Stefan Queins, Chris Rupp

3

Requirements-Engineering im Überblick – von der Idee zur Anforderung

Die Wünsche und Bedürfnisse der Stakeholder sind die Basis für die Systementwicklung – so weit, so gut. Dem Risiko, die Anforderungen Ihrer Kunden nicht zu erfassen und deshalb daran vorbei zu entwickeln, können Sie mit einem gut durchdachten Requirements-Engineering begegnen. Existiert dieses Risiko nicht, können Sie entwickeln, was Sie für richtig halten, ohne sich Gedanken über Ihre Kundenwünsche machen zu müssen. Diese Möglichkeit haben aber nur wenige glückliche Firmen.

Alle anderen, nicht unbedingt unglücklichen, Firmen sollten ein zielgerichtetes Requirements-Engineering durchführen. Da Zeit ja bekanntlich Geld ist, sollten Sie dafür nur genau so viel Aufwand investieren, wie Sie für den geforderten Umfang in der gewünschten Qualität benötigen. Hier gilt das Motto „So wenig wie möglich, so viel wie nötig".

Der Aufwand einer Tätigkeit während der Systementwicklung hängt entscheidend von einem zur Problemstellung passenden Vorgehen ab. Auch für das Requirements-Engineering sollten Sie sich zu Beginn eines Projekts Gedanken darüber machen, wie Ihre Prozesse zur Ermittlung und Herleitung von guten Anforderungen sowie deren Vermittlung und Verwaltung aussehen sollten und wie sich diese Requirements-Engineering-Prozesse in den Gesamtentwicklungsprozess einfügen. Zur Unterstützung Ihrer Überlegungen schicken wir ein paar allgemeine Aussagen zu Anforderungen vorweg, bevor wir die einzelnen Haupttätigkeiten des Requirements-Engineerings vorstellen.

3.1 Anforderungen ins Gesicht geschaut

In Kapitel 1 „In medias RE" wurde ja schon der Begriff der Anforderung definiert. Um systematisch mit den Anforderungen arbeiten zu können, müssen wir zunächst unterschiedliche Typen von Anforderungen differenzieren. Da wir in Projekten von einer Vielzahl von Anforderungen ausgehen, setzen wir diese in Zusammenhang. Diese Zusammenhänge helfen Ihnen bei allen Tätigkeiten, die sich mit Anforderungen beschäftigen, insbesondere dem Verstehen von Anforderungen. Da es Ihre Aufgabe als Requirements-Engineer oder Product-Owner ist, gute Anforderungen oder User-Storys zur Verfügung zu stellen, stellen wir Ihnen abschließend noch die Qualitätskriterien von Anforderungen im klassischen und agilen Umfeld vor. Diese helfen Ihnen zu bewerten, ob Sie bereits Anforderungen in der benötigten Qualität erzeugt haben. Ist dies noch nicht der Fall, so erhalten Sie Hinweise, in welche Richtung Sie Ihre Anforderungen verbessern sollten.

3.1.1 Typen von Anforderungen

Prinzipiell haben Sie eine Vielzahl von Möglichkeiten, Ihre Anforderungen zu klassifizieren. Diese Einteilungen sollten immer einem bestimmten Zweck dienen. Wir stellen Ihnen hier die Arten von Klassifizierungen vor, die uns bei unserer Arbeit mit Anforderungen unterstützen und uns deswegen wichtig erscheinen. Weitere Einteilungen, die besonders aus der Realisierung und dem Test der Anforderungen heraus benötigt werden, betrachten wir in diesem Buch nicht.

Die klassische Einteilung

Eine sehr einfache aber häufig verwendete Klassifikation teilt Anforderungen grob in die folgenden Klassen ein:

- **Funktionale Anforderungen** beschreiben das, was das System einer nutzenden Person oder einem Nachbarsystem an Funktionen unter gewissen Bedingungen zur Verfügung stellt.
- **Nicht-funktionale Anforderungen** subsumieren den gesamten Rest der Anforderungen.

Häufig werden die nicht-funktionalen Anforderungen in weitere Kategorien unterteilt. Dies sind unter anderem:

- Qualitätsanforderungen (Quality-of-Service-Requirements), die z. B. die Performance einer Funktion oder die Verfügbarkeit des gesamten Systems beschreiben,
- Anforderungen an die Technologie, nach denen sich die Realisierung des Systems richten muss.

Eine vollständige Liste der Typen von nicht-funktionalen Anforderungen finden Sie in Kapitel 13 „Nicht-funktionale Anforderungen“.

Die Durchführung der Haupttätigkeiten in der Analyse (und im weiteren Entwicklungsprozess) kann sich nach diesen Typen richten. So können z. B. nicht-funktionale Anforderungen anders als funktionale Anforderungen erhoben, analysiert oder dokumentiert werden. Mehr zu den Unterschieden für die einzelnen Haupttätigkeiten erfahren Sie in den entsprechenden Kapiteln dieses Buches.

Die agile Einteilung

Die zuvor beschriebene Einteilung von Anforderungen nach funktionalen und nicht-funktionalen Typen wird zumeist in einer klassischen Betrachtung von Anforderungen verwendet. In agil durchgeführten Projekten spielen im Wesentlichen die folgenden Begriffe zur Einteilung von Anforderungen nach ihrem Verfeinerungsgrad eine Rolle:

- User-Storys beschreiben in den meisten Fällen Funktionen, die von außerhalb des Systems von ihm erwartet werden und einem fachlichen Ziel der Nutzenden dienen.
- Akzeptanzkriterien erweitern User-Storys um Aussagen, wann eine User-Story als umgesetzt gilt. Sie können häufig als Anforderungen angesehen werden, die mehr Details zu einer User-Story beschreiben.
- Epics beschreiben Zusammenfassungen von User-Storys. Sie können damit auch als sehr grobe User-Storys angesehen werden.

Es existieren nur sehr wenige Richtlinien bezüglich des gewünschten Verfeinerungsgrades der Anforderungen im agilen Umfeld. Allerdings wird diese Unterscheidung beim Arbeiten mit Anforderungen in agilen Projekten, z. B. bei der Dokumentation von User-Storys in Form von Story-Mapping, ausgenutzt. Mehr zu den hier genannten Begriffen finden Sie in Kapitel 17 „Storytelling, User-Storys und Co.“.

Einteilung nach juristischer Verbindlichkeit

Die juristische Verbindlichkeit beschreibt den Grad der Bedeutung, den der Stakeholder den Angaben in der Spezifikation beimisst. In Zusammenhang mit Verträgen wird dieser Begriff oft verwendet, da die Verbindlichkeit festlegt, welche Teile des Vertrags rechtlich einklagbar sind. Ein weiterer Vorteil konsequenter Klassifizierung nach der Verbindlichkeit ist, dass Sie bei knappem Zeit- oder Kostenrahmen besser entscheiden können, welche Anforderungen für das Funktionieren des Systems unverzichtbar sind und welche beispielsweise auch in einer späteren Version des Systems realisiert werden können.

In einer agilen Entwicklung wird viel von diesen Notwendigkeiten durch den Product-Owner gesteuert. Er oder sie kann, hauptsächlich durch Priorisierung der Backlog-Items, die Reihenfolge der Bearbeitung und damit die „aktuelle Verbindlichkeit" festlegen.

Üblicherweise wird die rechtliche Verbindlichkeit in textuellen Anforderungen durch Verwendung bestimmter Modalverben ausgedrückt. Die standardsetzende Organisation IETF (Internet Engineering Task Force, [IETF]) empfiehlt in [NWG97] die Verwendung der Schlüsselwörter „must", „must not", „required", „shall", „shall not", „should", „should not", „recommended", „may" und „optional". Demgegenüber empfiehlt der Ingenieursverband IEEE in „Systems and software engineering – Life cycle processes – Requirementsengineering" [IEEE29148] die fünf Schlüsselwörter: „shall", „should", „may", „can" und „will".

Wir SOPHISTen sind allerdings der Meinung, dass nicht so viele Schlüsselwörter gebraucht werden. Unsere Erfahrung hat gezeigt, dass drei Schlüsselwörter genügen, um gute Anforderungen mit festgelegter juristischer Verbindlichkeit zu verfassen.

Verbindlichkeit	Deutsches Schlüsselwort	Englisches Schlüsselwort
Pflicht	Muss	Shall
Wunsch	Sollte	Should
Absicht	Wird	Will

Abbildung 3.1: Bewährte Schlüsselwörter für die rechtliche Verbindlichkeit

- „Muss (Shall)" klassifiziert eine rechtlich verbindliche, zwingend zu erfüllende Anforderung.
- „Sollte (Should)" wird für Anforderungen verwendet, von deren Umsetzung unter bestimmten Umständen abgesehen werden kann.
- „Wird (Will)" ermöglicht es Ihnen, zukünftige Rahmenbedingungen und Anforderungen bereits anzukündigen.

Unter Umständen kann es sich als nötig erweisen, als viertes Schlüsselwort „darf nicht" zu verwenden. Dieser Verbindlichkeit können zwei Bedeutungen zugrunde liegen. Zum einen können Sie damit ausdrücken, was das System nicht machen darf (Nicht-Anforde-

rungen, www.sophist.de/re7/kapitel3). Zum anderen können Sie damit Anforderungen definieren, die sich durch eine Umformulierung zu einer „Muss"-Anforderung machen lassen. Vergleichen Sie hierzu die beiden folgenden Anforderungen. Wir empfehlen die positive Formulierung mit dem Schlüsselwort „muss" zu verwenden.

Die Zeitdauer für das Entriegeln der Tür darf nicht länger als 2 Sekunden sein.

Die Zeitdauer für das Entriegeln der Tür muss kleiner als 2 Sekunden sein.

Neben den Anforderungen mit einer der drei Verbindlichkeiten existiert in einer Anforderungssammlung noch ein weiterer, wichtiger Typ von Eintrag: der Kommentar. Mit ihm können Sie nicht verbindliche Informationen zu den Anforderungen geben, um diese zu erläutern, mit Beispielen zu hinterlegen oder in Zusammenhang mit anderen Anforderungen zu setzen. In einem solchen Kommentar können auch Annahmen dokumentiert werden, auf die aufbauend die Anforderungen formuliert wurden und die noch von den Stakeholdern bestätigt werden müssen.

Sie sollten die für Ihr Projekt zu verwendenden Schlüsselwörter mit ihrer Bedeutung festlegen. Die Verwendung dieser Schlüsselwörter lernen Sie in Kapitel 19 „Schablonen für Anforderungen und User-Storys" kennen.

Einteilung nach der Verantwortlichkeit

In den meisten Fällen werden Sie als Requirements-Engineer oder Product-Owner mit den Aufgaben betraut sein, Vorgaben (z. B. Wünsche oder Ideen) aus den Anforderungsquellen zu ermitteln und diese zu analysieren, um daraus gültige Produktanforderungen herzuleiten.

