2 Requirements Engineering – kurz und knapp

Ihr Nutzen aus diesem Kapitel:

»It isn't that they can't see the solution. It is that they can't see the problem.« Der berühmte Autor Gilbert Keith Chesterton adressierte damit eine wesentliche Herausforderung. Requirements Engineering identifiziert systematisch Probleme und Ziele – bevor Lösungen vorschnell entwickelt werden. Was sollten Sie für die eigene Praxis direkt übernehmen? Welche Faustregeln helfen, um das Requirements Engineering richtig zu justieren? Worauf sollten Sie in Ihren Projekten achten? In diesem Kapitel fasse ich kurz die wichtigsten Gesetzmäßigkeiten des Requirements Engineering zusammen. Systematik heißt nicht Formalismus oder gar Dogmatismus, sondern muss sich pragmatisch auf vorliegende Probleme einstellen. Zunehmend arbeiten wir im RE agil, also flexibel und situativ. Daher zeige ich hier die Essenz des Requirements Engineering. Sie lernen, was Anforderungen sind, wie verschiedene Typen von Anforderungen ganz verschieden auf das Projekt wirken und wie Sie Requirements Engineering leben können. Ein durchgängiges Beispiel transportiert das Vorgehen immer wieder in der Praxis.

2.1 Was ist eine Anforderung?

Eine Anforderung beschreibt, was der Kunde oder Benutzer vom Produkt erwartet, also Bedingungen, Attribute, Ziele und vor allem Nutzen. Anforderungen sind definiert als:

- Eine Eigenschaft oder Bedingung, die von einem Benutzer (Person oder System) zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
- Eine Eigenschaft oder Bedingung, die ein System oder eine Systemkomponente erfüllen muss, um einen Vertrag, eine Norm oder andere, formell vorgegebene Dokumente zu erfüllen.
- Eine dokumentierte Repräsentation einer Eigenschaft oder Bedingung wie in den ersten beiden Punkten beschrieben.

Wenn wir hier von **Produkt** sprechen, umfasst dies Anwendungen, IT-Systeme, eingebettete Software bis hin zu großen IT-Lösungen. Auch Dienstleistungen sind

Produkte. Die **Benutzer** oder Anwender des Produkts sind diejenigen, die nach der Auslieferung damit in irgendeiner Form in Berührung kommen. Wir wollen dieses »in Berührung kommen« später nochmals aufgreifen (siehe Kap. 3), denn es ist bei der Ermittlung von Anforderungen wichtig, hier die Basis nicht zu sehr einzuschränken. Beispielsweise ist es relevant, zu unterscheiden, wer exakt **Kunde** ist (also vertraglich in die Entstehung eingebunden ist und dafür bezahlt) und wer das Produkt später nutzt.

Wert ist, wofür ein Kunde zahlt, und keine lange Funktionsliste. RE fokussiert Anforderungen auf den zu liefernden Wert und erreichbare Ziele. Anforderungen können mehrdeutig sein, sie können überspezifiziert sein, sie können unvollständig sein, sie können kontextspezifisch sein, sie können sich widersprechen, sie können nicht machbar oder schlichtweg falsch sein. In aller Regel jedoch sind es zu viele, um unter gegebenen Randbedingungen realisiert werden zu können. Das kommt deutlich am Beispiel eines Wunschzettels zum Ausdruck (Abb. 2–1).

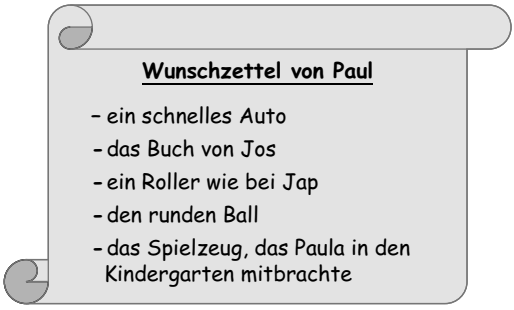


Abb. 2–1 *Anforderungen sind der »Wunschzettel« des Kunden.*

Das Problem in vielen Unternehmen ist, dass zu oft Funktionen und zu selten Träume adressiert werden. Als Ingenieure sind wir darauf geeicht, Lösungen zu finden. Wir definieren Funktionen und implementieren sie. Allerdings führen solche – angenommenen – Lösungen nicht immer zum Markterfolg und zu zufriedenen Kunden. Das überrascht uns, wo doch die Lösung so viele interessante Funktionen hat. Aber hatten wir wirklich ein Problem und einen Bedarf adressiert? Werden durch unser Produkt eine Vision und ein Traum wahr oder ersticken die Benutzer in Komplexität?

Wir stürzen uns viel zu schnell auf eine Lösung, weil es das ist, was wir dank Ausbildung und unter Projektdruck als Ergebnis sehen wollen. Ein Projekt, das von einer – angenommenen – Lösung aus startet, führt dazu, dass man einer Fata Morgana nachläuft, die sich ständig ändert. Wenn es uns gelingt, die Ziele und Anforderungen zu verstehen und systematisch umzusetzen, dann können wir jedes Projekt beherrschen.

Ein Produkt ist dann erfolgreich, wenn es den Bedürfnissen seiner Benutzer und seiner Umgebung gerecht wird. Anforderungen kommunizieren diese Bedürfnisse,

und Requirements Engineering ist die Disziplin, die die Behandlung von Anforderungen über den gesamten Lebenszyklus des Produkts hinweg umfasst.

