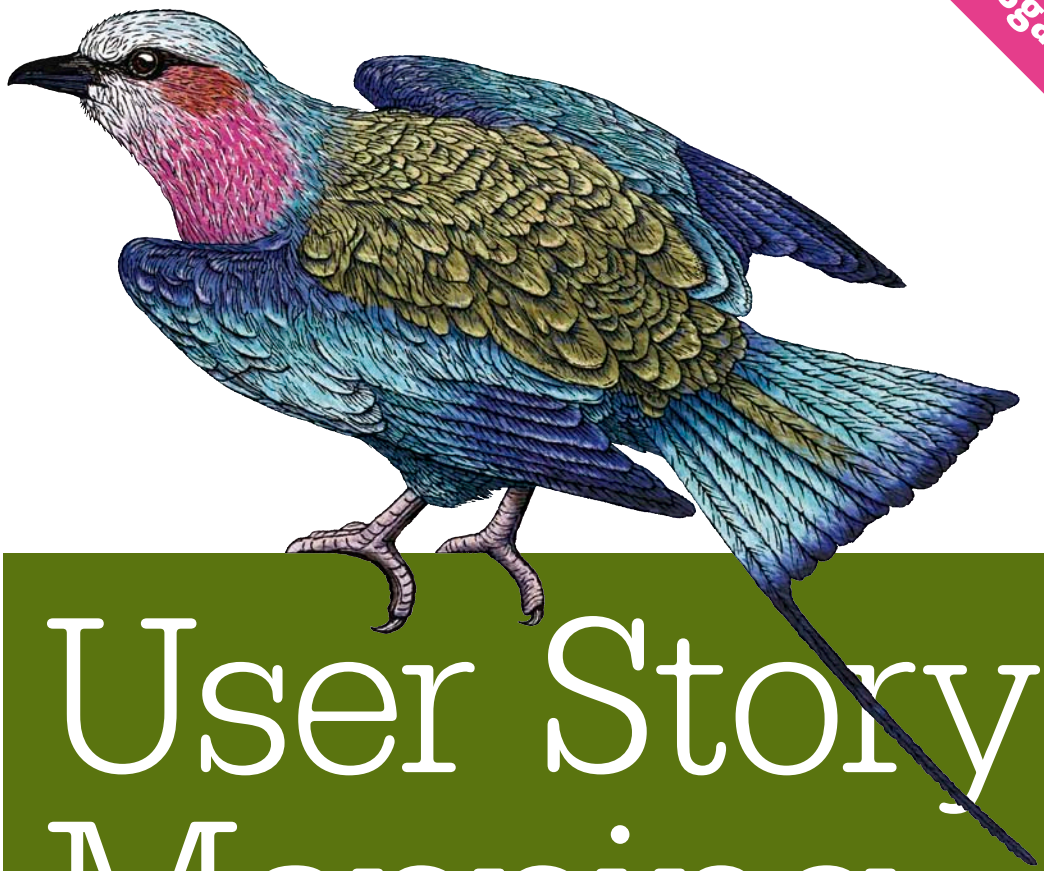
O'REILLY®

Deutsche Ausgabe



User Story Mapping

DIE TECHNIK FÜR BESSERES NUTZERVERSTÄNDNIS IN DER AGILEN PRODUKTENTWICKLUNG

Jeff Patton
mit Peter Economy
Übersetzung von Petra Hildebrandt

User Story Mapping

Jeff Patton

mit Peter Economy

Übersetzung von Petra Hildebrandt

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
E-Mail: kommentar@oreilly.de

1. Auflage 2015

Die Originalausgabe erschien 2014 unter dem Titel *User Story Mapping* bei O'Reilly Media, Inc.

Die Darstellung einer Gabelschwanzracke im Zusammenhang mit dem Thema User Story Mapping ist ein Warenzeichen des O'Reilly Verlags GmbH & Co. KG

Bibliografische Information Der Deutschen Nationalbibliothek Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.de* abrufbar.

Lektorat: Susanne Gerbert, Köln
Übersetzung: Petra Hildebrandt, Hamburg
Korrektorat: Eike Nitz, Köln
Umschlaggestaltung: Michael Oreal, Köln
Produktion: Andrea Miß, Köln
Satz: Reemers Publishing Services GmbH, Krefeld,
www.reemers.de,
Belichtung, Druck und buchbinderische Verarbeitung:
Mediaprint, Paderborn

ISBN: 978-3-95875-067-8

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Widmung

Für Stacy, Grace und Zoe, meine größten Unterstützerinnen, die all meinen Mühen einen Sinn geben.

Und im Angedenken an Luke Barrett, einen geschätzten Kollegen und Mentor. Er hat mein Leben verändert, so wie das ungezählter anderer.

Vorwort

Vorwort von Martin Fowler

Zu den positiven Auswirkungen des Aufstiegs der agilen Softwareentwicklung gehört die Verbreitung des Konzepts, eine große Menge Anforderungen in kleinere Stücke aufzuteilen. Diese Stücke – Stories – erlauben viel bessere Einblicke in den Fortschritt eines Entwicklungsprojektes. Wenn ein Produkt Story für Story entsteht und jede Implementation einer Story vollständig in das Softwareprodukt integriert ist, kann jeder sehen, wie das Produkt wächst. Dadurch, dass sie Stories verwenden, die aus Usersicht Sinn ergeben, können Entwickler ihr Projekt steuern, indem sie festlegen, welche Stories sie als Nächstes bearbeiten werden. Die bessere Sichtbarkeit fördert eine stärkere Mitwirkung von Userseite – Sie müssen nicht mehr ein Jahr oder länger darauf warten, dass Sie sehen können, was das Entwicklerteam so getrieben hat.