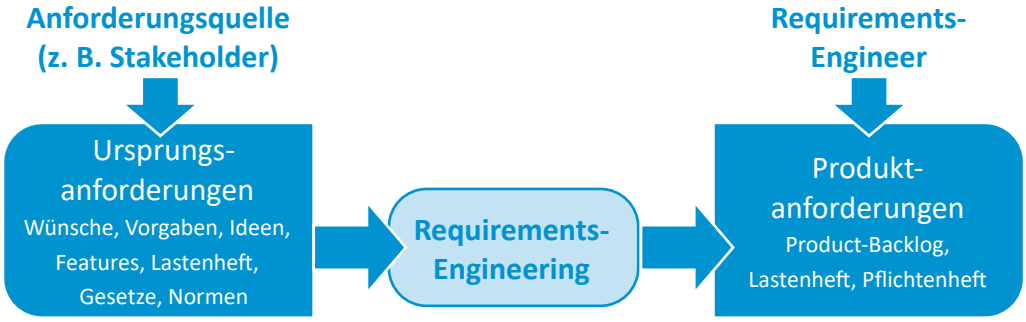


Abbildung 3.2: Ein- und Ausgaben des Requirements-Engineerings

Damit können wir für Anforderungen zwei unterschiedliche Verantwortliche festlegen: ErzeugerInnen der Ursprungsanforderungen (z. B. Stakeholder) und ErzeugerInnen von Produktanforderungen (häufig der Requirements-Engineer oder auch der Product-Owner). Bitte beachten Sie hierbei, dass die von einer auftraggebenden Organisation erzeugten Anforderungen z. B. in Form eines Lastenhefts als Eingabe in einen Requirements-Engineering-Prozess bei einer auftragnehmenden Organisation dienen können.

Diese Einteilung der Anforderungen soll Sie für einige wichtige Prinzipien beim Arbeiten mit Anforderungen sensibilisieren:

- Die Ursprungsanforderungen sind existent: Sie dürfen diese nicht ungefragt verändern. Je nach Rahmenbedingungen fließt Ihr Erkenntnisgewinn der Analyse lediglich in neu erstellte Produktanforderungen oder User-Storys ein und die Ursprungsanforderungen bleiben unverändert (sind damit aber veraltet, sofern sie dokumentiert wurden). Oder Sie sorgen dafür, dass die Anforderungen in ihren Quellen bei jedem Erkenntnisgewinn mit aktualisiert werden. Bei einem sehr leichtgewichtigen Vorgehen lassen Sie das neue gewonnene Wissen der Analyse lediglich in die Realisierung des Produktes einfließen und holen sich das O. K. des Stakeholders beim Sprintreview ab.
- Die Ursprungsanforderungen müssen analysiert werden: Legen Sie jede dieser Anforderungen auf die Goldwaage. Ist es wirklich das, was der Stakeholder will und was Ihr System leisten kann? Ist die Anforderung auch mit den gegebenen Randbedingungen umsetzbar?

Einteilung nach dem Betrachtungsgegenstand

Eine wichtige Information zur Einordnung von Anforderungen ist, auf welchen Betrachtungsgegenstand (z. B. das System oder eine Komponente des Systems) sie sich beziehen. Gehen wir davon aus, dass Sie sich die Entwicklung eines Systems zum Ziel gesetzt haben. Sie werden bestimmt zunächst die Anforderungen im Fokus haben, die die Eigenschaften von eben jenem System beschreiben. Eventuell wissen Sie aber auch schon, aus welchen Teilen sich das System zusammensetzen soll und wie sich diese Teile verhalten sollen. Dann könnten Sie auch Anforderungen an diese Teile des Systems beschreiben. Damit ändert sich der Betrachtungsgegenstand, auf den sich die Anforderung bezieht, weg vom System hin zu einem Teil des Systems.

Diese Änderung des Betrachtungsgegenstands wird häufig durch einen expliziten Architekturschritt unterstützt, bei dem Lösungen für das eigentliche Problem, die Anforderungen an das System, vorgegeben werden. Damit ergibt sich:

Anforderungen an Komponenten des Systems sind technische Lösungen für Anforderungen an das System.

Betrachten wir hierzu ein Beispiel aus unserem Smart-Home-System (SHS) mit Anforderungen, die sich auf unterschiedliche Betrachtungsgegenstände auf unterschiedlichen Architekturebenen beziehen:

Das SHS muss einer Person die Möglichkeit bieten, die Haustür zu entriegeln.

Die Kamera muss das Gesicht der Person fotografieren.

Die zentrale Steuerung muss eine Gesichtserkennung durchführen.

Ein Unterschied zwischen den drei Anforderungen ist der Betrachtungsgegenstand, auf den sich die jeweilige Anforderung bezieht. Es macht z. B. keinen Sinn, die erste Anforderung der zentralen Steuerung zuzuordnen, da an der Anforderung nicht nur diese Komponente beteiligt ist. Somit ist eine der wichtigsten Aufgaben eines Requirements-Engineers, den richtigen Betrachtungsgegenstand für eine gegebene Anforderung zu ermitteln und aus „zu großen" Anforderungen den richtigen Teil für seinen Betrachtungsgegenstand zu identifizieren.

Mehr zu dem Zusammenhang zwischen den Anforderungen auf System- und auf Komponentenebene erfahren Sie in Abschnitt 3.2.2 „Vom Wo und Wann des Requirements-Engineerings".

Neben den angesprochenen Betrachtungsgegenständen, die sich auf ein System oder Teile von ihm beziehen, existieren noch zwei weitere Betrachtungsgegenstände, mit denen wir in der Systementwicklung konfrontiert werden:

- Anforderungen an Tätigkeiten während der gesamten Lebenszeit des Systems,
- Anforderungen an weitere Lieferbestandteile, z. B. ein Benutzerhandbuch.

Gerade Anforderungen an Tätigkeiten während der Lebenszeit des Systems sind es wert, im Rahmen einer Systemanalyse genauer betrachtet zu werden. In der Praxis werden z. B. häufig anstelle der Systemanforderungen, die nur schwer eindeutig formulierbar sind, die Tests angegeben, wann das System bezüglich der entsprechenden Anforderung als abgenommen gilt. Aus diesen Anforderungen gilt es die eigentlichen Systemanforderungen abzuleiten.

3.1.2 Zusammenhänge zwischen Anforderungen

In dem vorigen Abschnitt haben wir Anforderungen nach ihrem Betrachtungsgegenstand unterschieden. Dabei wurde schon eine Art von Zusammenhang zwischen Anforderungen eingeführt: Aus Anforderungen an ein System entstehen Anforderungen an die Teile dieses Systems (z. B. Komponenten). Diese Komponentenanforderungen beschreiben technische Lösungen für die Systemanforderungen.

Aber auch bei Anforderungen, die sich auf genau einen Betrachtungsgegenstand beziehen, können wir zwischen der Problemebene und der Lösungsebene unterscheiden. Da wir dabei nicht den Betrachtungsgegenstand wechseln, sprechen wir hier von fachlichen Lösungen. Dabei lösen die sogenannten **verfeinerten Anforderungen** das Problem, das mit einer **abstrakteren Anforderung** beschrieben ist [Davis93].

Verfeinerte Anforderungen geben eine fachliche Lösung für eine abstraktere Anforderung vor.

Betrachten wir hierzu noch einmal das Beispiel von oben:

U1: Das SHS muss einer Person die Möglichkeit bieten, die Haustür zu entriegeln.

Wie genau der Ablauf, die Bedienung des Systems, sein soll, und was das System alles tun soll, können wir mit den folgenden verfeinerten Anforderungen (unvollständig) beschreiben.

E1: Das SHS muss eine Annäherung einer Person an die Haustür erkennen.

E2: Das SHS muss die Berechtigung zum Entriegeln der Haustür überprüfen.

E3: Falls die Berechtigung positiv war, muss das SHS die Tür entriegeln.

Diese drei Anforderungen können nun wiederum genauer durch verfeinerte Anforderungen beschrieben werden. Für die Anforderung E2 könnte man unter anderem vorgeben:

E4: Das SHS muss ein Bild des Gesichtes der Person erstellen.

E5: Das SHS muss eine Gesichtserkennung durchführen.

E6: Falls fünf Gesichtscharakteristika mit den Charakteristika einer berechtigten Person übereinstimmen, muss das SHS die Person zum Entriegeln der Tür berechtigen.

Mit diesen Anforderungen werden wiederum Lösungen vorgegeben. So soll z. B. die Berechtigung nicht über eine PIN, sondern über eine Gesichtserkennung erfolgen.

Dieser Zusammenhang zwischen verfeinerten und abstrakteren Anforderungen lässt sich auch im agilen Umfeld beobachten. Hier wird ja verlangt, in einer User-Story die Motivation für die geforderte Funktion des Systems anzugeben (siehe Kapitel 19 „Schablonen für Anforderungen und User-Storys“).

Als BewohnerIn möchte ich, dass das SHS die Berechtigung zum Entriegeln der Haustür überprüft, damit die Tür nur für berechtigte Personen entriegelt wird.

Diese User-Story enthält im ersten Teil die geforderte Funktionalität, die Berechtigung zu überprüfen. Im zweiten Teil des Satzes werden sogar zwei Begründungen (oder abstraktere Anforderungen) dafür angesprochen: das Entriegeln der Tür und die Sicherheitsanforderung, dass nur berechtigten Personen Zutritt gewährt werden soll. Wir haben also unter anderem eine implizite Qualitätsanforderung durch eine funktionale Anforderung verfeinert.

Die Verfeinerung einer Qualitätsanforderung wird expliziter in einem zweiten Beispiel betrachtet. Hier gehen wir von der folgenden (Nicht-)Anforderung aus:

Kein Unbefugter darf in das Haus gelangen.

Was das System dafür tun soll, kann durch eine verfeinerte Anforderung beschrieben werden:

Falls nach einer Entriegelung die Tür für 10 Sekunden nicht geöffnet wurde, muss das SHS die Tür verriegeln.

Beachten Sie, dass als Lösung einer Qualitätsanforderung eine funktionale Anforderung definiert wurde. Beide Anforderungen lassen sich zusammen in einer User-Story formulieren, wobei wiederum im zweiten Teil die abstraktere Anforderung, die Begründung, gegeben wird.

Als BenutzerIn möchte ich, dass das SHS während der Entriegelung den aktuellen Nummer-eins-Hit spielt, damit ich mich nicht langweile.

Allgemein gilt für verfeinerte Anforderungen:

- Die Verfeinerung von funktionalen Anforderungen beschreibt den Ablauf von Benutzerinteraktionen und Systemaktionen oder gibt zusätzliche Eigenschaften der Funktion an (z. B. Qualitätsanforderungen).
- Die Verfeinerung von Qualitätsanforderungen beschreibt, wie das System die geforderte Eigenschaft sicherstellen soll.