Beispiel:
Apple unter Steve Jobs zeigte, wie man mit wertorientiertem Requirements Engineering hervorragende Produkte macht. Steve Jobs gelang es, Träume zu verkaufen und diese Träume in Funktionen zu übersetzen. Er schockierte seine Ingenieure regelmäßig mit der einfachen Frage: Kann man im Design noch etwas weglassen? Unnötige Funktionen sind Ballast und verursachen Kosten im gesamten Lebenszyklus. Ein Produkt war für Steve Jobs erst gut genug, wenn jede Funktion einen Wert lieferte – und die Komplexität auf ein Minimum reduziert war.

Wir trennen daher klar zwischen **Anforderung** und **Lösung**. Eine Anforderung beschreibt ein Bedürfnis oder einen Nutzen, der erreicht werden soll. Sie beschreibt nicht, wie dieser Nutzen zu realisieren ist. Diese Implementierungssicht wird durch die Lösung beschrieben. Abbildung 2–2 veranschaulicht diesen Unterschied durch die Trennung zwischen Problemraum (oberer Teil: Marktanforderungen, Lastenheft etc.) und Lösungsraum (unterer Teil: Lösungsspezifikation, Pflichtenheft, Design, Fachkonzept etc.). Der Problemraum ist zunächst immer nur unscharf umrissen und wird im Verlauf der Lösungskonzeption eingeschränkt. Wert existiert nur im Auge des Betrachters. Daher sind Lösungen oftmals am erfolgreichsten, wenn sie nicht alle Anforderungen des Kunden adressieren. Der Erfinder und Unternehmer Henry Ford hat das früh begriffen und festgestellt: »Wenn ich die Leute gefragt hätte, was sie wollen, hätten sie gesagt: schnellere Pferde.« Steve Jobs hat Apple damit erfolgreich gemacht, dass er innovative Lösungen lieferte und Altlasten rigoros hinterfragt und reduziert hat. Legendär sind die frühen iPhone-Telefone, die zwar kaum telefonieren konnten, aber sich hervorragend verkauften, weil sie ganz andere neue Bedürfnisse adressierten.

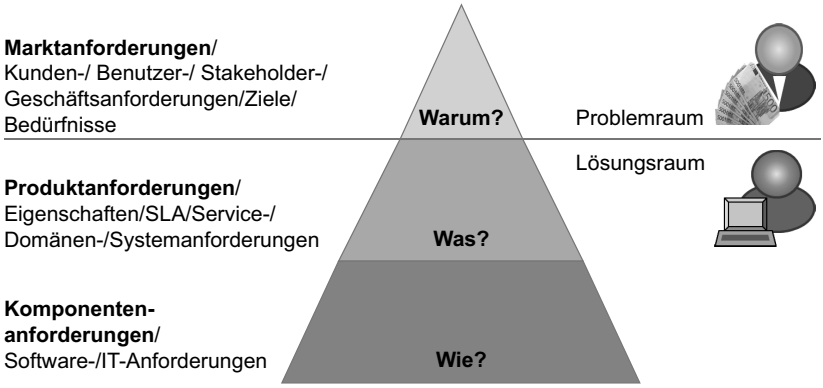


Abb. 2–2 *Anforderungen und Lösungen*

Es gibt nicht die »Anforderung« schlechthin. Zu einer Anforderung gehört immer die Perspektive, aus der sie beschrieben wird. Eine Anforderung ist eine Bedingung oder eine Fähigkeit, die ein Benutzer benötigt, um ein Problem zu lösen oder um ein Ziel zu erreichen. Das heißt, sie hängt von der Perspektive ab. Ein Benutzer kann der Kunde sein, der für die Lösung bezahlt, aber es kann auch ein Entwickler sein, der daraus eine Architektur ableitet. Entsprechend unterschiedlich sind die Schwerpunkte und Inhalte, die durch diese Anforderung beschrieben werden. Was dem einen die Anforderung ist, ist dem anderen die Lösung. Man trennt daher in der Praxis unterschiedliche Arten von »Anforderungen«, beispielsweise Marktanforderungen oder Komponentenanforderungen, und vermeidet, von einer »Anforderung« ohne Präzisierung zu sprechen.

2.2 Perspektiven: vom Markt zur Realisierung

Drei verschiedene Sichten auf Anforderungen werden im Laufe der Lösungskonzeption unterschieden (Abb. 2–2):

- Marktanforderungen
- Produktanforderungen
- Komponentenanforderungen

Diese drei Sichten entstehen durch Verfeinerung bzw. Abstraktion. Offensichtlich ist diese Dreiteilung rekursiv: Eine Komponentenanforderung an einen Lieferanten ist dort wiederum eine Marktanforderung.

Marktanforderungen

Marktanforderungen beschreiben Anforderungen an ein Produkt aus der Sicht des Kunden. Sie werden daher oft auch als Kunden-, Benutzer- oder Geschäftsanforderungen oder als Bedürfnisse bezeichnet. Sie beschreiben den Nutzen und die Erfahrungen mit dem Produkt in der Sprache des Kunden oder Benutzers, also **warum** ein Projekt überhaupt durchgeführt wird. Einziger Maßstab an Wert und Erfüllungsgrad ist daher die Wahrnehmung oder Spezifikation des Kunden. Marktanforderungen werden im Lastenheft dokumentiert.

Beispiel:
Marktanforderung_1:
Der Datentransfer muss geschützt erfolgen, um Missbrauch zu verhindern.

