12 Agiles Requirements Engineering mit User Stories

Agile Methoden gehen viele Dinge in der Softwareentwicklung anders an: Es wird mehr direkt miteinander geredet und weniger Dokumente und Tickets hin und her geschoben. Es geht mehr um fertige Software, die dem Anwender tatsächlich nützt, als um die Einhaltung von definierten Vorgaben. Es geht mehr um ein gemeinsames Erarbeiten der Spezifikation und ein gemeinsames Hochziehen des Produktes als um das sture Abarbeiten von Aufträgen. Änderungen werden jederzeit positiv behandelt, wenn sie das Produkt besser machen, und es wird nicht eisern an einem Plan festgehalten. Agile Methoden stellen also Zusammenarbeit und Flexibilität deutlich in den Vordergrund.

Workshops unterstützen diese Werte perfekt! Workshops sind genau dazu da, um gemeinsam im Team strukturiert, effizient und trotzdem flexibel Ergebnisse zu erarbeiten. Entsprechend häufig kommen sie auch zum Einsatz und viele der bisher beschriebenen Methoden können in einem agilen Umfeld sofort eingesetzt werden: Jedes agile Team braucht eine Vision für sein Produkt, ein Big Picture, auf das es hinarbeiten kann. Diese Produktvision kann natürlich in Form eines Elevator Pitch erarbeitet oder als Produktbox dargestellt werden. In einem gemeinsamen Workshop erschafft das Team Personas für die wichtigsten Anwendergruppen. So lernt das Team in kurzer Zeit viel über die Anwender und bekommt als positiven Nebeneffekt noch eine Teambuilding-Session dazu. SMARTe Ziele ermöglichen es dem agilen Team, jederzeit zu prüfen, ob es noch auf dem richtigen Weg ist. Evidenzbasiertes Arbeiten, also das ständige Prüfen und Messen, ist ebenfalls eine Grundlage vieler agiler Methoden. Ein gutes agiles Team hat immer die Risiken im Blick und verfolgt diese aktiv über ein Risiko-Backlog. Gemeinsam arbeitet sich das Team in die Fachprozesse der Anwender ein und entwickelt Schritt für Schritt ein System, das die Anwender in ihrer täglichen Arbeit optimal dabei unterstützt, ihre Ziele zu erreichen.

Viele agile Teams gehen dabei stark iterativ und inkrementell vor. Sie verschaffen sich erst einmal einen Überblick, was alles zu tun ist, und füllen das

Backlog mit Dingen, die den Anwendern helfen. Je nützlicher dabei etwas ist, desto früher kommt es dran. Jedes dieser Backlog Items beschreibt in Form einer kurzen Geschichte, wie der Anwender ein neu zu schaffendes Feature nutzt, um eine seiner Aufgaben besser zu lösen. Für diese kleinen Geschichten hat sich der Name »User Story« eingebürgert. Zu jeder Story notiert sich das Team, wer was wozu benötigt. Viele Teams schreiben dies in der bereits in Abschnitt 11.2 vorgestellten Form nieder: »*Als <Rolle> möchte ich <Aktion>, damit <Nutzen>*« [Cohn, 2004]. Dies ist aber nur eine Möglichkeit, dies zu dokumentieren. Jedes Team kann selbst entscheiden, wie es seine Anforderungen am besten beschreibt, Hauptsache es wird klar, wer was und vor allem wozu braucht.

Für jede Anforderung notiert sich das Team, was für den Anwender wichtig ist, sodass sie tatsächlich Nutzen bringt. Dabei geht es nicht darum, die gewünschte Lösung möglichst detailliert zu beschreiben, sondern darum, die Anforderung des Kunden zu verstehen und Kriterien zu finden, anhand derer das Team erkennen kann, ob eine bestimmte Lösung tatsächlich gut und ausreichend ist oder nicht. Da diese Kriterien dazu dienen, festzuhalten, wann der Anwender eine Lösung akzeptiert, nennt man sie auch »Akzeptanzkriterien«.

Die Anforderungen im Backlog werden dann eine nach der anderen umgesetzt. Sobald etwas fertig ist, zeigt das Team es sofort den Anwendern und holt Feedback ein. Da jede Story einen kleinen Nutzen für die Anwender beinhaltet, möchten diese meist die neue Produktversion sofort einsetzen. Das ist ein wesentlicher Aspekt agiler Methoden. Produkte werden nicht erst ausgeliefert und eingesetzt, wenn nach vielen Monaten der Entwicklung alles fertig ist, sondern sobald ein Produkt das Leben der Anwender leichter macht.