Wir haben bei Anforderungen häufig einen Wechsel von Problemstellung und Lösung.

Bitte machen Sie sich immer den Unterschied bewusst, denn

- die Problemstellung ist meist länger stabil als ihre Lösungen, und
- zu einer Problemstellung finden Sie normalerweise mehr als eine mögliche Lösung.

Doch wie können Sie erkennen, ob zwei Anforderungen in einer Verfeinerungsbeziehung zueinander stehen? Bei der Formulierung als User-Story ist der Zusammenhang schon immanent gegeben. Bei klassisch formulierten Anforderungen hilft die Queins'sche Um-zu-Regel. Können Sie die beiden Anforderungen mit einem „Um ... zu"-Konstrukt in einem Satz formulieren, handelt es sich um eine Verfeinerung. Betrachten Sie als Beispiel die oben genannten Anforderungen.

***Um** die Berechtigung zum Öffnen der Tür **zu** überprüfen, muss das SHS eine Gesichtserkennung durchführen.*

Die eigentliche Anforderung, die Gesichtserkennung, wird in der Anforderung selbst mit dem ersten Teil des Satzes begründet.

Diese Art der Überprüfung führt in leicht abgewandelter Form auch bei der Verfeinerung von Qualitätsanforderungen zum Ziel:

Damit kein Unbefugter in das Haus gelangen kann, muss das SHS die Tür automatisch nach 10 Sekunden verriegeln.

Weitere Beispiele zu abstrakten und verfeinerten Anforderungen finden Sie in Kapitel 9 „Das SOPHIST-*RE*gelwerk" und in Kapitel 12 „Anforderungen analysieren". Dort wird auch beschrieben, wie Sie die abstrakteste Ebene von Anforderungen finden und wie detailliert Sie diese verfeinern sollten.

Zusammenfassend lässt sich (jedoch leider) sagen, dass keine Vorgaben bezüglich des benötigten Verfeinerungsgrades von Anforderungen gemacht werden können. Der benötigte Verfeinerungsgrad hängt nicht von dem Typ des Betrachtungsgegenstands ab und kann sich je nach Anforderung in einer Spezifikation unterscheiden. Um aber die

Spezifikation nicht unendlich in die Tiefe wachsen zu lassen, gilt der folgende Merksatz:

Geben Sie Verfeinerungen für die Anforderungen an, für die Sie eine fachliche Lösung vorgeben oder ausschließen möchten.

Dieser Satz gewinnt auch dadurch an Bedeutung, dass Sie durch ihn die Vollständigkeit Ihrer Spezifikation definieren können. Diese Vollständigkeit ist eines von vielen Qualitätskriterien, denen gute Anforderungen genügen sollten. Sie werden im nächsten Abschnitt vorgestellt.

3.1.3 Gute und perfekte klassische Anforderungen

Die perfekte Anforderung bzw. Spezifikation wird es in der Realität kaum geben. Doch sollten wir aufzeigen, wie eine solche Spezifikation aussehen würde, um damit eine Richtung bzw. ein Ziel für Ihre Arbeit als Requirements-Engineer vorzugeben. Als Unterstützung zum Erreichen dieses Ziels dient sowohl das SOPHIST-*RE*gelwerk (siehe Kapitel 9 „Das SOPHIST-*RE*gelwerk“) als auch das Anwenden der in Kapitel 12 „Anforderungen analysieren“ beschriebenen Tätigkeiten.

Auch das dritte Adjektiv in der Überschrift dieses Abschnitts bedarf einer Erklärung: Wir wollen im Folgenden unter „klassischen“ Anforderungen all die Anforderungen verstehen, die in einem klassischen Vorgehen entstehen bzw. genutzt werden. Sie unterscheiden sich von ihrer Formulierung und ihren Qualitätskriterien von denen, die in einer agilen Entwicklung genutzt werden. Dort werden die Anforderungen (meist als User-Story inklusive ihrer Akzeptanzkriterien formuliert) mehr als Kommunikationsversprechen angesehen und bilden die Basis für weitere Gespräche über diese Anforderungen. Im Gegensatz dazu benötigt die „perfekte“ Anforderung im klassischen Vorgehen keine weiteren Erläuterungen, da die für die Entwicklung benötigten Informationen in Spezifikationen beschrieben sein sollten.

Es gibt sehr viele Listen von Qualitätskriterien, denen eine klassische Anforderung oder Spezifikation genügen sollte ([CPRE20]; [IEEE29148]). Bei genauer Durchsicht dieser Kriterien fallen jedoch Zusammenhänge und Dopplungen auf, weswegen wir eine Liste von Qualitätskriterien zusammengestellt haben, die uns ausreichend für die Bewertung von Anforderungen scheint.

Abbildung 3.3: Qualitätskriterien für Anforderungen

Dabei haben wir die Qualitätskriterien zwei Bereichen zugeordnet: Bezieht sich das Kriterium auf eine einzelne Anforderung oder auf eine Menge von Anforderungen (eine Anforderungsspezifikation)?

Qualitätskriterien für eine Anforderung

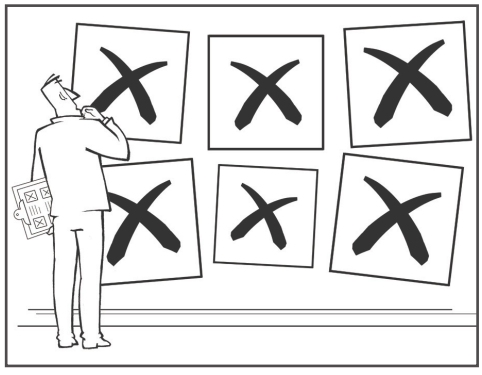
Das wohl wichtigste Kriterium haben wir an den Anfang gestellt: Eine Anforderung sollte **notwendig** sein. Das heißt, sie muss genau das beschreiben, was das System wirklich leisten soll. Dazu muss der Betrachtungsgegenstand der Anforderung das betrachtete System sein und die Forderung muss gewünscht sein. Sie muss also der Erfüllung eines Ziels für das System dienen.

Um die Notwendigkeit richtig einschätzen zu können, sollte eine Anforderung **eindeutig** formuliert sein. Dies stellt im besten Fall sicher, dass die empfangende Person der Anforderung genau das unter der Anforderung versteht, was die erstellende Person intendierte. Weiterhin ist eine Eindeutigkeit wichtig, um die Prüfbarkeit einer Anforderung zu gewährleisten: Nur wenn sie eindeutig formuliert ist, ist es für die Testfallerstellenden möglich zu beschreiben, wann das System die Anforderung erfüllt und wann nicht.

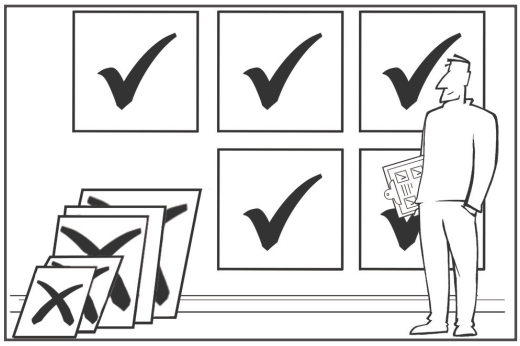
Unter der **Vollständigkeit** einer Anforderung verstehen wir, dass die geforderte Eigenschaft/Funktionalität des Systems umfassend beschrieben wird. Dies bedeutet, dass Sie z. B. alle Bedingungen, unter denen eine Funktion durchgeführt werden soll, in der Anforderung beschreiben.

Die beste Anforderung nutzt niemandem, wenn sie von den Empfangenden nicht verstanden wird. Deshalb sollten Anforderungen **verständlich** geschrieben sein. Dies klingt einfach, wird aber anspruchsvoller, wenn Sie unterschiedliche Gruppen von Konsumenten mit Ihren Anforderungen bedienen müssen. So könnte sich z. B. das Entwicklungsteam mit einer sehr formalen Ausdrucksweise wohlfühlen, die bei fachlichen Ansprechpersonen aber zu Langeweile beim Lesen und damit zu einer ablehnenden Haltung führen kann.

Weiterhin sollte jede Anforderung **abgestimmt** sein. Diese Abstimmung richtet sich an unterschiedliche Rollen, die Verantwortung für die Anforderungen übernehmen. So sollten die Anforderungen von den Anforderungsgebenden in Bezug darauf bewertet werden, ob sie ihre Wünsche korrekt widerspiegeln. Falls das betrachtete System mit anderen Systemen zusammenarbeitet, bietet sich eine Abstimmung der Schnittstellen zwischen den Systemen an. Letztlich müssen auch die Konsumenten der Anforderungen (Test-, Entwicklerteam …) ihre Zustimmung zu den Anforderungen geben. Damit wird z. B. überprüft, ob die Anforderungen innerhalb der Rahmenbedingungen des Projekts (Zeit, Aufwand) realisierbar und testbar sind. Falls nicht, müssen die Anforderungen in Abstimmung mit den Stakeholdern geändert werden oder die Rahmenbedingungen müssen angepasst werden.



Qualitätskriterien für eine Anforderungsspezifikation



Auch für eine Anforderungsspezifikation eines Systems fordern wir, dass diese **vollständig** ist. Doch müssen wir diese Forderung etwas genauer betrachten, um sie erreichbar formulieren zu können. Dazu unterscheiden wir zwischen einer Vollständigkeit in der Breite (alle Anforderungen auf der abstraktesten Ebene sind bekannt) und in der Tiefe (die abstrakten Anforderungen sind genügend detailliert). Zur Entscheidung, wann die Anforderungen genügend detailliert sind, verweisen wir auf den vorherigen Abschnitt.

Mit dieser Definition von Vollständigkeit stellen wir auch einen Teil einer weiteren Eigenschaft einer Anforderungsspezifikation sicher: Sie sollte **angemessen** sein. Beschreiben Sie nicht weniger, aber auch nicht mehr Anforderungen in Ihrer Spezifikation, als für Ihr Projekt benötigt werden. Dies kann von Vorgaben wie der Erfüllung von Standards oder von dem für die Entwicklung gewählten Vorgehensmodell abhängen (siehe Kapitel 4 „RE ist nicht gleich RE“ und Kapitel 16 „Wegweiser – Anforderungen dokumentieren und vermitteln“).

Das nächste Kriterium für eine Anforderungsspezifikation ist offensichtlich: Die Anforderungen in ihr sollten **konsistent** zueinander sein. Sie dürfen nichts Widersprüchliches fordern. Wir gehen zusätzlich noch einen Schritt weiter und fordern, dass die Anforderungen ein „rundes Gesamtbild“ des Systems ergeben sollten. Wenn Sie an einer Stelle eine Fehleingabe mit einem roten Bildschirmhintergrund angezeigt haben wollen, so sollte dies in allen Eingabefunktionen so behandelt werden.