Die Zerstückelung hat aber auch negative Konsequenzen. Dazu gehört, dass man schnell den Überblick darüber verlieren kann, was die Software als Ganzes tun soll. Man kann ein Durcheinander an Stücken herausbekommen, die nicht in ein zusammenhängendes Ganzes passen. Oder man baut ein System, das den Usern nicht wirklich etwas nutzt, weil einem zwischen all den Details der Sinn für das, was im Kern benötigt wird, abhanden gekommen ist.

Story Mapping ist eine Methode, mit der der Gesamtzusammenhang hergestellt werden kann, den ein bloßer Haufen von Stories oft vermissen lässt.

Und das war's auch schon – die Beschreibung dieses Buches in einem Satz. Dieser Satz trägt in sich ein großes Versprechen. Der Blick auf das große Ganze hilft dabei, mit Usern effektiv zu kommunizieren, er hilft allen Beteiligten, zu vermeiden, unnütze Features zu produzieren, und er dient als Orientierungshilfe für eine in sich stimmige User Experience. Wenn ich mit meinen Kollegen bei ThoughtWorks darüber spreche, wie sie ihre Stories entwickeln, nennen sie Story Mapping regelmäßig als Schlüsselmethode. Häufig haben sie die Methode in Workshops mit Jeff gelernt, denn er hat die Methode entwickelt und kann sie am besten kommunizieren. Dieses Buch gibt mehr Menschen die Möglichkeit, die Methode direkt an der Quelle zu erlernen.

Aber dieses Buch ist nicht nur etwas für Leute, die einen Titel wie »Business-Analyst« auf der Visitenkarte oder ihrem Online-Profil stehen haben. Eine der größten Enttäuschungen für mich war in diesem Jahrzehnt, in dem agile Methoden immer mehr Anwendung fanden, dass viele Programmierer Stories als eine Kommunikations-Einbahnstraße vom Analysten zu ihnen ansehen. Von Anfang an waren Stories als Zündfunken für *Konversationen* gedacht. Wenn man sich tatsächlich effektive Softwarelösungen für bestimmte Tätigkeiten ausdenken will, muss man diejenigen, die die Software schreiben, als lebenswichtige Ideengeber für die Einsatzmöglichkeiten betrachten, denn es sind die Programmierer, die am besten wissen, was Software leisten kann. Programmierer müssen verstehen, was ihre User zu erreichen versuchen, und sollten kollaborativ Stories entwickeln, die diese Bedürfnisse der User abbilden. Ein Programmierer, der Story Mapping beherrscht, ist besser in der Lage, den Kontext des Users zu erkennen, und kann dabei helfen, die Software zu umreißen – und somit seine Arbeit besser zu machen.

Als Kent Beck (auf den der Gedanke der »Story« zurückgeht) seine Ideen zur Softwareentwicklung erarbeitete, bezeichnete er Kommunikation als Schlüsselwert für effektive Teams. Stories sind die Bausteine der Kommunikation zwischen den Entwicklern und denen, die ihre Arbeit benutzen werden. Story Maps organisieren und strukturieren diese Bausteine und verbessern damit den Kommunikationsprozess – der den wohl wichtigsten Part der Softwareentwicklung darstellt.

Vorwort von Alan Cooper

In Mary Shelleys berühmtem Science-Fiction-Roman *Frankenstein* erschafft der wahnsinnige Dr. Frankenstein eine Kreatur aus verschiedenen Stücken toter Menschen und erweckt diese Kreatur mit der damals neuartigen Technologie der Elektrizität zum Leben. Natürlich wissen wir, dass das eigentlich nicht möglich ist. Man kann kein Leben erschaffen, indem man zufällige Körperteile zusammennäht.

Und doch ist es das, was Softwareentwickler immer wieder versuchen: Sie erweitern eine Software um gute Features, eins nach dem anderen, und dann wundern sie sich, wieso so wenige User ihr Produkt gut finden. Der Kern des Problems liegt darin, dass Entwickler ihre Konstruktionsmethode als Design-Tool benutzen, die beiden aber nicht untereinander austauschbar sind.

Es ist absolut sinnvoll, dass Programmierer die Software Feature für Feature *produzieren*. Das ist eine hervorragende Strategie, die sich jahrelang bewährt hat. Was sich allerdings auch über die Jahre gezeigt hat, ist, dass der Ein-Feature-nach-dem-anderen-Ansatz – setzt man ihn als Designmethode für das Verhalten und den Umfang eines digitalen Produkts ein – zu einem Frankenstein'schen Monster(-Programm) führt.

Wenn sie auch eng verwandt sind, so unterscheiden sich die Praxis des Softwaredesigns und die der Erstellung ebendieser Software doch deutlich voneinander und werden üblicherweise von verschiedenen Personen mit unterschiedlichen Skills absolviert. Zahllose Stunden damit zuzubringen, User zu beobachten und Verhaltensmuster zu mappen, wie das die Interaktionsdesigner tun, würde die meisten Programmierer in den Wahnsinn treiben. Umgekehrt wäre stundenlanges Brüten über Algorithmen eine viel zu einsiedlerische Tätigkeit für die meisten Designer.

