1 Einleitung

1.1 Meine Vision für die Anforderungsermittlung

> *Anforderungsermittlung bedeutet, herauszufinden, was unsere Kunden wirklich brauchen. Dafür müssen wir unsere Kunden verstehen und begreifen, was sie tun. Wir müssen ihre Bedürfnisse kennen und herausfinden, wie ihnen Software helfen kann, was diese können muss und wie die wesentlichen Funktionen laufen sollen.*

Ein tiefes Verständnis der Probleme, Ziele und Bedürfnisse unserer Kunden ist das Fundament für alle Anforderungen. Basierend darauf können wir gemeinsam mit Anwendern und Entwicklern herausfinden, was die Anforderungen an die neu zu erstellende oder zu ändernde Software sind. Dieses Verständnis für den Kunden und seine Anforderungen müssen wir dann zu denjenigen transportieren, die die Software bauen. Diese müssen es so umsetzen, dass das System am Ende unseren Kunden hilft, ihre Ziele zu erreichen und Probleme zu lösen. Das Verständnis für den Kunden, seine Ziele und Probleme ermöglicht mir als Requirements Engineer, die richtigen Fragen zu stellen, und dem Entwickler, die richtigen Entscheidungen bei der Umsetzung zu treffen.

Meiner Meinung nach ist es nicht immer zwingend notwendig, alles bis ins kleinste Detail vorab auszuspezifizieren. Wenn die Entwickler Erfahrung in der Fachdomäne und mit der eingesetzten Technologie haben, kann ich Detailentscheidungen getrost ihnen überlassen. Haben die Entwickler diese Erfahrung nicht oder ist ein neuer Umsetzungspartner mit im Boot, muss wesentlich genauer erhoben und spezifiziert werden. Es ist meine Aufgabe als Requirements Engineer, zu erkennen, welchen Grad an Spezifikation ein Projekt erfordert, und mein Vorgehen danach anzupassen. Ziel ist nicht die perfekteste Spezifikation oder die Verwendung einer bestimmten Methode, sondern Software, die unsere Kunden weiterbringt!

1.2 Von anderen lernen

Bevor wir uns ins Ermitteln von Anforderungen für Softwaresysteme stürzen, möchte ich Ihnen eine kleine Geschichte zum Thema Umgang mit Kunden und deren Anforderungen in einem völlig anderen Umfeld erzählen:

Erinnern Sie sich noch an Ihren letzten Autokauf? Bei mir war das vor gut 15 Jahren. Ich betrat den Schauraum des Autohändlers meines Vertrauens. Sofort sprang Herr G., der Topverkäufer vor Ort, auf, kam auf mich zu und begrüßte mich herzlich. Er fragte mich freundlich, wie er mir helfen könne. »Ich brauche ein neues Auto, mein altes ist vorige Woche zusammengefallen«, brachte ich es gleich auf den Punkt. »Oje! Na dann schauen wir mal. Wo wohnen Sie denn, Herr Unterauer?«, fragte mich Herr G. Es folgte ein nettes Gespräch, in dem er sich nach meiner Wohnsituation erkundigte, meinem Job, ob ich Kinder hätte, wohin ich in Urlaub fahre und was ich gerne in meiner Freizeit mache. Kaum mal ein Wort über ein Auto. Dafür hatte Herr G. am Ende ein sehr gutes Bild davon, wie mein Alltag so aussieht und wofür ich denn ein Auto überhaupt brauche.

Erst jetzt fragte er mich, was ich mir vorgestellt hätte. Ha! Ich hatte mir schon eine genaue Liste gemacht: Nicht allzu groß sollte es sein, nicht zu teuer, mit Klimaanlage, CD-Player, geringem Verbrauch und langer Lebensdauer. Ein Sportlenkrad und Nebelscheinwerfer wollte ich auch unbedingt haben. Den Allradantrieb, der eigentlich auf meiner Liste gestanden hatte, hatte ich gedanklich während des Gesprächs schon gestrichen. Herr G. hörte sich das alles an, machte sich Notizen und zog dann einige Prospekte hervor, die er mir in die Hand drückte. »Ich möchte Ihnen mal etwas zeigen, Herr Unterauer«, sagte er und führte mich zu einem schicken Mittelklassewagen. Er beschrieb mir die Ausstattung, die Vorzüge, die ewige Lebensdauer und garantierte Pannenfreiheit des Wagens. Das gefiel mir alles sehr gut! Als ich dann allerdings auf das Preisschild schielte, begann meine linke Gesichtshälfte unkontrolliert zu zucken. Der Wagen war zwar toll und hatte alles, was ich zuvor aufgezählt hatte, war aber aufgrund der Größe und Ausstattung viel zu teuer. Herr G. reagierte sofort und bugsierte mich sachte zu einem sportlich angehauchten Kleinwagen. Was soll ich sagen: Der war es! Herr G. plauderte beim Auto noch ein wenig mit mir, hatte aber natürlich längst gemerkt, dass wir den perfekten Wagen für mich gefunden hatten.