Das letzte Kriterium ist ein eher weiches, wenn auch sehr wichtiges. Die Anforderungsspezifikation sollte eine **klare Struktur** besitzen. Dies unterstützt in erster Linie die Verständlichkeit der Anforderungen, besonders dann, wenn wir von mehreren Tausend, natürlichsprachlich definierten Anforderungen reden. Bei der Dokumentation von so vielen Anforderungen werden Sie darauf angewiesen sein, den Kontext der Anforderung als vorausgesetzt anzunehmen, um diesen nicht bei jeder Anforderung voranstellen zu müssen. Der Kontext einer Anforderung kann sich aus ihrer Einordnung in die Struktur ergeben, wenn diese Struktur inhaltlich motiviert ist. Weiterhin wird auch die Sicherstellung der Konsistenz unterstützt, da die PrüferInnen der Anforderungen leichter die Stellen in der Spezifikation identifizieren können, wo auf Konsistenz zu prüfende Anforderungen zu finden sind. Letztlich dient eine gute Strukturierung Ihrer Anforderung auch der Erweiterbarkeit der Spezifikation. Mehr zur Strukturierung von Anforderungssammlungen findet sich in Kapitel 21 „Strukturen und Zustände“.

Weitere und abgeleitete Qualitätskriterien

Neben den bisher vorgestellten Qualitätskriterien existieren zahlreiche weitere Kriterien, die für Ihre Arbeit wichtig sein können. Wir haben hier nur einige aus der Literatur

bekannte aufgezählt. Diese zusätzlichen Kriterien lassen sich zwei Gruppen zuordnen.

- Abgeleitete Kriterien: Sie leiten sich aus den oben angegebenen Kriterien ab und sind hier der Vollständigkeit halber aufgezählt. Zu ihnen zählen:
  - Die **Korrektheit** einer Anforderung leitet sich aus den Kriterien **Eindeutigkeit**, **Vollständigkeit** und **Notwendigkeit** her.
  - Um **prüfbar** zu sein, sollte eine Anforderung zumindest **eindeutig** sein.
  - Eine **klare Struktur** hilft bei der **Erweiterbarkeit.**
- Organisatorische Kriterien: Sie sind für das Arbeiten mit Anforderungen wichtig und werden häufig für die Verwaltung von Anforderungen von einem Requirements-Management-Werkzeug sichergestellt (siehe Kapitel 21 „Strukturen und Zustände“). Diese Kriterien haben somit einen mehr formalen Charakter, der das inhaltliche Arbeiten eines Requirements-Engineers nicht direkt beeinflusst.
  - **Identifizierbar:** Die Vergabe eines eindeutigen Bezeichners einer Anforderung (z. B. durch eine ID) erlaubt das Referenzieren und Finden der Anforderungen. Dies ist unumgänglich, um z. B. Anforderungen mit ihrem Ursprung zu verknüpfen.
  - **Versionierbar:** Damit wird die Möglichkeit gegeben, eine alte Version einer Anforderung zu betrachten, um z. B. das Änderungsmanagement zu unterstützen

3.1.4 Qualität von agilen Anforderungen

Wir unterscheiden bei der Definition der Qualität von Anforderungen bewusst zwischen klassischen Anforderungen und den im agilen Umfeld benutzten Darstellungsarten, da für die Anforderungen im agilen Umfeld andere Gesetze gelten. Die Anforderungen, die dort auftauchen, gelten nicht als unumstößlich. Sie sollen vielmehr als Basis für eine Kommunikation zwischen Entwicklungsteam und Product-Owner dienen. Diese Tatsache hat natürlich auch Auswirkungen auf die Qualitätskriterien, die für diese Art von Anforderungen gelten sollten.

Qualitätskriterien für eine User-Story

Falls Sie versuchen, die Qualitätskriterien für Anforderungen im klassischen Umfeld auf eine User-Story anzuwenden, so werden Sie scheitern. Laut Definition beinhaltet eine User-Story die Gespräche zur präziseren Definition der beinhalteten Funktionalität. Dies widerspricht aber offensichtlich den Qualitätskriterien „Eindeutigkeit“ und „Vollständigkeit“ einer klassischen Anforderung.

Wir benötigen also andere Maßstäbe für gute User-Storys. Unserer Meinung nach hat Bill Wakes [Wake03] mit dem INVEST-Prinzip gute Qualitätskriterien für User-Storys eingeführt, da sie sowohl die Idee hinter den User-Storys sehr gut widerspiegeln als auch beim Arbeiten mit User-Storys fast ohne Abstriche einsetzbar sind.

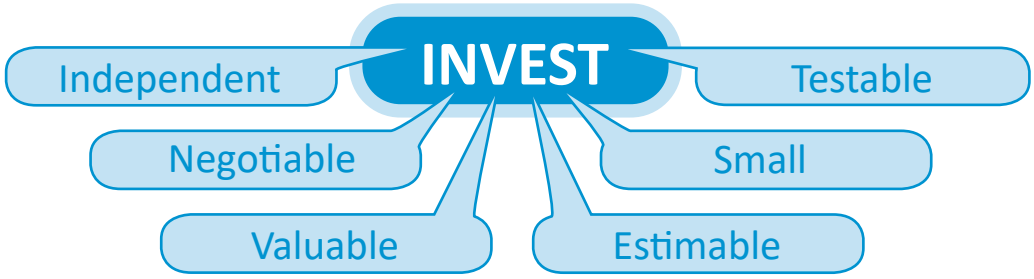
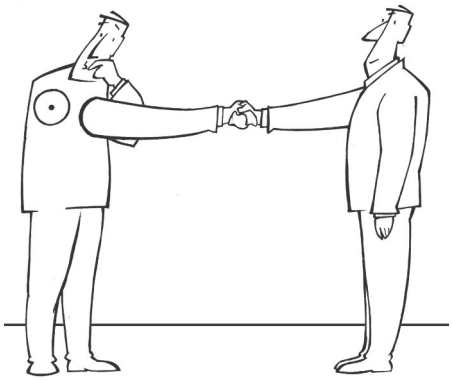


Abbildung 3.4: Das INVEST-Prinzip für eine User-Story

Damit User-Storys in einem Projekt umgeplant und neu priorisiert werden können, sollten sie **unabhängig** (independent) voneinander sein. Sind sie es nicht, so kann eine Umplanung einer User-Story Auswirkungen auf andere User-Storys haben, die bereits in der Entwicklung sind, und damit die dynamische Behandlung von User-Storys sehr erschweren. Leider ist dieses Kriterium eines der wenigen, das in der Realität nicht immer umsetzbar ist, da die Entwicklung in einem Sprint auf die im Sprint zuvor erzielten Ergebnisse aufsetzen kann.

Die **Verhandelbarkeit** (negotiable) hingegen ist schon durch die Idee hinter den User-Storys gegeben. Sie müssen nicht alle relevanten Informationen für eine geforderte Funktionalität in die User-Story hinein formulieren. Sowohl diese Informationen als auch der Umfang der in der User-Story betrachteten Funktionalität können sich im Rahmen von Besprechungen zwischen Entwicklerteam und Product-Owner ändern.

User-Storys sollten einen **fachlichen Wert** (valuable) für eine Person haben, die das System nutzt. Dieses Kriterium ist eine Verschärfung der oben eingeführten „Notwendigkeit". Sie resultiert aus dem Umstand, dass das Ergebnis eines SCRUM-Sprints ein „Shippable Product" sein soll. Eine User-Story soll also eine Funktion beschreiben, die nicht nur intern im System, sondern von einem Nutzenden des Systems benötigt wird.



Die nächsten beiden Kriterien resultieren aus der Anwendung der User-Storys als Instrument zur Projektsteuerung. Sie sollten **klein** (small) sein, sodass sie im Rahmen eines Sprints realisiert werden können. Des Weiteren hilft die kleine Größe bei der **Abschätzung** (estimable) des benötigten Aufwands zur Realisierung. Ist eine User-Story zu groß, so muss sie in kleinere User-Storys (verfeinerte Anforderungen) zerlegt werden. Die ursprüngliche User-Story kann als strukturierendes Element (oder als Epic) bestehen bleiben.

Das letzte Kriterium folgt wiederum aus der Natur der User-Storys und des agilen Gedankens. Eine User-Story sollte **testbar** (testable) sein. Nur dann kann das Team während des Sprints und der Product-Owner am Ende des Sprints entscheiden, ob eine User-Story korrekt umgesetzt wurde oder nicht. Dabei helfen ihnen die Akzeptanzkriterien, die als verfeinerte Anforderungen oder auch als Testfälle angesehen werden können (siehe Abschnitt 17.2.3 „Formulieren einer User-Story").

Qualitätskriterien für das Backlog

Im Rahmen einer agilen Entwicklung spricht man selten von einer Spezifikation als Sammlung von Anforderungen. Da eine User-Story als Anforderung angesehen werden kann, entspricht diese einer Sammlung von Anforderungen dem (Product-)Backlog, in dem die User-Storys gesammelt und priorisiert werden.

Für dieses Backlog gelten jedoch ähnliche Qualitätskriterien wie für eine Spezifikation im klassischen Umfeld. Deshalb verweisen wir für die Erklärung der Kriterien auf den vorigen Abschnitt.

Der Umfang eines Backlogs sollte angemessen sein, was im Allgemeinen weniger Einträge als im klassischen Bereich bedeutet. Weiterhin sollten natürlich die Einträge **konsistent** zueinander sein. Eine **klare Struktur** für das Backlog folgt im Allgemeinen durch die Zuordnung mittels Story-Mapping und die Anordnung gemäß der Priorisierung der User-Storys. Eine **Vollständigkeit** wird bei einem Backlog nicht verlangt, da dies der gewünschten Dynamik eines Backlogs widersprechen würde (siehe Kapitel 16 „Wegweiser – Anforderungen dokumentieren und vermitteln“)

3.2 Requirements-Engineering aus der Vogelperspektive

Nachdem wir im vorigen Abschnitt die Anforderungen, deren Qualität und Zusammenhänge genauer betrachtet haben, wollen wir in diesem Abschnitt auf die Entstehung dieser Anforderungen eingehen. Diesen Prozess bezeichnen wir im Folgenden als Requirements-Engineering und meinen damit alle Tätigkeiten, die aus den Eingaben in Ihre Entwicklung die für Sie benötigten Anforderungen herleiten.

Dazu betrachten wir zunächst die Verursacher einer Entwicklung, die die primäre Eingabequelle für Anforderungen sind. Sie werden auch als Stakeholder bezeichnet, da sie in erster Linie an dem neu entstehenden Produkt interessiert sind. Danach können wir darauf eingehen, wie sich Requirements-Engineering in verschiedene Typen von Entwicklungsprozessen eingliedert. Dies ist von der Stellung abhängig, die eine Organisationseinheit in der Gesamtentwicklung einnimmt, und davon, wie damit das Requirements-Engineering stattfindet. Zuletzt werden wir die wichtigsten Tätigkeiten beim Requirements-Engineering in einen logischen Ablauf bringen, um damit die nachfolgenden Kapitel dieses Buches in Zusammenhang zu setzen.

