Christof Ebert

Systematisches Requirements Engineering

Anforderungen ermitteln, dokumentieren, analysieren und verwalten

dpunkt.verlag

Christof Ebert ist Geschäftsführer der Vector Consulting Services. Zuvor war er zwölf Jahre in Führungsaufgaben bei einem IT-Marktführer tätig, zuletzt mit weltweiter Verantwortung für Softwareplattformen. Er arbeitet in verschiedenen industriellen Boards und Aufsichtsgremien, ist Professor an der Universität Stuttgart sowie an der Sorbonne in Paris und Autor mehrerer Bücher. Seit vielen Jahren ist er in den Herausgeberkomitees von führenden Zeitschriften wie »IEEE Software« und »Journal of Systems and Software«.

Folgen Sie ihm auf Twitter: @ChristofEbert

Kontakt: christof.ebert@vector.com, *www.christofebert.de*

Homepage des Buches: *www.vector.com/RE-Buch*

Der QR-Code rechts führt Sie schnell zur Homepage des Buches.

Christof Ebert

Systematisches Requirements Engineering

Anforderungen ermitteln, dokumentieren, analysieren und verwalten

6., überarbeitete und erweiterte Auflage

dpunkt.verlag

Christof Ebert
christof.ebert@vector.com, www.christofebert.de

Lektorat: Christa Preisendanz
Copy-Editing: Friederike Daenecke, Zülpich
Satz: lll-satz, www.drei-satz.de
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Fachliche Beratung und Herausgabe von dpunkt.büchern im Bereich Wirtschaftsinformatik:
Prof. Dr. Heidi Heilmann · heidi.heilmann@augustinum.de

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Buch 978-3-86490-562-9
PDF 978-3-96088-452-1
ePub 978-3-96088-453-8
mobi 978-3-96088-454-5

6., überarbeitete und erweiterte Auflage 2019

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Meinen Eltern Elfriede und Otto
und meinem Lehrer Rudolf Lauber.
Sie haben mir gezeigt,
dass ein *Werk* nur Wert schafft,
wenn die *Anforderungen* verstanden und umgesetzt sind.

Vorwort zur 6. Auflage

Es ist nicht genug zu wissen, man muss es auch anwenden.
Es ist nicht genug zu wollen, man muss es auch tun.
Johann Wolfgang von Goethe

Unternehmen ohne Requirements Engineering laufen im Nebel ständig gegen Hindernisse. Jene mit durchschnittlichem Requirements Engineering lernen, wo die Hindernisse sind; Unternehmen mit gutem Requirements Engineering bauen Hindernisse für andere auf. Oft erkennt man diese Hindernisse erst, wenn es zu spät ist. Dieses Buch ist aus der Praxis für die Praxis geschrieben und zeigt Ihnen, wie Sie Hindernisse frühzeitig aus dem Weg schaffen und damit Wert für sich und Ihr Unternehmen kreieren.

Anforderungen sind der Schlüssel zum Erfolg. Projekte scheitern, weil ihre Ziele und Anforderungen unklar sind. Produkte kommen am Markt nicht an, weil sich niemand Gedanken über die Ziele und Randbedingungen gemacht hat. Ständige Nachbesserungen sind teuer, zu komplex oder sind inakzeptabel. Das weiß jeder, und trotzdem tappt auch in diesem Jahr wieder ein Drittel aller Projekte in genau diese Falle. Offensichtlich ist es nicht genug zu wissen, man muss es auch wollen und tun.

Systematisches Requirements Engineering entscheidet über Erfolg oder Misserfolg eines Projekts und Produkts. Dieses Buch beschreibt praxisorientiert und systematisch das gesamte Requirements Engineering – von der Konzeption bis zur Evolution eines Projekts oder Produkts. Es adressiert Requirements Engineering in einem breiten Kontext, sodass sich ganz unterschiedliche Anwendungsbereiche wiederfinden, sei es Software und IT, aber auch Hardware, Systemtechnik oder Serviceentwicklung.