Obwohl ich mich kaum losreißen konnte, setzten wir uns als Letztes vor den Computer von Herrn G. Jetzt ging die technische Detailarbeit los. Welches Getriebe, welche Motorisierung, welches Radio, welche Sitzbezüge, Felgen, Lackfarbe und ungefähr noch 20.000 weitere Fragen hatte Herr G. auf seiner Checkliste. Da musste ich jetzt durch. Eine halbe Stunde später verließ ich verschwitzt, aber glücklich das Autohaus. In meiner Tasche hatte ich den Kaufvertrag für mein nagelneues Auto.

Wir können für die Ermittlung von Anforderungen in Softwareprojekten eine Menge von diesem Beispiel aus dem »richtigen Leben« lernen:

1. **Mit Zielen und Nutzen beginnen**
   Starten Sie wie Herr G. mit der Frage nach den Zielen und dem »Warum brauchen Sie das?«. Lernen Sie die Welt der Stakeholder kennen. Nur wenn wir das Ziel kennen, können wir die optimale Lösung finden. Unsere Stakeholder kommen mit einem bunten Mix aus Zielen, Nutzen, Problemen und konkreten technischen Lösungsideen zu uns. Es ist unsere Aufgabe, die Stakeholder etwas zu bremsen und herauszuarbeiten, was der Zweck des Systems ist, welche Ziele damit erreicht und welche Probleme gelöst werden sollen.
2. **Anforderungen schrittweise verfeinern**
   Wenn Sie wissen, was die Ziele sind, erarbeiten Sie die Anforderungen vom Groben zum Feinen. Die zentrale Frage ist hier: »Was soll das System können?« Beginnen Sie mit groben Anforderungsblöcken, präzisieren Sie sie für die Planung und Budgetierung so weit, dass Sie sie schätzen können, und spezifizieren Sie sie für die Umsetzung dann detailliert aus.
3. **Ständig dazulernen**
   Als Requirements Engineer lernen wir zu Beginn die Sprache der Stakeholder und was diese wollen. Unsere Stakeholder lernen, was sie wirklich brauchen und was mit der gegebenen Technologie alles möglich ist. Alle gemeinsam lernen wir, wie wir am besten zusammenarbeiten können, damit wir die Ziele erreichen. Da dieses Lernen im ganzen Projektverlauf weitergeht, ist es völlig natürlich, dass sich die Anforderungen ändern. Unsere Stakeholder dachten vielleicht zu Beginn des Projektes, sie würden für eine bestimmte Aufgabe einen gedruckten Bericht benötigen, schlussendlich stellt sich aber heraus, dass eine reine Bildschirmanzeige die bessere Lösung ist. Rechnen Sie also fix mit Änderungen!

PS: Ich habe meinen Wagen mittlerweile schon 15 Jahre und er bringt mich immer noch ohne Pannen oder größere Reparaturen überall hin. Danke Herr G.! Wenn uns das in unseren Softwareprojekten ebenso gelingt, sind wir schon einen großen Schritt weiter.

1.3 Anforderungen schrittweise ermitteln

In meiner Praxis hat sich folgendes schrittweise Vorgehen zum Ermitteln von Anforderungen bewährt:

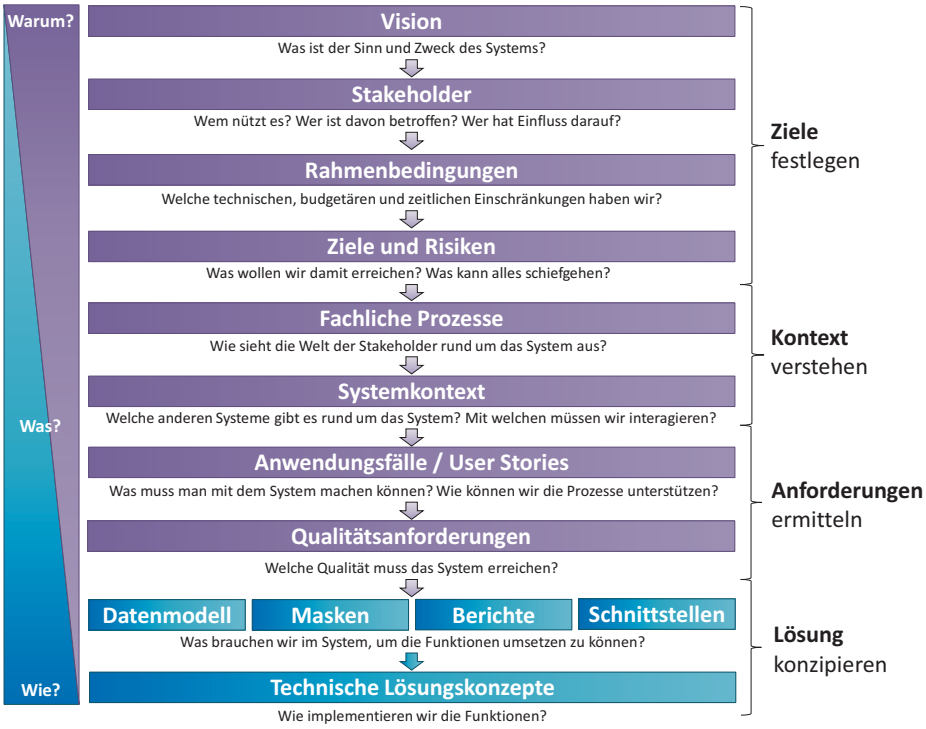

Abb. 1–1 *Anforderungen schrittweise ermitteln und verfeinern*

Beginnen Sie damit, herauszufinden, was der Auftraggeber mit dem System erreichen will. Lernen Sie seine Welt kennen und verstehen. Je besser Ihnen dieser erste Schritt gelingt, desto bessere Fragen werden Sie stellen können und desto besser wird das neue System den Anwendern helfen, ihre Probleme zu lösen und Ziele zu erreichen. Erst danach beschreiben Sie, was das System können und wie es aussehen soll. Das Ergebnis ist ein System, das genau diejenigen Funktionen und Eigenschaften bietet, die die Anwender tatsächlich brauchen. Nicht mehr und nicht weniger. Es wird weder etwas gebaut, das dann nicht verwendet wird. Das wäre Verschwendung. Noch wird etwas weggelassen, was benötigt würde. Das würde zu mangelhaftem Nutzen führen. Das System erfüllt exakt die tatsächlichen Bedürfnisse der Anwender bei geringstmöglichen Entwicklungskosten.

Die linke Seite in Abbildung 1–1 zeigt die Fragen, die uns im Laufe der Zeit leiten. Beginnend mit dem »Warum?« über das »Was?« kommen wir erst am Ende zum technischen »Wie?« Diese Trennung verläuft nicht immer klar und eindeutig. Wenn Sie sich in einer frühen Phase in einer technischen Detaildiskussion

verzetteln, ist das kein Drama. Notieren Sie die wichtigsten Punkte für später und kehren Sie dann wieder zur ursprünglichen Fragestellung zurück.

Der Mittelteil von Abbildung 1–1 beschreibt die einzelnen Themen, die Sie Schritt für Schritt bearbeiten. Beginnen Sie mit dem Systemzweck, den wesentlichen Geschäftszielen und den großen Funktionsbereichen. Diese werden oft gemeinsam mit den wichtigsten Rahmenbedingungen vom Projektsponsor vorgegeben. Aus dem Zweck und den Funktionsbereichen können Sie ableiten, wer die Stakeholder sind. Mit den Stakeholdern können Sie erarbeiten, welche konkreten Ziele erreicht und welche Probleme an der aktuellen Situation gelöst werden sollen. Beschreiben Sie auf grober Ebene die aktuelle Systemlandschaft und die fachlichen Prozesse, in die das neue System eingebettet sein wird.

Gemeinsam mit den Stakeholdern entwerfen Sie, was das System können muss und wie die fachlichen Prozesse der Anwender unterstützt werden können. Diese als Anwendungsfälle spezifizierten Funktionen bilden das zentrale Element der funktionalen Spezifikation. Zu den Anwendungsfällen ergänzen Sie die notwendigen Daten, Masken, Berichte und Schnittstellen (siehe Abb. 1–2).