3.2.1 Ursachen und Quellen von Anforderungen

Requirements-Engineering sollte keinen Selbstzweck darstellen. Vielmehr dient es dazu, Vorgaben an ein zu entwickelndes Produkt oder an Teile davon zu machen und damit den Inhalt eines Entwicklungsprojekts vorzugeben. Bitte beachten Sie: In diesem Abschnitt werden wir von einem Entwicklungsprojekt reden, wenngleich die hier getroffenen Aussagen für alle Typen von Vorhaben gültig sind.

Im Allgemeinen setzt sich ein Entwicklungsprojekt zur Erzeugung eines verkaufbaren Produkts aus mehreren kleineren Entwicklungsprojekten zusammen. Oftmals entwickelt eine auftragnehmende Organisation einen Teil des Produkts im Rahmen des Gesamtentwicklungsprojekts einer auftraggebenden Organisation. Jedes dieser Entwicklungsprojekte, unabhängig von seiner Einordnung, hat einen oder mehrere VerursacherInnen, die in erster Linie für die Inhalte des Projekts Verantwortung tragen. Sie werden manchmal auch als Sponsoren des Projekts bezeichnet.

In diesem Abschnitt stellen wir Ihnen die potenziellen GeberInnen von Anforderungen vor. Welche von diesen in einem konkreten Projekt als VerursacherInnen für dieses Projekt auftreten, hängt natürlich von der jeweiligen Organisation und der individuellen Aufgabenverteilung ab. Trotzdem werden Sie auch für Ihr Projekt die nachfolgend beschriebenen Stakeholder, explizit oder implizit, finden. Mehr zum Aufspüren dieser für Ihr Projekt relevanten Stakeholder finden Sie in Kapitel 6 „Ziele, Informanten und Fesseln“.

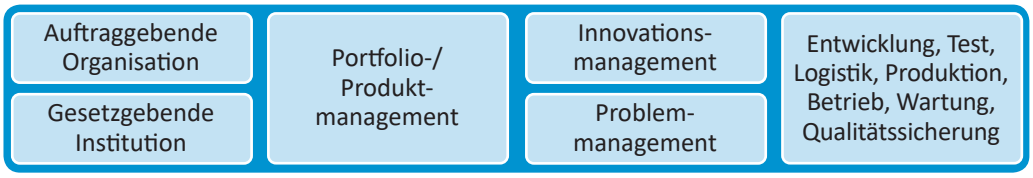


Abbildung 3.5: Typische Quellen von Anforderungen

Auftraggebende Organisation

In einer Beziehung zwischen auftraggebender und auftragnehmender Organisation ist die auftraggebende Organisation die wohl prominenteste Verursacherin eines Entwicklungsprojekts. Sie gibt der auftragnehmenden Organisation das Geld für die Entwicklung (und vielleicht auch für nachfolgende Tätigkeiten wie die Produktion) und bestimmt damit maßgeblich die Anforderungen an das zu entwickelnde Produkt.

Portfolio-/Produktmanagement

Das Portfolio-/Produktmanagement hingegen ist oftmals die Ursache eines Entwicklungsprojekts oder zumindest eine zusätzliche Anforderungsquelle. Die folgenden Tätigkeiten sind relevant, um die Anforderungen an ein Produkt bzw. Entwicklungsprojekt zu erheben:

- Definieren Sie die strategische Weiterentwicklung der Produkte durch das Festlegen der Neuerungen. Anwendungen aus dem Bereich der Smart-Eco-Systeme und der Digitalisierung bieten ein großes Potenzial für solche Neuerungen oder auch für Erweiterungen des Produktportfolios (siehe Kapitel 24 „Die digitale REvolution“).
- Planen Sie Neuerungen, zum Beispiel mithilfe von Features, in die bevorstehenden Entwicklungsprojekte ein.
- Bewerten Sie die geplanten Tätigkeiten aus Business-Sicht (Kosten-Nutzen-Analyse).

Abhängig von der Organisationsform oder dem in der Organisation gewählten Vorgehensmodell existiert für diese Aufgaben eine Vielzahl von Begriffen. Im klassischen Kontext wird häufig zwischen einem Portfolio-, Produkt- und Release-Management unterschieden [Breuer99]. In agil ausgerichteten Organisationen kommt häufig der SAFe-Ansatz (Scaled Agile Framework, [Mathis18] und [Knaster18]) zum Einsatz. Dort übernehmen die Portfolio-, Large-Solution- und Program-Management die Aufgaben zur Planung der Produkte auf hoher Ebene.

Die Bewertung der geplanten Tätigkeiten aus Business-Sicht erscheint auf den ersten Blick wenig mit Anforderungen zu tun zu haben. Jedoch können erst durch diese Bewertungen Anforderungen an neue Produkte priorisiert werden oder sogar wegfallen, wenn diese Anforderungen keine Potenziale am Markt versprechen.

Dazu werden unterschiedliche Priorisierungstechniken verwendet, z. B. die Portfolio-Matrix der Boston Consulting Group [BCG20] oder die Portfolio-Matrix von McKinsey [McKinsey20]. In beiden Fällen handelt es sich um Ein-Kriteriums-Klassifikation nach dem Kriterium „Markterfolg“. Priorisierung hilft Ihnen vor und während des Requirements-Engineerings, die richtigen Entscheidungen bezüglich der Funktionalitäten Ihres zukünftigen Produkts zu treffen. Weitere Informationen zu Priorisierungstechniken und ihrer Anwendung erfahren Sie unter www.sophist.de/re7/kapitel3. Im agilen Umfeld ist der Ansatz „Weighted Shortest Job First“ [WSJF] weit verbreitet.

Innovationsmanagement

Eine wichtige Eingabequelle für die Tätigkeiten des Produktmanagements liefert das Innovationsmanagement. Es hat die Aufgabe, Neuerungen für ein Produkt so weit zu betrachten, bis sowohl die Risiken und die Kosten für ein Entwicklungsprojekt als auch der Nutzen für die Organisation abschätzbar sind. Damit diese Betrachtung von Neuerungen nicht zu einem Glücksspiel wird, werden auch für diese eigentlich schwer planbaren Tätigkeiten Prozesse definiert. Diese Prozesse erzeugen ähnliche Ergebnisse wie ein konventioneller Produktentstehungsprozess [Ehrlenspiel09], werden jedoch häufig mit Mechanismen zur kontinuierlichen Bewertung ergänzt und müssen keine Aspekte aus dem Markt mit berücksichtigen. Im agilen Umfeld werden Innovationen oft im Rahmen eines Living-Lab-Prozesses, wie in Kapitel 5 „Wegweiser: Wissen ermitteln“ beschrieben, entwickelt. Im klassischen Umfeld ist der Stage-Gate-Prozess nach Cooper [Cooper10] ein typischer Vertreter. Er bietet mit den Toren (den Gates) die Möglichkeit, die in einem Abschnitt (einem Stage) erarbeiteten Ergebnisse zu bewerten. Nur bei positivem Befund wird die Arbeit in dem nachfolgenden Abschnitt weitergeführt. Aus dem Ergebnis eines Innovationsprojekts können Anforderungen abgeleitet werden, die als Eingabe in ein Entwicklungsprojekt dienen.

Gesetzgebende Institutionen

Gesetzgebende Institutionen, oder besser Gesetze, Normen und Standards, liefern Anforderungen auf zahlreichen Ebenen. Heutzutage existieren kaum Produkte, die nicht einem Gesetz unterliegen, sich nach Normen richten oder Standards bei der Entwicklung berücksichtigen müssen.

Diese Texte beschreiben unter anderem folgende Arten von Anforderungen:

- Welche Eigenschaften und Funktionen muss das Produkt aufweisen?
- Wie ist das Produkt zu testen?
- Welche Tätigkeiten müssen bei der Entwicklung durchgeführt werden?
- Welche Beschreibungen müssen zusätzlich zu dem eigentlichen System erzeugt werden?
- Wie muss die Nachverfolgbarkeit der Produkteigenschaften gewährleistet werden?

Einige prominente Vertreter der Gesetze, Normen und Standards sind im Bereich der Automobilentwicklung A-SPICE [ASPICE07] oder auch die ISO26262 der funktionalen Sicherheit [ISO26262 18]. Im Bereich der Medizintechnik müssen sich Entwicklungen nach den Vorgaben der [GxP] richten.

Primär dienen diese Vorgaben als zusätzliche Anforderungen, nach denen sich die Entwicklung richten muss. Allerdings kann die Änderung eines Gesetzes auch eine Anpassung eines bestehenden Produkts im Rahmen eines Entwicklungsprozesses auslösen. Im Allgemeinen wird dieses jedoch mit einer Erweiterung der Produkteigenschaften (ausgelöst durch das Produktmanagement) einhergehen.

Problemmanagement

Eine ähnliche Stellung wie die gesetzgebenden Institutionen nimmt das Problemmanagement ein. Es liefert Änderungswünsche an ein Produkt, die hauptsächlich in späteren Lebenszyklusphasen dieses Produkts identifiziert wurden. Diese können zu zusätzlichen oder geänderten Anforderungen führen. Sie können auch den Bedarf für ein neues Entwicklungsprojekt auslösen, das dann die gewünschten Änderungen im Produkt umsetzt. Oder sie werden durch ein Change-Management in ein laufendes Projekt eingesteuert (siehe Abschnitt 22.5. „Change-Management“). Beispiele für die späteren Lebenszyklusphasen sind:

- Produktion, wenn das entwickelte System nicht in der benötigten Qualität und Zuverlässigkeit produziert werden kann.
- Transport/Logistik, wenn sich z. B. Einschränkungen aufgrund des Transportweges ergeben.
- Integration des System in ein Gesamtsystem, was darauf hindeutet, dass die Anforderungen von Ihrem Auftraggeber nicht korrekt waren.
- Betrieb, wenn z. B. festgestellt wird, dass mit dem aktuell verfügbaren System die Ziele der Systemnutzenden nicht erfüllt werden können.

Entwicklung, Test, Logistik, Produktion, Betrieb, Wartung, Qualitätssicherung

Mit dieser letzten Gruppe von Stakeholdern stellen wir die Organisationseinheiten vor, die wichtige AnforderungsgeberInnen sind, auch wenn sie nur indirekt über das Problemmanagement neue Entwicklungsprojekte anstoßen.

Aus den Tätigkeiten in den einzelnen Disziplinen können neue Anforderungen an das Produkt folgen. Beispiele hierfür sind:

- Die Logistik muss unter anderem den Transport des fertig entwickelten Systems sicherstellen, wodurch z. B. eine Zerlegbarkeit des Systems für den Transport gefordert wird.
- Die Produktion hat im Allgemeinen gewisse Einschränkungen, die die Realisierung des Systems aus Sicht der Produzierbarkeit beeinflusst.
- Die Wartung kann eine Austauschbarkeit von Komponenten ebenso wie die Protokollierung von wartungsrelevanten Daten fordern.