Auch in seiner sechsten Auflage ist dieses Buch eine Referenz für Ihren Schreibtisch. So ist diese komplett überarbeitete Auflage auch für bisherige Leser lohnend. Neue Beispiele, Tipps, Abbildungen und Inhalte schaffen Abwechslung zu den früheren Auflagen. Agiles Requirements Engineering hat nun ein eigenes Kapitel, und eine Fallstudie zu IoT macht die praktische Umsetzung lebendig.

Die meisten Projekte umfassen Änderungen existierender Systeme. Das Buch adressiert diese Situation und bewegt sich nicht akademisch auf der »grünen

Wiese«. Der Kanon des *IREB Certified Professional Requirements Engineer* wird abgedeckt und dort erweitert, wo nach Ansicht des Autors im Lehrplan Inhalte fehlen, beispielsweise bei der Schätzung und im Test. Die Checklisten wurden erneuert, denn man lernt in Projekten ständig dazu. Ein Selbsttest hilft bei der Bewertung Ihrer Fähigkeiten im Requirements Engineering. Der Nutzen und ROI von Requirements Engineering wird an verschiedenen Stellen herausgestellt. Damit haben Sie konkrete Ansatzpunkte, wie Sie mit Ihren eigenen Herausforderungen umgehen können. Die vorgestellten Vorlagen und viele aktuelle Tipps sowie Hinweise auf Trainings sind auf der Homepage dieses Buches verfügbar: *www.requirements-excellence.com*.

Mein Dank geht an die Mitarbeiter und Kunden von Vector Consulting, mit denen wir die genannten Praktiken umsetzen und verbessern. Besonders danken möchte ich Kai Rüdele für seine Impulse. Danke an Christa Preisendanz und den dpunkt.verlag für die gewohnte Geduld und Professionalität.

Al Davis hatte mich vor 25 Jahren als Chefredakteur von *IEEE Software* dazu angeregt, Requirements Engineering in der Industrie zu verankern. Dank an Al und die vielen Kollegen, die dieses Thema in der Praxis antreiben, wie Ian Alexander, Dan Berry, Anthony Finkelstein, Don Gause, Michael Goedicke, Martin Glinz, Matthias Jarke, Neil Maiden, Barbara Paech, Klaus Pohl, Mary Popendieck, Suzanne Robertson, Ian Sommerville und Karl Wiegers.

Viele Leser der früheren Auflagen haben mir wertvolle Tipps für diese Überarbeitung gegeben. Wie bereits bisher stehe ich Ihnen, verehrte Leserinnen und Leser, während und nach der Lektüre des Buches für Fragen und Unterstützung gerne zur Verfügung. Um die Sätze kurz zu formulieren, verwende ich im Folgenden übrigens die männliche Form – wohl wissend, dass gerade das Requirements Engineering stark durch Frauen geprägt wird. Ich hoffe, die Kolleginnen werden es mir nachsehen.

Produkte sind, was wir liefern. Wert ist, was die Kunden wahrnehmen. Das läuft oft stark auseinander. Mehr Requirements bedeuten nicht unbedingt mehr Wert, aber definitiv mehr Kosten und Komplexität. Zu viele oder falsche Anforderungen ruinieren Budgets, Termine und die Qualität. Man startet mit unklaren Bedürfnissen und Zielen, rudimentären Ideen, Luftschlössern aus dem Vertrieb, nicht bewerteten Abhängigkeiten und vagen Vermutungen. Später rächt sich diese Nachlässigkeit in Gestalt von Änderungen und Nacharbeit. Entwickeln Sie nicht nur Anforderungen, sondern verfolgen Sie das Ziel, für Ihre Kunden Wert zu schaffen – und für Ihre Mitbewerber Hindernisse.

Ihnen, liebe Leser und Kunden, wünsche ich dabei viel Erfolg! Dieses Buch gibt Ihnen mit lösungsorientiertem Requirements Engineering die nötigen Impulse. Es liegt ganz an Ihnen, ob Sie nur wissen oder auch wollen und tun – und damit Erfolg haben.