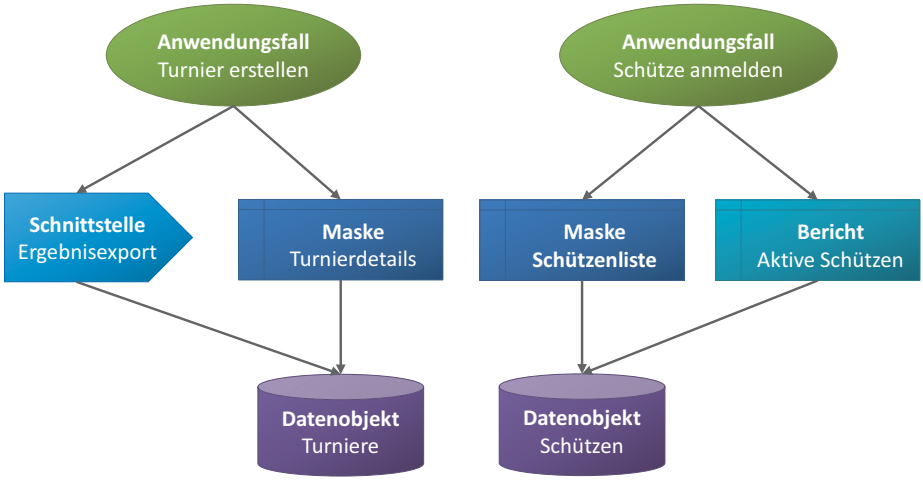

Abb. 1–2 *Systemfunktionen als Anwendungsfälle spezifizieren und dafür Daten, Masken, Berichte und Schnittstellen beschreiben*

All diese Informationen zum »Warum?« und »Was?« bilden gemeinsam eine solide Grundlage für den Entwurf des technischen Lösungskonzeptes.

1.4 Anforderungsermittlung als iterativer Prozess

Diese Vorgehensweise scheint auf den ersten Blick dem klassischen »Big Design Upfront«-Ansatz zu folgen, bei dem erst das gesamte System vollständig und bis ins Detail spezifiziert werden muss, bevor mit der Umsetzung begonnen werden darf. So können Sie es zwar machen, ich empfehle es aber nicht.

Besser ist es, sich an agilen Prinzipien zu orientieren[1]. Das bedeutet, Sie beginnen mit dem Systemzweck, den Zielen und den Stakeholdern. Die darauffolgenden Schritte werden stark überlappend durchgeführt, indem Sie für alle Ebenen immer erst eine Liste mit Überschriften sammeln und deren Inhalte und Details im Laufe der Zeit ausarbeiten, sobald die Umsetzung bevorsteht.

Beispielsweise erstellen Sie mit den Stakeholdern zuerst eine Liste der zu unterstützenden Geschäftsprozesse. Diese priorisieren Sie nach Nutzen. Die wichtigsten 1–5 Prozesse, die schon im ersten Release unterstützt werden müssen, sehen Sie sich nun genauer an. Nur für diese Top-5-Prozesse leiten Sie Anwendungsfälle und genaue Anforderungen ab. Diese werden dann im ersten Release umgesetzt. Während der Umsetzung beginnen Sie mit der Spezifikation des nächsten Release. Das heißt, Sie wählen aus der Liste der Geschäftsprozesse die nächsten aus, spezifizieren sie genauer, leiten Anwendungsfälle ab usw.

Das Grundprinzip dieser Vorgehensweise ist: Wir spezifizieren so spät wie sinnvoll möglich. James Shore spricht hier vom »latest responsible moment« [Shore, 2007]. Es wird lediglich das genau ausspezifiziert, was in der nächsten Zeit umgesetzt wird. Alles andere wird nur grob abgesteckt [Bergsmann, 2018] (siehe Abb. 1–3).

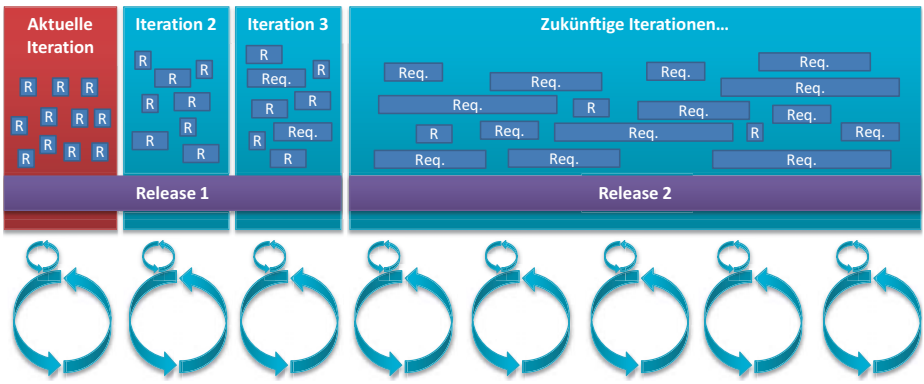

Abb. 1–3 *Je näher die Umsetzung, desto genauer wird spezifiziert (Grafik angelehnt an [Ambler & Lines, 2012] und [Bergsmann, 2018]).*