Die Qualitätssicherung nimmt häufig die Rolle der Prozesshüterin ein. Sie fordert (und überprüft), dass die Tätigkeiten gemäß den internen und externen Vorgaben durchgeführt werden. Interne Vorgaben existieren meist in Form von Prozessbeschreibungen, die für ein spezielles Entwicklungsprojekt adaptiert werden müssen. Im Rahmen eines agil durchgeführten Projekts übernimmt der Scrum-Master die Prozesshüter-Rolle. Ein anderer Teil der Vorgaben kann durch die Betrachtung in einer Definition of Done (DoD) sichergestellt werden und wird dann vom Product-Owner bei einem Sprint-Review abgenommen.

Bis auf wenige Ausnahmen wurden in diesem Abschnitt nicht-funktionale Anforderungen angesprochen, die nicht unbedingt aus Sicht der auftraggebenden Organisation oder des Produktmanagements gefordert werden. Umso wichtiger ist es, auch diese weiteren Stakeholder in die Anforderungsermittlung zu integrieren.

3.2.2 Vom Wo und Wann des Requirements-Engineerings

Das Requirements-Engineering dient dazu, Anforderungen zu spezifizieren, die in einem Entwicklungsprojekt benötigt werden. Wir möchten diese sehr allgemeine Aussage konkretisieren und betrachten dazu den Einsatz von Requirements-Engineering in verschiedenen Situationen. Obwohl wir im nachfolgenden Kapitel drei detaillierte Szenarien definieren, die wir im weiteren Verlauf des Buchs verwenden, werden wir hier ein übergreifendes Szenario darstellen.

Dafür gehen wir von einem Unternehmen aus, das ein Produkt für Endkunden anbietet. Dieses Unternehmen vergibt als auftraggebende Organisation die Entwicklung des Produkts an eine auftragnehmende Organisation. Bei dieser findet neben der eigenen Entwicklung eine Vergabe für eine Komponente des Gesamtsystems an eine unterauftragnehmende Organisation statt.

Um Requirements-Engineering in diesen Gesamtentwicklungsprozess einzuordnen, müssen wir noch zwei weitere Charakteristiken dieses Projekts betrachten:

- die Art des Vertrages zwischen der auftraggebenden und auftragnehmenden Organisation und
- die Notwendigkeit der Nachvollziehbarkeit in der Entwicklung.

Wir werden im Folgenden zwei Extrema betrachten: ein klassisch und ein agil durchgeführtes Projekt. Bei einem klassisch durchgeführten Projekt wird häufig ein Vertrag mit einem Festpreis abgeschlossen, für den die auftragnehmende Organisation das gewünschte System sehr genau beschreibt. Darauf aufbauend verpflichtet sich die auftragnehmende Organisation, dieses System zu einem Festpreis zu entwickeln. Zusätzlich gehen wir davon aus, dass bei diesem klassischen Projekt Regularien bestehen, die die Nachvollziehbarkeit in der Entwicklung durch eine ausführliche Dokumentation erfordern.

Eine andere Art der Vertragsgestaltung wird bei einer angestrebten agilen Zusammenarbeit zwischen der auftraggebenden und der auftragnehmenden Organisation vorausgesetzt. Bei einem auf „Time & Material" basierenden Vertrag werden Entwicklungsaufwände und kein vorher definiertes Gewerk bezahlt. Dabei gehen wir von dem Szenario aus, dass die auftragnehmende Organisation den Product-Owner für das Entwicklungsprojekt stellt. Den Product-Owner aus der auftraggebenden Organisation zu stellen ist zwar aus Sicht der Agilität der erstrebenswerte Zustand, jedoch trifft man heutzutage diese sehr enge Art der Zusammenarbeit nur selten an.

Requirements-Engineering bei der auftraggebenden Organisation

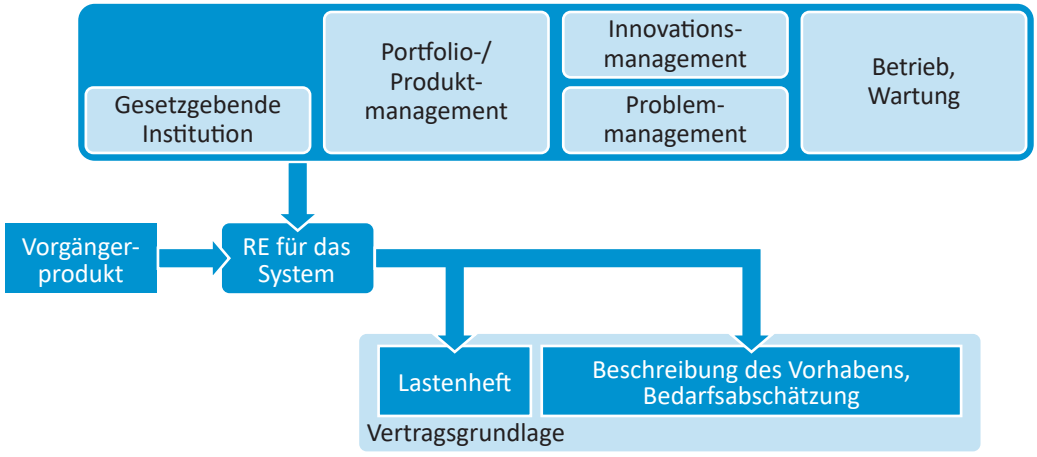


Abbildung 3.6: RE bei einer auftraggebenden Organisation

Bei einer auftraggebenden Organisation kommt der Wunsch nach der Entwicklung eines neuen Produkts häufig aus dem Produkt- oder Innovationsmanagement. Aus deren Wünschen und Ideen muss das Requirements-Engineering nun Anforderungen an das neue System erzeugen. Diese müssen bei einem Festpreisprojekt in einem Lastenheft spezifiziert werden (Fall 1) oder sie dienen als Grundlage für eine Beschreibung des Vorhabens und für die Abschätzung der Bedarfe (Time & Material, Fall 2). Im zweiten Fall wird weniger Aufwand entstehen, da sich die auftraggebende Organisation bei den

Anforderungen noch nicht präzise festlegen muss. In beiden Fällen müssen sowohl die aktuell geltenden Gesetze und Vorgaben als auch Erkenntnisse aus dem Betrieb und der Wartung des Systems, vielleicht organisiert durch ein Problemmanagement, berücksichtigt werden. Eventuell können die Beschreibungen eines Vorgängerprodukts, im besten Fall das damals erzeugte Lastenheft, als zusätzliche Eingabequellen dienen.

Je agiler die zukünftige Zusammenarbeit, desto geringer der Aufwand für das initiale Requirements-Engineering.

Requirements-Engineering bei der auftragnehmenden Organisation im klassischen Kontext

Der Einsatz des Requirements-Engineerings gestaltet sich bei einer auftragnehmenden Organisation komplexer als bei der auftraggebenden Organisation, da sie sich bei einer Top-down-Entwicklung des Systems auch mit Anforderungen an die Teile des Systems beschäftigen muss.

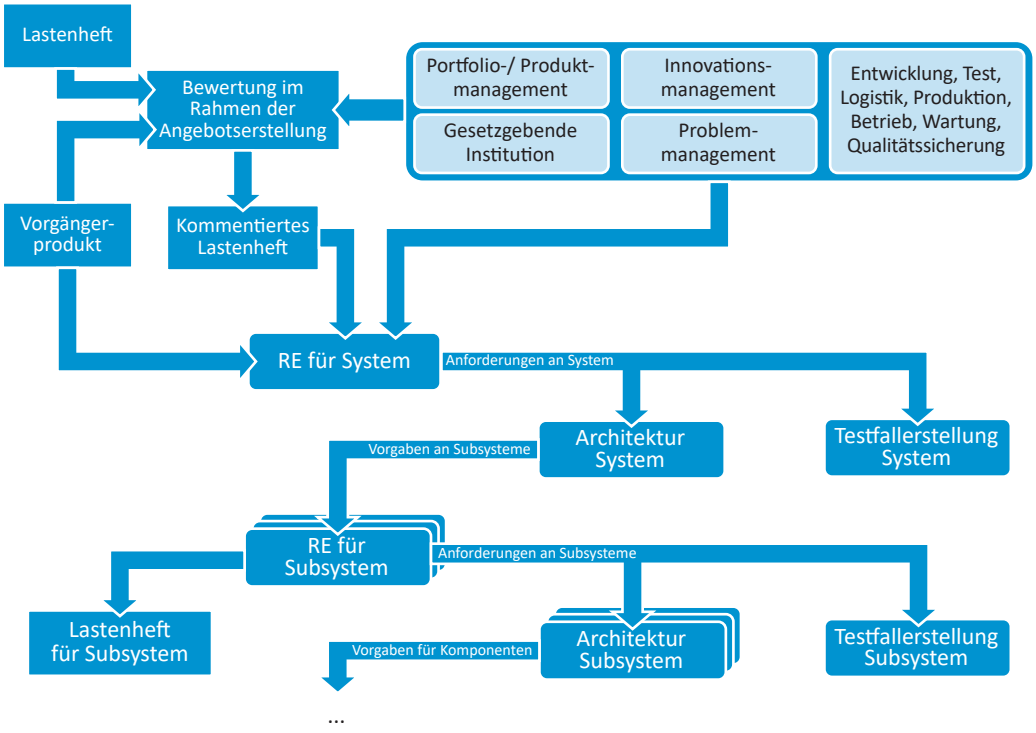


Abbildung 3.7: Klassisches RE der auftragnehmenden Organisation

Zunächst wird die auftragnehmende Organisation die Anforderungen der auftraggebenden Organisation in der Angebotsphase bewerten. Die Ergebnisse dokumentiert sie im

Allgemeinen durch eine Kommentierung des Lastenhefts. Dieses löst das urprüngliche Lastenheft als Vertragsgrundlage ab und bildet die Eingabe in die Entwicklung.

Nun beginnt ein Wechsel zwischen Requirements-Engineering- und Architektur-Tätigkeiten. In einem Top-down-Vorgehen entstehen zunächst die Anforderungen an das betrachtete System, die häufig in einem Pflichtenheft zur Verfügung gestellt werden. Sie werden danach in einem Architekturschritt den Teilen des Systems, den Subsystemen, zugewiesen. Dadurch entstehen Vorgaben an diese Subsysteme, die nun wiederum als Eingabe in einen Requirements-Engineering-Schritt für jedes Subsystem dienen. Bei einer Eigenentwicklung schließt sich nun ein Architekturschritt für die Subsysteme und nachfolgend ein Requirements-Enegineering für die gefundenen Komponenten an. Bei einer Entwicklung durch eine unterauftragnehmende Organisation kann die Ausgabe des Requirements-Engineerings ähnlich wie in Abbildung 3.7 als Lastenheft für das zugelieferte Subsystem angesehen werden. Parallel zu den hier beschriebenen Schritten können aus den Anforderungen Testspezifikationen für die jeweiligen Betrachtungsgegenstände hergeleitet werden.