Christof Ebert
Bengaluru, September 2018

Stimmen zum Buch

»... ein hervorragendes Buch für den praxisnahen Einstieg in die vielschichtigen Themenkomplexe der Anforderungsanalyse und des Anforderungsmanagements.«

Chip.de (Juli 2010 zur 2. Auflage)

*

»›Systematisches Requirements Engineering‹ ist die wertvollste Anleitung für Anforderungen, die Sie finden können. Christof Ebert deckt die gesamte Landschaft von Praktiken ab, die ein Requirements-Ingenieur, Projektmanager oder Produktmanager kennen sollte. Für Praktiker und Manager gleichermaßen kann ich dieses Buch nicht hoch genug empfehlen. Ich war schon immer ein Fan von seinen Schriften – und werde es auch weiterhin sein!«

Alan M. Davis, Entrepreneur und Professor
University of Colorado, Colorado Springs

*

»Inzwischen ein Klassiker für den systematischen Umgang mit Anforderungen. Geschrieben von einem Praktiker für die Praxis – einfach, verständlich und anwendbar! Dass der Autor sein Metier versteht, durfte ich in einem gemeinsamen Praxisprojekt hautnah erleben.«

Hans Leibbrand
ehem. Chief Operating Officer und Vorstand, Thales

*

»Mit den Anforderungen werden die Weichen gestellt für den Projekterfolg – oder Misserfolg. Aus fehlerhaften, unvollständigen oder widersprüchlichen Anforderungen wird niemals gute Software entstehen. Das vorliegende Buch hilft, den richtigen Einstieg in die Softwareentwicklung zu finden und die vielen Klippen zielgerichtet zu vermeiden.«

Peter Liggesmeyer, Direktor Fraunhofer IESE
ehem. Vorsitzender der Gesellschaft für Informatik

*

»Christof Ebert schafft es, sowohl Frischlingen als auch alten Hasen Neues beizubringen. Anschaulich und ohne Besserwisserei zeigt er, wie Requirements Engineering im Unternehmen optimal aufgesetzt wird. Ein Muss für Entscheider und alle, die Erfolg bei Softwareprojekten haben wollen.«

Gerhard Mack, Chief Operating Officer und Vorstand
Vodafone – Kabel Deutschland

*

»Christof Ebert hat ein Talent dafür, zum Kern der Sache zu kommen und die einfache (aber schwer erkennbare) Wahrheit freizulegen. Danke für die immer klar und elegant formulierten Ratschläge!«

Suzanne Robertson, Gründerin und Principal
The Atlantic Systems Guild Ltd.

*

Inhaltsverzeichnis

1 Motivation

Ihr Nutzen aus diesem Kapitel:

»*Wenn Du nicht weißt, wohin Du gehen willst, kannst du jeden Weg nehmen.*« Alice im Wunderland wusste es, wie auch viele Denker vor und nach ihr. Klare Ziele werden erreicht, unklare Ziele werden sicher verfehlt. In diesem Kapitel zeige ich, wie Sie mit Requirements Engineering Ziele schrittweise klären und kommunizieren. Insbesondere möchte ich den Nutzen eines systematischen Requirements Engineering darstellen. Beantworten Sie die folgenden Fragen einfach spontan und ehrlich: Sind Ihre Anforderungen strukturiert und testbar dokumentiert? Hat Ihr derzeitiges Projekt einen expliziten Business Case, der auch geprüft wird? Gibt es für jede Anforderung eine kurze Begründung, die den Nutzen und Wert beschreibt? Falls nicht, ist das Buch das Richtige für Sie. Falls ja, lesen Sie die Fragen nochmals und gehen Sie in sich.

1.1 Warum ein Buch über Requirements Engineering?

»*Am Anfang wurde das Universum erschaffen. Das machte viele Leute sehr wütend und wurde allenthalben als Schritt in die falsche Richtung angesehen.*« Douglas Adams hatte in seinem fünfbändigen Werk »Per Anhalter durch die Galaxis« präzise erkannt, warum im Leben, im Universum und dem ganzen Rest nicht immer alles so klappt, wie wir wollen. Die Anforderungen sind häufig unpräzise – oder fehlen.