1. Eine gute Einführung in agile Methoden finden Sie z.B. in [Beck et al., 2001] und [Schwaber, 2004].

Auf diese Weise vermeidet man, dass Dinge spezifiziert werden, die am Ende doch nicht gebraucht oder völlig anders umgesetzt werden. Einer meiner Kunden erzählte beispielsweise, er hatte in einem seiner Projekte Hunderte Stunden in ein 210-seitiges Spezifikationsdokument investiert. Umgesetzt wurde lediglich ein Drittel, den Rest hatten die Entwickler nicht einmal mehr gelesen. Solche »write only«-Spezifikationen wollen wir vermeiden.

Darüber hinaus verteilt sich der Aufwand für die Anforderungsermittlung besser auf das gesamte Projekt (siehe Abb. 1–4, [Bergsmann, 2018]).

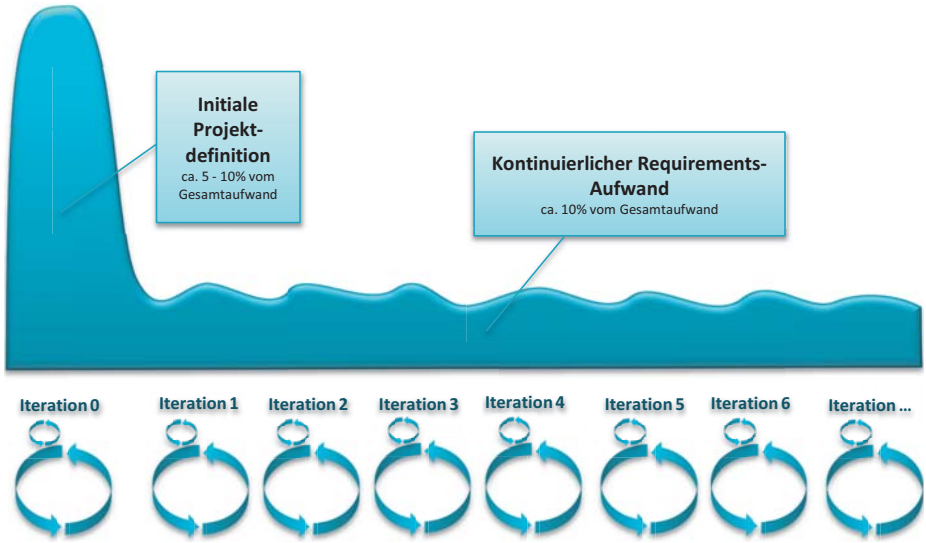

Abb. 1–4 *Aufwand für Anforderungsermittlung verteilt sich über das ganze Projekt.*

Ein weiterer wichtiger Aspekt, den wir bei der Ermittlung von Anforderungen berücksichtigen müssen, ist, dass wir und unsere Stakeholder zu Beginn gar nicht alle Anforderungen kennen. Wir können sie also auch mit noch so viel Aufwand nicht ermitteln. Unser Vorgehen muss uns also ermöglichen, ständig dazuzulernen und das bisher Erreichte zu hinterfragen und zu korrigieren. Wir müssen es dafür unseren Stakeholder so einfach wie möglich machen, Feedback zu geben und möglichst eng mitzuarbeiten [Grau & Lauenroth, o.J.]. Je besser eine Methode das Feedback und die Zusammenarbeit mit unseren Stakeholdern unterstützt, desto besser hilft sie uns in unseren Projekten.