Der Auslöser für ein Projekt wie in unserem Beispiel ist die auftraggebende Organisation. Die auftragnehmende Organisation könnte ebenfalls auf ein Vorgängerprodukt aufsetzen und wird auch gesetzliche Vorgaben bei der Entwicklung berücksichtigen müssen. Die weiteren in Abbildung 3.7 dargestellten Rollen dienen als Geberinnen von Anforderungen. Hier kommen insbesondere die neben der Entwicklung an der Beauftragung des Systems beteiligten internen Abteilungen hinzu (z. B. Logistik und Produktion).

Bitte beachten Sie, dass in unserem Beispiel nur zwei Architekturebenen dargestellt wurden (System, Subsystem). In der Realität besteht ein komplexes System aus mehr Architekturebenen und das Wechselspiel setzt sich nach unten fort. Auch die Vergabe an eine auftragnehmende Organisation kann für einen Betrachtungsgegenstand auf einer beliebigen Architekturebene durchgeführt werden. Ausführlicher wird das Wechselspiel im Rahmen eines System-Engineerings in Abschnitt 23.2 „Das Twin-Peaks-Modell“ vorgestellt.

Requirements-Engineering bei der auftragnehmenden Organisation im agilen Kontext

Auch im Requirements-Engineering spiegelt sich die Eigenschaft einer agilen Entwicklung, Informationen erst zum Bedarfszeitpunkt zu ermitteln, wider. Während der Entwicklung stößt das Team auf eine Lücke in den Anforderungen und erfragt diese dann beim Product-Owner, der in unserem Beispiel durch die auftragnehmende Organisation gestellt wird. Dieser wird zur Beantwortung der Fragen auf seine Stakeholder in der auftraggebenden Organisation zurückgreifen. Diese spielen natürlich auch bei all seinen Tätigkeiten eine Rolle, die sich mit der Definition der User-Storys beschäftigen. Daneben muss die Entwicklung aber auch die Vorgaben der internen Stakeholder berücksichtigen. In Abbildung 3.8 ist dies durch die Unterscheidung der Stakeholder in die beiden an der Entwicklung beteiligten Organisationen ausgedrückt.

Das Requirements-Engineering stellt sich bei einer auftragnehmenden Organisation im Rahmen einer agilen Zusammenarbeit mit der auftraggebenden Organisation also sehr viel überschaubarer dar, da hier keine expliziten Anforderungsdokumente für die im Rahmen eines agilen Projekts betrachteten Gegenstände (System, Subsystem ...) gefordert werden.

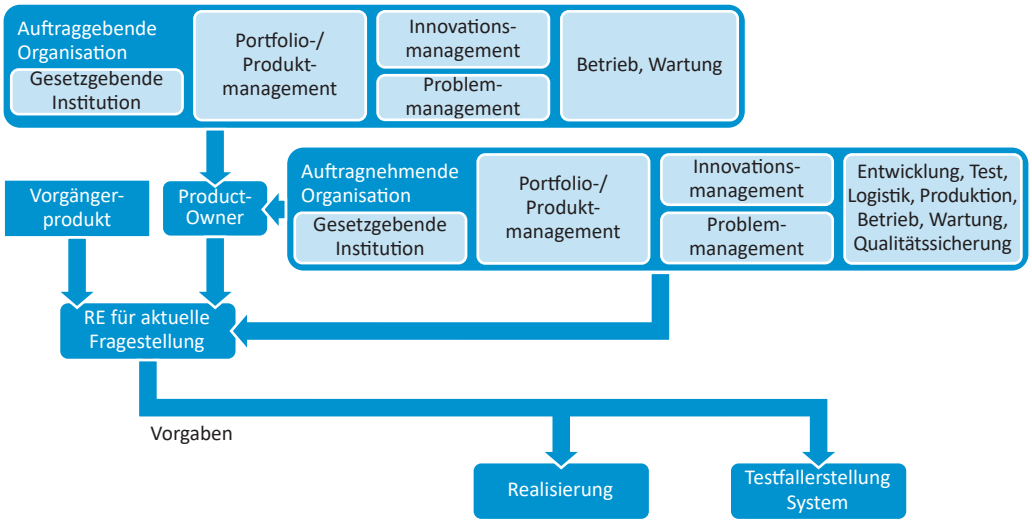


Abbildung 3.8: RE bei der auftragnehmenden Organisation in einer agilen Entwicklung

3.2.3 Requirements-Engineering im Überblick

Nach der groben Einordnung des Requirements-Engineerings in die Entwicklungsprozesse bei einer auftraggebenden und einer auftragnehmenden Organisation, betrachten wir nun Requirements-Engineering detaillierter.

Wir setzen dazu die wichtigsten Tätigkeiten in einen logischen Zusammenhang, wohl wissend, dass die Intensität, mit der Sie diese Tätigkeiten durchführen, immer von den Gegebenheiten in Ihrem Projekt abhängt. Aber auch die Reihenfolge der Tätigkeiten in Ihrem Projekt wird sich nicht nur nach diesem logischen Zusammenhang richten. Sie werden z. B. Tätigkeiten immer wieder zu unterschiedlichen Zeitpunkten oder aufgrund von Änderungen wiederholt durchführen müssen.

Eine Übersicht über die Tätigkeiten finden Sie in Abbildung 3.9. Sie sind in Haupttätigkeiten zusammengefasst. Da hier nur ein logischer Zusammenhang dargestellt werden soll, verweisen wir für die Anwendung in einem klassischen oder agilen Kontext auf Kapitel 4 „RE ist nicht gleich RE“.

Die weiter oben beschriebenen Anforderungsquellen liefern die Eingaben in die erste Haupttätigkeit, das Ermitteln des benötigten Wissens. Damit erhalten Sie die Vorgaben für das neue System, aus denen dann die zu realisierenden Anforderungen hergeleitet werden. Diese zweite Haupttätigkeit stellt sicher, dass die zuvor ermittelten

Anforderungen auf die in Abschnitt 3.1 „Anforderungen ins Gesicht geschaut“ eingeführten Qualitätskriterien überprüft und verbessert werden und so als Eingabe in den weiteren Entwicklungsprozess dienen. Dazu müssen diese Systemanforderungen zu den konsumierenden Personen vermittelt werden, z. B. über eine Dokumentation dieser Anforderungen. Wann immer Anforderungen dokumentiert werden, kommt die letzte Haupttätigkeit ins Spiel: das Verwalten der dokumentierten Anforderungen. In diesen vier Haupttätigkeiten werden unterschiedliche Techniken eingesetzt, die wir in Abbildung 3.9 in „Übergreifende Techniken und Einflussfaktoren“ zusammengefasst haben.

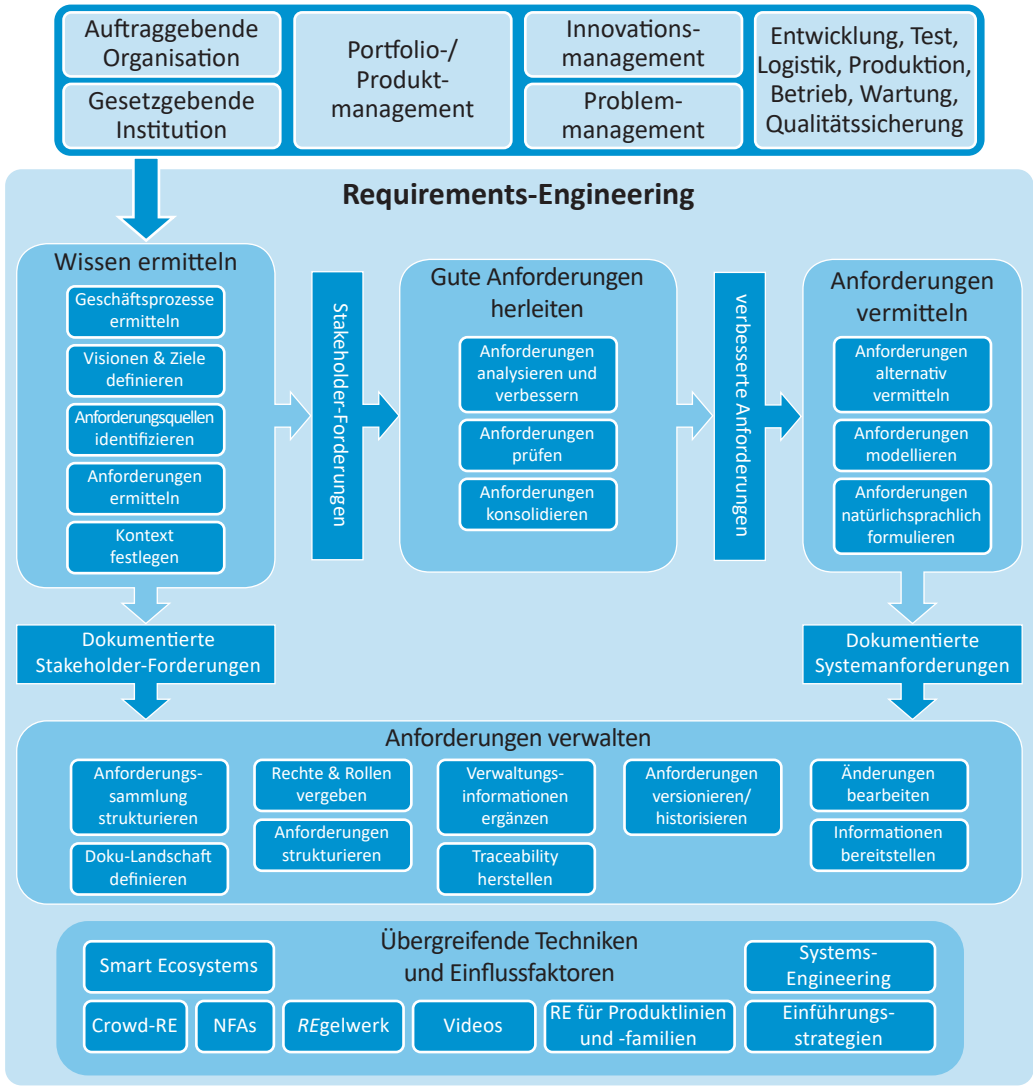


Abbildung 3.9: Logischer Zusammenhang der RE-Tätigkeiten

Wissen ermitteln

Die erste Haupttätigkeit „Wissen ermitteln“ legt den Grundstein für das weitere Requirements-Engineering. Hier wird die Richtung für das Projekt dadurch vorgegeben, dass die Visionen und die Ziele für das Projekt definiert werden, sofern sie nicht bereits vorliegen. Dies muss mit den Sponsoren und den Stakeholdern des Projekts gemeinsam geschehen. Da diese aber nicht immer alle zu Beginn des Projekts bekannt sind, müssen sie zunächst identifiziert werden. Beide Tätigkeiten sind in Kapitel 6 „Ziele, Informanten und Fesseln“ beschrieben. Dort wird zusätzlich die Kontextabgrenzung eingeführt, die eng mit der Ermittlung der Informanten zusammenhängt. Wenn festgelegt wird, dass eine Aufgabe innerhalb des Systems realisiert werden soll, so müssen die Anforderungsquellen identifiziert werden, die Informationen zu dieser Aufgabe liefern.