Wir Menschen entwickeln uns, weil wir aus Anforderungen Lösungen machen. In den Sechzigerjahren überlegte man sich, wie man im All schreiben kann. Kugelschreiber funktionierten aufgrund der fehlenden Schwerkraft nicht so gut. Die Amerikaner entwickelten einen aufwendigen »Space Pen«, der ganz aus Metall gefertigt und gekapselt in einem weiten Temperaturbereich auf verschiedenen Oberflächen und ohne Schwerkraft funktioniert. Die Russen mit den gleichen Anforderungen nutzten dafür einen einfachen Bleistift.

Komplexe Anforderungen brauchen keine komplizierten Lösungen. Die kleine Anekdote aus den unergründlichen Weiten des Weltalls zeigt uns noch mehr. Funktionale Anforderungen benötigen zur praxisorientierten Umsetzung flankierende Qualitätsanforderungen und einschränkende Randbedingungen. Da fällt der Bleistift natürlich durch, da Holz und Graphit in der sauerstoffreichen Atem-

luft eines Raumschiffes ein Brandrisiko darstellen. Abgebrochene Bleistiftminen sind zudem in der Schwerelosigkeit gefährlich, denn sie könnten eingeatmet werden oder ins Auge gelangen. Die Amerikaner setzten jedenfalls den Space Pen mit Erfolg ein – allerdings zum Preis von drei Dollar pro Stück, und auf der Erde.

Oft zerbrechen wir uns vorschnell den Kopf über eine Lösung, ohne verstanden zu haben, welches Problem wir lösen müssen. Wir gehen zu Besprechungen, ohne zu hinterfragen, was sie bringen. Wir entwickeln Funktionen für ein Softwaresystem und wissen nicht, welchen Wert sie für die Käufer darstellen. Wir optimieren und bemühen uns ständig, bessere Produkte zu entwickeln – und fühlen uns wie im Hamsterrad. Während des Projekts wundern wir uns, dass sich die Anforderungen ständig ändern. Dabei war niemals klar, was wir eigentlich konkret erreichen wollen – und was nicht.

Abbildung 1-1 zeigt ein typisches Beispiel einer Anforderung. Sie ist Prosa, weil das leichter zu schreiben ist, und wirft mehr Fragen auf, als sie beantwortet:

- Wie hängen diese unstrukturierten Funktionen zusammen?
- Bringt die Anforderung Kundennutzen und damit Profit?
- Kann die Anforderung überhaupt umgesetzt werden?
- Wie ist der Status der Anforderung?
- Wer ist für die Anforderung verantwortlich?
- Wie wird die Anforderung validiert?
- Wie wird die Anforderung umgesetzt?

REQ_0815

```
iHome wird mit einem Control Panel
gesteuert, das als App für Smartphone und
Tablet verfügbar ist. Die typischen
Parameter wie Licht und Temperatur werden
dargestellt. Der Benutzer stellt die
Parameter auf dem Dashboard ein.
```

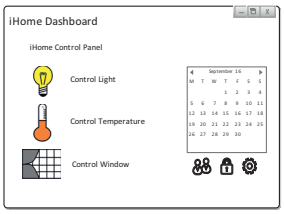


Abb. 1-1 *Eine Anforderung – so viele Fragen*

Anforderungen an Software werden zunehmend komplexer. Abbildung 1-2 zeigt die Entwicklung verschiedener Softwaresysteme, die wir untersucht haben.[1] Auf der waagrechten Achse sind die Jahreszahlen angegeben, während senkrecht das Softwarevolumen in tausend Objektcodebefehlen dargestellt ist. Diese Darstellung erschien uns als die einzig praktikable, wenn wir so unterschiedliche Systeme wie Apps für Mobiltelefone und eingebettete Software vergleichen wollen. Der Umfang der Software verdoppelt sich alle zwei bis vier Jahre. Mit diesem

1. Quellen der Daten: Eigene Studien des Autors, Vector Benchmarking Datenbank, NASA, Codeanalysen bei Windows und Linux. Konversionen soweit nötig gemäß den anerkannten Korrekturfaktoren [Hatton2005, Ebert2007].