1.5 Workshops und andere Techniken

In diesem Buch geht es um den Einsatz von Workshops zur Ermittlung von Anforderungen. Neben Workshops gibt es natürlich noch zahlreiche andere Methoden und Techniken zum Ermitteln von Anforderungen, die Sie ergänzend oder alternativ zu Workshops einsetzen können. Die folgenden stellen eine kleine Auswahl dar [IREB C.-A. , 2012; Ebert, 2019]:

- **Diskussion und Besprechung**
  Die einfachste Form, ein Thema zu behandeln, ist eine simple Besprechung. Alle Teilnehmer sitzen an einem Tisch und jeder soll sich zum Thema äußern. Ein Moderator achtet darauf, dass sich alle einbringen, dass die Diskussion beim Thema bleibt, und protokolliert die Ergebnisse.
- **Interview und Fragebogen**
  Ein Interview ist eine mehr oder weniger strukturierte mündliche Befragung. Interviews können eingesetzt werden, wenn ein Thema mit einzelnen Personen im Detail beleuchtet oder die Meinungen verschiedener Personen unabhängig voneinander eingeholt und verglichen werden sollen. Ein Beispiel für den Einsatz von Interviews als Ergänzung zu Workshops ist die Detailausarbeitung von Anwendungsfällen, wie in Abschnitt 11.4 beschrieben.

  Ein Fragebogen ist eine schriftliche Befragung. Der große Vorteil dabei ist, dass er statistisch auswertbare und vergleichbare Ergebnisse von einer großen Menge an Personen möglich macht. Setzen Sie ihn immer dann ein, wenn Sie rasch viele Personen, z.B. potenzielle Anwender oder Kunden, zu einem klar definierten Thema befragen wollen.
- **Feldbeobachtung und »In die Lehre gehen«**[2]
  Bei der Feldbeobachtung sehen Sie sich als Requirements Engineer einen Ablauf in freier Wildbahn an. Daraus leiten Sie dann Anforderungen ab. Ein Beispiel für den Einsatz einer Feldbeobachtung als Ergänzung zu Workshops finden Sie in Abschnitt 9.3.

  Das »In die Lehre gehen« geht eine Stufe weiter als die Feldbeobachtung. Hier sehen Sie nicht nur zu, sondern führen den Prozess selbst aus. Natürlich lernt man so am meisten über die Anforderungen an das neue System.
- **FMEA (Failure Mode and Effects Analysis)**
  Bei einer FMEA bewerten Sie für alle Komponenten eines Systems, welche Fehler auftreten können, welche Auswirkungen diese Fehler haben und wie oft sie auftreten werden. Dafür leitet man dann Gegenmaßnahmen ab. FMEA sind im sicherheitskritischen Bereich üblich und durch Normen gefordert [Spillner, A., Roßner, T., Winter, M. & Linz, T., 2014].

2. Geläufiger ist für diese Methode die englische Bezeichnung »Apprenticing«.

- **Systemarchäologie**
  Wenn man ein bestehendes Altsystem ablösen oder erweitern muss, ist es oft notwendig, sich in die Dokumentation oder den Quellcode des Altsystems hineinzugraben. Diese Technik der Gewinnung von Anforderungen nennt man Systemarchäologie.
- **Dokumentanalyse**
  Viele Anforderungen kommen aus gesetzlichen Vorgaben, Normen und Verträgen. Auch solche Anforderungen können wir nicht in Workshops ermitteln, hier müssen wir die Dokumente selbst analysieren und Anforderungen daraus ableiten.
- **Prototypenbau**
  Manche Dinge können uns die Stakeholder einfach nicht sagen, solange sie nicht etwas sehen, auf das sie sich stützen können. Ich nenne das das »I know it when I see it«-Prinzip[3]. Wir müssen erst einen Prototyp bauen, anhand dessen wir dann die Anforderungen mit den Stakeholdern diskutieren können. Einfache Formen von Prototypen als Basis für die Diskussion in den Workshops finden Sie in den Abschnitten 14.1 und 15.1.
- **User Stories**
  Aus der agilen Welt kommt die Methode, Anforderungen mit kleinen Beschreibungen, den User Stories, zu spezifizieren. Diese zeigen, was bestimmte Anwender mit dem System machen wollen und wozu. Sie werden immer mit der Satzschablone »Als <Rolle> möchte ich <Aktion>, damit <Nutzen>« beschrieben. Ergänzt werden diese kurzen User Stories durch Akzeptanzkriterien. In Abschnitt 11.1 werden obige Satzschablone und Akzeptanzkriterien beim Ermitteln von Anwendungsfällen verwendet.

Je nach Quelle der Anforderung, Gruppengröße und Art der Anforderung müssen Sie die richtige Ermittlungstechnik auswählen und einsetzen. Kommen Ihre Anforderungen von Personen, müssen Sie andere Techniken einsetzen, als wenn Sie Anforderungen aus Dokumenten oder bestehenden Systemen ableiten wollen. Wenn Sie eine Gruppe von Personen einbinden müssen, sind andere Techniken gefragt, als wenn Sie nur einen einzigen Experten oder einen anonymen großen Anwenderkreis haben.