Grundprinzipien des agilen Requirements Engineering

Ein agiles Team sollte im Requirements Engineering diesen Prinzipien folgen:

- **Wir erarbeiten Anforderungen gemeinsam**, statt dass der Product Owner sie der Entwicklung als Auftrag über den Zaun wirft.
- **Wir reden miteinander**, statt Ticket-Ping-Pong zu betreiben.
- **Wir beschreiben Anforderungen aus Sicht unserer Kunden**, statt technische Funktionen zu spezifizieren.
- **Wir dokumentieren Anforderungen flexibel, wie wir es brauchen**, statt schwergewichtige Dokumentvorlagen zu produzieren.
- **Wir spezifizieren rollierend immer die nächstwichtigsten Sachen**, statt Big-Design-Upfront durchzuführen.
- **Wir ändern alles jederzeit, wenn es das Produkt besser macht**, statt ein dickes Spezifikationsdokument stur umzusetzen.
- **Wir sind fertig, sobald der Nutzen erreicht ist**, nicht wenn der Plan vollständig abgearbeitet wurde.

- **Sobald etwas** nützt, können unsere Kunden es einsetzen, statt auf große Releases zu warten und zu zittern.
- **Wir holen uns während der Umsetzung permanent Feedback**, statt erst im Betrieb Bugs und Change Requests zu erhalten.

Rollen im agilen Requirements Engineering

Im agilen Requirements Engineering ist Zusammenarbeit sehr wichtig. Anforderungen werden in unterschiedlichen Workshops erarbeitet und verfeinert. Dabei sind je nach konkretem Vorgehensmodell verschiedene Rollen beteiligt:

- **Product Owner**
  Dem Product Owner (PO) gehört das Backlog. Er ist dafür verantwortlich, dass das Produkt die Erwartungen der Stakeholder erfüllt. Requirements Engineering ist dementsprechend eine seiner Hauptaufgaben. Der Product Owner definiert die Produktvision, leitet daraus konkrete Ziele für das Produkt ab, erarbeitet die Anforderungen, priorisiert diese und gibt Feedback zur Umsetzung. Er macht das natürlich nicht alles alleine, aber ist dafür verantwortlich, dass es passiert.
- **Team**
  Das agile Entwicklungsteam unterstützt den Product Owner beim Erkunden und Verfeinern der Anforderungen und entwirft Lösungen dafür. Das Team ist in ständigem Kontakt mit dem PO, gibt Feedback zu Anforderungen, liefert Lösungsideen, schätzt Aufwände, setzt die Anforderungen um und zeigt sie dem Kunden, um Feedback einzuholen.
- **Agile Master**
  Viele agile Methoden sehen eine coachende und begleitende Rolle vor. Der Agile Master sorgt dafür, dass das Team optimal arbeiten und sich ganz auf die Produktentwicklung konzentrieren kann. Vielfach übernimmt der Agile Master dafür die Rolle des Moderators in Workshops. Auch im Requirements Engineering kann er den Product Owner und das Team auf diese Weise unterstützen und Workshops vorbereiten und moderieren. PO und Team können sich dadurch auf die inhaltliche Arbeit fokussieren, den Ablauf steuert der Agile Master.

Diese Aufgabenteilung auf die einzelnen Rollen ist nur eine grobe Richtschnur. In der Praxis ist alles erlaubt, was funktioniert, Hauptsache das Team erstellt ein tolles Produkt. Das Fundament für jede Entwicklung ist aber auch in agilen Teams ein professionelles Requirements Engineering. Nur mit dem richtigen Verständnis, was die Anwender tatsächlich brauchen und welche Stories für sie umgesetzt werden müssen, kommt etwas heraus, das von der ersten Version an die Kunden begeistert.

12.1 User Stories mit einer Impact Map erarbeiten

Impact Mapping ist eine einfache Methode, wie man Kandidaten für User Stories finden kann. Die Grundidee ist, den Fokus erst einmal weg von der Software, hin auf den Anwender und seine Ziele zu richten [Adzic, 2012].