Aber wenn diese beiden unterschiedlichen Ansätze – Design und Entwicklung – zusammenarbeiten, kommt eine elektrische Spannung auf, die das Potenzial besitzt, ein lebendes, atmendes Produkt zu erschaffen. Teamarbeit haucht dem Monster Leben ein und bringt die Leute dazu, es zu lieben.

Zwar ist die Idee der Zusammenarbeit weder neu noch sonderlich aufschlussreich, aber es ist dennoch sehr schwer, sie tatsächlich effektiv umzusetzen. Die Arbeitsweise der Entwickler – ihr Tempo,

ihre Sprache und ihr Rhythmus – unterscheidet sich sehr von der der Interaktionsdesigner.

Die Fachleute beider Gebiete sind stark, kompetent und innerlich diszipliniert, aber sie haben beide die gleiche Schwäche: Es ist wirklich schwer, ein Designproblem in Programmierersprache auszudrücken, und es ist ebenso schwer, ein Entwicklungsproblem in der Sprache der Designer zu formulieren. Die beiden Schwesterdisziplinen haben keine gemeinsame Sprache. Und genau dort, auf der Kreuzung zwischen den beiden Disziplinen, finden wir Jeff Patton.

Jeffs Methode des Story Mapping ist für Entwickler gut verständlich, und sie ist ebenso gut verständlich für die Designer. Story Mapping ist der Rosetta-Stein des digitalen Zeitalters.

Trotz gegenteiliger Behauptungen ist die agile Entwicklung kein sonderlich gut geeignetes Designwerkzeug. Es handelt sich um eine Art, über Entwicklung nachzudenken, die designfreundlich ist – was eine gute Sache ist –, aber für sich genommen, führt sie nicht zu einem Produkt, das die Nutzer toll finden. Auf der anderen Seite kann man oft erleben, dass gute, wohldokumentierte Designs in die Hände von Entwicklern gelegt werden, die dann – agil oder nicht – die inneren Werte dieses Designs in der Umsetzung vollständig plätten.

Pattons Story-Mapping-Ansatz ist ein Brückenschlag über diesen Abgrund. Beim Interaktionsdesign geht es darum, die Wahrheit des Users zu finden und als Erzählung abzubilden. In der Softwareentwicklung geht es darum, diese Erzählung in kleine, funktionale Stücke zu zerlegen und diese wiederum zu implementieren und zu integrieren. Es passiert ganz schnell, dass in diesem komplexen Prozess der Wesenskern der Erzählung flöten geht. Ja, die Funktionen wurden implementiert, aber der Patient stirbt auf dem OP-Tisch.

Indem es die User Stories als Landkarte auslegt, behält das Design seine narrative Struktur, kann aber dennoch für eine effektive Implementation dekonstruiert werden. Die Geschichte des Designers, die eine formalisierte Fassung der User Story ist, bleibt durch die gesamte Entwicklung hindurch intakt.

Die klassische Welt der Konzerne hat bewiesen, dass es nahezu unmöglich ist, mit Teams von zwei- oder dreihundert Leuten Produkte herzustellen, die jemand wirklich richtig gut findet. Die Startup-Community wiederum hat bewiesen, dass ein Team aus vier oder

fünf Personen *sehr wohl* kleine Produkte machen kann, die die Leute mögen. Aber selbst solche Produkte werden irgendwann groß und verlieren diesen inneren Funken. Die Herausforderung, der wir uns stellen müssen, ist, große Software zu erschaffen, die die Leute lieben. Große Software bedient große Gruppen von Nutzern, die komplexe, kommerziell relevante Aufgaben damit verrichten. Es ist unglaublich schwer, solche Software so zu gestalten, dass ihre Benutzung Spaß macht und leicht zu erlernen ist.

Die einzige Methode, auf die wir große Software produzieren können, die kein Frankenstein'sches Monster ist, ist, indem wir lernen, das Softwaredesign und die Entwicklung miteinander zu vereinen. Und niemand weiß das besser als Jeff Patton.

Vorwort von Marty Cagan

Ich hatte das große Glück, mit vielen der besten Technologieteams der Welt arbeiten zu dürfen. Mit Menschen, die die Produkte erschaffen haben, die wir jeden Tag benutzen und gern haben. Teams, die im Wortsinn die Welt verändert haben.

Man hat mich auch hinzugezogen, um Firmen zu helfen, bei denen es nicht so gut lief. Startups, die Bodenhaftung bekommen mussten, ehe ihnen das Geld ausging. Größere Firmen, die versuchten, ihre frühen Innovationen zu replizieren. Teams, die es nicht schafften, etwas Wertvolles zum Unternehmen beizutragen. Führungskräfte, die darüber frustriert waren, wie lange es dauerte, bis aus einer Idee Realität wurde. Techniker, die über ihre Produkt-Owner verärgert waren.

Ich habe dabei gelernt, dass es einen wesentlichen Unterschied dazwischen gibt, wie die besten Firmen ihre Technologie-Produkte erschaffen, und wie der Rest es macht. Und damit meine ich nicht kleine Unterschiede. Ich meine alles, angefangen davon, wie sich Führungskräfte verhalten, über das Empowerment von Teams und die Art und Weise, wie Teams zusammenarbeiten, bis hin zur Unternehmenskultur und der Art und Weise, auf die Produktentwicklung, Design und Technik gemeinsam daran arbeiten, effektive Lösungen für ihre Kunden zu erschaffen.

