3 Requirements-Ermittlung und -Dokumentation

3.1 Ein kurzer Überblick

[IREB RE@Agile Primer] LE 3.1.12

Gutes Requirements Engineering hilft, dass keine relevanten Aspekte vergessen werden. Bei nicht agilen Methoden erfolgt dies oft durch eine Dokumentation aller potenziellen Anforderungen der Stakeholder. In agilen Ansätzen nutzt man dagegen die direkte Kommunikation und rasche Feedbackschleifen, z. B. durch inkrementelle Produktentwicklung oder Prototypen. Der oft zitierte zweite Leitsatz des Agilen Manifests beinhaltet genau dieses Thema: »Funktionierende Software mehr als umfassende Dokumentation« [Agile Manifesto].

Jede Organisation kann profitieren, wenn bewusst überlegt wird, was schriftlich erfasst und was besprochen oder was in Form von Prototypen oder Inkrementen gezeigt werden kann. Die besten Ergebnisse werden erzielt, wenn Dokumentation, Kommunikation und Prototyping bzw. inkrementelle Entwicklung entsprechend den Rahmenbedingungen und der Kultur einer Organisation und der Art des umzusetzenden Projekts ausgewogen und mit gesundem Menschenverstand angewendet werden.

Es gibt einige Artefakte und Vorgehensweisen, die für eine erfolgreiche Entwicklung sowohl im nicht agilen als auch im agilen Requirements Engineering sehr wichtig sind und die in den nächsten Abschnitten behandelt werden:

- Mit Visionen oder Zielen beginnen
- Immer die wichtigsten Stakeholder identifizieren und kennen
- Den Umfang explizit festlegen und vom Kontext abgrenzen
- Ein Verständnis der Funktionalitäten und der Geschäftsbegriffe schaffen, um funktionale Anforderungen zu erfassen (einschließlich Funktionalität und Daten)
- Qualitätsanforderungen und Randbedingungen berücksichtigen, da sie Designentscheidungen stark beeinflussen. Werden diese ignoriert oder erst zu spät erkannt, kann viel Überarbeitungsaufwand entstehen oder ein Produkt, das den Erwartungen des Kunden nicht entspricht.

3.1.1 Anforderungsarten

[IREB RE@Agile Primer] LE 3.1.4

Die Anforderungen werden auf Basis der Vision und der Ziele erfasst und sind durch das Kontextmodell begrenzt.

IREB definiert drei unterschiedliche Anforderungen: funktionale Anforderungen, Qualitätsanforderungen und Randbedingungen (Constraints), die in den nachfolgenden Abschnitten dieses Kapitels näher erörtert werden (siehe auch [IREB CPRE FL]).

3.1.2 Requirements-Dokumente vs. Product Backlog

[IREB RE@Agile Primer] LE 3.1.1

Requirements-Ermittlung und -Dokumentation gehören zu den Kernaufgaben des Requirements Engineering (vgl. Kap. 2). Anforderungen können nicht für sich alleine stehen, um passende Produkte oder Lösungen daraus zu schaffen. Alles, was im weiteren Verlauf als Anforderung an das Produkt benötigt wird, soll in einer priorisierten Liste organisiert und dokumentiert werden.

Die Requirements-Sammlung in Form einer priorisierten Liste ist das zentrale Artefakt für Product Owner, Requirements Engineers oder Business-Analysten unabhängig von der verwendeten Methode. Diese Sammlung muss nicht unbedingt dokumentenbasiert sein, sondern kann Repository-basiert, in einer Datenbank oder auch auf Papier erstellt werden. Auch die Darstellungsart kann nach Bedarf variieren zwischen Text, Bildern, Diagrammen, Modellen etc. Die Bezeichnungen sind ebenfalls vielfältig wie z.B. je nach Detaillierungsgrad: Benutzeranforderungsspezifikation, System- oder Softwareanforderungsspezifikation, Product Backlog, Sprint Backlog. Allen diesen Artefakten liegt gemeinsam zugrunde, dass sie für Ermittlung, Dokumentation, Abstimmung, Validierung und Management der Anforderungen verwendet werden.

Im agilen Umfeld wird als Requirements-Sammlung typischerweise das Product Backlog und das Sprint Backlog eingesetzt, die mit einer geeigneten Software oder in ganz einfacher Form auch mit Karteikarten oder Haftnotizen gemanagt werden können. Verschiedene Techniken unterstützen den Product Owner beim Management der Anforderungen wie z.B. die DEEP-Qualitätskriterien für die Formulierung, Priorisierungstechniken, Schätzmethoden oder Detailbeschreibungstechniken.

Wichtig für das Verständnis des Requirements Engineering ist es, zu erkennen, dass ein Requirement kein bestimmtes Artefakt auf einer definierten Granularitätsebene ist. In der Praxis tauchen oft Aussagen auf wie »Ein Requirement ist immer eine sehr detaillierte Beschreibung, eine User Story ist dagegen eine grobe

Beschreibung« oder »Ein Requirement ist die übergeordnete grobe Beschreibung und der PO bzw. die Entwicklerin leitet sich daraus die Use Cases und User Stories ab«. Solche und ähnliche Statements sind nicht korrekt. Ein Requirement ist jede Anforderung an das System, unabhängig davon, auf welcher Abstraktionsebene oder in welcher Detailliertheit diese beschrieben ist. In diesem Sinne sind z. B. User Stories, Epics, Features, Use Cases, aber auch die übergeordneten Ziele gleichermaßen als Requirements zu bezeichnen.

Ein Requirement (Anforderung) ist …

1. *Ein Bedürfnis, das von einem Stakeholder wahrgenommen wird.*
2. *Eine Fähigkeit oder Eigenschaft, die ein System haben soll.*
3. *Eine dokumentierte Darstellung eines Bedarfs, einer Fähigkeit oder Eigenschaft.*

[IREB Glossar]

Diese Definition gilt für ALLE Artefakte, die diese Inhalte repräsentieren, unabhängig von ihrer Bezeichnung oder Darstellungsform.

Unabhängig davon, wie die Requirements dokumentiert werden, ist es erforderlich, alle Arten von Requirements zu erfassen:

- Visionen und Ziele (siehe Abschnitt 3.2.2)
- Umfang und Kontext des Systems (siehe Abschnitt 3.2.3)
- Funktionale Anforderungen (siehe Abschnitte 3.4.1 und 3.4.2)
- Qualitätsanforderungen (siehe Abschnitt 3.4.3.1)
- Randbedingungen (siehe Abschnitt 3.4.3.2)
- Glossar (Definition von relevanten Begriffen und Abkürzungen)

Der Detaillierungsgrad kann natürlich auch je nach Situation variieren.

Es ist keine gute Alternative, wenn nur eine verbale Kommunikation zwischen den Stakeholdern erfolgt und keine schriftlichen Anforderungen vorhanden sind, da schriftliche Dokumente oft die Grundlage für Verhandlungen, Akzeptanztests, rechtliche Belange und vieles mehr sind. Je mehr die beteiligten Personen miteinander kommunizieren, desto weniger Schriftlichkeit ist in der laufenden Kommunikation notwendig. Die Ergebnisse aus der Kommunikation (die Anforderungen) müssen jedoch in einer passend gewählten Form erfasst werden.

3.1.3 Granularität funktionaler Requirements

[IREB RE@Agile Primer] LE 3.1.5

Funktionale Anforderungen können (und sollten) auf verschiedenen Abstraktionsebenen diskutiert und dokumentiert werden, da die Stakeholder Anforderungen auch auf verschiedenen Ebenen der Granularität von sehr grobkörnig bis sehr detailliert und feinkörnig spezifizieren.

Nicht agile und agile Methoden verwenden unterschiedliche Begriffe für diese Artefakte auf den verschiedenen Abstraktionsebenen.

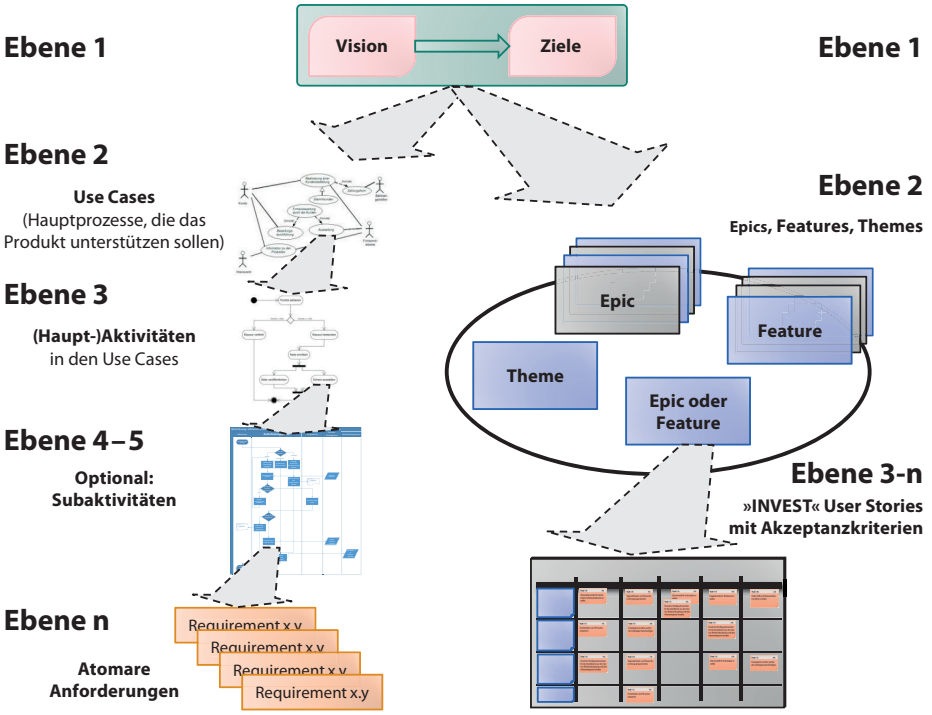

Abb. 3–1 *Anforderungsgranularität (nach [IREB RE@Agile Primer])*

Wenn man den nicht agilen Ansatz (linke Seite der Abbildung) betrachtet, wird das System z. B. in Use Cases zerlegt, um Visionen und Ziele zu verfeinern (Ebene 2). Ein Use-Case-Diagramm bietet eine übergeordnete verhaltensorientierte Sicht über die benötigte Funktionalität bzw. auch über die Akteure oder Benutzer. Bei umfangreichen Anwendungsfällen wird man eine weitere Zerlegung durchführen und weitere Details auf untergeordneten Ebenen definieren (siehe Ebene 3, 4, 5 etc.). Dies ist ein mögliches Beispiel. In der Praxis nutzt man zur Verfeinerung nach Bedarf z. B. auch Features, Businessobjekte, Datenflüsse, Prozessabläufe oder Komponenten bestehender Systeme.

