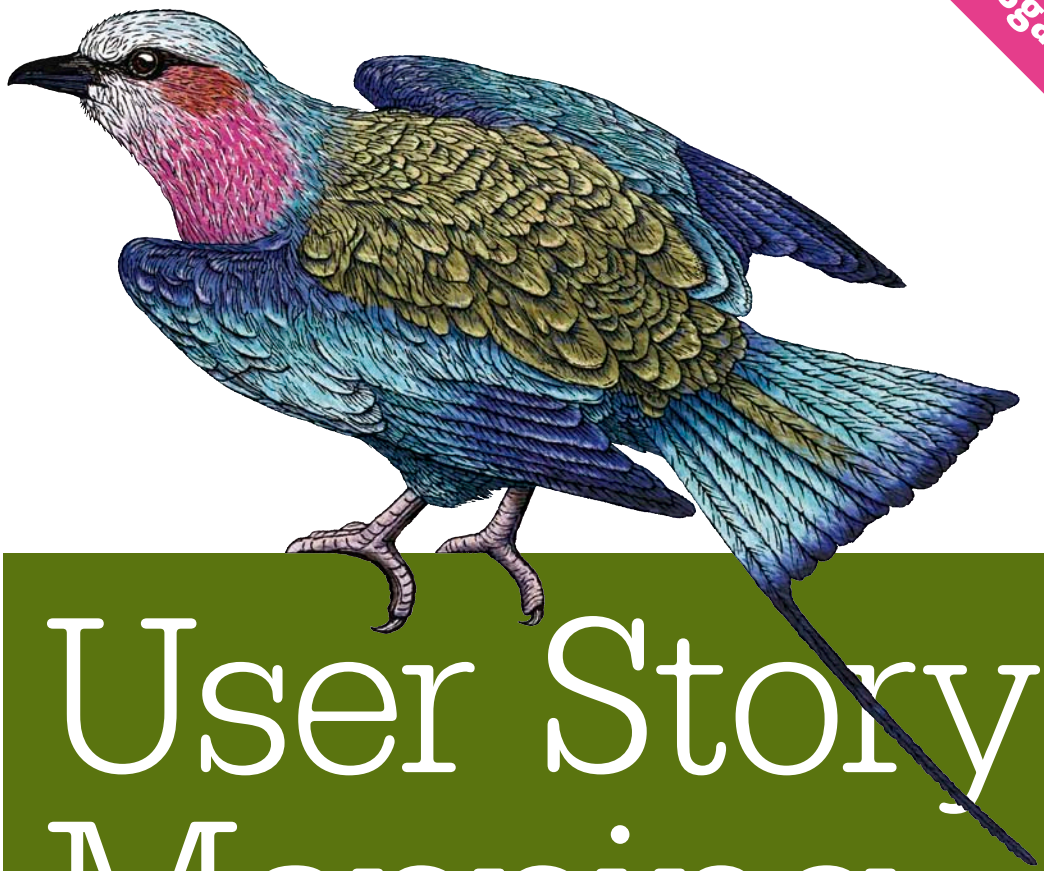
O'REILLY®

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,