Dieses Buch trägt den Titel *User Story Mapping*, aber ihr werdet schon bald merken, dass es um sehr viel mehr geht als nur diese einfache, aber wirkungsvolle Technik. Dieses Buch reicht bis in den Kern der Frage hinein, wie Teams zusammenarbeiten, kommunizie-

ren und schlussendlich gute Dinge finden, die es sich lohnt zu produzieren.

Viele von euch hatten noch nie die Gelegenheit, aus der Nähe mitzuerleben, wie ein starkes Produktteam arbeitet. Alles, was ihr möglicherweise kennt, ist das, was eure jetzige oder vorige Firma tut. Deswegen möchte ich versuchen, hier einen kleinen Eindruck davon zu vermitteln, wie stark sich die besten Teams vom Rest unterscheiden.

Mit einer dankbaren Verneigung vor Ben Horowitz' *Good Product Manager, Bad Product Manager* habt ihr hier einen flüchtigen Einblick in einige der entscheidenden Unterschiede zwischen starken Produkt-Teams und schwachen Teams:

> Gute Teams haben eine überzeugende Vision ihres Produktes, die sie mit der Leidenschaft eines Missionars verfolgen. Schlechte Teams sind Söldner.
>
> Gute Teams beziehen ihre Inspiration und Produktideen aus Scorecard-KPIs, aus der Beobachtung sich abmühender Kunden, aus der Analyse der Daten, die Kunden bei der Nutzung des Produktes generieren, und aus der gewohnheitsmäßigen Anwendung neuester Technik bei der Problemlösung. Schlechte Teams holen bei Kunden und Sales-Abteilungen Anforderungen ein.
>
> Gute Teams wissen, wer ihre Key Stakeholder sind, sie kennen die Begrenzungen, innerhalb deren diese Stakeholder operieren, und sie sind darauf ausgerichtet, Lösungen zu erschaffen, die nicht bloß für die Kunden und Nutzer funktionieren, sondern auch innerhalb der Beschränkungen des jeweiligen Business. Schlechte Teams holen bei Stakeholdern Anforderungen ein.
>
> Gute Teams sind in den zahlreichen Techniken ausgebildet, mit denen man schnell Produktideen austesten kann, um zu bestimmen, welche es sich wirklich zu entwickeln lohnt. Schlechte Teams halten Meetings ab, um priorisierte Roadmaps zu erzeugen.
>
> Gute Teams brainstormen liebend gern mit klugen Köpfen aus dem gesamten Unternehmen. Schlechte Teams sind beleidigt, wenn jemand von außerhalb ihres Teams es wagt, vorzuschlagen, dass sie irgendetwas tun sollten.
>
> In guten Teams arbeiten Produktentwicklung, Design und Technik Seite an Seite, und sie begrüßen das Geben und Nehmen zwischen Funktionalität, User Experience und der Technik, die diese ermöglicht. Schlechte Teams hocken in ihren jeweiligen Funktionsbereichen und

erwarten, dass die anderen ihre Dienste in Form von Dokumenten und angesetzten Meetings anfordern.

Gute Teams testen konstant neue Ideen aus, um Innovation zu ermöglichen, aber auf eine Weise, die den Unternehmensgewinn und die Marke schützt. Schlechte Teams warten immer noch auf die Erlaubnis, einen Test durchzuführen.

Gute Teams bestehen darauf, dass sie über die nötigen Fähigkeiten verfügen, um erfolgreiche Produkte zu machen, wie zum Beispiel ein starkes Interaktionsdesign. Schlechte Teams wissen nicht mal, was Interaktionsdesigner sind.

Gute Teams sorgen dafür, dass ihre Techniker Zeit haben, die Discovery-Prototypen täglich zu testen, damit sie ihre Einsichten dazu beitragen können, wie das Produkt besser werden kann. Schlechte Teams führen den Technikern die Prototypen während der Sprint-Planung vor, damit sie eine Einschätzung abgeben können.

Gute Teams setzen sich jede Woche direkt mit Endnutzern und Kunden auseinander, um diese besser zu verstehen und die Reaktionen der Kunden auf ihre neuesten ideen zu erleben. Schlechte Teams denken, sie seien der Kunde.

Gute Teams wissen, dass viele ihrer Lieblingsideen am Ende nicht für den Kunden funktionieren werden, und dass selbst die, die es schaffen könnten, mehrere Iterationen brauchen werden, um an den Punkt zu gelangen, an dem sie das gewünschte Ergebnis liefern. Schlechte Teams machen nur das, was auf der Roadmap eingezeichnet ist, und sind zufrieden damit, Termine einzuhalten und die Qualität zu sichern.

Gute Teams begreifen, wie wichtig Geschwindigkeit ist, und dass eine schnelle Iteration der Schlüssel zur Innovation ist. Sie wissen, dass die Geschwindigkeit aus dem Einsatz der richtigen Techniken resultiert, nicht aus Zwangsarbeit. Schlechte Teams beklagen sich, sie seien so langsam, weil ihre Kollegen nicht hart genug arbeiteten.

Gute Teams gehen Verpflichtungen mit hoher Integrität ein, nachdem sie eine Anfrage evaluiert und sichergestellt haben, dass sie eine lebensfähige Lösung anbieten können, die tatsächlich für den Kunden und sein Business funktioniert. Schlechte Teams beklagen sich, ihr Unternehmen sei verkaufsorientiert.

Gute Teams instrumentieren ihre Arbeit, damit sie sofort herausfinden können, wie ihr Produkt benutzt wird, und auf diesen Daten basierende Anpassungen vornehmen können. Schlechte Teams betrachten Analysetools und Reporting als »nice-to-have«.