Für die genaue Festlegung und Beschreibung der Schritte und Funktionalität eines Use Case gibt es verschiedene Möglichkeiten:

- Mündliche Beschreibung (Klartext)
- Anwendungsfallvorlagen (halbstrukturiertes Format oder in Form von narrativen[1] Abläufen)
- Aktivitätsdiagramm (oder jede andere Art von grafischem Funktionsmodell wie Datenflussdiagramme, BPMN, Kontextdiagramm, Kontrollflussdiagramm, Zustandsdiagramm, Geschäftsobjektmodell usw.). Ebene 3 in Abbildung 3–1 könnte beispielsweise den Use Case mit UML-Aktivitätsdiagrammen weiter verfeinern.

Je nach Problemkomplexität können und sollen weitere Granularitätsebenen verwendet werden, beispielsweise die Zerlegung in Teilaktivitäten (siehe Ebene 4 in Abb. 3–1) oder mit textbasierten Anforderungsbeschreibungen einzelner Aktivitäten (siehe Ebene 5 in Abb. 3–1). Mit diesem Vorgehen werden grobkörnige Kundenanforderungen schrittweise verfeinert hin zur erwarteten Lösungsfunktionalität.

Der oben genannte Ansatz ist ein Top-down-Prozess der Anforderungsanalyse und Zerlegung. In der Realität werden Requirements oft auch zuerst auf einer detaillierten Lösungsebene geäußert und spezifiziert und die allgemeineren Ziele kristallisieren sich erst später heraus. Daher kann der Requirements-Entstehungsprozess in der Realität durchaus verschieden sein: top-down, bottom-up, outside-in, inside-out oder eine Kombination daraus.

Der entscheidende Punkt ist, dass es Methoden und Techniken im Requirements Engineering gibt, um alle diese Diskussionen und Spezifikationen an ihren unterschiedlichen Ebenen und Zeitpunkten zu unterstützen. Dadurch wird mit fortschreitendem Wissen über die Anforderungen auch eine sinnvolle Hierarchie der Anforderungen erstellt, die hilft, den Überblick zu bewahren.

Bei geringer Komplexität der Use Cases kann auch nur eine der Optionen angewendet werden (z. B. könnten nur ein paar Sätze zur Beschreibung ausreichen). Bei mittlerer Komplexität können weitere Optionen wie z. B. Use-Case-Templates mit einer genaueren Beschreibung der Prozessschritte innerhalb des Use Case verwendet werden (siehe auch Abschnitt 3.3.3). Dies erleichtert das Verstehen der Anforderungen. Bei komplexen Anforderungen können grafische Formate wie z. B. Aktivitätsdiagramme eingesetzt werden, um mehrere Szenarien oder komplexe parallele oder verzweigte Abläufe oder Sequenzen darzustellen.

Wenn man den agilen Ansatz (rechte Seite der Abb. 3–1) betrachtet, findet man eine ähnliche Struktur nur mit anderen Begriffen aus der agilen Terminologie wie z. B. Epics (siehe auch Abschnitt 3.2.5), Themes (siehe auch Abschnitt 3.9.1), Features (siehe auch Abschnitt 3.4.1) oder User Stories (siehe auch Abschnitt 3.4.2).

Hier werden übergeordnete Geschäftsziele und komplexe Funktionalität als Epics oder Features zusammengefasst. Themes unterstützen die Gruppierung.

---

1. Von lateinisch narrare »kundtun, erzählen«.

Wenn die Anforderungen komplexer sind und z. B. mehr als einen Sprint zur Umsetzung benötigen würden, müssen diese komplexen Anforderungen ebenfalls in kleinere Einheiten zerlegt werden. Die Kriterien für gute User Stories sind auch in den Abschnitten 2.1.2 und 3.4.2 angeführt. Generell sollen diese der Definition of Ready entsprechen (siehe auch Abschnitt 5.2).

Hinsichtlich der Erfüllung der INVEST-Qualitätskriterien (siehe Abschnitt 5.1.3) sind besonders »E« (schätzbar/estimable), »S« (klein genug/small)) und »T« (testbar/testable) wichtig.

Auch im agilen Umfeld ist es so, dass das Vorgehen nach Bedarf eine strenge Top-down-Verfeinerung von Epics in Features und dann in User Stories sein kann oder umgekehrt auch zuerst eine Sammlung von User Stories, die dann bottom-up durch Themes gruppiert werden und von denen dann die Epics zur besseren Übersicht später abgeleitet werden. Wie auch immer, es hängt von der Art eines Projekts und der beteiligten Akteure ab, welches Vorgehen schlussendlich sinnvoll und zielführend ist.

Die agile Vorgehensweise bietet Artefakte für das Requirements Engineering, um sowohl das »Big Picture« als auch detaillierte »umsetzungsbereite« Anforderungen zu spezifizieren, zu diskutieren und zu priorisieren.

3.1.4 Grafische Modelle und textuelle Beschreibungen

[IREB RE@Agile Primer] LE 3.1.6

Die Wahl der passenden Granularität bzw. Detaillierungsebene ist ein wichtiger Aspekt im Requirements Engineering. Ein anderer wichtiger Aspekt ist die Wahl der passenden Notation für die Beschreibung der funktionalen Anforderungen. Dies ist unabhängig voneinander.

Es gibt (ebenfalls wieder unabhängig vom gewählten Vorgehensmodell) verschiedene Möglichkeiten der Beschreibung von Requirements:

- **Textform (natürliche Sprache)**
  Zum Beispiel mit der User-Story-Satzschablone oder als einfacher natürlichsprachlicher Satz oder auch als etwas formellere textbasierte Struktur (wie z. B. Gherkin – siehe auch Abschnitt A.3 im Anhang), die klar ausdrücken, was das System tun soll. Das Problem bei natürlicher Sprache ist, dass sie manchmal nicht ausreichend genau und eindeutig ist und daher von anderen Lesern auch leicht fehlinterpretiert werden kann.

  Andererseits sind textbasierte Anforderungen in der Regel schneller zu spezifizieren, einfacher zu verwalten und werden von vielen Stakeholdern auch (vermeintlich) leichter verstanden.

- **Grafische Notationen**
  Grafische Modelle sind formaler und vermeiden so unterschiedliche Interpretationen oder Missverständnisse.

  Es gibt viele unterschiedliche grafische Darstellungsformen wie z. B. UML-Aktivitätsdiagramme, BPMN-Diagramme, Flussdiagramme, Zustandsdiagramme oder Sequenzdiagramme. Die jeweiligen grafischen Notationen decken typischerweise unterschiedliche Aspekte ab: Aktivitätsdiagramme oder BPMN-Diagramme sind für eher lineare Prozesse geeignet, die aufeinanderfolgende Schritte, Alternativen oder Schleifen darstellen. Zustandsdiagramme eignen sich sehr gut, um asynchrone Ereignisse und Abläufe darzustellen. Sequenzdiagramme bieten sich für Schnittstellenspezifikationen und auch für »Specification by Example« an, da sie konkrete Szenarien für eine Interaktion zeigen, ohne jedoch zu versuchen, vollständig zu sein.

Beide Varianten (Textform und grafische Notation) haben ihre Vor- und Nachteile. Die Wahl sollte durch das Umfeld des Projekts und die zentralen Ziele des Requirements Engineering bestimmt werden. Die Darstellungsformen können auf jeder Ebene und auch bei jeder einzelnen Ausprägung eines Requirements nach Bedarf gewählt werden (z. B. einen Use Case grafisch beschreiben und den nächsten textuell) und lassen sich auch mischen (einen Teil eines Use Case als Text spezifizieren und einen anderen grafisch). Wichtig ist, dass man sich in der Wahl der Beschreibungstechniken nicht zu stark einschränkt, sondern nach Bedarf aus dem vollen Spektrum der Möglichkeiten schöpft.

Es ist alles erlaubt, was hilft, ein gemeinsames Verständnis aller Stakeholder zu erreichen sowie Unvollständigkeit und Missverständnisse zu vermeiden.

3.1.5 Definition von Begriffen, Glossare und Informationsmodelle

[IREB RE@Agile Primer] LE 3.1.7

Es ist wichtig, dass ein klares Verständnis aller Begriffe in der Spezifikation von Anforderungen (egal ob Text und grafisches Modell) erreicht wird.

Bei vielen einschlägig fachlichen, betriebswirtschaftlichen oder technischen Begriffen zur Beschreibung der Anforderungen und anderen Artefakten haben die Beteiligten eine sehr klare Vorstellung davon, was der jeweilige Begriff zu bedeuten hat. Bei manchen ist die Auslegung jedoch unterschiedlich. Daher ist für eine erfolgreiche Zusammenarbeit ein klares Verständnis der Begriffe erforderlich.

Gerade im Product Backlog beim agilen Vorgehen steht oft die Spezifikation der Funktionalität im Vordergrund und die Definition von Geschäftsbegriffen tritt in den Hintergrund.

Es ist daher notwendig, ein Verzeichnis aller relevanten Begriffe, die in den Prozessen, Epics, User Stories oder anderen Artefakten verwendet werden, zu erstellen. Dies gilt für alle Arten von Projekten (sowohl agil als auch nicht agil) und wird leider oft vergessen.

Die minimale Form ist eine Liste von (textlich geschilderten) Begriffen und Abkürzungen, die in der Regel zum leichten Auffinden und Ändern alphabetisch sortiert werden. Bezeichnungen, die dafür verwendet werden, sind »Glossar«, »Wörterbuch« oder einfach »Liste von Definitionen«.

Bei komplizierten Zusammenhängen oder sehr umfangreichen Anwendungsdomänen kann es helfen, dieses Glossar zu strukturieren. Dazu kann man im einfachsten Fall Begriffe in Kategorien, Klassen oder Entitäten gruppieren oder bei komplexeren Zusammenhängen diese Beziehungen dann auch grafisch darstellen. Dazu können Datenmodelle, Informationsmodelle, Entity-Relationship-Diagramme oder UML-Klassendiagramme verwendet werden. Zusätzlich zu der Definition von Begriffen umfassen diese grafischen Modelle auch die relevanten statischen Beziehungen zwischen den Begriffen.

3.1.6 Akzeptanz- und Abnahmekriterien

[IREB RE@Agile Primer] LE 3.1.9

Anforderungen müssen validiert, geprüft oder getestet werden können. Andernfalls ist die Anforderung nur von beschränktem Nutzen bzw. führt meist zu Mehraufwänden, weil dann typischerweise noch Missverständnisse oder Unklarheiten auftreten.

Daher benötigt jede Anforderung, egal ob im nicht agilen oder agilen Umfeld, eine Reihe von Kriterien, die getestet werden können, um festzustellen, ob die Anforderung auch erfüllt ist.