Wachstum steigt auch der Umfang der Spezifikationen an. Gab es Anfang der Neunzigerjahre beispielsweise einige wenige Steuergeräte in einem Neuwagen mit ungefähr hundert Seiten an Spezifikationen, so sind es heute bereits fünfzig und mehr Steuergeräte mit über 100.000 Seiten an Spezifikationen. Diese schnell wachsende Komplexität fordert systematisches Requirements Engineering (RE), um die Qualität und Kosten nachhaltig kontrollieren zu können.

Ein gutes Beispiel für diese nicht hinterfragte künstliche Komplexität sind Migrationsprojekte. Die Anspruchsträger sind sich oft schnell darin einig, dass alle Funktionen des existierenden Altsystems übertragen werden müssen. Ein Fehler. Erstens kann sowieso niemand mehr alle existierenden Altfunktionen im Zusammenhang beschreiben. Und zweitens ist gerade ein neues System die einzige Chance, alte Funktionen und Workflows über Bord zu werfen. Dass es anders geht, zeigen Start-ups und exzellente Unternehmen wie Apple. Bei ihnen lautet die erste Frage nicht, welche Komplexität noch zusätzlich entwickelt werden soll, sondern wie das Produkt hinreichend schlank bleibt.

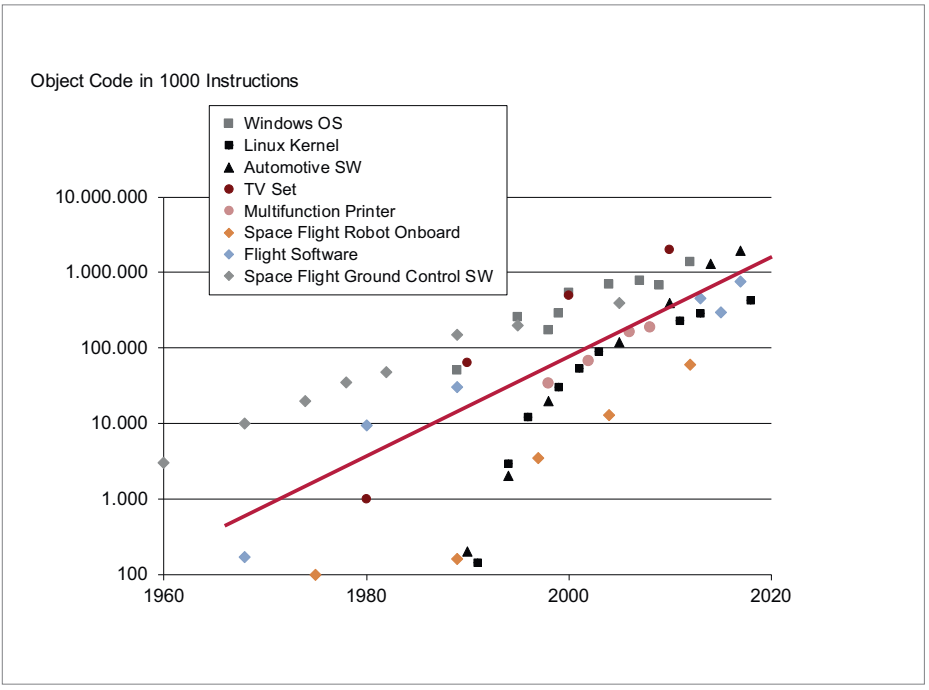


Abb. 1-2 *Komplexität von Softwaresystemen über die Zeit*

Erfahrene Projektmanager und Entwickler wissen, dass es erprobte Methoden sowie werkzeugunterstützte Hilfsmittel für das Requirements Engineering gibt. Häufig fehlt ihnen aber der Überblick über die Theorie und Praxis des Requirements Engineering, um die für ihre Situation passenden Methoden, Verfahren

und Hilfsmittel auszuwählen, sowie die notwendige Kenntnis im Detail, um sie produktiv nutzen zu können.