> Gute Teams integrieren und releasen fortlaufend, denn sie wissen, dass ein gleichbleibender Strom kleinerer Releases für eine erheblich stabilere Lösung für ihren Kunden sorgt. Schlechte Teams testen manuell am Ende einer schmerzlichen Integrationsphase und releasen dann alles auf einmal.
>
> Gute Teams machen sich einen Kopf über ihre Referenzkunden. Schlechte Teams machen sich einen Kopf über ihre Mitbewerber.
>
> Gute Teams feiern, wenn sie signifikante Auswirkungen auf die KPIs ihres Business erzielen. Schlechte Teams feiern, wenn sie endlich irgendwas abliefern.

Mir ist klar, dass ihr euch vielleicht fragt, was das alles mit Story Maps zu tun hat. Ich glaube, ihr wärt überrascht. Und genau deswegen bin ich ein Fan von Story Maps.

Ich habe nur wenige agile Experten getroffen, die ich für qualifiziert halte, einem Produktteam dabei zu helfen, sich so sehr zu verbessern, dass es das leisten kann, was seine Firma braucht und verdient. Jeff Patton ist einer davon. Ich habe erlebt, wie er mitten in der Produkt-Discovery mit Teams in den Schützengraben stieg und die Ärmel hochkrempelte. Ich stelle ihn gern Firmen vor, denn er ist effektiv. Teams mögen ihn, weil er Ahnung hat, aber ein bescheidener Typ ist.

Die Tage, in denen Produktmanager Anforderungen zusammenstellten und dokumentierten, in denen Designer sich abmühen mussten, um auch nur einen Hauch Lippenstift auf den Look auftragen zu können, und in denen sich die Techies im Keller verschanzten und Code tippten, sind bei den besten Teams lange vorbei. Und es ist an der Zeit, dass sie auch in eurem Team vorbei sind.

Über dieses Buch

> Live in it, swim in it, laugh in it, love in it / Removes embarrassing stains from contour sheets, that's right / And it entertains visiting relatives, it turns a sandwich into a banquet.
>
> Tom Waits, »Step Right Up«

Dieses Buch sollte eigentlich nur ganz kurz werden, eine Broschüre quasi.

Ich begann damit, über eine simple Verfahrensweise zu schreiben, die ich *Story Mapping* nannte. Ich erstelle – wie eine Menge anderer Leute auch – einfache Maps, die es erleichtern, mit anderen zusammenzuarbeiten und nachzuempfinden, wie es sich anfühlt, ein Produkt zu benutzen.

Story Mapping sorgt dafür, dass wir uns auf die Benutzer (User) und ihre Erfahrungen (User Experience) konzentrieren. Das Ergebnis ist eine bessere Konversation und schlussendlich ein besseres Produkt.

Das Erstellen einer Map ist ganz einfach. Zusammen mit den anderen Beteiligten erzähle ich die Geschichte (Story) eines Produkts und schreibe jeden wichtigen Schritt, den ein User absolviert, auf Klebezettel, die ich von links nach rechts anordne. Dann schauen wir uns diese Schritte noch einmal komplett an, sprechen über die Details jedes einzelnen Schrittes und notieren diese auf weiteren Klebezetteln, die wir vertikal unter dem jeweiligen Schritt anbringen. Damit erhalten wir eine einfache, rasterähnliche Struktur, die die Story von links nach rechts erzählt, und die damit verbundenen Details von oben nach unten auflistet. Das geht ganz fix und macht Spaß. Mithilfe dieser Details verfügen wir dann auch über ein besseres Backlog der Stories bei unseren agilen Entwicklungsprojekten.

Wie schwierig konnte es wohl sein, ein Buch darüber zu schreiben?

Nun, es stellte sich heraus, dass auch ganz einfache Dinge ausgesprochen kompliziert sein können. Und zu beschreiben, warum man eine Story Map anlegen sollte, was dabei vorgeht und auf wie viele verschiedene Weisen man sie benutzen kann, hat eine ganze Menge Seiten gefressen. Da war doch etwas mehr an dieser simplen Verfahrensweise dran, als ich gedacht hatte.

Wenn ihr einen agilen Entwicklungsprozess einsetzt, füllt ihr vermutlich Backlogs mit User Stories. Ich hatte angenommen, dass es Zeitvergeudung sei, in diesem Buch etwas zu Stories zu schreiben, da sie doch so weit verbreitet sind. Aber damit lag ich falsch. Seit Kent Beck vor rund eineinhalb Jahrzehnten das erste Mal User Stories schrieb, sind sie beliebter denn je – und werden umso häufiger falsch verstanden und eingesetzt. Das macht mich traurig. Und vor allem macht es all die Vorteile zunichte, die wir durch Story Mapping gewinnen könnten.

Deswegen möchte ich in diesem Buch so viele Missverständnisse hinsichtlich Stories und ihrer Benutzung in der agilen und schlanken Softwareentwicklung ausräumen wie möglich. Und das ist der Grund dafür, dass ich, in den Worten von Tom Waits, »aus dem Sandwich ein Buffet gemacht habe«.

Wieso ich?