Im nicht agilen Umfeld werden dafür oft Begriffe wie »Qualitätskriterien«, »Abnahmekriterien« oder auch nur »Testfälle« verwendet. Beim agilen Vorgehen sind es häufig Begriffe wie »Akzeptanzkriterien« (für User Stories) oder »Erfolgskriterien« (für Epics oder Themes).

Im Grunde sind diese Kriterien oder Testfälle auch nichts anderes als Requirements (siehe auch Abschnitt A.2 im Anhang). Die ursprünglichen Anforderungen und die dazu passend definierten Testfälle ergänzen sich und ergeben zusammen eine sinnvolle Gesamtspezifikation.

Abhängig von der Granularität der Anforderungen müssen entsprechende Akzeptanzkriterien definiert werden:

- Auf höheren Ebenen (Vision, Ziele und Epics) werden in der Regel Erfolgskriterien definiert, da man manchmal nur feststellen kann, ob die benötigte Funktionalität/Fähigkeit vorhanden ist oder nicht – was bedeutet, dass die Anforderung erfüllt wurde.
- Auf niedrigeren Abstraktionsniveaus können Akzeptanzkriterien verwendet werden, um zu beschreiben, wie die Lösung getestet werden soll, damit sie schließlich vom Kunden akzeptiert wird.

Tipp: Es ist auch auf höheren Abstraktionsebenen zu empfehlen, sich Gedanken über sinnvolle Akzeptanzkriterien zu machen. Dies kann sehr hilfreich für die weitere Spezifikation und Vervollständigung von Anforderungen sein. Die Frage, die man sich immer stellen sollte, ist: »Wie wird diese Anforderung nach Fertigstellung vom Kunden getestet?« Wenn man selbst oder der Kunde darauf keine Antwort weiß, sollte man folgende Anschlussfrage stellen: »Ist es Ihnen egal, wie diese Anforderung am Ende der Entwicklung umgesetzt oder aussehen wird? Wenn nicht, nennen Sie mir bitte die wesentlichen Punkte, die Ihnen nicht egal sind.«

Beispiel für die Ermittlung von Akzeptanzkriterien auf höheren Ebenen:

Ausgangspunkt ist eine Sammlung von Epics, bei denen z.B. folgendes Epic enthalten ist:

Epic: »Die Zeiterfassung wird effiziente Erfassungs- und Auswertungsmöglichkeiten für alle Zeitarten (Tagesarbeitszeit, Fehlzeiten etc.) bereitstellen.«

Die Frage an die Kundin ist: »Wie werden Sie diese Anforderung nach Fertigstellung testen bzw. abnehmen?«

Kundin: »So genau weiß ich das nicht. Ich werde schauen, ob ich die Zeiten eingeben und einen Bericht über die Tagesarbeitszeit inkl. Angabe der Fehlzeiten ausdrucken kann.«

Frage an die Kundin: »Ist es Ihnen egal, wie die Maske zur Erfassung der Arbeitszeit aussieht?«

Kundin: »Nein, ich möchte gerne eine Listenansicht, in der ich die Zeiten für mehrere Tage gleichzeitig eingeben kann. Außerdem soll automatisch eine Warnung kommen, wenn nicht die erforderliche Pausenzeit eingegeben wird.«

Frage an die Kundin: »Ist es Ihnen egal, wie der Bericht, der ausgedruckt wird, aussieht?«

Kundin: »Nein, ich möchte gerne eine Spaltenansicht mit Datum, Kommt-Zeit, Geht-Zeit, Pausendauer sowie Fehlzeiten. Außerdem möchte ich die Liste nach Mitarbeitende und Monat sortieren und gruppieren können und unten soll noch eine Summe der Zeiten über den Kalendermonat ausgegeben werden.«

Diesen Dialog könnte man noch beliebig weiter fortsetzen.

Er soll zeigen, dass durch einige wenige gezielte Fragen aus einer recht unklaren übergreifenden Anforderung in Form eines Epics durchaus hilfreiche Erkenntnisse gewonnen werden können, was sich die Kundin letztendlich darunter vorstellt.

→

Diese binnen weniger Minuten gesammelten wertvollen Zusatzinformationen könnten nun auf zwei Arten weiterverarbeitet und dokumentiert werden:

a) Wenn sich das Projekt noch in einer Phase der Ideenfindung oder groben Spezifikation befindet, könnten diese Punkte als Akzeptanzkriterien formuliert zum Epic dazu notiert und vorerst nicht weiter verfeinert werden (um sich nicht in Details zu verlieren).
b) Wenn diese Fragen im Rahmen der Anforderungsverfeinerung (z.B. als Vorbereitung für ein klar definiertes Sprint Backlog) stattfinden, dann ist es sinnvoll, diese Hinweise z.B. als User Stories zu notieren und auch noch durch weitere Fragen zu ergänzen und zu verfeinern.

3.1.7 Definition of Ready & Definition of Done

[IREB RE@Agile Primer] LE 3.1.10

Die »Definition of Ready« (DoR) und »Definition of Done« (DoD) sind Artefakte, die den Prozess der Entwicklung unterstützen und die Qualität der Anforderungen und Produktinkremente sicherstellen sollen.

Die DoR ist ein Quality Gate für den *Eingang* des agilen Entwicklungsprozesses. Sie soll helfen, die Qualität, den Wert und auch den Detaillierungsgrad und Informationsgehalt der Anforderungen, die im Sprint umgesetzt werden, zu verbessern und eine Überlastung des Teams mit unqualifizierten Anforderungen zu vermeiden.

Die DoD ist im [Scrum Guide] als »Commitment« definiert. Sie ist ein Quality Gate für den *Ausgang* des agilen Entwicklungsprozesses.

Detailliertere Informationen finden sich in den Abschnitten 5.2 und 5.3.

3.1.8 Prototyp vs. Inkremente

[IREB RE@Agile Primer] LE 3.1.11

Viele Stakeholder wollen keine Dokumente schreiben oder lesen, sie möchten kurzfristige Erfolge in Form von etwas »Greifbarem« sehen. Hierzu bieten sich Demonstrationen eines lauffähigen (Teil-)Systems an, um Feedback einzuholen und die Bedürfnisse zu erkennen. Dabei können Prototypen eingesetzt und mit Produktinkrementen gearbeitet werden (siehe auch Abschnitt 3.7).

Der Begriff »Spike« (siehe auch Abschnitt 3.8.1) wird im agilen Kontext für eine Entwicklungsiteration (oder einen Teil davon) verwendet, die sich explizit damit befasst, das Verständnis für einen komplexen Bereich zu verbessern (z.B. Systemarchitektur oder Integration von Fremdkomponenten) und damit das Risiko zu reduzieren. Es kann sich auf die Validierung einer oder mehrerer Aufgaben oder einer ganzen Iteration beziehen.

Prototyping ist eine mögliche Technik innerhalb von Spikes. Hier stehen das Sammeln von Wissen und Erfahrungen im Vordergrund und nicht das Erstellen von funktionierendem Code.

3.1.9 Ermittlung

[IREB RE@Agile Primer] LE 3.2.1

Für die Ermittlung (engl. »Elicitation« [IREB Glossar]) von Anforderungen[2] gibt es viele verschiedene Techniken: Interviews, Fragebögen, Beobachtungstechniken, artefaktbasierte Nachforschung (Systemarchäologie, Dokumentenanalyse, Wiederverwendung von bestehenden Systemen etc.), Prototyping und auch Kreativitätstechniken wie Brainstorming oder Design Thinking.

In der agilen Entwicklung setzt man beim Requirements Engineering oft auf eine intensive Kommunikation zwischen allen Stakeholdern (einschließlich der Entwicklung), um Anforderungen zu ermitteln. Dies kann als gute Mischung zwischen Interviews und kreativem Brainstorming angesehen werden. Techniken wie der »On-Site-Customer« fokussieren z. B. durch Besprechungen zwischen dem Product Owner und den Entwicklerinnen im Rahmen des Backlog Refinement ebenfalls auf eine gute Anforderungsermittlung. Ziel ist es in allen Fällen, besser zu verstehen, was der Kunde wirklich will und benötigt. Es kann hilfreich sein, beim agilen Vorgehen neben der direkten Kommunikation nach Bedarf auch andere der oben genannten Techniken anzuwenden.

Der Einsatz von Prototypen und Produktinkrementen ist ebenfalls eine gute Möglichkeit, mehr über die Anforderungen zu erfahren (siehe auch Abschnitt 3.7). Durch die Präsentation eines Produktinkrements im Sprint-Review werden die Beteiligten zu neuen Ideen angeregt, die dann wieder in das Product Backlog aufgenommen werden können.

Das Requirements Engineering in der agilen Entwicklung kann viel von den bereits etablierten Vorgehensweisen des RE, wie z. B. verschiedenen Ermittlungstechniken, profitieren. Das agile Team kann diese Techniken bewusst nach Bedarf auswählen und anwenden, um schneller und effektiver Anforderungen zu erkennen und zu ermitteln. Die intensive Kommunikation zwischen Team und Stakeholdern sowie das schnelle Feedback über das Bereitstellen von Produktinkrementen in agilen Vorgehensweisen sind grundsätzlich gute Basistechniken. Für gutes Requirements Engineering benötigt es jedoch mehr. Man sollte nicht übersehen, dass es noch weitere Aspekte zu beachten gibt: Bei Projekten mit Hunderten oder Tausenden von Stakeholdern wird eine rein verbale Kommunikation nicht mehr ausreichen. Für die Anregung zu und Ermittlung von innovativen unterbewussten Anforderungen sind oft zusätzliche Kreativitätstechniken erforderlich. Um neue Technologieideen zu generieren und zu untersuchen, sind Spikes hilfreich (siehe Abschnitt 3.8.1). Wenn man nicht genug Zeit für intensive verbale Erhebung und Diskussion hat (z. B. unter Zeitdruck), dann können ergänzende Techniken wie

2. Das Thema »Anforderungsermittlung« wird nur implizit in einzelnen Stellen im Text angesprochen. Für allgemein gebräuchliche Techniken und Methoden zur Anforderungsermittlung wie Brainstorming, Interviews etc. wird hier auf die einschlägige Literatur, z. B. [Pohl & Rupp 2021], verwiesen.

z. B. Requirements-Erhebungstabellen (Listen mit den wichtigsten zu erhebenden Requirements-Attributen) oder Requirements-Erhebungskarten (vorbereitete Karten, die ebenfalls die wichtigsten Requirements-Attribute enthalten – von manchen Methodenberatern auch Snowcards genannt) angewendet werden. Dies hilft Zeit zu sparen durch strukturiertes Vorgehen beim Erheben der Requirements.