Viele Projekte umfassen Änderungen an Bestehendem. **Marktanforderungen adressieren gerade auch solche Änderungen und nicht nur Projekte und Produkte auf der »grünen Wiese«.** Änderungen verlangen eine genaue Abstimmung der Bedürf-

nisse und Nutzen mit den jeweiligen Zielgruppen oder Marktsegmenten. Häufig realisieren wir Funktionen oder Änderungen, die interessant scheinen, deren Markt aber zu klein ist. Hier ist eine Priorisierung aus betriebswirtschaftlicher Sicht wichtig.

Marktanforderungen sind Bestandteil von Verträgen, Entwicklungsaufträgen, Projektplänen, Teststrategien etc. Sie dienen als Basis für Abschätzung, Planung, Durchführung und Verfolgbarkeit der Projekttätigkeiten. Sie sind in der Sprache und im Kontext des Kunden formuliert. Wenn wir bei der Ermittlung der Marktanforderungen nicht aufpassen, haben wir die gleichen Schwierigkeiten, mit denen auch Eltern sich auseinandersetzen müssen, die einen Wunschzettel ihres Sprösslings in der Hand halten (Abb. 2–1). Es gibt Widersprüche, Inkonsistenzen und verborgene Prioritäten. Die Anforderungen sind zu umfangreich, und das Budget ist limitiert.

Marktanforderungen machen Wünsche erfüllbar und erlebbar. Das Ziel der Anforderungsermittlung ist es, aus verschiedenen Perspektiven möglicher Stakeholder und deren Vorgaben eine tragfähige Basis realisierbarer Anforderungen zu entwickeln. Abbildung 2–3 zeigt exemplarisch drei Benutzergruppen und deren Wünsche, Bedürfnisse und Geschäftsvorgaben als Punkte. Gutes RE schafft gemeinsam mit dem Produktmanagement einen Schnitt dergestalt, dass einige wesentliche Funktionen so ausgewählt werden, dass möglichst viel Wert erreicht wird, bei gleichzeitiger Optimierung der Kosten. Das erfordert Verhandlungsgeschick und Durchsetzungskraft, denn wir können es nicht jedem Beteiligten recht machen. Wesentlich ist, dass wir dabei klar trennen, was unrealistische Wünsche sind, die oftmals nur als Testballons platziert werden, was tatsächliche Bedürfnisse sind und was die Geschäftsvorgaben sind. Letztere sind Pflicht, Bedürfnisse werden priorisiert und gegenübergestellt, und Wünsche fallen in der Regel heraus, wenn kein Business Case nachvollziehbar ist. Wir sprechen im Requirements Engineering von einer Kundenbeziehung. Dabei kann der Kunde ein externer Benutzer sein oder eine interne Abteilung.

Beispiel:

Marktanforderungen im Consumer-Bereich, wie bei einer digitalen Kamera, gehen sehr konkret auf Benutzungsaspekte ein, also auf die Lebensdauer der Akkus oder die Pixelzahl und damit auf die Bildschärfe und -auflösung. Diese Anforderungen an die digitale Kamera werden im Unternehmen, das sie herstellt, in Produktanforderungen übersetzt, die dann die Sprache des internen Produktmarketings oder Produktmanagements sprechen. Schließlich werden sie in Anforderungen an Komponenten übersetzt und sprechen die Sprache von Entwicklungsingenieuren oder Einkäufern, die diese Komponenten entwickeln oder beschaffen. Änderungen der Anforderungen werden hinsichtlich ihres potenziellen Einflusses auf bestehende Pläne und Produkte abgeschätzt, geprüft und in

→

die bestehenden Anforderungen aufgenommen. Anforderungen werden durch das ganze Projekt hindurch kontrolliert, um ihren Status zu kennen und um beurteilen zu können, wie weit das Projekt – aus Kundensicht – fortgeschritten ist.

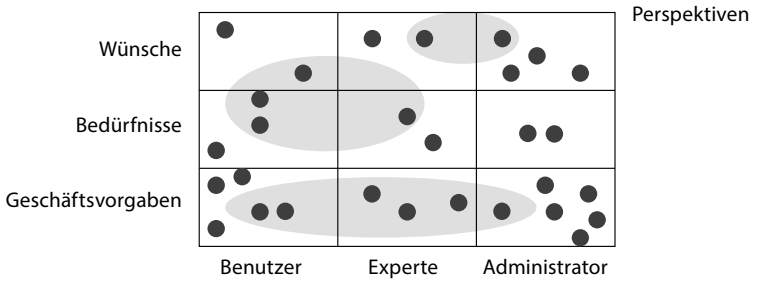


Abb. 2–3 *Aus verschiedenen Perspektiven belastbare Geschäftsanforderungen entwickeln*

Produktanforderungen

Produktanforderungen beschreiben Anforderungen an ein Produkt aus der Sicht der Realisierung einer späteren Lösung. Produktanforderungen beschreiben, **was** verschiedene Benutzer mit dem Produkt machen können und wie Marktanforderungen und Kundenbedürfnisse in ein Produkt umgesetzt werden. Sie beschreiben eine Eigenschaft in der Sprache des Produkts und werden daher auch als Funktionen, Eigenschaften oder Systemanforderungen bezeichnet. Sie definieren den Lösungsraum und die Prioritäten. Produktanforderungen werden im Pflichtenheft dokumentiert.

Beispiel:
Produktanforderung_1:
Jede einzelne Transaktion zwischen baulich getrennten Komponenten wird individuell verschlüsselt.