Schlussendlich hängt die Wahl der Ermittlungsmethode auch von der Neuheit und Komplexität des Systems ab. Muss etwas völlig Neues konzipiert werden, benötigen Sie eher kreative Techniken und solche, die die Stakeholder stark einbinden und ein exploratives Herausfinden der Anforderungen ermöglichen. Wenn im Gegensatz dazu bestehende Konzepte weiterentwickelt werden oder die Stakeholder die Anforderungen bereits kennen, kann auf einfache befragende Techniken zurückgegriffen werden.

3. Diese Phrase geht auf den US-Richter Stewart Potter zurück, der sie 1964 in einem Prozess wegen Pornografie verwendete. Naja, passt auch für Software, wie ich finde.

1.6 Wo Workshops Sinn machen … und wo nicht

Workshops sind besonders gut geeignet, wenn Sie die Anforderungen mit einer Gruppe von Stakeholdern erarbeiten müssen. Dies ist in vielen Projekten der Fall und darauf konzentriert sich auch dieses Buch.

Workshops liefern hohe Akzeptanz der erarbeiteten Lösungen

Da die Anforderungen nicht im stillen Kämmerlein von einigen wenigen definiert, sondern von den Betroffenen selbst gemeinsam erarbeitet werden, ist die Akzeptanz der gefundenen Lösungen größer.

Workshops ermöglichen unmittelbares Feedback

Alle Beteiligten können in einem Workshop sofort Feedback geben. Dies ermöglicht es, schneller bessere Lösungen zu entwickeln, als wenn einzelne Personen die Anforderungen entwerfen und dann Feedback dazu einholen.

Workshops vereinen Ermittlung und Abstimmung

Im Requirements Engineering müssen Anforderungen nach der Ermittlung und Dokumentation mühsam abgestimmt werden. Dies geschieht meist, indem ein Dokument an alle relevanten Stakeholder zum Review geschickt wird. Diese prüfen und kommentieren das Dokument und schicken es dem Requirements Engineer zurück, der die Dutzenden, oft auch Hunderte von Anmerkungen einarbeiten muss – alles in allem ein sehr langwieriger und aufwendiger Prozess. In Workshops sind alle Beteiligten gemeinsam bei der Anforderungsdefinition dabei. Dies ersetzt Abstimmung und Review der am Ende erstellten schriftlichen Spezifikation nicht völlig. Die Anzahl der zu ändernden und neu auszuhandelnden Punkte sinkt aber in der Regel dramatisch. Damit wird der gesamte Prozess deutlich schneller und weniger aufwendig.

Aber: Näher an der Software brauchen wir ergänzende Methoden

Workshops sind vor allem in den frühen Phasen der Anforderungsermittlung, wenn es stark um fachliche und Prozessaspekte geht, sehr gut geeignet. Je mehr wir uns dem System selbst und der Technik, z. B. an den Schnittstellen zu anderen Systemen, annähern, desto kleiner wird der Kreis der Beteiligten an den Workshops und Meetings. Wie bereits erwähnt, ist aber jedes Projekt anders und Sie sollten immer wieder neu hinterfragen, welche Methode in der jeweiligen Situation am besten geeignet ist!

Wie es im Buch nun weitergeht …

In den Kapiteln 4 bis 17 werden konkrete Methoden vorgestellt, wie Sie die jeweiligen Themen in Workshops mit Ihren Stakeholdern erarbeiten können.

Am Beginn jedes Kapitels finden Sie die vorgestellte Methode an einem Beispiel angewandt, danach die Beschreibung, wie Sie sie im Workshop Schritt für Schritt einsetzen. Da ich begeisterter Bogenschütze bin, habe ich als Beispielsystem eine fiktive Applikation für die Abwicklung von Bogenturnieren mit Namen »MyBowTurnament« gewählt.

Abb. 1–5 *Als begeisterter Bogenschütze muss die Beispielapplikation natürlich eine Anwendung zur Abwicklung von Bogenturnieren sein.*

Bevor wir uns nun in die Anforderungsermittlung stürzen, möchte ich Ihnen die Grundkenntnisse vermitteln, wie Sie Anforderungsworkshops erfolgreich planen und durchführen.