Ein wichtiger Aspekt bei der Requirements-Ermittlung besteht darin, alle Quellen von Anforderungen zu kennen und diese in die Ermittlung einzubeziehen. Dies wird durch eine klare Zieldefinition (siehe auch Abschnitt 3.2.2) die Stakeholder-Analyse (siehe auch Abschnitt 3.2.4) und die Systemkontextanalyse (siehe auch Abschnitt 3.2.3) sichergestellt.

[IREB Advanced Level RE@Agile – Practitioner/Specialist] LE 3.2

RE@Agile strebt »Just-in-Time-Anforderungen« an [IREB RE@Agile Primer]. Um zu bestimmen, welche Anforderungen ausreichend detailliert definiert werden sollten, ist eine Übersicht erforderlich.

Die Vision und die Ziele (siehe Abschnitt 3.2.2) reichen möglicherweise nicht aus, um solche Entscheidungen zu treffen. Daher sollte sich der Product Owner einen möglichst breiten Überblick über alle Anforderungen des Systems verschaffen und nur bei den wichtigsten Anforderungen auch in die Tiefe gehen. Dies wird oft als »T-Ansatz« bezeichnet, da dieser breite Überblick dem horizontalen Balken des Buchstabens T entspricht, während der Drilldown[3] der Anforderungen, die den höchsten Geschäftswert versprechen, dem vertikalen Balken des Buchstabens T gleichkommt.

Unterschiedliche Kriterien können verwendet werden, um die Vision oder die Ziele zu zerlegen und daraus Anforderungen auf hoher Ebene zu erhalten, die – zusammen betrachtet – den beabsichtigten Umfang abdecken. Manche Methoden schlagen eine prozessorientierte Dekomposition vor, d. h. die Aufteilung der Funktionalität in Geschäftsprozesse, Anwendungsfälle oder große Stories (siehe auch Abschnitt 3.3). Diese Art der Dekomposition erfüllt die ersten drei Kriterien von INVEST (siehe Abschnitt 5.1.3), d. h., die resultierenden Prozesse sind unabhängig (independent), verhandelbar (negotiable) und – am wichtigsten – werthaltig (valuable).

Alternative Dekompositionsstrategien könnten auf Geschäftsobjekten, auf Subsystemzerlegung eines bestehenden Systems (d. h. ein versionsbasierter oder designbasierter Ansatz), Hardware- oder geografische Verteilung von Subsystemen basieren.

Während die Ergebnisse einer prozessorientierten Dekomposition meist als Use Cases oder User Stories bezeichnet werden, werden die resultierenden Teile der alternativen Dekomposition oft Epics, Themes oder Features genannt.

3. Als Drilldown wird im Allgemeinen die Navigation in hierarchischen Daten bezeichnet.

Wie auch immer die großen Funktionsbrocken heißen, sie bieten eine gute Grundlage für eine (grobe) Schätzung und können möglicherweise auch schon eingeplant werden, wodurch eine Roadmap oder ein vorläufiger Zeitplan für Iterationen (oder Sprints – in der Scrum-Terminologie) erstellt wird.

Alle diese High-Level-Anforderungen müssen detailliert werden, bevor sie von den Entwicklerinnen implementiert werden können (siehe auch Abschnitt 3.4 ff.).

Für die Kommunikation und Dokumentation werden Stories, Epics, Features oder Themes häufig auf physischen Karten festgehalten, die im Product Backlog gesammelt werden. Alternativ stehen viele Tools zur Verfügung, um diese Anforderungen elektronisch zu erfassen. Sobald Anforderungen auf höherer Ebene in eine Reihe von Anforderungen auf niedrigerer Ebene verfeinert werden, werden Anforderungshierarchien erstellt. Im Idealfall kann der Product Owner die verschiedenen Ebenen verwalten, ohne dass die übergeordneten Gruppierungen danach verloren gehen.

Story Maps (siehe Abschnitt 6.5.3) sind eine Möglichkeit, Karten zweidimensional anzuordnen und zu visualisieren, sodass Epics und Features auch dann noch sichtbar sind, wenn sie zu User Stories verfeinert wurden.

3.1.10 Dokumentation

[IREB RE@Agile Primer] LE 3.2.2

Agile Methoden verwenden oft den Begriff »Backlog« (siehe auch Abschnitt 6.5.1) für das Requirements-Management-Artefakt, das die Anforderungen der Requirements-Dokumentation erfüllt.

Im agilen Kontext kann das Backlog als ein Kern der Dokumentation von Requirements gesehen werden. Im Vergleich zu nicht agilen Projekten ist dies jedoch durch die Dokumentation der Anforderungen als Story Cards, die eine Priorität, einen Geschäftswert und ggf. weitere Informationen enthalten, ein eher informeller Ansatz.

Für die Förderung und Verbesserung der Kommunikation zwischen allen Stakeholdern ist die Dokumentation der Anforderungen eine wichtige Aktivität. Gerade im agilen Umfeld wird der Formalismus für die Dokumentation minimiert, indem Details verbal kommuniziert werden. Das Team sollte sich abhängig vom Projektumfeld auf einen adäquaten Dokumentationsgrad verständigen. Dies kann von vielen Faktoren abhängen, wie Größe der Projekte, Anzahl der beteiligten Akteure, rechtliche Einschränkungen oder Kritikalität des Projekts.

Ziel sollte sein, eine übermäßige Dokumentation zu vermeiden, aber trotzdem ein Mindestmaß an nützlicher Dokumentation sicherzustellen.

Die Arbeit mit einem »lebenden« Backlog ist zwar ein effizienter Weg der Dokumentation, aber nicht immer ausreichend. Wenn z.B. aufgrund rechtlicher Rahmenbedingungen eine langfristige Rückverfolgbarkeit zwischen Anforderungen, Design, Quellcode und Testdaten erforderlich ist, ist eine entsprechende Archivierung und Nachvollziehbarkeit auch im Backlog notwendig. Backlog-Objekte dürfen dann nicht gelöscht werden und müssen zur Dokumentation aufbewahrt werden. In diesen Fällen ist es empfehlenswert, anstatt physischer Tafeln und Karten zur Backlog-Verwaltung eine passende Software zu verwenden.

Generell ist das Dokumentieren von Anforderungen kein Selbstzweck. Es sollte die Kommunikation zwischen allen Beteiligten, insbesondere zwischen der Kundschaft (oft durch den Product Owner repräsentiert) und der Entwicklung, erleichtern. Gerade in agilen Projekten ist eine gute verbale Kommunikation ein wichtiges Mittel, um Dokumentationsaufwände zu minimieren. Dennoch muss auch bei agilen Projekten eine angemessene Dokumentation von Anforderungen und begleitenden Informationen (z.B. über Entscheidungen und Annahmen) vorhanden sein. Es gibt noch weitere Dokumentationsarten und -zwecke.

Aus der Requirements-Engineering-Sicht kann man folgende Arten von Dokumentationen unterscheiden:

- **Dokumentation für rechtliche Zwecke**
  In bestimmten Projekten oder Branchen (z.B. in einem sicherheitskritischen Umfeld wie Medizin, Bahntechnik, Automotive, Flugbranche) ist es notwendig, gewisse Dokumentationsvorschriften einzuhalten (z.B. der Nachweis von spezifizierten Anforderungen und Testfällen für ein System sowie deren Zusammenhang), um sich rechtlich abzusichern und die Haftungen zu reduzieren.

  Hier gilt: *Die rechtlich notwendige Dokumentation muss von den zugrunde liegenden Normen oder Gesetzen abgeleitet werden und ist ein untrennbarer Bestandteil des Produkts.*

- **Dokumentation zum Zweck der Nachhaltigkeit (Bewahrung)**
  Gewisse Informationen über ein System sind es wert, erhalten zu bleiben, damit diese für spätere Aufgaben (z.B. Wartung) verfügbar sind. Dies können z.B. die zentralen Ziele oder Anwendungsfälle oder auch Nichtanwendungsfälle des Systems sein (inkl. Begründung).

  Diese Dokumentation kann zum gemeinsamen »Wissensspeicher« werden, der dem Team oder der Organisation hilft, eigene »Speicherkapazitäten« freizubekommen oder die Diskussionen über früher getroffene Entscheidungen zu vereinfachen (»Warum haben wir uns damals entschieden, das nicht zu implementieren?«).

  Hier gilt: *Das Team entscheidet, was zum Zweck der Nachhaltigkeit dokumentiert wird.*

- **Dokumentation für Kommunikationszwecke**
  Verbale oder direkte Face-to-Face-Kommunikation ist durch die Interaktivität ein wichtiges Werkzeug agiler Methoden. In der Praxis gibt es aber verschiede-

ne Situationen, die verbale Kommunikation verhindern oder erschweren (z. B. verteilte Projekte, Zeitrestriktionen der involvierten Personen). Außerdem sind Informationen manchmal so komplex, dass verbale Kommunikation ineffizient ist. Ein geschriebener Text oder ein Diagramm kann immer wieder gelesen werden (z. B. wenn ein komplexer Algorithmus nicht gleich verstanden wird oder auch wenn einzelne Punkte von den Lesenden wieder vergessen wurden). Manchmal bevorzugen Stakeholder es auch, eine schriftliche Dokumentation zu lesen, anstatt den Sourcecode oder die laufende Software zu reviewen.

Das Prinzip der Dokumentation für Kommunikationszwecke:
Diese Dokumentation wird dann erstellt, wenn Stakeholder oder die Entwicklerinnen darin einen Wert für die Kommunikation sehen. Das Dokument kann wieder verworfen werden, wenn die Kommunikation erfolgreich durchgeführt wurde.

- **Dokumentation zum Zweck des Nachdenkens**
  Ein oftmals vergessener Aspekt der Dokumentation ist, dass diese auch immer ein Mittel ist, um den Denkprozess der Person, die den Text verfasst, zu verbessern und zu unterstützen. Auch wenn die Dokumentation später im Prozess weggeworden wird, so bleibt doch der Nutzen der Verbesserung und Unterstützung. Zum Beispiel zwingt das Schreiben von Use Cases dazu, über konkrete Interaktionen zwischen dem System und den Aktoren nachzudenken, die dann z. B. auch Ausnahmen und alternative Szenarien umfassen. Das Niederschreiben von Use Cases kann daher auch als Werkzeug verstanden werden, um das eigene Wissen und Verständnis für das System zu überprüfen.

  Das Prinzip der Dokumentation zum Zweck des Nachdenkens:
  Die Person, die den Text schreibt, entscheidet, welche Form der Dokumentation für ihren Prozess des Nachdenkens am besten geeignet ist. Sie muss dies auch nicht anpassen. Wenn der Prozess beendet ist, kann die Dokumentation wieder verworfen werden.