Produktanforderungen betrachten das Softwareprodukt oder den Dienst innerhalb eines größeren Kontextes, also der Umgebung, in der das System einmal arbeiten muss. Das kann ein PC sein, vor dem ein einzelner Benutzer sitzt (z.B. bei einem Computerspiel), eine Rechnerumgebung mit verschiedenen Benutzern (z.B. bei einem Ressourcenplanungssystem in einem Unternehmen), eine interaktive Online-Umgebung mit sehr vielen unbekannten Benutzern (z.B. bei einem Online-Buchungssystem), ein gemischtes Hardware-Software-System (z.B. bei einer Gebäudeautomatisierung) oder auch ein eingebettetes System, bei dem man kaum noch an Software denkt (z.B. ein Getränkeautomat, der in die Logistikkette eines Getränkelieferanten eingebaut ist).

Komponentenanforderungen

Komponentenanforderungen beschreiben Anforderungen an eine Komponente eines Produkts. Sie erläutern aus der Sicht der Realisierung und der späteren Lösung, *wie* Produktanforderungen durch eine Komponente des Produkts (z. B. Benutzerschnittstelle, Betriebssystem) adressiert werden.

Beispiel:

Komponentenanforderung_1:
Der Datenaustausch an der externen Schnittstelle xyz wird mit 128 Bit PGP-verschlüsselt.

Komponentenanforderungen dienen zur rekursiven Verfeinerung einer Produktanforderung oder eines Systems in handhabbare Teile. Aus der Sicht eines Lieferanten, der diese Komponente liefert, ist dies wiederum eine Marktanforderung. Komponentenanforderungen werden wie die Produktanforderungen auch im Pflichtenheft (in IT-Projekten oftmals auch Fachkonzept genannt) spezifiziert.

Viele Komponenten, wie Hardware und externe Software-Stacks, werden zugekauft. Das reduziert Kosten, verbessert die Qualität und erlaubt, sich auf den eigenen Wertbeitrag zu fokussieren. Daher müssen diese Anforderungen an externe Komponenten frühzeitig präzise spezifiziert werden, um danach die Entwicklungsarbeit verteilen und später die Ergebnisse integrieren zu können.

Die drei Sichten von Markt, Produkt und Komponenten sind nicht im Voraus oder von außen definiert, sondern hängen von der Lösungsstruktur ab.

Beispiel:

Der Benutzer oder Kunde stellt die folgende Anforderung:

M-Req-1: Das System verwaltet die Kundendaten im Format Name, Vorname, Adresse.

Der Requirements-Ingenieur spezifiziert dazu eine Lösung mit der folgenden Komponentenanforderung:

K-Req-1: Die Datenbank zur Verwaltung der Kundendaten wird mit Oracle 10 realisiert.

Offensichtlich sind dies zwei Anforderungen an die Datenhaltung, die aus verschiedenen Perspektiven beschrieben sind, nämlich Nutzung und Realisierung. Nun könnte aber auch der Kunde bereits eine technische Anforderung zur Realisierung haben, da er eine mySQL-Umgebung einsetzt und Kompatibilität sowie günstigere Lizenzkosten will. Er spezifiziert:

→

M-Req-2: Die Datenbank zur Verwaltung der Kundendaten wird mit mySQL realisiert.

Nun wird aus der bisherigen Komponentenanforderung (K-Req-1) eine Marktanforderung (M-Req-2). Ebenso gibt es Situationen, in denen das Lastenheft und die Geschäftsanforderungen so vage sind, dass das Pflichtenheft auch als Problembeschreibung fungiert. Anstatt über solche Feinheiten zu debattieren, ist es wichtig, dass die Anforderungen immer mit ihrer jeweiligen Quelle spezifiziert werden. Dann ist zu jedem Zeitpunkt klar, wie es zur Anforderung kam und welche Freiheitsgrade für eine Änderung bestehen, was die Realisierung erleichtert. M-Req-2 lässt sich sehr viel schwerer beeinflussen als die konzeptionell gleiche K-Req-1, da die M-Req-2 direkt vom Kunden stammt und daher die Lösung bereits definiert.

Anforderungen und Lösungen bedingen sich gegenseitig, und dürfen nicht vermischt werden. Die zwei Sichten auf Anforderungen und Lösungen sind in Abbildung 2–4 anhand typischer Arbeitsergebnisse konkretisiert. Horizontal sind Funktionen und Ziele aus Sicht des Markts (in B2C) oder eines Auftraggebers (in B2B) beschrieben. Vertikal ist die Lösung als spätere Realisierung dargestellt. Verschiedene Komponenten tragen dazu bei, dass Funktionen umgesetzt werden. Offensichtlich ist das keine 1:1-Beziehung, sondern eine bunt gemischte n:m-Darstellung.

Werden diese beiden sehr verschiedenen Sichten vermischt, führt das zu einem Strukturbruch (siehe auch Abb. 5). Dieser Strukturbruch ist einer der häufigsten Gründe für verkorkste Projekte und inflationäre Komplexität. Bereits in den späten Jahrzehnten des vergangenen Jahrhunderts war das als »Softwarekrise« bekannt. Aufgelöst wurde diese Krise erst durch modernes Requirements Engineering, das es mit seinen Perspektiven und der Verfolgbarkeit ermöglichte, eine Sicht in eine andere zu übertragen. Modelle helfen dabei, den Strukturbruch zu beherrschen und Konsistenz zwischen den Sichten (Problem vs. Lösung) zu erreichen (siehe Kap. 5). Beispielsweise muss die Systemumgebung sehr frühzeitig definiert werden. Ein Architekturmodell wiederum hilft bei der Identifizierung von Komponentenanforderungen.