Zu jedem Ziel stellt man sich die Frage: »Wer kann dabei helfen, dieses Ziel zu erreichen?« Für jede der so gefundenen Personen oder Rollen überlegt man, was sie tun kann, um dem Ziel näher zu kommen. Dabei wird man auch viele Dinge finden, die nichts mit der zu bauenden Software zu tun haben. Vielfach können Ziele besser und effizienter durch organisatorische Änderungen, Anpassungen der Geschäftsprozesse oder kleine Changes in bereits vorhandenen Tools erreicht werden. Eine Impact Map ermöglicht es dem Team, auch diese Punkte zu finden und dem Kunden vorzuschlagen. Nur wenn man keine andere, einfachere und kostengünstigere Lösung findet, bauen wir wirklich Software.

Haben wir herausgefunden, was die relevanten Personen und Rollen tun können, um den Zielen näher zu kommen, können wir überlegen, mit welchen Funktionen der zu schaffenden Software wir sie dabei unterstützen können. Das sind dann Kandidaten für User Stories.

Als letzten Schritt im Impact Mapping definiert man Metriken, anhand derer die Zielerreichung gemessen werden kann. Nach jedem umgesetzten und zum Kunden ausgelieferten Feature wird gemessen, ob das Ziel schon erreicht ist. Wenn ja, ist man mit der Umsetzung fertig und eventuell noch weitere geplante Features können gestrichen werden. Falls nicht, setzt man so lange weitere Features um, bis das Ziel erreicht ist.

Beispiel für eine Impact Map für die Bogenturnier-Anwendung

Das Beispiel zeigt, wie für die Bogenturnier-Anwendung Kandidaten für User Stories mithilfe einer Impact Map aus den Zielen erarbeitet werden können:

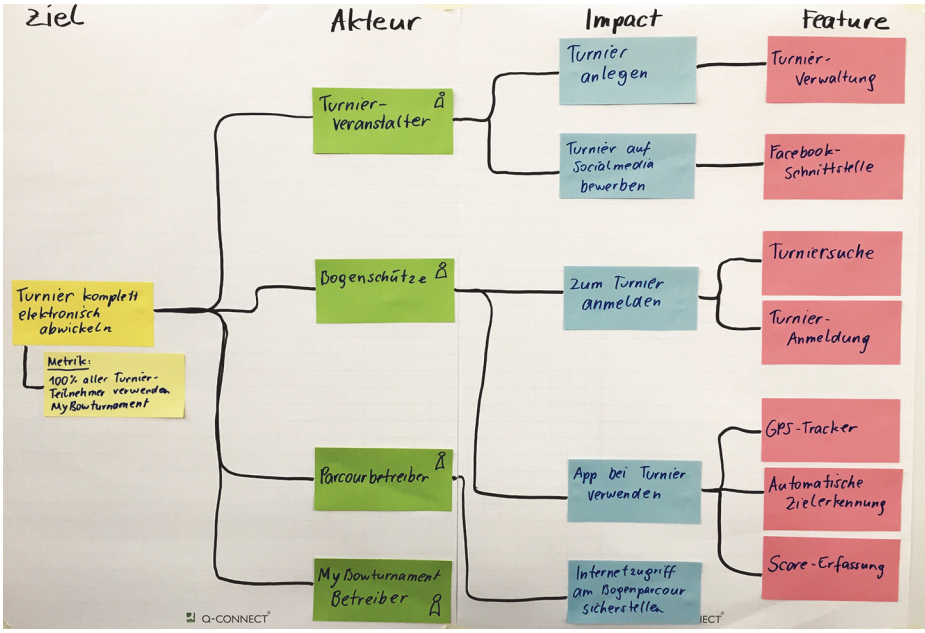

Abb. 12–1 *Die Impact Map zeigt, wer bei der Erreichung der Ziele wie helfen kann und was dafür in der neuen Software gebaut werden muss.*

Methode	Impact Mapping
Teilnehmer	■ Product Owner und agiles Team ■ Unbedingt teilnehmen sollten Vertreter der zukünftigen Anwender ■ Evtl. Experten für beteiligte Fremdsysteme
Hilfsmittel	■ Flipcharts und Stifte ■ Pinnwand und Pinnwandnadeln ■ Post-its und Moderationskarten
Gruppengröße	Maximal 7 Personen
Zeitbedarf	2–4 Stunden
Vorbereitung	Vision für das Produkt definieren

So finden Sie Kandidaten für User Stories

1. **Das Ziel festlegen**
   Ziele beantworten die Frage »Wozu?«. Im ersten Schritt legen Sie also fest, was die Stakeholder erreichen wollen.
2. **Unterstützende Akteure identifizieren**
   Finden Sie nun Personen und Rollen, die etwas dazu beitragen können, dass das Ziel erreicht wird.