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:

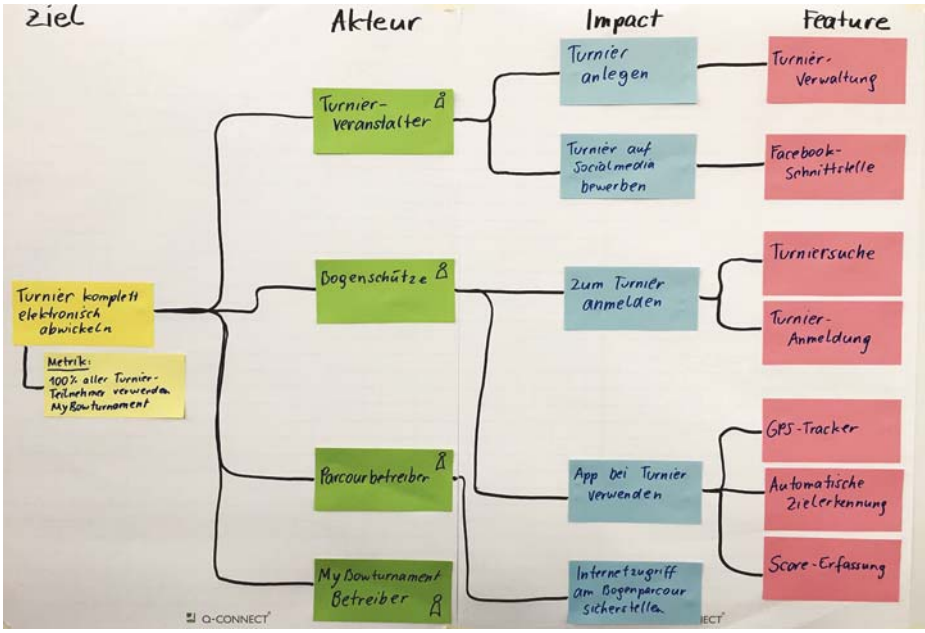

Abb. 12–1 *Die Impact Map zeigt, wer bei der Erreichung der Ziele wie helfen kann und was dafür in der neuen Software gebaut werden muss.*

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«.

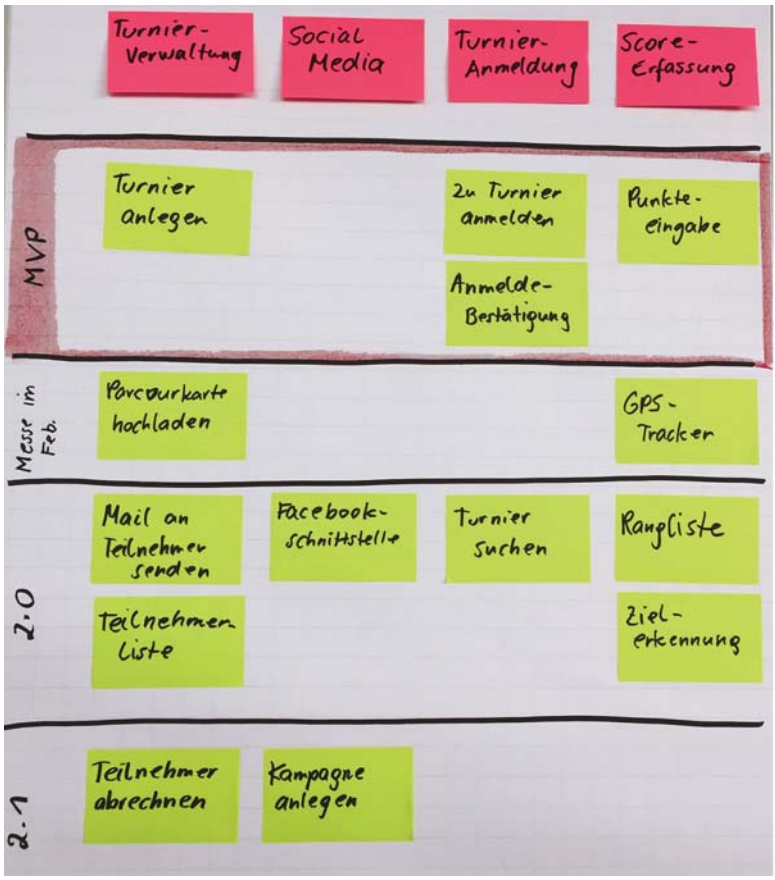

Abb. 12–3 *Das Minimum Valuable Product (MVP) zeigt das kleinstmögliche Produkt, das Wert für den Kunden hat. Swimlanes zeigen die weiteren Releases.*