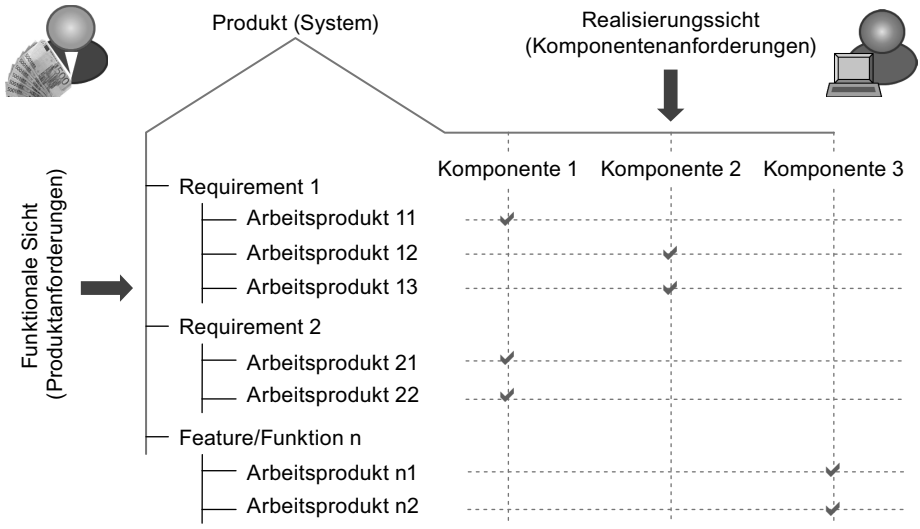


Abb. 2–4 *Komponentenanforderungen werden aus Produktanforderungen abgeleitet.*

Die hier beschriebene Systematik und Methodik gilt für jede Art von Produkten: Software, Services, Hardware, Mechatronik, künstliche Intelligenz – oder auch für ganz banale Konsumgüter und moderne Bauwerke. Wir unterscheiden kein Requirements Engineering anhand von Branchen oder Produkten. Requirements Engineering hat sich aus einem interdisziplinären Diskurs entwickelt, beispielsweise aus Patterns von (Gebäude-)Architekten und aus der Serviceorientierung in der Medizin. Insofern wollen wir hier auch keine künstlichen Gräben zwischen greifbaren Systemen und virtuellen Diensten aufreißen. Aus der Sicht des Requirements Engineering ist die Vorgehensweise identisch. Und Hand aufs Herz: Wo ist die Trennung in der Lösung? Die enge Verknüpfung von Systementwicklung und Dienstentwicklung, die gerade im Requirements Engineering explizit adressiert werden muss, zeigt Abbildung 2–5.

Beispiel: Serviceorientierung

Viele heutigen Dienste bestanden früher aus Hardware oder Software. Betrachten wir multimodale Transportlösungen, die verschiedene Verkehrsträger verschmelzen: Bis vor einigen Jahren standen an den Haltestellen von Bussen und Bahnen Telefonzellen, damit man anrufen konnte, um sich abholen zu lassen oder um ein Taxi zu bestellen. Heute nutzt man einen Mobilitätsservice, der einem die beste Verbindung von A nach B mit unterschiedlichen Verkehrsträgern darstellt. Das verknüpft Hardware (Zugsignalisierung) und Software (Apps und Betriebsleitzentralen) zu Diensten – mit einem serviceorientierten Requirements Engineering (siehe auch Abschnitt 11.5).

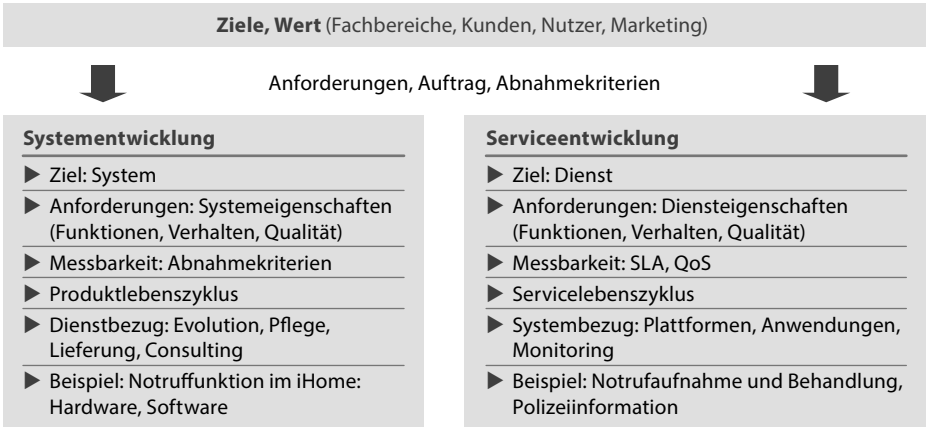


Abb. 2–5 *Requirements Engineering für Systeme und Dienste*

2.3 Arten von Anforderungen

Man unterscheidet drei Arten von Anforderungen (Abb. 2–6):