Agile Methoden können wesentlich davon profitieren, wenn diese vier Arten der Dokumentation unterschieden und im entsprechenden Kontext angewendet werden. Vor allem die Unterstützung für die Kommunikation und das Nachdenken werden oftmals unterschätzt.

[IREB Advanced Level RE@Agile – Practitioner/Specialist] LE 3.6

Für das Projektteam sind Story Cards oft ausreichend zur Erinnerung an die laufenden Diskussionen, um als Grundlage für die Entwicklung zu dienen [Cohn 2010]. Für Entwicklerinnen, die kontinuierlich an einem Projekt beteiligt waren, reicht oft der Code selbst aus, um zu verstehen, was zuvor getan wurde.

Neben den direkt an der Projektentwicklung Beteiligten gibt es jedoch viele andere Stakeholder – der Kunde, das Unternehmensumfeld, Supportteams, verwandte Projektteams usw. –, denen Code und User Stories nicht genügend Kontext und

Struktur bieten, um zu verstehen, wie das Produkt schließlich aussehen bzw. sich auf sie auswirken wird. Diese Stakeholder repräsentieren daher unterschiedliche Zielgruppen für die Dokumentation.

Darüber hinaus wird das Produkt, das aus einem Entwicklungsprojekt hervorgeht, selbst (hoffentlich) ein Leben über das Projekt hinaus haben und kann sich tatsächlich über mehrere Entwicklungsprojekte hinweg weiterentwickeln.

Requirements-Engineering-Artefakte können in der Regel nur dann als relevant für das Projekt oder als Teil der Produktdokumentation klassifiziert werden, wenn sie für die zukünftige Verwendung aufbewahrt werden sollen. Die Trennung von Produktbelangen und Projektbelangen kann eine gute Ausgangsbasis für eine nachhaltige Dokumentation darstellen.

In Übereinstimmung mit den agilen Prinzipien sollte nur dann Aufwand für die Dokumentation getrieben werden, wenn diese Dokumentation einen Stakeholder hat, der diese benötigt und von dem bei der Entwicklung dieses Artefakts regelmäßig Feedback eingeholt wird. Techniken zur Minimierung des Dokumentationsaufwands insbesondere für produktorientierte Artefakte können z.B. die Erstellung dedizierter, Wiki-orientierter Dokumentationssysteme sein, die sich im Laufe der Zeit weiterentwickeln.

Dokumentation soll kein Selbstzweck sein, sie soll die Kommunikation zwischen den Stakeholdern, die Nachhaltigkeit und rechtliche Anforderungen unterstützen!

3.1.11 Artefakte

Die in diesem Buch beschriebenen Artefakte, Techniken und Methoden zur Ermittlung und Dokumentation von Anforderungen sind bewusst in mehreren Blöcken zusammengefasst und in einer Reihenfolge angeführt, die bei ihrer Verwendung einer Vorgehensweise vom Groben ins Detail entspricht.

Die Verfeinerung von Anforderungen erfolgt in fünf Ebenen:

- **Übergeordnete Sicht und Kontext**
  (Abschnitt 3.2, »Übergeordnete Artefakte«)
- **Prozesse**
  (Abschnitt 3.3, »Geschäftsprozesse und Systemverhalten«)
- **Funktionen und Qualitätsanforderungen**
  (Abschnitt 3.4, »Funktionale und nicht funktionale Sicht«)
- **Gestaltung des User Interface**
  (Abschnitt 3.5, »Benutzerschnittstelle«)
- **Technische Anforderungen**
  (Abschnitt 3.6, »Systemschnittstelle« und Abschnitt 3.8, »Entwicklersicht«)

In einem Projekt ist es oft zielführend, in dieser Reihenfolge vorzugehen, da eine zu frühe und vielleicht noch nicht notwendige Festlegung auf Details den möglichen Lösungsraum für die Umsetzung zu stark einschränkt. Dies ist auch eine gute Herangehensweise, um sich nicht schon am Anfang durch eine zu starke Detailsicht zu »zerfransen« oder den Projektfokus zu verlieren.

In der Praxis können und sollen die Techniken und Methoden nach Bedarf an jeder beliebigen Stelle und zu jedem beliebigen Zeitpunkt im Entstehungsprozess angewendet werden, und zwar so, wie es für die jeweilige Situation am effektivsten ist. Erfahrungsgemäß sind zusätzlich zum erstmaligen Durchlauf durch die genannten Spezifikationsebenen zumindest ein bis zwei Iterationen über die Ebenen Prozesse, Funktionen und User Interface erforderlich, um die Requirements vollständig und konsistent zu erhalten.

Eine Frage ist auch oft, welche Beschreibungsebene man für eine von einem Stakeholder genannte Anforderung wählen soll? Soll man z. B. die Anforderung »Tagesarbeitszeit buchen« als Prozessschritt, als Szenario, als User Story oder Use Case oder schon als GUI-Element spezifizieren? Es gibt hier keine eindeutige Regel, denn dies hängt von der Komplexität des Systems ab und zusätzlich auch davon, welche Bedeutung das Element im Gesamtkontext des Systems hat.

Nachfolgend sind für diese Fragestellung zwei kleine Beispiele und Möglichkeiten zur Spezifikation beschrieben:

Situation A: ERP-System-Implementierung

Das ERP-System besteht aus vielen Modulen, von denen die Zeiterfassung nur ein kleines ist. Es gibt viele übergeordnete Businessprozesse, in denen das Requirement »Tagesarbeitszeit buchen« als eines von Tausenden anderen Requirements vorkommt. Die »Zeiterfassung« wird als untergeordneter, wenig kritischer Systemteil eingestuft.

In diesem Fall wäre es empfehlenswert, das Requirement »Tagesarbeitszeit buchen« als Schritt in einem Subprozess oder Szenario zu modellieren. Eine weitere Verfeinerung vor der Implementierung wird in diesem Fall wahrscheinlich nicht durchgeführt, sondern die Details dieses »kleinen« Punkts werden der Entwicklung überlassen.

Situation B: Zeiterfassungssystem-Implementierung

In einer Produktentwicklung soll explizit ein neues Zeiterfassungssystem als Kernprodukt entwickelt werden. Im neu zu implementierenden System gibt es einige Businessprozesse. Die Anforderung »Tagesarbeitszeit buchen« ist dabei eine Kernfunktion.

In diesem Fall wird »Tagesarbeitszeit buchen« möglicherweise sogar ein Hauptprozess oder Hauptszenario sein und zum besseren Verständnis grafisch modelliert und mit einer Textspezifikation detaillierter beschrieben werden. Des Weiteren wird man sich hier auch noch Gedanken darüber machen, wie denn das Design dieser Funktion auf der Benutzeroberfläche aussehen soll, und auch schon die Maskengestaltung zumindest grob vorgeben.

3.1.12 Ein Blick auf das große Ganze

Zur Strukturierung und Einordnung von Requirements-Artefakten kann man grundsätzlich drei Ebenen unterscheiden:

- Die High-Level-Sichtweise, um den Überblick über das Vorhaben zu erlangen bzw. zu behalten
- Die Strukturierungsebene, auf der unterschiedliche Artefakte in einen Zusammenhang gebracht werden
- Die Detailebene mit den feingranularen Inhalten

In Abbildung 3–2 werden diese drei Ebenen dargestellt und mit den drei Sichten »Kunde«, »Entwickler*in« und »Management« kombiniert. In dieser Matrix werden die wichtigsten in der agilen Softwareentwicklung verwendeten Requirements-Artefakte im Überblick angegeben.

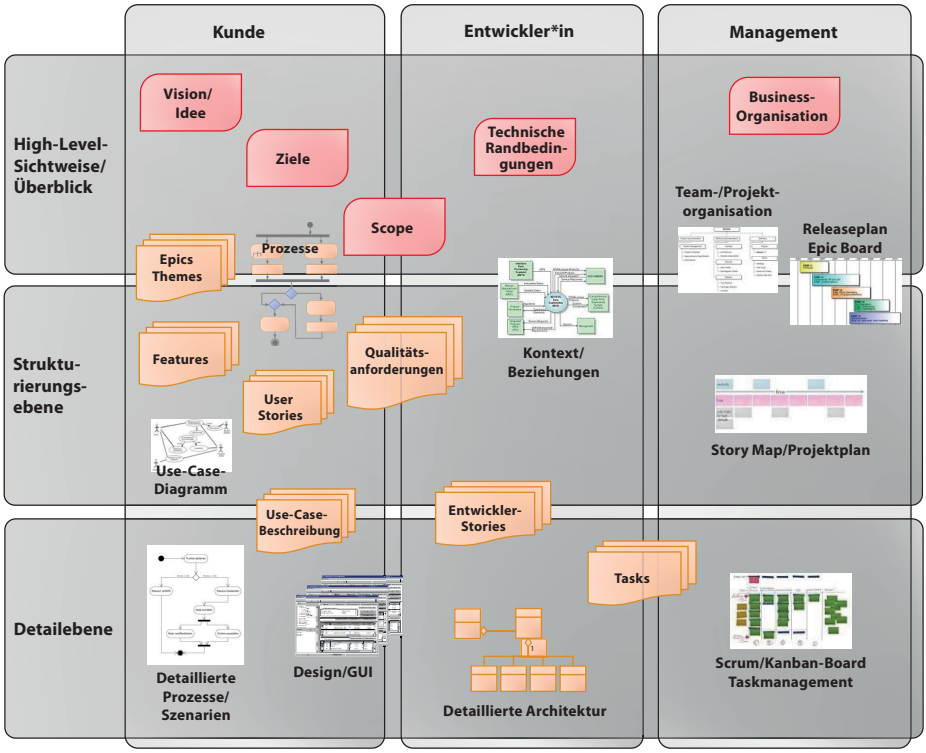

Abb. 3–2 *Überblick über Requirements-Artefakte im agilen Umfeld*

Die Zusammenhänge im Requirements Engineering sind fast immer komplexer, als es auf den ersten Blick aussieht (siehe Abb. 3–3). Um teamübergreifende Anforderungen, Einsatzszenarien bei den Anwendern und das große gemeinsame Ziel optimal im Blick behalten zu können, müssen die oft verwendeten User Stories sinnvoll

um weitere Requirements-Artefakte ergänzt werden. Jedes Artefakt steht dabei mit vielen verschiedenen anderen Artefakten in Beziehung.