Ich stelle gern Sachen her. Was mich motiviert, ist, dass es mir Freude macht, Software zu erschaffen und zu sehen, wie Leute sie benutzen und davon profitieren. Ich bin ein wenig zögerlich, wenn es um Methodologie geht. Ich habe festgestellt, dass ich erst lernen

muss, wie Prozesse und Verfahren funktionieren, um sie besser einsetzen zu können. Nach über 20 Jahren in der Softwareentwicklung lerne ich gerade erst, wie ich weitergeben kann, was ich selbst gelernt habe. Und ich weiß, dass das, was ich anderen beibringe, eine bewegliche Größe ist. Was ich verstehe, verändert sich jede Woche wieder. Wie man es am besten erklärt, verändert sich nahezu ebenso schnell. All das hat mich jahrelang davon abgehalten, ein Buch zu schreiben.

Aber die Zeit ist reif dafür.

Stories und Story Maps sind eine unglaublich gute Idee, von der schon sehr viele Leute profitiert haben. Sie haben das Leben der Entwickler besser gemacht, und ebenso die Produkte, die diese entwickelt haben. Aber während sie manchen zu einem besseren Dasein verholfen haben, ringen auch immer mehr Leute mit Problemen, die sie mit Stories haben. Ich möchte dazu beitragen, dass das ein Ende hat.

Dieses Buch ist etwas, das ich beitragen kann, um zu helfen. Und wenn es auch nur das Arbeitsleben einer Handvoll Leute verbessert, habe ich einen Grund zu feiern.

Wenn du Schwierigkeiten mit Stories hast, ist dieses Buch für genau Dich

Eine Menge Firmen und Organisationen benutzen heutzutage agile und schlanke (»lean«) Prozesse, und damit auch Stories. Dabei passiert es schnell, dass man in eine oder mehrere Fallen tappt, sofern man falsche Vorstellungen von Stories hat. Und zwar Fallen wie diese:

- Stories ermöglichen es uns, die Aufmerksamkeit auf das Erstellen kleiner Teile des Ganzen zu lenken und dabei den großen Gesamtzusammenhang aus dem Auge zu verlieren. Das Ergebnis ist nicht selten ein »Frankenstein-Produkt«, bei dem jedem, der es benutzt, klar wird, dass es aus nicht zueinander passenden Teilen zusammengestückelt ist.
- Beim Erstellen eines umfangreichen Produktes werden viele kleine Bestandteile produziert, so dass die Beteiligten sich fragen, *wann das Ganze jemals fertig wird bzw. was genau dabei eigentlich herauskommen wird*. Wenn du der Ersteller bist, fragst du dich das auch.

- Da Stories sich um den Austausch (Conversations) drehen, benutzen manche diesen Ansatz, um nichts aufzuschreiben. Danach vergessen sie, worüber sie sich ausgetauscht und worauf sie sich geeinigt haben.
- Da gute Stories Akzeptanz-Kriterien haben sollten, bemüht man sich, solche Kriterien festzuhalten, aber es gibt kein gemeinsames Verständnis darüber, was eigentlich hergestellt werden soll. *In der Folge können Teams ihre Arbeiten nicht im vorher geplanten Zeitrahmen abschließen*.
- Gute Stories sollten aus der User-Perspektive formuliert sein. Da es aber eine Menge Teile gibt, die der User nie sehen wird, argumentieren Teammitglieder: »*Unser Produkt hat keine User, deswegen greifen User Stories hier nicht.*«

Wenn ihr schon mal in eine dieser Fallen gestolpert seid, werde ich versuchen, die Fehlannahmen zu beseitigen, die dazu geführt haben. Ihr werdet lernen, wie man im Gesamtzusammenhang denkt (das »Big Picture«), wie man im Großen wie im Kleinen plant und schätzt, wie man produktive Konversationen darüber führt, was User zu erreichen versuchen, und was eine gute Software tun kann, um ihnen dabei zu helfen.

Wer sollte dieses Buch lesen?

Du natürlich. Ganz besonders, wenn du es gekauft hast. Ich persönlich denke, das war eine weise Investition. Wenn du es dir nur ausgeliehen hast, bestell dir gleich dein eigenes Exemplar und gib dieses hier zurück, sobald dein Buch ankommt.

Tatsächlich ist die Lektüre dieses Buches aus verschiedenen Gründen besonders vorteilhaft für Leute mit ganz spezifischen Aufgaben:

- *Produktmanager und User-Experience- (UX-) Spezialisten, die für kommerzielle Softwarehersteller tätig sind*, sollten dieses Buch lesen, damit sie in die Lage versetzt werden, die Kluft zu überbrücken zwischen vollständigen Produkten und User Experience auf der einen und dem Nachdenken über taktische Pläne und Punkte aus dem Backlog auf der anderen Seite. Wenn ihr es schwierig findet, von der Vision, die euch antreibt, eine Brücke zu den Details zu schlagen, die euer Team herstellen kann, können euch Story Maps dabei helfen. Wenn ihr Schwierigkeiten habt, anderen dabei zu helfen, sich in die Erfahrungen der User hineinzudenken und diese mitzufühlen, kann Story

Mapping euch helfen. Wenn ihr keinen Plan habt, wie ihr gute UX und gute Produktdesign-Practices einbinden könnt, ist dieses Buch eine Hilfestellung. Wenn ihr daran arbeitet, Lean-Startup-Experimente bei der Arbeit einzusetzen, wird dieses Buch euch dabei helfen.

