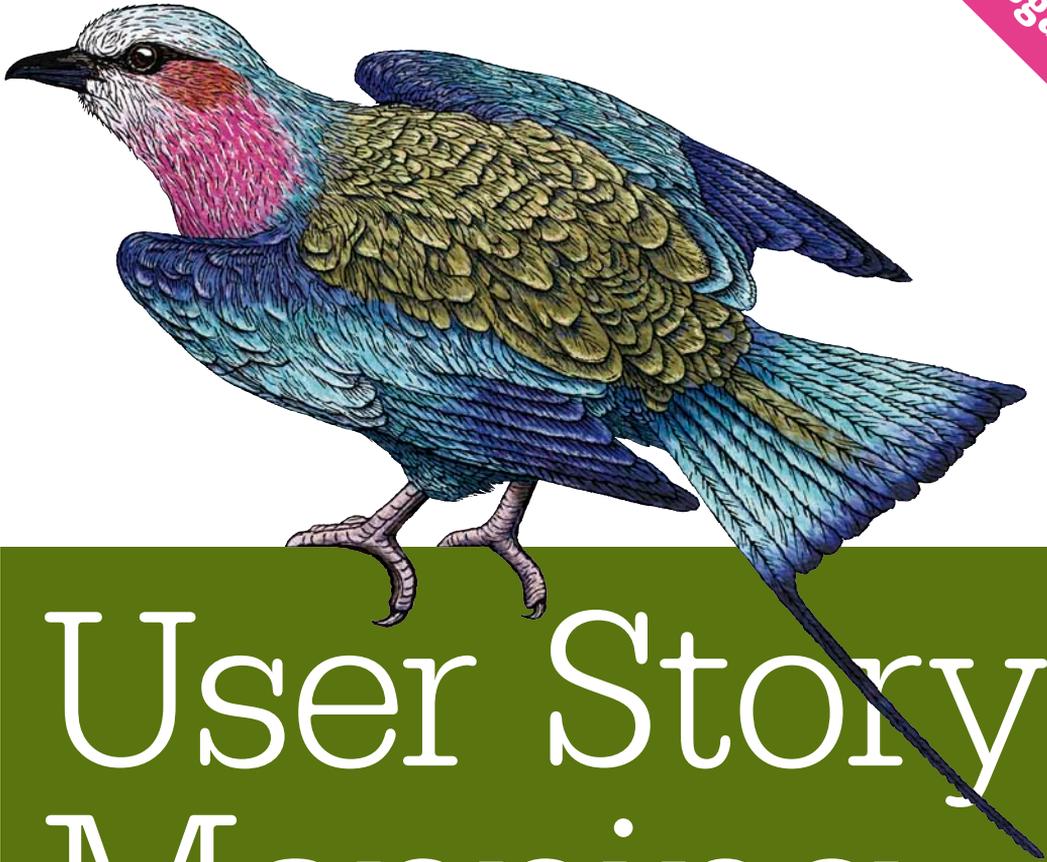


O'REILLY®

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

Die Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Verlag, Autoren und Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen. Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen. Der Verlag richtet sich im Wesentlichen nach den Schreibweisen der Hersteller. Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Alle Rechte vorbehalten einschließlich der Vervielfältigung, Übersetzung, Mikroverfilmung sowie Einspeicherung und Verarbeitung in elektronischen Systemen.

Kommentare und Fragen können Sie gerne an uns richten:

O'Reilly Verlag
Balthasarstr. 81
50670 Köln
E-Mail: kommentar@oreilly.de

Copyright:

© 2015 O'Reilly Verlag GmbH & Co. KG
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
Korrektur: 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

Inhalt	V
Widmung	XI
Vorwort	XIII
Über dieses Buch	XXI
Hier geht's los	XXIX
1 Das große Ganze	1
Das Wort mit »A«	1
Stories erzählen, nicht Geschichten schreiben	3
Die ganze Geschichte erzählen	4
Gary und die Tragödie des flachen Backlogs	5
Talk and Doc	7
Umreißt eure Idee	9
Beschreibt eure Kunden und User	10
Erzähl die Stories deiner User	11
Details und Möglichkeiten erforschen	15
2 Planen, (um) weniger zu produzieren	23
Mapping hilft großen Gruppen, gemeinsames Verständnis herzustellen.	24
Mapping hilft euch, die Löcher in eurer Geschichte zu entdecken	28
Es ist immer zu viel (zu tun)	29

Definiert das Minimum für ein Minimum Viable Product Release	30
Definiert eine Release Roadmap	31
Priorisiert Outcomes, nicht Features	33
Es ist Magie – wirklich.	33
Die Sache mit dem MVP	37
Das neue MVP ist gar kein Produkt!	39
3 Planen, (um) schneller zu lernen	41
Diskutiert die Chancen	42
Validiert das Problem	43
Benutzt Prototypen, um etwas zu lernen	44
Was User Wollen	45
Lernt aus euren Builds	46
Iteriert bis zum MVP	49
Wie man es falsch macht	50
Validiertes Lernen	51
Minimiert eure Experimente. Wirklich.	53
Zusammenfassung	54
4 Planen, (um) rechtzeitig fertig zu werden	55
Redet mit dem Team	57
Die Kunst des gekonnten Schätzens	58
Plant, Stück für Stück zu produzieren	59
Macht nicht aus jedem Slice ein Release	60
Das andere Geheimnis gekonnter Schätzungen	61
Managt euer Budget	62
Iterativ UND inkrementell	67
Strategien: Eröffnungs-, Mittel- und Endspiel	68
Markiert eure Entwicklungsstrategie in einer Map.	69
Es geht um das Risiko.	70
Wie geht es weiter?	71
5 Ihr wisst schon, wie es geht	73
1. Schreibt eure Story auf – einen Schritt nach dem anderen	73
2. Organisiert eure Story	78
3. Entdeckt alternative Stories	79
4. Komprimiert die Map und erzeugt einen Backbone	81