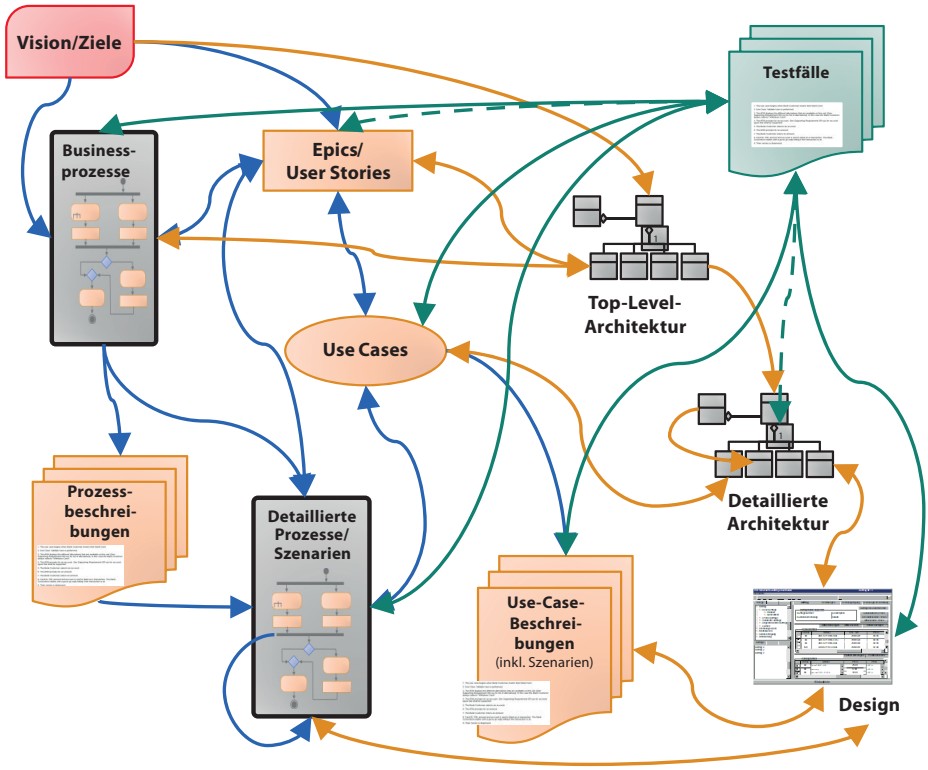

Abb. 3–3 *Komplexe Zusammenhänge im Requirements Engineering*[4]

Diese Artefakte und Beziehungen sollten in jedem Projekt berücksichtigt und gemanagt werden. Passiert dies nicht, sind sie zwar nicht mehr explizit sichtbar, aber trotzdem immer noch vorhanden. Das Nichtbeachten dieser Situation führt dann oft dazu, dass Ineffizienzen wie z. B. Leerlaufzeiten und Fehler, vergessene Anpassungen und Testfälle oder Inkonsistenzen zwischen Artefakten entstehen. Die Folge ist, dass dann z. B. der Tester nicht oder erst verzögert informiert wird, wenn sich an den Anforderungen etwas ändert, und daher die Testfallerstellung erst verspätet beginnen kann oder die falschen Testfälle erstellt werden.

4. Aus Gründen der Übersichtlichkeit sind in dieser Grafik nur die wichtigsten Requirements-Artefakte und Beziehungen dargestellt. Code-Artefakte, externe Rechtsgrundlagen und andere ggf. noch relevante Artefakte wurden weggelassen, müssen in der Praxis jedoch ebenfalls berücksichtigt werden.

Unabhängig davon, ob eine agile oder nicht agile Vorgehensweise in einem Projekt angewendet wird: Die komplexen Beziehungen zwischen den Projektartefakten bestehen in jedem Projekt und müssen entsprechend berücksichtigt werden.

In agilen Projekten versucht man, diese Zusammenhänge und Komplexitäten weniger durch explizite Dokumentation in passenden Artefakten und durch Verwaltung in geeigneten Tools zu beherrschen, sondern durch intensive persönliche Kommunikation. Die Kommunikationstheorie hat jedoch festgestellt – und auch in der Praxis zeigt sich dies immer wieder –, dass sich der Kommunikationsaufwand mit der Anzahl der Kommunikationsknoten und Beziehungen exponentiell erhöht. Schon bei einer einfachen und nicht alle Beziehungen eines Projekts umfassenden Abbildung wird ersichtlich, dass es nicht einmal mit sehr großem Kommunikationsaufwand möglich ist, alle Zusammenhänge zwischen den Kundenanforderungen und den restlichen Projektartefakten zu bewahren und konsistent zu halten. In der Praxis wird dies kaum ausschließlich auf Basis der persönlichen Kommunikation funktionieren.

Was in Projekten ohne explizite Dokumentation der Artefaktbeziehungen oft passiert, ist, dass ein Großteil der eigentlich für eine nachhaltige Entwicklung notwendigen Informationen nur unzureichend oder inkonsistent vorhanden ist. Speziell für Nichtteammitglieder sind diese Informationen nicht nutzbar, da sie zwar in den Köpfen von Teammitgliedern vorhanden sind, jedoch nur mit großem Aufwand an andere weitergegeben werden können. Außerdem können sie nicht strukturiert analysiert und geprüft werden. Das Wissen ist in den Köpfen »eingebunkert«. In vielen Projekten führt dies dann zu Problemen.

Die wesentlichen Beziehungen und Inhalte der durchgeführten Kommunikation sollten daher auch dokumentiert werden, um dem Wissensverlust entgegenzuwirken (siehe Grundprinzip 3 »Risiko und zeitlicher Abstand zur Umsetzung steuern Detailgrad« in Abschnitt 2.3 sowie Abschnitt 3.1.3). Des Weiteren sollten Aufwände, die zur Aufrechterhaltung der Beziehungen zwischen den Artefakten nötig sind, durch den Einsatz von passenden Werkzeugen (Requirements Management, Modellierung, Codeverwaltung, Testmanagement, Prozessautomatisierung etc.) reduziert oder vermieden werden.

3.2 Übergeordnete Artefakte

3.2.1 Zusammenhänge und Abhängigkeiten

[IREB Advanced Level RE@Agile – Practitioner/Specialist] LE 2.4

Bei der Definition von Vision und Zielen, Stakeholdern und Systemgrenze (siehe nachfolgende Abschnitte) sind wechselseitige Abhängigkeiten zu berücksichtigen:

- Die relevanten Stakeholder formulieren die Vision und die Ziele. Daher kann die Identifizierung eines neuen relevanten Stakeholders Auswirkungen auf die Vision oder die Ziele haben.
- Vision und Ziele können verwendet werden, um die Identifizierung neuer Stakeholder durchzuführen, indem Sie z. B. fragen: Welcher Stakeholder könnte an der Erreichung der Vision/der Ziele interessiert sein oder ist von der Erreichung der Vision/der Ziele betroffen?
- Vision und Ziele können verwendet werden, um einen ersten Umfang zu definieren, indem gefragt wird: Welche Elemente sind notwendig, um die Vision/die Ziele zu erreichen?
- Eine Änderung der Systemgrenze (und damit des Scopes) kann Auswirkungen auf die Vision/die Ziele haben. Wird ein Aspekt aus dem Anwendungsbereich entfernt, muss überprüft werden, ob das System noch in der Lage ist, die Vision/die Ziele zu erreichen.
- Stakeholder definieren die Systemgrenze. Daher kann sich die Identifizierung eines neuen relevanten Stakeholders auf die Systemgrenze und Scope auswirken.
- Eine Änderung des Scopes (z. B. zur Erfüllung eines Ziels) erfordert die Zustimmung der relevanten Stakeholder.

Diese gegenseitigen Abhängigkeiten sollten genutzt werden, um alle drei Elemente auszubalancieren und die Auswirkungen der Änderung eines der drei Elemente auf das andere zu untersuchen.

Aufgrund dieser engen Abhängigkeiten zwischen Vision und Zielen, Stakeholdern und Umfang sollten alle diese Elemente zeitgleich und auf kohärente Weise behandelt werden.

3.2.2 Vision und Ziele

[IREB RE@Agile Primer] LE 3.1.2
[IREB Advanced Level RE@Agile – Practitioner/Specialist] LE 2.1

Definition	Die **Vision** (lat. visio, »sehen«) ist als kreatives Wunschbild oder »Gesamtziel« eine sehr allgemeine, oft noch unkonkrete Darstellung oder Richtschnur dessen, was erreicht werden soll. Ein **Ziel** ist ein künftig angestrebter definierter Zustand bzw. ein Sachverhalt, den der Stakeholder erreichen möchte. Ziele sind konkreter gefasst und sollen klar messbar formuliert sein.
Anwendung	Vision und Ziele werden verwendet, um die strategische Ausrichtung und die groben Leitlinien des zu entwickelnden Produkts oder einer übergreifenden Produktlinie oder eines Produktportfolios zu beschreiben. Vision und Ziele sollen in jedem Entwicklungsprojekt vorhanden sein und sind im Grunde Requirements-Engineering-Artefakte auf höchster Ebene, wobei auf dieser Ebene oft nur grobe Aussagen möglich sind, die dann in tieferen Ebenen in detailliertere Sichten aufgeteilt werden.[a] Die Vision und Ziele sollen so früh wie möglich zwischen allen relevanten Stakeholdern vereinbart werden und sind für alle Entwicklungsaktivitäten von höchster Bedeutung.
Mitwirkende	■ Projektentscheider*innen auf Businessebene (z. B. Geschäftsführer*in, CEO, Portfoliomanager*in) ■ Entscheider*innen des Auftraggebers bzw. der Auftraggeberin ■ Produktmanager*in, Product Owner
Eigenschaften	■ Oberste Ebene ■ Grobe, übergreifende Aussagen ■ Leicht verständliche und umfassende Orientierung (»gemeinsame Mission«) für das gesamte (agile) Projekt ■ Inhaltliche Anforderungssicht und planende Sicht manchmal gemischt ■ **Wichtig als Leitplanken für alle untergeordneten Ebenen** ■ In jedem Projekt erforderlich ■ Anzahl: Eine Vision und ca. 3–7 übergeordnete Ziele, die sich alle Projektbeteiligten merken können ■ Achtung: Ziele können auch widersprüchlich sein und in Konflikt zueinander stehen. Dies sollte aufgelöst werden.
Testbarkeit	■ Vision: nein ■ Ziele sollten prüfbar oder zumindest bei der Abnahme nachvollziehbar formuliert sein.
Zeitpunkt	**Zum Projektstart**, nach Bedarf zwischendurch Anpassung
Vorlaufzeit bis Umsetzung	Wenige Monate bis ein Jahr **Hinweis:** Längere Zeiträume als ein Jahr sind meist nicht sinnvoll. Hier empfiehlt es sich, das Vorhaben in kleinere Einheiten aufzuteilen, die binnen max. einem Jahr zur Umsetzung gelangen können.