- Funktionale Anforderungen
- Qualitätsanforderungen
- Randbedingungen

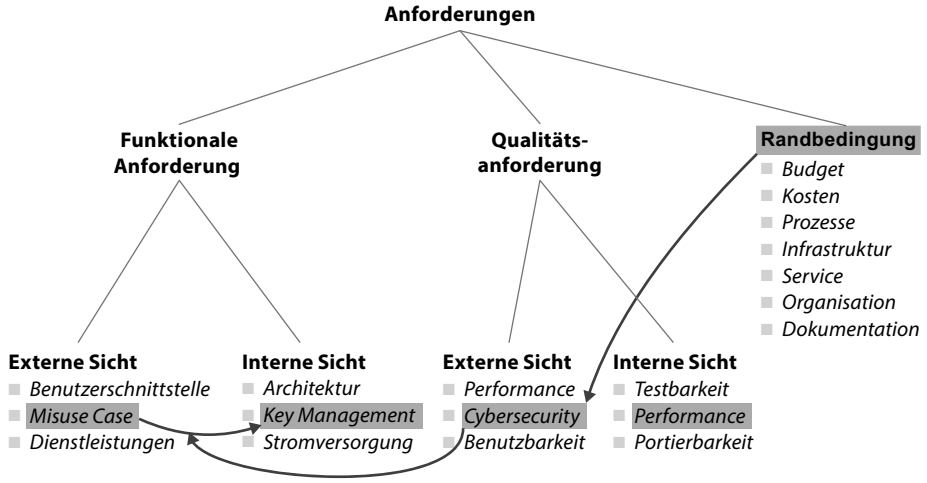


Abb. 2–6 *Unterschiedliche Typen von Produktanforderungen*

Funktionale Anforderung

Eine funktionale Anforderung beschreibt eine vom System oder einer Systemkomponente bereitzustellende Funktion des betrachteten Systems. Sie erläutert in der Sprache des Systems, was das System tun soll, beispielsweise die Berechnung einer Ausgangsgröße aus Eingangsgrößen durch Anwendung eines Algorithmus.

Beispiel:
Der Datenaustausch an der externen Schnittstelle xyz wird mit 128 Bit PGP-verschlüsselt.

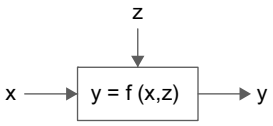
Funktionale Anforderungen beschreiben funktionsorientiert und in der Sprache des Produkts, was das Produkt tut. Sie lassen sich in der Entwicklung verfolgen und auch validieren. Man kann für eine bestimmte Funktion einen Testfall schreiben, der später in verschiedenen Testphasen geprüft wird.

Beispiele für solche funktionalen Anforderungen sind Funktionen, Ablaufbeschreibungen und Szenarien, die angeben, wie ein System auf bestimmte Eingangsgrößen oder Eingaben zu reagieren hat (Abb. 2–7). Dazu gehören auch Daten- oder Schnittstellenanforderungen, die nötig sind, um das zu entwickelnde System in einer bestimmten Umgebung einsetzen zu können. Geschäftsprozesse und Workflows sind ebenfalls funktionale Anforderungen.

Ein **Geschäftsprozess** beschreibt eine Folge zusammengehöriger Aktivitäten, die schrittweise ausgeführt werden, um ein geschäftliches oder betriebliches Ziel zu erreichen. Er beschreibt als Anforderung, wie das System intern operiert, um die Anforderungen der Umwelt zu erfüllen. Zur Vereinfachung werden Geschäftsvorfälle genutzt, die den Geschäftsprozess als Instanz konkretisieren. Geschäftsvorfälle sind Vorgänge, die die Vermögenszusammensetzung in einem Unternehmen beeinflussen oder verändern.

Workflows als Anforderungen beschreiben eine inhaltlich abgeschlossene, zeitlich zusammenhängende Folge von Aktivitäten, die zur Bearbeitung eines betriebswirtschaftlich relevanten Objekts notwendig sind und deren Funktionsübergänge von einem Informationssystem gesteuert werden. Der Workflow beschreibt eine Prozesssicht, während der Geschäftsprozess die Sicht auf betriebswirtschaftliche Faktoren betrachtet.

Ein **Anwendungsfall** oder **Use Case** schließlich als Beispiel für eine funktionale Anforderung beschreibt den Bezug einer Systemleistung durch die Außenwelt. Anwendungsfälle vermitteln also, was die Umwelt vom System erwartet. Sie sind daher vor allem zur Beschreibung von Abläufen an Schnittstellen geeignet.



- x: Eingangsgröße
- y: Ausgangsgröße
- z: Randbedingungen, z.B. Datenformate, Ereignisse
- f: Algorithmus, Abbildung von Eingangsgröße unter gegebenen Randbedingungen auf Ausgangsgröße z.B. berechnen, übertragen, wandeln
- Quantifizierung z.B. 10V, 200Nm/s etc.

Abb. 2–7 *Funktionale Anforderungen*

Qualitätsanforderung

Eine Qualitätsanforderung beschreibt eine qualitative Eigenschaft, die das betrachtete System oder einzelne Komponenten des Systems aufweisen müssen (siehe auch Abschnitt 3.5). Qualitätsanforderungen (manchmal auch **nichtfunktionale Anforderungen** genannt) ergänzen die funktionalen Anforderungen.[1] Beispiele sind Zuverlässigkeit, Verfügbarkeit, Wartbarkeit, funktionale Sicherheit und Cybersecurity.[2]

Beispiel:
Die Verschlüsselung und Entschlüsselung einer Transaktion muss innerhalb von einer Millisekunde abgeschlossen sein.

Qualitätsanforderungen sind nur aus der Systemsicht beschreibbar. Oft werden nur die funktionalen Anforderungen hinreichend präzise spezifiziert, während die Qualitätsanforderungen vage bleiben. Bestimmte funktionale Anforderungen können spezifische Qualitätsanforderungen bedingen, beispielsweise die Robustheit gegenüber unzulässigen Eingaben oder Signalrauschen.