3. **Einfluss (»Impact«) der Akteure festhalten**
   Halten Sie fest, wie die Akteure beeinflussen können, ob das Ziel erreicht wird. Was können sie tun? Wie können sie ihr Verhalten ändern, damit man dem Ziel näher kommt?
4. **Features und Eigenschaften ableiten**
   Überlegen Sie nun, welche Features das Team bauen und welche Eigenschaften das System erfüllen muss, damit die Akteure dabei unterstützt werden, dem Ziel näher zu kommen.
5. **Metriken aufstellen**
   Finden Sie Metriken, mit denen Sie messen können, ob und wie weit Sie dem Ziel näher gekommen sind.

Leitfragen für das Impact Mapping

Impulsfragen zum Finden der Ziele finden Sie in Abschnitt 7.1. Metriken für die Ziele können Sie finden, indem Sie folgende Fragen stellen:

- Woran erkennen wir, dass das Ziel erreicht ist bzw. wir uns dem Ziel nähern? Wie kann man das messen?
- Welche Bedingungen müssen erfüllt sein, damit das Ziel erreicht ist? Wie können wir diese Bedingungen mit Zahlen beschreiben?

Mithilfe folgenden Fragen ist es einfacher, herauszufinden, wer bei der Erreichung der Ziele helfen kann:

- Wer hat das Ziel genannt? Für wen ist es wichtig? Wer hat Interesse daran, es zu erreichen?
- Wer sind die Anwender unseres Systems?
- Wer kann etwas dazu beitragen, dass wir dem Ziel näher kommen? Wer noch?
- Welche Hindernisse stehen der Erreichung des Ziels im Wege? Wer kann diese Hindernisse aus dem Weg räumen?
- Wer kann verhindern, dass wir das Ziel erreichen? Wie?
- Wer hat sonst noch Einfluss darauf, dass wir das Ziel erreichen?

Stellen Sie folgende Fragen, um zu erarbeiten, was die gefundenen Personen oder Rollen tun können, um den Zielen näher zu kommen:

- Wie kann die Person/Rolle helfen, das Ziel zu erreichen? Wie könnte sie helfen, zumindest einen kleinen Schritt näher zu rücken?
- Wie könnte die Person/Rolle ihr Verhalten ändern, um das Ziel zu erreichen?
- Wie könnte die Person/Rolle helfen, Hindernisse aus dem Weg zu räumen?

Folgende Fragen helfen abzuleiten, was das System können muss, um die Personen und Rollen dabei zu unterstützen, das Ziel zu erreichen:

- Welche Funktionen brauchen die Personen/Rollen, um ihren Beitrag zur Zielerreichung zu leisten?
- Welche Eigenschaften (z. B. Performance, Sicherheit) muss das System aufweisen, damit die Personen/Rollen ihren Beitrag zur Zielerreichung leisten können?

12.2 User Stories mit einer Story Map herunterbrechen

Eine Impact Map generiert Ideen für Features, die erst sortiert, strukturiert und in kleinere Pakete heruntergebrochen werden müssen, bevor sie umgesetzt werden können. Hierbei kann eine Story Map helfen.

Story Mapping ist eine Technik, um User Stories in zwei Dimensionen darzustellen [Patton, 2014]. Dabei werden horizontal die Aktivitäten des Benutzers in der Reihenfolge, wie sie zeitlich ausgeführt werden, geordnet. Die Reihenfolge entspricht also dem Ablauf der Arbeit mit dem System. Diese horizontal angeordneten Elemente nennt man den »Backbone« der Story Map. Vertikal werden darunter für den jeweiligen Schritt User Stories sortiert nach Business Value angereiht.

Beispiel für eine Story Map für die Bogenturnier-Anwendung

Abb. 12–2 *Stories in einer Story Map strukturieren und herunterbrechen*

Die Umsetzung der in der Story Map strukturierten User Stories erfolgt somit von links oben nach rechts unten.