→

Hinweise	Vision und Ziele können auch als sogenanntes »Elevator Statement«[b] – also eine kurze knackige Aussage über Inhalt und Zweck des Vorhabens – verfasst werden. Im agilen Umfeld wird oft auch der Begriff »Produktvision« oder »Produkt-Vision-Statement« verwendet, um zu betonen, dass jedes Ergebnis des Entwicklungsprozesses einen eigenen Geschäftswert haben sollte, der die Produktvision unterstützt. Jede Anforderung soll zum Erreichen der Vision und Ziele beitragen [Scrum Guide]. **Tipp:** Erstellen Sie ein Plakat oder veröffentlichen Sie die Vision und Ziele auf der Startseite der Projekt-Website, um sie laufend präsent zu haben. Manchmal wird anstelle des Begriffs »Ziel« auch der Begriff »Problem« (das von einem bestimmten System gelöst werden soll) verwendet. Aus RE-Perspektive sind die Begriffe direkt miteinander verbunden: Ein Problem ist eine Aussage über eine unerwünschte vorhandene Eigenschaft, während ein Ziel die gewünschte Zukunft ausdrückt.

a. Im RE@Agile sowie [IREB Glossar] wird explizit unterschieden zwischen Ziel und Anforderung (Aussage über eine gewünschte Systemeigenschaft). Die Unterscheidung ist in der Praxis oft nicht ganz klar und vielfach »Geschmackssache«, daher sehen wir hier der Einfachheit halber Vision und Ziele als High-Level-Anforderungen.

b. Als »Elevator Statement« oder auch »Elevator Pitch« (dt. etwa »Aufzugsgespräch«) werden kurze, informative Darstellungen einer Idee, eines Vorhabens, einer Leistung oder auch eines Produkts bezeichnet. Der Begriff verdeutlicht, dass die Darstellung so effektiv erfolgen soll, dass die Aussage innerhalb einer Fahrt in einem Aufzug (also in ca. 30–45 Sekunden) wiedergegeben werden kann.

Vision und Ziele sind die »Leitplanken« für das Projekt und alle weiteren Requirements.

In jedem Entwicklungsprozess sollten die Vision und Ziele die Ausgangsbasis für die Produktdefinition sein. Die Lösung ist dann erfolgreich, wenn Vision und Ziele umgesetzt bzw. erreicht sind.

Bei Vision und Zielen kann man zwischen verschiedenen Sichtweisen, z.B. technische Sicht, Geschäftssicht, unterscheiden. Die technische Sichtweise berücksichtigt Aspekte, die meist produktbezogen sind, wie z.B. Technologie, Architektur, Wartbarkeit oder Betriebsinfrastruktur. Die Geschäftssicht betrachtet primär die Organisation, Prozesse und wirtschaftlichen Aspekte.

Beispiele für Visionen:

- **Technische Vision**
  »Mit der neuen Softwarelösung für die Arbeitszeiterfassung werden wir technologisch gegenüber unserem direkten Mitbewerber führend sein.«
- **Wirtschaftliche Vision**
  »Die Nutzer können durch den Einsatz des Systems die Produktivität im Bereich Zeiterfassung vervielfachen.«

→

Beispiele für Ziele:

- **Technisches Ziel**
  »Bis Ende des nächsten Jahres ist das neue System auf Microsoft Windows und iOS verfügbar.«
- **Wirtschaftliches Ziel**
  »Durch die Ablösung der papierbasierten Zeiterfassung werden die Aufwände für die Zeiterfassung und monatliche Analyse und Auswertung der Daten um mindestens 50% reduziert.«

Vision und Ziele beziehen sich in der Formulierung grundsätzlich auf das Gesamtsystem bzw. Gesamtvorhaben.

Abgeleitet von den übergeordneten Zielen werden manchmal auch Ziele mit unterschiedlicher Granularität und Zeitintervall vereinbart (z.B. Ziele für die Roadmap von einem Jahr, Ziele für das nächste Release von drei Monaten und Ziele für einen Sprint für die nächsten zwei bis vier Wochen (abhängig von der Sprint-Länge). Wichtig dabei ist, dass die untergeordneten (Teil-)Ziele immer im Einklang mit dem Gesamtziel bzw. der Gesamtvision stehen.

In Vision und Zielen können organisatorische, zeitliche, finanzielle, qualitative und funktionale Aspekte enthalten sein. Eine Trennung ist auf dieser Ebene nicht so wichtig. Viel wichtiger ist, dass die wesentlichen Ziele dargestellt und an alle Stakeholder klar kommuniziert werden (z.B. auch zeitlich eingeordnet auf einer Visions-/Ziele-Roadmap). Eine saubere Aufteilung in eine inhaltliche Sicht (Prozesse, Funktionen, Qualitätseigenschaften etc.) und eine Managementsicht (Projektorganisation, Aufwand, Aufgabenplanung etc.) soll dann auf den untergeordneten Ebenen erfolgen.

Die Ziele von verschiedenen Stakeholdern können zueinander widersprüchlich sein. Hier ist ein Ausgleich zwischen den Beteiligten erforderlich, um eine gemeinsam vereinbarte Vision oder ein gemeinsames Ziel zu erreichen. Alternativ kann man auch entscheiden, Varianten eines Produkts anzubieten (z.B. ein Zeiterfassungssystem für Projektdienstleister und eine Variante für Produktionsbetriebe) oder eine Trennung in verschiedene Produkte vorzunehmen (z.B. eine eigene Desktop-Zeiterfassung und eine mobile App zur Zeiterfassung).

Vision und Ziele für das aktuelle Entwicklungsvorhaben sind normalerweise abgeleitet aus einer übergeordneten Vision und Zielen, wie z.B. Businessvision und -ziele, Geschäftsstrategie, können aber auch durch eine persönliche Vision und Ziele maßgeblicher Stakeholder beeinflusst werden. Sie sind wichtig, um allen Beteiligten und insbesondere auch den Entwicklerinnen deutlich zu machen, worauf bei dem Vorhaben besonders Wert gelegt wird und was die wichtigsten Bedürfnisse und Anliegen der Auftraggeberin sind. Außerdem können hier evtl. auch schon Vorgaben für grundlegende Architektur- und Designentscheidungen abgeleitet oder in eine bestimmte Richtung gelenkt werden.

Jedes weitere inhaltliche Requirement und jeder Managementaspekt im weiteren Projektverlauf soll implizit oder explizit gegen die definierte Vision und vereinbarten Ziele geprüft werden, um festzustellen, ob ein Beitrag dazu geleistet wird.

Eine Anforderung ohne Beziehung zu einem Ziel liegt außerhalb des Scopes und sollte bewusst ausgegrenzt bzw. ggf. in ein Nachfolgevorhaben ausgelagert werden. Wenn solche Anforderungen trotzdem bestehen bleiben und von Stakeholdern »verteidigt« werden, ist dies entweder ein Indikator für »Goldplating«[5] oder dafür, dass die Ziele bzw. der Scope des Projekts noch nicht richtig definiert ist. In beiden Fällen ist Anpassungsbedarf gegeben.

Vision und Ziele sind daher sehr wichtige Artefakte für den Projekterfolg und müssen sehr sorgfältig formuliert werden. Sie bestimmen den Rahmen für alle Projektaktivitäten, ohne die Kreativität in der Umsetzung unnötig einzuschränken.

Ändern von Vision und Zielen

Wichtig ist, dass auch Vision und Ziele nicht als »in Stein gemeißelte« Gebote gesehen werden. Die Projektsituation (z. B. andere oder neue Stakeholder oder gravierende Änderungen am Systemkontext) oder die Sichtweise der Auftraggeberin kann sich durchaus ändern und auch die Vision und Ziele können und sollen angepasst werden. Dabei ist zu beachten, dass bei einer Änderung der obersten Vision und Ziele alle untergeordneten Requirements und auch der Zusammenhang mit der übergeordneten Vision und den Zielen neu geprüft werden muss. Dies kann bei komplexen Systemen mit viel Aufwand verbunden sein und dazu führen, dass viel bereits getane Arbeit umsonst gemacht wurde.

Vision und Ziele können durchaus auch eine gewisse Dynamik haben und sich an Veränderungen oder ein anderes Verständnis der Stakeholder anpassen. Dies soll jedoch kein Freibrief zum laufenden Ändern der Produktvision sein, sondern mit Bedacht und nur bei wirklich relevanten Änderungen durchgeführt werden.

Da es sich bei Vision und Zielen um sehr zentrale und übergeordnete Elemente für das gesamte Vorhaben handelt, sollte die Visions- und Zielformulierung wohl überlegt sein und – soweit dies möglich ist – auch durch Reviews und Diskussionen mit den Stakeholdern möglichst frühzeitig am Beginn eines Projekts festgelegt und abgesichert werden.

Es muss den Entscheidungsträgern klar sein, dass Änderungen an zentralen Projektgrundlagen oder Zielen – vor allem dann, wenn sie massive Auswirkungen auf die technische Architektur des Systems haben – mit sehr hohen Kosten verbunden sind!

---

5. Der Begriff »Goldplating« wird verwendet, wenn an einem Projekt oder einer Aufgabe über den Punkt hinaus gearbeitet wird, an dem der Nutzen wieder abnimmt. Es bedeutet das Hinzufügen von Funktionen, die im ursprünglichen Plan oder in der Produktspezifikation nicht vorgesehen waren. Grundsätzlich heißt das jedoch nicht, dass keine neuen Features mehr hinzugefügt werden dürfen; sie können jederzeit hinzugefügt werden, solange sie dem definierten Entwicklungsprozess folgen und die Auswirkungen der Änderung im Projekt berücksichtigt werden (in Anlehnung an [Wikipedia: Goldplating]).

Beispiel für eine gravierende Zieländerung:

- **Geändertes technisches Ziel**
  »Bis Ende des nächsten Jahres wird das neue System auf Microsoft Windows und iOS **sowie auf Android, Windows Phone und Linux** verfügbar sein.«

Zur Vereinfachung des Änderungsmanagements bei Vision und Zielen kann man auch mit unterschiedlichen Zeithorizonten und daran angepasst auch unterschiedlichen Änderungsverhalten arbeiten. So kann zum Beispiel mit unterschiedlichen Visions- und Zieldokumenten für Sprints, Releases und einen langfristigen Produktausblick gearbeitet werden. Auch das bewusste Fokussieren auf einen bestimmten abgegrenzten Zeitraum von z. B. einem ½ Jahr und das anschließende Evaluieren und ggf. auch komplette Ändern der Entwicklungsrichtung ist legitim, wenn dies allen Beteiligten klar ist (z. B. bei Lean-Startup-Ansätzen).

Wie auch immer man bezüglich Vision und Zielen vorgeht, es sollte möglichst klar und frühzeitig mit allen Stakeholdern abgestimmt sein.