Zur besseren Verfolgbarkeit sollten Qualitätsanforderungen in funktionale Anforderungen »übersetzt« werden. Das erleichtert die Verfolgbarkeit und Validierung. Beispielsweise lässt sich Wartbarkeit durch Lesbarkeitsindizes und Reviews prüfen. Anforderungen an die Sicherheit lassen sich durch Reviews und Verweise auf einschlägige Standards prüfen. Mit wachsender Produkthaftung und expliziten Vertragsstrafen, die sich auf Nichterfüllung von Anforderungen beziehen, wächst das Interesse bei Kunden und Lieferanten, alle Anforderungen so präzise zu definieren, dass sie eindeutig implementiert, geprüft und abgenommen werden können.

---

1. Wir verwenden in diesem Buch konsequent die Bezeichnung »Qualitätsanforderungen«.
2. Im Englischen auch als RAMSS (Reliability, Availability, Modifiability, Safety, Security) bezeichnet.

Bei den funktionalen Anforderungen und Qualitätsanforderungen unterscheiden wir eine interne (Entwicklung) und eine externe (Kunde, Benutzer) Sichtweise (Abb. 2–6, unten). Die interne Sichtweise beschreibt Anforderungen, die vor allem aus Entwicklungssicht eine Rolle spielen. Dazu gehören beispielsweise konkrete Anforderungen zum Stromverbrauch oder zur Architektur, aber auch Qualitätsanforderungen, wie die Validierbarkeit. Die externe Sichtweise spiegelt die Kunden- oder Benutzersicht wider. Dabei geht es um direkten Zusatznutzen aus der Sicht dessen, der dafür bezahlen soll. Die meisten explizit beschriebenen Anforderungen in einem Software- oder Systemprojekt fallen in diese Kategorie der funktionalen Anforderungen aus Benutzersicht.

Randbedingung

Randbedingungen sind Anforderungen, die die Art und Weise einschränken, wie das betrachtete System realisiert werden kann. Randbedingungen ergänzen die funktionalen Anforderungen und die Qualitätsanforderungen. Beispiele sind Kosten, Geschäftsprozesse und Gesetze.

Beispiel:
Die Verschlüsselung einer Transaktion muss den gesetzlichen Anforderungen des BSI genügen.

Randbedingungen beschreiben Grenzen, Vorgaben und Beschränkungen, beispielsweise durch Geschäftsprozesse oder Infrastruktur. Häufig umfasst dies heutzutage eine ganze Anzahl von Lieferanten, Unterauftragnehmern oder Offshore-Outsourcing-Partnern. Bereits hier werden die Randbedingungen des späteren Projekts definiert, denn schließlich müssen der Preisrahmen und die geplanten Umsatzzahlen im Marketing lange vor der exakten Spezifikation bekannt sein.

Auch organisatorische Randbedingungen spielen eine Rolle, obwohl sie nur ungern als Anforderungen zugegeben werden. Jedoch gibt es seit Jahren Untersuchungen über den Einfluss der Organisationsform auf die Architektur. Melvin Conway hat daraus bereits vor knapp 50 Jahren ein Gesetz abgeleitet, das bis heute nach ihm benannt ist [Conway1968]. Das Gesetz von Conway (Conway's Law) besagt, dass die Struktur und Architektur eines Produkts die organisatorische Struktur widerspiegeln. Eine Matrixorganisation unterstützt beispielsweise eine modulare Struktur. Ist die Organisation eher informell, wie beispielsweise bei Start-ups, wird man vergeblich nach klaren Schnittstellen innerhalb des Produkts suchen.

Gesetzliche Vorgaben oder Standards sind Randbedingungen, die direkt zu funktionalen Anforderungen oder Qualitätsanforderungen führen. Viele Ausschreibungen im System- und Softwarebereich fordern heute eine hinreichende Prozessfähigkeit des Lieferanten. Hintergrund dafür ist, dass die Kunden immer weniger in der Lage sind, genau zu beurteilen, ob die Lieferanten tatsächlich auf der Höhe

der Zeit sind und ob sie ihre Zusagen auch einhalten können. Die Produkthaftung verlangt beispielsweise, dass der Lieferant den Stand der Technik beherrscht. *Dieses Buch beschreibt den Stand der Technik im Requirements Engineering.*

2.4 Was ist Requirements Engineering?

Requirements Engineering (RE) ist das disziplinierte und systematische Vorgehen (d.h. »Engineering«) zur Ermittlung, Dokumentation, Analyse, Prüfung, Abstimmung und Verwaltung von Anforderungen unter kundenorientierten, technischen und wirtschaftlichen Zielvorgaben. Das Ziel von RE ist es, qualitativ gute – nicht perfekte – Anforderungen zu entwickeln und sie in der Umsetzung risiko- und qualitätsorientiert zu verwalten. Systematisches RE macht den Unterschied aus zwischen einem erfolgreichen Produkt und einer Sammlung irrelevanter Funktionen.

Diese Systematik mit sechs konkreten Aktivitäten zeigt Abbildung 2–8. Wir haben bewusst keine zeitlichen oder kausalen Abhängigkeiten dargestellt, denn das ist, wie Sie später sehen werden, gar nicht so einfach.