Geschäftsprozesse, die durch das betrachtete System unterstützt werden sollen, sind ebenfalls Treiber für eine Systementwicklung. In vielen Fällen sind diese Prozesse allerdings nicht so bekannt oder dokumentiert, dass sie als formale Eingabequelle in das Requirements-Engineering dienen könnten. Deswegen haben wir uns diesem Thema in Kapitel 7 „Geschäftsprozesse ermitteln und verfeinern“ angenommen.

Mit den bisherigen Informationen sind Sie nun in der Lage, die Anforderungen an das betrachtete System aus den Anforderungsquellen zu ermitteln. Dieser Schritt ist in Kapitel 8 „Anforderungsermittlung“ beschrieben und liefert die Anforderungen, die die Vorgabe für die nächste Haupttätigkeit bilden.

Gute Anforderungen herleiten

Um gute Anforderungen herzuleiten, sollten Sie zunächst die zuvor ermittelten Vorgaben analysieren, um ein möglichst umfassendes Bild der Anforderungen an das zu entwickelnde System zu bekommen (Kapitel 12 „Anforderungen analysieren“). Je nach Vorgehensmodell muss dieses Bild nun auf den Prüfstand gestellt werden. VertreterInnen unterschiedlicher Rollen in Ihrer Entwicklung können die Anforderungen einem Review unterziehen:

- Aus Sicht der anforderungsgebenden Personen werden die Anforderungen bezüglich der Erfüllung der Erwartungen an das System überprüft.
- Für den Test des entwickelten Systems werden die Anforderungen daraufhin überprüft, ob aus ihnen entsprechende Testfälle abgeleitet werden können.
- Die Entwicklung (insbesondere die Architektur) wird die Realisierbarkeit der Anforderungen überprüfen.

In Abhängigkeit von Ihrer Anwendungsdomäne können noch weitere Überprüfungen notwendig werden. So kann im Automotive-Umfeld die Überprüfung auf Erfüllung der ISO-Norm 26262 (funktionale Sicherheit) gefordert sein. Unterschiedliche Techniken zur Überprüfung der Anforderungen finden Sie in Kapitel 14 „Prüftechniken für Anforderungen“.

In beiden Tätigkeiten können Widersprüche in den Anforderungen zutage treten. Diese sollten behoben werden, indem Sie die widersprüchlichen Anforderungen konsolidieren und damit eine konsistente Menge von Anforderungen für den kommenden Entwicklungsumfang erhalten (Kapitel 15 „Anforderungskonflikte“).

Anforderungen vermitteln

Mit den bisher betrachteten Haupttätigkeiten sind Sie in der Lage, sich einen guten Überblick über die von Ihnen benötigten Anforderungen zu verschaffen. Da aber die Anforderungen jemand anderem bei seiner Arbeit helfen sollen, benötigen wir eine Haupttätigkeit zur Vermittlung der gewonnenen Anforderungen.

Hierbei gehen wir davon aus, dass Sie entweder die Anforderungen dokumentieren wollen oder Ihr Wissen über die Anforderungen in anderer Art und Weise weitergeben wollen. Für den zweiten Fall verweisen wir auf das Kapitel 17 „Storytelling, User-Storys und Co.“.

Falls Sie eine Sammlung von dokumentierten Anforderungen erstellen müssen oder wollen, haben Sie die Wahl zwischen zwei Alternativen:

- Sie können die Anforderungen an das System natürlichsprachlich formulieren. Hierfür stellen wir Ihnen in Kapitel 19 „Schablonen für Anforderungen und User-Storys “ mehrere Schablonen zur Verfügung. Diese erlauben es Ihnen, klare und präzise Anforderungen zu formulieren.
- Sie können sich entscheiden, die Anforderungen modellbasiert zu dokumentieren. Hierfür existieren mehrere Diagrammarten, die wir Ihnen in Kapitel 18 „Anforderungen modellieren“ vorstellen. Damit können Sie hauptsächlich Anforderungen bezüglich der Funktionen des Systems beschreiben. Aber auch die in Ihrem System wichtigen Begriffe und fachlichen Daten lassen sich modellbasiert darstellen.

In der Praxis verspricht ein gemischter Ansatz aus beiden Alternativen den höchsten Mehrwert. Auch auf diese Verbindung zwischen den beiden Darstellungsarten geht Kapitel 18 „Anforderungen modellieren“ ein.

Anforderungen verwalten

Sobald wir die Dokumentation von Anforderungen betrachten, kommt eine neue Herausforderung auf Sie zu: das Verwalten dieser dokumentierten Anforderungen. Hierfür müssen Sie einige Entscheidungen treffen:

- Aus welchen Dokumenten setzen sich meine benötigten Spezifikationen zusammen und wie sind diese im Einzelnen aufgebaut?
- Wie ist das Product-Backlog organisiert? Welche Informationen sind neben dem Backlog notwendig?
- Wer darf welche Aktionen in den Anforderungen durchführen (Vergabe von Rechten und Rollen)?

- Welche zusätzlichen Informationen benötigen Sie zum Verwalten der Anforderungen? Dies können unter anderem sein: AutorIn der Anforderung, letztes Änderungsdatum, aktueller Zustand der Anforderung.
- Wie soll die Nachverfolgbarkeit (Traceability) sichergestellt werden? Welche Typen von Anforderungen sollen mit welchen anderen Typen von Anforderungen in Verbindung gebracht werden?
- Welches Versionierungskonzept möchten Sie Ihren Anforderungen zugrunde legen? Können z. B. zwei Versionen einer Anforderung zu einer Zeit relevant sein und müssen diese Versionen auch wieder zusammengeführt werden können (Branch-Merge)? Oder müssen Sie vielleicht auf Vorgängerversionen Ihrer Anforderungen zugreifen (Verwendung der Historie einer Anforderung).
- Welche Informationen müssen in Ihrem Projekt in welcher Form bereitgestellt werden? Hier müssen Sie festlegen, welche Ausschnitte aus der Anforderungsmenge mit welchen zusätzlichen Informationen jemand anderem bereitgestellt werden müssen (View).
- Wie werden Sie mit Änderungen an den Anforderungen umgehen? Wie möchten Sie in Ihrem Projekt Änderungen an den Anforderungen kenntlich machen?

Diese Punkte werden in Kapitel 21 „Strukturen und Zustände“ und Kapitel 22 „Attribute, Traces, Historie“ adressiert, in denen wir Ihnen mit zahlreichen Beispielen Erfahrungen aus der Praxis beim Verwalten von Anforderungen geben. Dies sollte Ihnen eine gute Basis für die Entscheidungen in Ihrem Projektkontext geben. Während Sie die vorherigen Haupttätigkeiten auch in einer agilen Entwicklung vollständig, wenn auch nur implizit, durchführen, werden Sie viele der Tätigkeiten zum Verwalten von Anforderungen in einem solchen Vorgehen gar nicht oder nur rudimentär durchführen. Auch hier verweisen wir auf Kapitel 4 „RE ist nicht gleich RE“.

Übergreifende Techniken und Einflussfaktoren

In diesem Bereich haben wir all das zusammengefasst, was in vielen der vorgestellten Tätigkeiten eine Rolle spielt bzw. das Arbeiten im Requirements-Engineering insgesamt beeinflusst.

Dazu gehört offensichtlich die Betrachtung der nicht-funktionalen Anforderungen (Kapitel 13 „Nicht-funktionale Anforderungen“), da diese z. B. besonders ermittelt und dokumentiert werden können. Aber auch die in Kapitel 9 „Das SOPHIST-*RE*gelwerk“ vorgestellten Regeln zur Untersuchung von Anforderungen lassen sich in vielen Tätigkeiten anwenden. So können Sie diese Regeln beim Ermitteln, beim Analysieren oder auch beim Vermitteln von Anforderungen beherzigen.

Dem CrowdRE (Kapitel 10 „CrowdRE“) als Ermittlungstechnik kommt heutzutage eine immer größere Bedeutung zu, besonders bei Systemen mit einer Vielzahl von evtl. noch unbekannten NutzerInnen.

Ähnlich verhält es sich mit dem Einsatz von Videos zur Ermittlung und Dokumentation unterschiedlicher Informationen, die im Rahmen des Requirements-Engineerings relevant sind. So können Sie mithilfe von kleinen Videosequenzen den Kontext Ihres Systems, das zu lösende Problem oder sogar schon Anforderungen darstellen. Das Kapitel 27 „Videos im RE“ widmet sich dieser relativ neuen Technik im Requirements-Engineering.

Dass unsere Systeme immer komplizierter werden, ist ein seit Jahren bekanntes Phänomen. Dieser Herausforderung kann man begegnen, indem man den Zusammenhang zwischen ähnlichen Systemen explizit betrachtet und in der Entwicklung ausnutzt. Der Grundstein dafür wird im Requirements-Engineering gelegt. Kapitel 25 „RE für Produktlinien und -familien“ beschäftigt sich mit dieser Disziplin und stellt Ihnen die wichtigsten Prinzipien beim Arbeiten mit Varianten im Rahmen von Produktlinien und -familien vor. Dies hilft Ihnen, Zeit und Kosten schon bei der Betrachtung der Anforderungen in ähnlichen Systementwicklungen zu sparen.

Einsparungen erreichen Sie aber auch durch eine Anpassung der Entwicklungsprozesse, und damit auch des Requirements-Engineerings, an die Art Ihrer Systeme. Zum Beispiel wird ein geschäftsprozessorientiertes Softwaresystem andere Entwicklungstätigkeiten als ein mechanisches System verlangen. Wir haben in Kapitel 23 „Systems-Engineering“ und Kapitel 24 „Die digitale REvolution“ die Auswirkungen dieser Typen von Systemen auf das Requirements-Engineering betrachtet.

Die beste Methodik im Requirements-Engineering (oder in anderen Disziplinen) macht ohne die Mitarbeit der betroffenen Personen keinen Sinn. Somit müssen Festlegungen für das Arbeiten mit Anforderungen immer einhergehen mit der Einführung bzw. mit der Vermittlung dieser neuen oder geänderten Methoden. Mit diesem herausfordernden Schritt in der Verbesserung des Requirements-Engineerings in Ihrer Organisation oder Ihrem Projekt beschäftigt sich Kapitel 26 „Einführungsstrategien“.