Definition und Messbarkeit

Eine Vision kann durchaus auch »visionär« sein und muss nicht immer klar messbar formuliert sein. Sie soll vielmehr auch einen motivierenden Charakter haben. Aus der oft allgemein und unkonkret angegebenen Vision leiten sich dann konkrete Ziele ab. Diese Ziele sollen im Gegensatz zur Vision »SMART« formuliert sein. Leider gibt es in der Literatur keine einheitliche Definition dafür. Nachfolgend einige gängige SMART-Varianten:

	[Wikipedia: SMART]	[Scrum.org]	[Wolf & Roock 2021] [IREB RE@Agile AL]	Beschreibung
S	Specific	Specific	Specific	Ziele müssen eindeutig definiert sein (nicht vage, sondern so präzise wie möglich).
M	Measurable	Measurable	Measurable	Ziele müssen messbar oder zumindest überprüfbar sein (Messbarkeitskriterien).
A	Achievable, Accepted, Attainable	Assignable	Achievable	Ziele müssen von den Empfängern akzeptiert werden (auch: angemessen, attraktiv, ausführbar) und sie müssen erreichbar sein.
R	Reasonable, Relevant	Realistic	Relevant	Das gesteckte Ziel muss relevant, angemessen und realistisch sein.
T	Time bounded	Time related	Time based	Zu jedem Ziel gehört eine klare Terminvorgabe, bis wann das Ziel erreicht sein muss.

Die gute Formulierung der Ziele wird am einfachsten über eine Checkliste geprüft, die die Kriterien für die Formulierungsansätze beinhaltet. Die Verwendung von Schablonen oder Formularen mit entsprechend vorbereiteten Feldern ist auch möglich.

Neben dem SMART-Schema gibt es noch verschiedene andere Ansätze zur Zielformulierung:

- **PAM-Zielformulierung** [Robertson2003]
  Die Stakeholder beantworten folgende Fragen nach ihren Zielen:
  - P = Purpose
    Eine Ein-Satz-Beschreibung des Zwecks des Vorhabens.
  - A = Advantage
    Was ist der Geschäftsvorteil/Mehrwert?
  - M = Measure
    Wie würden wir den Geschäftsvorteil messen?
- **Product Vision Box/Design the Box** [Highsmith2004]
  Nach diesem Ansatz wird eine (physische) Box für das Produkt erstellt, die die zentralen Vorteile/Ideen des Produkts für potenzielle Kunden präsentiert. Der visionäre Anteil spielt hier eine große Rolle.

  Der Ursprung liegt bei Bill Shackelford und Jim Highsmith (einer der ursprünglichen Unterzeichner des Agilen Manifests und Vater des »Adaptive Software Development« (ASD)). Die Vision-Box-Technik wird auch »Design the Box«-Technik genannt und wird als Teil der Visionsphase (engl. Envision Phase) der Produktentwicklung gesehen.

  Zur Vereinfachung des Prozesses können sich die Teilnehmenden bzw. Teammitglieder auch an den klassischen W-Fragen orientieren. Die Ergebnisse werden dann auf bzw. mit der Box präsentiert:
  - Wer ist der Zielkunde?
  - Was will das Produkt erreichen? Wie macht es das Leben der Benutzerin einfacher oder besser?
  - Wann wird erwartet, dass das Produkt geliefert wird?
  - Wo wird das Produkt verwendet? Unter welchen spezifischen Umständen?
  - Warum sollten Benutzerinnen dieses Produkt anstelle anderer Softwarelösungen für das Problem verwenden, das es anspricht?
  - etc.

  Eine andere Möglichkeit zur Unterstützung sind Textschablonen (in Anlehnung an Highsmith):

  Zur Lösung von *[Warum, Bedarf, Problem]*
  für *[Zielkunde]* benötigen wir
  das *[Produktname]* mit der *[Produktkategorie],*
  weil es *[wesentlicher Vorteil, Kaufgrund, Feature]*
  und sich von *[primäre Wettbewerbsalternative]*
  unterscheidet durch *[Erklärung der primären Differenzierung]*.

  Da die Box nur einen begrenzten Platz bietet, hat dies den Vorteil, dass man sich fokussieren muss. Dies hilft, die erarbeiteten Inhalte knapp und aussagekräftig darzustellen.

- **Nachrichten aus der Zukunft**
  Diese Technik gibt es in verschiedenen Ausprägungen. Zum Beispiel kann man einen kurzen Artikel schreiben, der am Tag nach einer erfolgreichen Produkteinführung veröffentlicht wird. Darin stehen die Gründe, warum das Produkt entwickelt wurde und was es so einzigartig macht. Man kann auch ein fiktives Interview in der Zukunft gestalten, um sich gedanklich aus der Realität zu lösen und visionäres Denken zu ermöglichen.

  Ein fiktiver Bericht über eine negative Produktentwicklung oder das Scheitern des Projekts, in dem angegeben ist, welche Gründe oder fehlende Features zum Scheitern geführt haben, kann ebenfalls hilfreich sein.

  Generell wird damit ein Ändern der Denkmuster bewirkt.

- **Vision Board**
  Ein Vision Board ist eine in vielen Lebensbereichen und Branchen verbreitete Methode, um Vision und Ziele zu visualisieren. Typischerweise ist es eine grafische Collage mit Zielen, die evtl. auch nach kurz-, mittel- und langfristigen Visionen sortiert ist.

  Eine mögliche Vorgehensweise zur Erstellung eines Vision Board ist nachfolgend dargestellt:

  - Brainstorming/Ideen sammeln
  - Grobe Strukturierung (z. B. Mindmap)
  - Visualisierung (z. B. in Schlüsselworten, Merksätzen, Zitaten, Skizzen, Grafiken, Fotos etc.)
  - Board gestalten: analog/digital, Hochformat/Querformat, zeitlich (kurz-, mittel, langfristig) horizontal/vertikal etc.
  - Unübersehbar platzieren (räumlich oder auch digital z. B. als Bildschirmhintergrund)

- **Canvas-Techniken**
  Es gibt verschiedene Techniken mit Canvas (= »Leinwand«) wie Business Model Canvas, Project Canvas, Chancen-Canvas oder auch Product Canvas. Es sind alles Techniken, mit denen man fokussierte, strukturierte und leicht verständliche visuelle Darstellungen erstellen kann. Meist sind sie jedoch größer angelegt und beinhalten mehr Informationen als reine Vision Boards. In [Pichler 2013] wird ein einfaches Canvas-Schema vorgestellt:

<table>
<tr><td colspan="2">Vision:</td><td>Produktname:</td></tr>
<tr><td>Kundenpersonas
bzw. deren Probleme, die gelöst werden sollen:</td><td>Big Picture:
■ Benutzerinteraktionen
■ Features
■ Epics
■ Szenarien
■ Storyboards
■ Workflows
■ etc.</td><td>Produktdetails:
■ Priorisierte Ziele
■ Umsetzungen bzw. Stories zur Zielerreichung</td></tr>
</table>

Für ein Projekt bzw. eine Produktentwicklung sollten nicht mehr als fünf bis sieben übergeordnete Ziele formuliert werden. Wenn es notwendig erscheint, kann auch abstrahiert und mehrere Ziele zu einer übergeordneten Vision bzw. einem Ziel zusammengefasst werden.

Die Produktvision oder -ziele sind die abstraktesten Anforderungen und eine grundsätzliche Orientierungshilfe für das ganze Entwicklungsvorhaben. Sie können nicht ohne weitere Verfeinerung (z.B. Befragung und Bedarfsermittlung von Stakeholdern) in die Umsetzung aufgenommen werden. Bei der Verfeinerung soll jede Detailanforderung dahingehend bewertet und geprüft werden, ob sie auch einen Beitrag zur Erfüllung der definierten Ziele leistet. Wenn nicht, deutet das auf einen geringen Nutzen bzw. wirtschaftlichen Wert hin. Vor allem in agilen Vorgehensweisen mit ihrer starken Nutzen- und Wertorientierung sind Zieldefinitionen daher ein sehr wichtiges Artefakt.

Abschließend zeigt Abbildung 3–4 noch einen Überblick über die Zusammenhänge von Vision und Zielen.

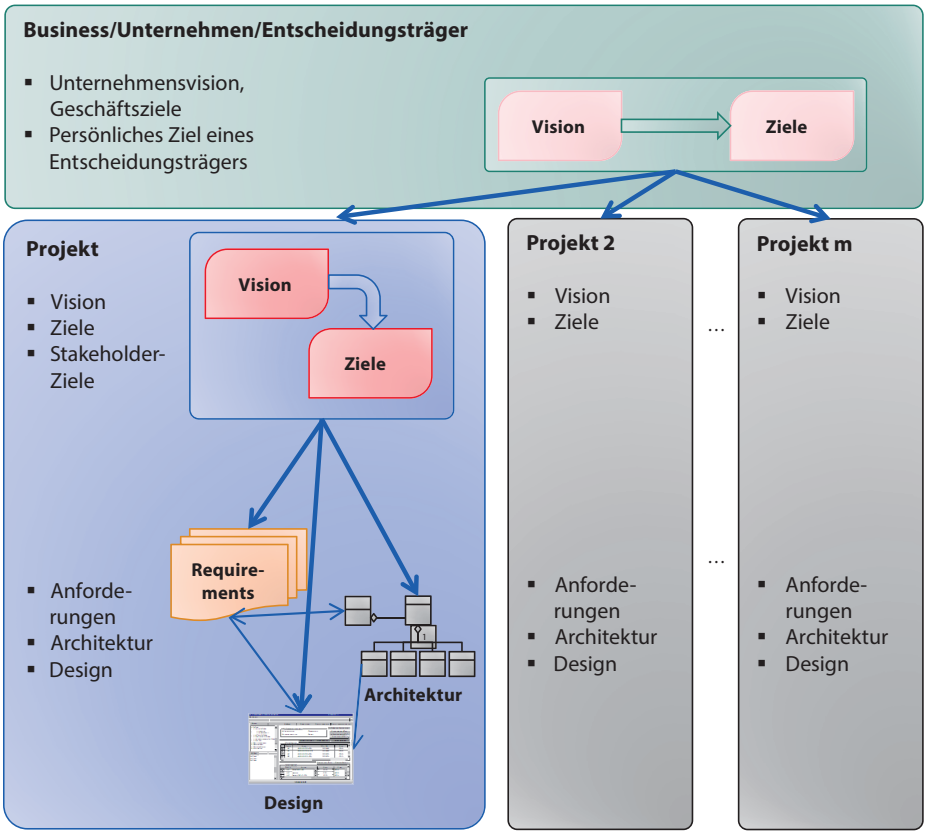

Abb. 3–4 *Vision und Ziele im Zusammenhang*