Methode	Story Mapping
Teilnehmer	■ Product Owner und agiles Team ■ Evtl. Vertreter der zukünftigen Anwender
Hilfsmittel	■ Flipcharts und Stifte ■ Pinnwand und Pinnwandnadeln ■ Post-its und Moderationskarten
Gruppengröße	Maximal 7 Personen
Zeitbedarf	2 Stunden
Vorbereitung	■ Kandidaten für Stories z.B. mit einer Impact Map generieren ■ Anwendungsfälle identifizieren ■ Prozesse der Anwender analysieren

So strukturieren Sie Stories in einer Story Map

1. Spannen Sie als Erstes den Backbone der Story Map auf. Elemente des Backbone sind oft die Schritte in einem Prozess oder Anwendungsfall oder größere Features. Achten Sie dabei darauf, die Backbone-Elemente gleich in der richtigen Reihenfolge anzuordnen, also beispielsweise nach zeitlichem Ablauf oder nach Business Value.
2. Brechen Sie nun die Backbone-Elemente in kleinere Pakete herunter und ordnen Sie diese vertikal unterhalb des entsprechenden Backbone-Elementes an. Verwenden Sie für die kleineren Pakete eine andere Farbe an Post-its als für den Backbone.
3. Reihen Sie die vertikalen Pakete jeweils innerhalb eines Backbone-Elementes nach Kundennutzen.

MVP (Minimum Valuable Product)

Oft beinhaltet die Story Map mehr Dinge, als in der gegebenen Zeit tatsächlich umgesetzt werden können. Hier kann es helfen, sich erst einmal zu überlegen, was denn das Minimum ist, das wir ausliefern müssen, damit der Kunde etwas erhält, was einen Wert für ihn hat. Genau das ist das MVP, das »Minimum Valuable Product«.

Im Workshop ziehen Sie dazu zuerst eine rote Linie horizontal durch die Mitte des Flipcharts oder der Pinnwand. Danach lassen Sie die Teilnehmer alle kleinen Pakete jedes Backbone-Elementes, die nicht unbedingt notwendig sind, unter die rote Linie hängen. Alles, was dann noch über der Linie übrigbleibt, gehört zum MVP.

Swimlanes für Releases, Teams etc.

Die Story Map kann man sehr einfach um eine Team- oder Releaseplanung erweitern. Ziehen Sie dafür eine Linie je Team oder Release, so wie Sie es beim MVP gemacht haben. Durch die horizontalen Linien entstehen Bahnen, in die dann die entsprechenden Elemente einsortiert werden können. Da diese Bahnen an die Schwimmbahnen in einem Schwimbecken erinnern, nennt man sie »Swimlanes«.

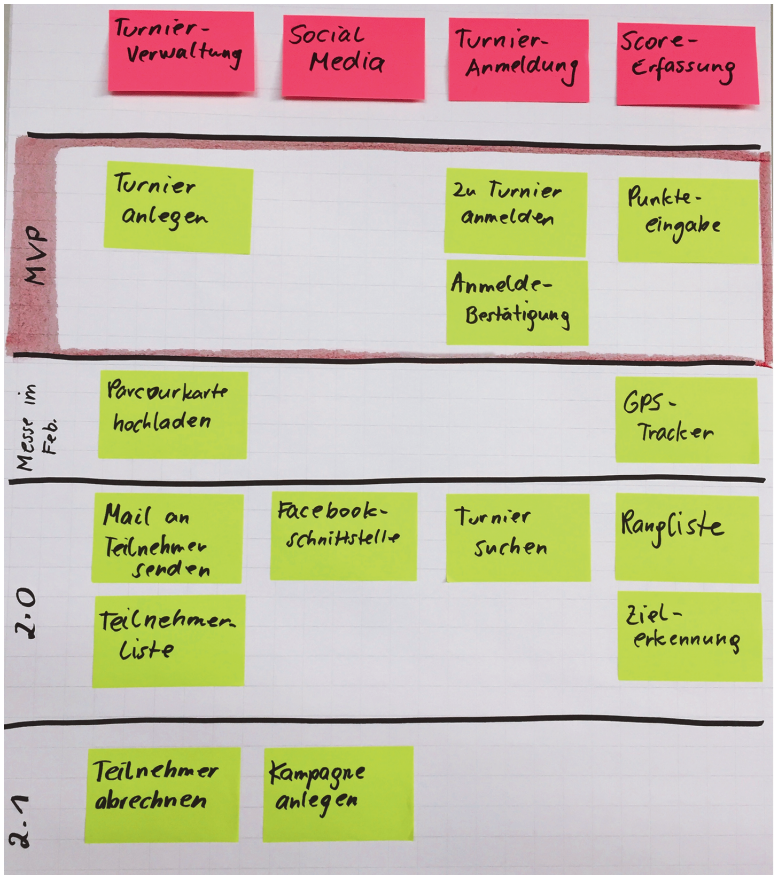

Abb. 12–3 *Das Minimum Valuable Product (MVP) zeigt das kleinstmögliche Produkt, das Wert für den Kunden hat. Swimlanes zeigen die weiteren Releases.*