5. Gruppierd Tasks, die euch helfen, einen bestimmten Outcome zu erzielen	83
Fertig! Ihr habt alle wichtigen Konzepte gelernt!	85
Do Try This at Home (oder bei der Arbeit)	85
Die Map dreht sich ums Jetzt, nicht ums Später	87
Probiert es wirklich aus	89
Mit Software ist es schwieriger	90
Die Map ist nur der Anfang	92
6 Die wahre Geschichte der Stories	97
Kents verstörend einfache Idee	97
Einfach ist nicht leicht	99
Ron Jeffries und die 3 Cs	101
Worte und Bilder	103
Das war's schon.	105
7 Bessere Stories erzählen	107
Connextras Tolles Template	107
Template-Zombies und der Schneepflug	112
Checkliste: Worüber ihr euch wirklich unterhalten solltet . .	115
Macht Urlaubsfotos	118
Das ist eine Menge Zeug	119
8 Nicht alles steht auf der Karte	121
Unterschiedliche Leute, unterschiedliche Konversationen . .	122
Wir brauchen eine größere Karteikarte	123
Strahler und Kühltruhen	126
Dafür ist das Werkzeug nicht gedacht	129
9 Die Karteikarte ist nur der Anfang	135
Habt eine klare Vorstellung davon, was ihr konstruiert	136
Entwickelt eine mündliche Tradition des Geschichtenerzählens	137
Inspiziert das Ergebnis eurer Arbeit	138
Es geht nicht um Euch	140
Entwickelt, um zu lernen.	141
Es ist nicht immer Software	142
Plant, zu lernen, und lernt, zu planen	143

10 Wir backen uns eine Story	145
Ein Rezept kreieren	146
Den großen Kuchen aufteilen	147
11 Steine brechen	153
Auf die Größe kommt es immer an	153
Stories sind wie Steine	155
Epen sind große Steine, die manchmal benutzt werden, um Menschen damit zu schlagen	157
Themen organisieren Story-Gruppen	158
Vergesst diese Begriffe und konzentriert euch darauf, Stories zu erzählen	159
Beginnt mit Chancen (Opportunities)	160
Entdeckt eine Minimum Viable Solution	161
Vertieft euch in die Details jeder einzelnen Story im Delivery-Prozess	163
Redet weiter, während ihr produziert	165
Evaluert jedes Stück	166
Evaluert mit Usern und Kunden	167
Evaluert mit Business-Stakeholdern	169
Evaluert auch nach dem Release weiter	170
12 Steinebrecher	173
Wertvoll – benutzbar – realisierbar	174
Ein Discovery-Team benötigt für den Erfolg viele andere Personen	177
Die drei Amigos	178
Produkt-Owner als Produzenten	182
Es ist kompliziert	183
13 Beginnt mit Chancen	185
Führt Konversationen über Chancen	185
Tiefer graben, wegwerfen oder darüber nachdenken	187
Chance sollte kein Euphemismus sein	192
Story Mapping und Chancen (Opportunities)	192
Seid wählerisch	199

14 Mit Discovery gemeinsames Verständnis aufbauen . . .	201
Bei Discovery geht es nicht um das Schreiben von Software	201
Vier essenzielle Schritte der Discovery	203
Discovery-Aktivitäten, Diskussionen und Artefakte	220
Discovery dient der Herstellung von gemeinsamem Verständnis.	221
15 User-Discovery für validiertes Lernen	223
Wir liegen meistens falsch	223
Die schlechte alte Zeit	225
Einfühlen, Fokussieren, Ideen Sammeln, Prototypen Bauen, Testen	226
Wie man etwas Gutes versaut	230
Kurze Zyklen validierten Lernens	232
Wie Lean Startup Thinking das Produktdesign verändert . .	233
Stories und Story Maps?	239
16 Verfeinern, Definieren, Produzieren	241
Karteikarten, Konversationen, mehr Karten, noch mehr Konversationen	241
Schneiden und Polieren	242
Der Story-Workshop	243
Sprint- oder Iterationsplanung?	246
Menschenmengen kollaborieren nicht	250
Aufteilen und Ausdünnen	251
Benutzt eure Story Map während der Delivery	257
Benutzt eine Map, um Fortschritte zu visualisieren	258
Benutzt einfache Story Maps in Story-Workshops	259
17 Stories sind genau genommen wie Asteroiden	265
Zerbrochene Steine wieder zusammenfügen	267
Übertreibt es nicht mit dem Mapping	269
Zerbrecht euch nicht den Kopf über Kleinkram	270
18 Lernt aus jedem Build	273
Review im Team	273
Review mit anderen aus eurem Unternehmen	277
Genug	279

Lernt von Usern	281
Lernt aus euren Releases	281
Outcomes nach Zeitplan	282
Benutzt eine Map, um zu evaluieren, ob ihr bereit für den Release seid	283
Das Ende. Oder?	285
Danksagung	287
Literatur	291
Index	293

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 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 Schlüsselmethod. 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,