- *Produkt-Owner, Business-Analysten und Projektmanager in IT-Organisationen* sollten dieses Buch lesen, um zwischen den internen Usern, Interessengruppen (Stakeholdern) und Entwicklern vermitteln zu können. Wenn ihr viele unterschiedliche Interessengruppen in der Organisation unter einen Hut bringen müsst, helfen euch Story Maps dabei. Benutzt die Stories, einfachen Übungen und Verfahrensweisen (»Practices«) aus diesem Buch, um eure Teams dabei zu unterstützen, besser zu werden.
- *Agile und Lean Process Trainer (Coaches)*, die Einzelpersonen und Teams helfen wollen, besser zu arbeiten, sollten dieses Buch lesen. Und wenn ihr das tut, dann macht euch bewusst, welche Fehlannahmen hinsichtlich Stories in eurer Organisation kursieren. Benutzt die Stories, einfachen Übungen und Verfahrensweisen (»Practices«) aus diesem Buch, um eure Teams dabei zu unterstützen, besser zu werden.
- *Alle anderen*. In der agilen Entwicklung schauen wir oft auf Rollen wie Produkt-Owner oder Business-Analysten, die einen Großteil der Arbeit mit Stories lenken sollen, aber ein wirkungsvoller Einsatz von Stories setzt voraus, dass *alle* die Grundlagen kapieren. Wenn die Leute die Grundzüge nicht verstehen, kommen schnell Beschwerden auf wie »die Stories sind nicht gut geschrieben« oder »sind zu umfangreich« oder »haben nicht genug Details«. Dieses Buch wird auch dazu Hilfestellungen geben, aber nicht unbedingt so, wie ihr denkt. Ihr und alle anderen werden erkennen, dass Stories keine Methode sind, mit der man bessere Anforderungskataloge schreibt, sondern eine Methode, um sich besser zu organisieren und auszutauschen. Mit diesem Buch lernt ihr zu verstehen, welche Art Konversationen ihr führen solltet, damit ihr die Informationen, die ihr braucht, dann bekommt, wenn ihr sie benötigt.

Ich hoffe, ihr könnt euch mit einer oder mehreren der beschriebenen Gruppen identifizieren. Falls nicht, reicht dieses Buch an jemanden weiter, der das kann.

Falls ja, lasst uns anfangen.

Ein paar Konventionen in diesem Buch

Ich gehe davon aus, dass dies nicht das erste Buch zu Softwareentwicklung ist, das ihr in eurem Leben in Händen haltet, von daher sollte euch nichts überraschen.

Die Überschriften in den einzelnen Kapiteln weisen euch den Weg durch das Thema.

Benutzt sie, um euch zurechtzufinden, oder überspringt das, was euch derzeit nicht interessiert.

Kernaussagen sehen so aus. Stellt euch vor, dass ich das hier ein wenig lauter als den anderen Text sage.

Wenn ihr nur durchblättert, lest die Kernaussagen. Wenn sie euch ansprechen oder nicht absolut selbsterklärend sind, lest den Text davor und dahinter. Das sollte sie verständlich machen.

Die Inhalte der Kästen beschreiben...

- *Konzepte, die interessant, aber nicht kritisch für das Verständnis sind.* Es handelt sich um unterhaltsame Ablenkungen, zumindest hoffe ich das.
- *Rezepte für bestimmte Verfahrensweisen (Practices).* Die Rezepte sollen euch helfen, in diese Practices hineinzufinden.
- *Stories und Beispiele von anderen*. Hier findet ihr ein paar gute Ideen, die ihr in eurer Organisation ausprobieren könnt.

Dieses Buch ist in spezifische Abschnitte unterteilt. Ihr könnt es abschnittsweise lesen oder die Abschnitte verwenden, um Ideen für eine spezifische Herausforderung zu finden, mit der ihr gerade zu tun habt.

Wie dieses Buch aufgebaut ist

Vor einiger Zeit habe ich einen coolen neuen Farblaserdrucker gekauft. Ich öffnete den Karton und mir leuchtete oben auf dem Drucker eine Broschüre entgegen, auf der in großen roten Buchstaben »Read This First« prangte. »Soll ich das *wirklich* lesen?«, fragte ich mich, denn ich befolge ungern Anweisungen. Aber ich bin

froh, dass ich es dann doch gemacht habe, denn es war eine ganze Menge Plastik-Schutzteile an verschiedenen Stellen im Drucker angebracht, die für einen sicheren Transport gesorgt hatten – und wenn ich den Drucker mit diesen Teilen in Betrieb genommen hätte, hätte er kaputt gehen können.

Das klingt jetzt vielleicht ein bisschen weit vom Thema ab, ist es aber nicht.

Dieses Buch enthält ein »Read This First«-Kapitel, das zwei entscheidende Konzepte und das zugehörige Vokabular erläutert, die ich im gesamten restlichen Buch benutzen werde. Ich möchte, dass ihr diese Konzepte verinnerlicht habt, bevor ihr loslegt. Wenn ihr mit Story Mapping beginnt, ohne sie verstanden zu haben, kann ich nicht für eure Sicherheit garantieren.

Story Mapping aus 3.000 Metern Höhe

Die Kapitel 1–4 geben euch eine Draufsicht auf das Thema Story Mapping. Wenn ihr mit Stories schon ein Weilchen arbeitet und schon mal mit einer Story Map gespielt habt, sollte euch dieser Abschnitt die Grundlagen vermitteln, um gleich loslegen zu können.

In Kapitel 5 findet ihr eine ausgetüftelte Übung, mit der ihr die Schlüsselkonzepte erlernt, die ihr braucht, um eine großartige Story Map zu erstellen. Probiert es mit einer Gruppe in eurem Büro aus – jeder, der mitmacht, wird es kapieren. Und ich verspreche, dass die Maps, die ihr danach für eure Produkte erstellt, erheblich besser sein werden.