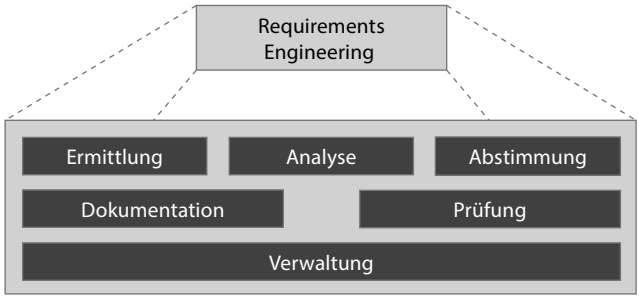


Abb. 2–8 *Aktivitäten des Requirements Engineering*

Requirements Engineering ist eine fachübergreifende Disziplin. Requirements Engineering bedient sich der Erfahrungen aus der Systemtechnik, der Psychologie, der Betriebswirtschaftslehre, dem Marketing, dem Produktmanagement, dem Projektmanagement und natürlich der Informatik.

Requirements Engineering ist eine Kerndisziplin aller Ingenieurwissenschaften und damit auch der Softwaretechnik und der Systemtechnik. Requirements Engineering ist nicht originär für die Softwaretechnik, und viele Methoden sind in andere Systeme transferierbar. Dieses Buch betrachtet das RE aus diesem Grund ganzheitlich und branchenübergreifend. Wir sprechen hier über Systementwicklung, denn häufig wird Software als Bestandteil eines größeren Systems geliefert [INCOSE 2015]. Von einem *System* ist die Rede, wenn es sich um eine Verbindung von Hardware, Software, Prozessen und Personen handelt, die gemeinsam die Fähigkeit haben, ein bestimmtes Ziel zu erreichen oder bestimmte Eigenschaften auszuprägen.

Requirements Engineering schafft eine gemeinsame Basis zu Zielen und Anforderungen zwischen den Benutzern und den Entwicklern eines Produkts. Oft sind

die Kunden nicht die Benutzer, was im RE zu massiven Konflikten führen kann. Kunden und Benutzer sind auch nicht notwendigerweise immer außerhalb der eigenen Firma angesiedelt. Damit spielt RE eine Schlüsselrolle während der gesamten Produktentwicklung (Abb. 2–9).

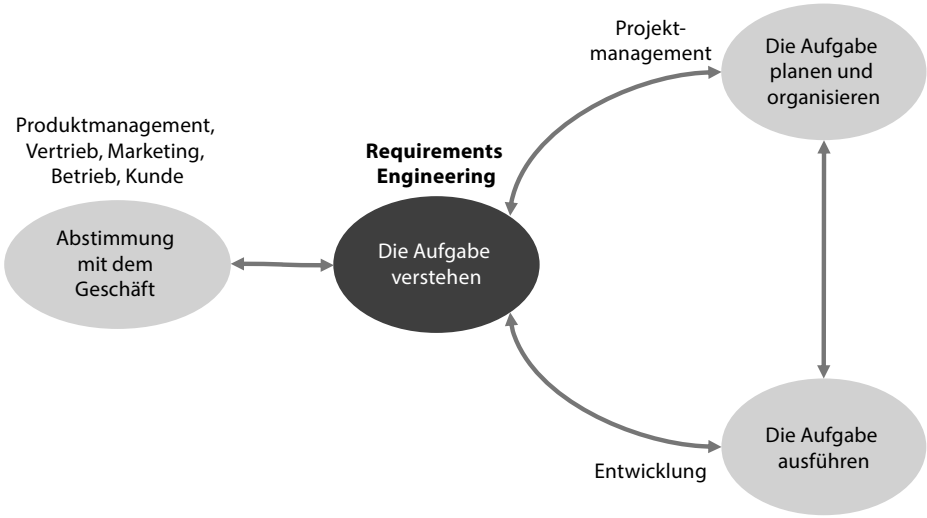


Abb. 2–9 *Requirements Engineering im Kontext der Produktentwicklung*

Requirements Engineering bringt verschiedene Interessen und Sichtweisen unter einen Hut. Es begrenzt die Probleme, die im Projekt sowieso auftreten. RE ist vor diesem Hintergrund eine Sisyphusarbeit. Egal, wo man anpackt, überall tun sich Lücken, Unklarheiten und Unschärfen auf. Wir können mit den Anforderungen nicht alle Inhalte endgültig klären. Das führt zu Problemen an den Stellen, die wir mit weniger Energie verfolgen. Ob die Probleme in Ihrem Unternehmen auftreten oder auf der Kundenseite, ist nahezu egal. Es trifft Sie. Ein Kunde, der zu lange warten muss und der nicht erhält, was er will, ist unzufrieden – selbst, wenn das Projekt im Budget abgeschlossen hat. Das Gleiche gilt, wenn Sie eine Funktion, die dem Kunden wichtig ist, herunterpriorisiert haben. RE hat daher viel mehr »politische« und psychologische Aspekte, als man gemeinhin wahrhaben will.

Requirements Engineering bestimmt die Wertschöpfung im gesamten Lebenszyklus. Abbildung 2–10 zeigt, wie Anforderungen zielorientiert die Bereiche des Unternehmens auf den Markt und den Kunden einstellen. Im Marketing werden Kaufkriterien bewertet. Der Vertrieb schafft mit dem Marketing und der Produktentwicklung eine Wertvorstellung, die dann durch die Entwicklung umgesetzt wird. Nachhaltiger wirtschaftlicher Erfolg bei Software entsteht durch ein funktionierendes Servicemodell. Damit wird der Wert gesichert und neue Kaufabsichten werden stimuliert. Die verbindenden Pfeile sind die Anforderungen in verschiedenen Stadien: ein Kreislauf, wie ihn erfolgreiche Unternehmen vorleben.