Das Buch füllt diese Lücke. Es liefert umsetzungsorientiert Theorie und Praxis des Requirements Engineering. Die gängigen Verfahren der Anforderungsanalyse sind beschrieben. Die Leser erhalten Einblick in die Art und Weise, wie Anforderungen ermittelt, entwickelt, dokumentiert und im Projekt verfolgt werden. Die grundsätzlichen Methoden, Verfahren, Werkzeuge und Notationen des Requirements Engineering werden übersichtlich behandelt. Sie werden durch konkrete Beispiele aus der Projektarbeit illustriert. Notationen und Modelle sind in der Regel mit UML 2.0 beschrieben. Fallstudien demonstrieren die konkrete Umsetzung und Erfahrungen aus der Praxis.

1.2 Projekte scheitern wegen unzureichender Anforderungen

Zu viele Projekte scheitern, und Produkte erreichen die Marktziele nicht. Unzureichendes Requirements Engineering ist immer unter den Top-3-Ursachen. Eine aktuelle Studie der Standish Group zeigt, dass ein gutes Drittel aller Projekte erfolgreich abgeschlossen wird. Ein Fünftel wird abgebrochen, und der Rest kommt zwar zu einem Abschluss, aber nur unter Aufgabe von ursprünglichen Zielen (Abb. 1-3) [Standish2018]. RE erhält im Schnitt nur 2–5 % des Projektaufwands, aber Fehler in den Anforderungen haben die größten Effekte [Gartner2018]. **Die meisten abgebrochenen Projekte hatten nur ungenügend geklärte initiale Anforderungen und konnten Änderungen der Anforderungen nicht beherrschen** [Anthopoulos2016, Standish2018, Charette2017, Tan2011, Ebert 2014a, Ebert2007].

Beispiel:

Viele Projekte scheitern, da sich Anforderungen ändern, aber die Projektstruktur und IT-Architektur das nicht berücksichtigen. Aktuelles Beispiel ist das amerikanische Versicherungssystem »Obamacare«. An das Projekt wurden politisch hohe Erwartungen geknüpft, sollte es doch erstmals einen Basisschutz für alle amerikanischen Bürger schaffen. Doch die Webseite mit der Online-Registrierung lieferte nie die erwartete Performance. Kundendaten verschwanden oder mussten wiederholt eingegeben werden. Schließlich funktionierte die Webseite rudimentär, stimulierte aber politische Gegner, damit das gesamte Programm zu hinterfragen. Die Mehrkosten der IT liegen im Bereich von knapp 500 Mrd. US$. Die Gründe für das Scheitern waren zu viele Anspruchsträger, die mitwirkten, sich stark ändernde Anforderungen, eine unzureichende Validierungsstrategie und ein monolithisches System, das ungeeignet für Teillösungen und inkrementelle Änderungen war. Viele eGovernment-Projekte leiden unter diesem Dilemma, es allen Anspruchsträgern recht machen zu wollen. Damit haben sie ein so weiches Ziel, dass das System nie fertig wird [Anthopoulos2016].

Die wesentlichen Befunde aus Kundenprojekten, bei denen wir zur Unterstützung und Moderation gerufen wurden, sind im Folgenden aufgelistet – als Warnung, aber auch, damit Sie sich selbst prüfen können:

- Unbrauchbare Lastenhefte und Ausschreibungen (z.B. oberflächlich und missverständlich beschriebene Anforderungen)
- Implizite Anforderungen (Beispiel: Kunde erwähnt Funktionen nicht, da sie für ihn selbstverständlich sind, aber der Lieferant weiß das nicht.)
- Fehlende Anforderungen (z.B. schwammige Anforderungen, die zwar nötig sind, aber nicht geklärt werden, da sie teuer werden könnten; unklare Ausrichtung des Projekts: Was wird nicht geliefert?)
- Unsicherheiten und Unklarheiten (Beispiel: Schätzungen und Pläne basieren auf nicht verstandenen Risiken und oberflächlich dokumentierten Anforderungen.)
- Unzureichendes Änderungsmanagement (Beispiel: Kunde meldet sich beim Projektmanager oder Entwickler: »Wir brauchen das und das noch.«)
- Inkonsistente Dokumentation (Beispiel: Testfälle setzen auf einer anderen Basis auf als die Entwicklung.)
- Varianz und Komplexität (z.B. Mehrfachentwicklungen, die in der Codebasis später inkonsistent werden und einzeln nachverfolgt werden müssen)
- Fehlendes Wissensmanagement (Beispiel: Projektmanager übernimmt neues Kundenprojekt und hat nicht das implizite Wissen zum Kunden und dessen Hintergrund.)

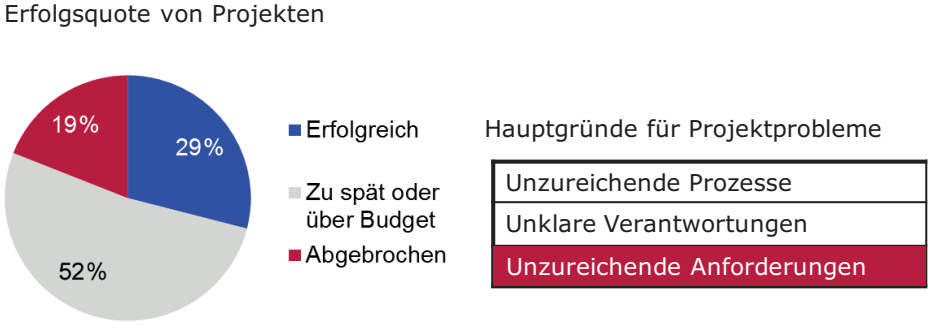


Abb. 1-3 *Unzureichendes Requirements Engineering reduziert den Projekterfolg.*

Ein wichtiger Grund dafür, dass Projekte ihre Ziele nicht erreichen, liegt in nicht sauber formulierten Zielen. Abbildung 1-3 zeigt im unteren Teil diesen Zusammenhang und unterstreicht schon aufgrund der Datenlage die immense – und wachsende – Bedeutung eines guten Requirements Engineering. Bei einem Großteil aller abgebrochenen Projekte war unzureichendes RE ein wesentlicher Grund für das Scheitern. Häufiger wurden nur noch »unzureichende Prozesse« und

»unklare Verantwortungen« genannt, aber das sind auch offensichtliche Allgemeinplätze, die man sich in jedem verkorksten Projekt gut vorstellen kann.

Technologische Herausforderungen lassen sich beherrschen. Schlechtes Management nicht. In Krisenzeiten, wie 2001 und 2008, geht die Erfolgsquote zurück. Dann werden vermeintlich unnötige Ausgaben, wie für Requirements Engineering, reduziert – mit durchschlagenden Konsequenzen.

Erfolg ist machbar. Hier die wichtigsten Erfolgsfaktoren aus der Industriepraxis:

- Ergebnisorientierte Vorgaben
- Zielorientierte Prozesse
- Kompetentes Produkt- und Projektmanagement
- Standardisierte und optimierte Infrastruktur
- Agile Entwicklung

Es geht nicht um eine spezielle Methode, sondern darum, dass diszipliniert gearbeitet wird. Wir wollen in diesem Buch darauf eingehen, welche Techniken des RE Sie einsetzen sollten, um mit Ihren Projekten und Produkten zu den Gewinnern zu gehören.

Auf was muss man beim RE achten? Aus unterschiedlichen Praxiserfahrungen lassen sich die wichtigsten Risiken im RE ableiten [Ebert2014a]. Die Risiken zu kennen, heißt, dass man sich darauf vorbereiten kann, um sie beim nächsten Mal zu vermeiden. Die folgende Liste hatte ich ursprünglich mit weiteren sehr erfahrenen RE-Praktikern erstellt [Lawrence2001]. Sie wurde hier nochmals aktualisiert.