User Stories verinnerlichen

Die Kapitel 6–12 erzählen die Geschichten hinter den Stories, wie Stories wirklich funktionieren und wie man sie gekonnt in agilen und schlanken Projekten einsetzt. Innerhalb von Story Maps gibt es viele kleine Stories, die man nutzen kann, um seine tagtägliche Entwicklungsarbeit voranzutreiben. Selbst den Veteranen des agilen Development unter euch kann ich versprechen: Ihr könnt noch etwas über Stories lernen, das ihr bislang nicht wusstet. Und wenn Stories neu für euch sind, findet ihr hier genug Fachkenntnisse, um auch die agilen (Alles-)Besserwisser in eurem Büro noch zu überraschen.

Bessere Backlogs

In Kapitel 13–15 tauchen wir tief in den Lebenszyklus einer Story ein. Ich werde spezifische Verfahrensweisen vorstellen, die euch dabei helfen, Stories und Story Maps einzusetzen, angefangen bei den ganz großen Gelegenheiten, und euch die Entdeckerarbeit zeigen, die euch hilft, ein Backlog voller Stories zu identifizieren, die ein umsetzbares Produkt beschreiben. Ihr erfahrt, wie Story Maps und eine Menge andere Practices euch auf jedem Schritt dieses Weges unterstützen.

Bessere Builds

In Kapitel 16–18 geht es um die taktische Anwendung von Stories, Iteration um Iteration oder Sprint um Sprint. Ihr werdet erfahren, wie man sich seine Stories bereitlegt, ihnen Aufmerksamkeit widmet, während sie im Build sind, wie man sie wirklich abarbeitet und wie man aus jeder einzelnen Story etwas lernt, das man in funktionierende Software umsetzt.

Ich habe festgestellt, dass die letzten Kapitel vieler Bücher zum Thema Softwareentwicklung nur Schrott in Form von Extras enthalten. Meistens kann ich sie ignorieren. Blöderweise habe ich keine solchen Kapitel geschrieben. Ihr müsst schon das ganze Buch lesen. Mein einziger Trost für euch ist, dass ihr aus jedem Kapitel ein paar nützliche Kleinigkeiten mitnehmen werdet, die ihr sofort umsetzen könnt.

Also, los geht's.

Hier geht's los

Dieses Buch hat keine Einleitung.

Ja, ihr habt richtig gelesen. Jetzt fragt ihr euch eventuell: »Wieso hat Jeffs Buch keine Einleitung? Hat er vergessen, eine zu schreiben? Wird er nachlässig nach all den Jahren? Hat der Hund sie gefressen?«

Nein, ich habe nicht vergessen, eine Einleitung zu diesem Buch zu schreiben. Und nein, ich lasse nicht geistig nach, zumindest glaube ich das nicht. Und mein Hund hat sie auch nicht gefressen (obwohl das Meerschweinchen meiner Tochter ziemlich verdächtig dreinschaut). Ich finde bloß schon seit Langem, dass Autoren zu viel Zeit auf das Bemühen verwenden, mich zu überzeugen, dass ich ihr Buch lesen sollte, und ein Großteil dieser Überzeugungsarbeit findet in der Einleitung statt. Mit dem eigentlichen Thema des Buches geht es meist erst in Kapitel 3 los. Und ich bin sicher nicht der Einzige, der gewohnheitsmäßig die Einleitung überblättert.

Dieses Buch beginnt tatsächlich hier.

Und ihr dürft diesen Teil nicht überblättern, denn es ist der wichtigste Teil. Tatsächlich wäre ich schon glücklich, wenn ihr bloß zwei Dinge aus diesem Buch mitnehmt. Und diese beiden Punkte findet ihr genau hier, in diesem Kapitel:

- Die Benutzung von Stories hat nicht das Schreiben besserer Stories zum Ziel.
- Produktentwicklung hat nicht die Herstellung von Produkten zum Ziel.

Lasst es mich erklären.

Stille Post

Vermutlich erinnert ihr euch aus Kinderzeiten an dieses Spiel: Stille Post. Man flüstert jemandem etwas ins Ohr, der es der nächsten Person weiterflüstert und so weiter, durch die ganze Runde, bis die letzte Person in der Gruppe laut sagt, was sie verstanden hat, und alle in Gelächter ausbrechen. Wir spielen dieses Spiel immer noch bei uns zu Hause, wenn meine Kinder am Esstisch sitzen. Tipp für Eltern: Das ist eine gute Methode, um Kinder bei Tisch bei Laune zu halten, die von der Unterhaltung der Erwachsenen angeödet sind.

In der Welt der Erwachsenen spielen wir das Spiel auch, allerdings flüstern wir nicht miteinander. Wir schreiben ellenlange Texte und produzieren offiziell aussehende Präsentationen, die wir irgendwem in die Hand drücken, der daraus dann irgendwas vollkommen anderes herausliest, als wir beabsichtigt hatten. Und diese Person erschafft aus diesem Dokument eine Reihe weiterer Dokumente, die sie an andere Personen weitergibt. Nur dass wir im Gegensatz zu dem Kinderspiel nicht alle am Ende lachen.

Wenn man Menschen schriftliche Anweisungen gibt, interpretieren sie diese unterschiedlich. Wenn ihr das nur schwer glauben könnt (es ist schließlich alles niedergeschrieben, nicht wahr?), habe ich hier für euch ein paar Beispiele dafür, wie Anweisungen furchtbar falsch verstanden werden können.