Risiko 1: Fehlende Anforderungen

Häufig werden bestimmte Anforderungen übersehen, und es werden nur greifbare und nachvollziehbare Funktionen spezifiziert. Aber Anforderungen haben verschiedene Ausprägungen, wie wir gesehen haben. Neben den funktionalen Anforderungen gibt es Qualitätsanforderungen und Randbedingungen. Neben den Produktanforderungen gibt es auch Marktanforderungen und Komponentenanforderungen. Nur die Hinterfragung aller dieser Typen macht die Anforderungsdokumentation vollständig. Wichtig wird diese Vollständigkeit vor allem auch bei der Testspezifikation. Testfälle müssen alle diese Kategorien von Anforderungen abdecken.

Aus vertraglichen Gründen sollte man dem Kunden das liefern, was er will, und nicht das, was er braucht. Interpretieren Sie also nicht, was denn »passen könnte«, denn Sie kennen die Welt des Kunden und seine konkreten Bedürfnisse niemals so gut wie er selbst. Im Zweifelsfall zählt, was vertraglich abgestimmt wurde. Das ist vor allem dort wichtig, wo verschiedene Anspruchsträger auf Kundenseite mitwirken und wo wir Anforderungen priorisieren. Ein Lieferant sollte im Interesse einer nachhaltigen Kundenbeziehung im Vorfeld klären, was

der Kunde wirklich braucht, um dann vor Projektbeginn eine Abstimmung zu erreichen zwischen dem, was gebraucht wird, und dem, was gewünscht und damit vertraglich festgehalten wird. Eine wirksame Basis für erfolgreiches Kundenmanagement ist es, zuallererst den Business Case des Kunden zu verstehen. Dabei geht es darum, zu erkennen, was der Kunde – anders – machen will, wenn er erst einmal das gewünschte Produkt in den Händen hält. Den Business Case zu verstehen bedeutet, dass man als Produkt- oder Projektmanager erkennt, welche Funktionen oder Anforderungen an das Projekt den größten Nutzen bringen.

Risiko 2: Falsche Anforderungen

Anforderungen sind grundsätzlich unvollständig und fehlerhaft. Sie sind unvollständig, da wir das System nicht in jedem Detail vorab spezifizieren können – und dies auch niemand bezahlen wollte. Sie sind fehlerhaft, weil bei jeder Arbeit Fehler entstehen.

Wir Menschen machen pro zehn Zeilen Text ungefähr einen inhaltlichen Fehler, den wir nicht sofort entdecken. Die Hälfte dieser Fehler entdecken wir bei einer Schlussdurchsicht – sofern wir uns die Zeit dafür nehmen. Die andere Hälfte bleibt im Dokument und muss durch zusätzliche Techniken aufgedeckt und behoben werden. Das ist gerade bei Anforderungen kritisch, denn viele Fehler werden erst spät bei Test und Abnahme des Produkts entdeckt, und dann sind Korrekturen aufwendig.

Typische Fehler sind sowohl handwerklicher Natur, wie vage und ungenaue Beschreibungen, Widersprüche, Inkonsistenzen, Lücken, als auch inhaltlicher Natur, wie Denkfehler, falsche Priorisierung, falsche Abstraktionen und Vermischung von Was und Wie.

Fehler entstehen durch nicht beherrschte Komplexität. Kunden, der Vertrieb und das Marketing spezifizieren Funktionen, die niemand braucht. Entwickler versuchen, Anforderungen, die sie vermeintlich verstanden haben, mit Leben zu füllen, und entwickeln so Funktionen, die nie vereinbart wurden. Branchenübergreifend ist ungefähr die Hälfte der gelieferten Funktionalität ohne Wert und wird dementsprechend kaum oder nie verwendet (Abb. 1-4). Das gilt für ein Fahrzeug genauso wie für eine Office-Software. Jede unnötige Funktion bringt Abhängigkeiten von anderen Funktionen, Sonderfälle, Ausnahmesituationen – und damit zusätzliche Entwicklungs-, Korrektur- und Testaufwände, die sich über die Lebensdauer mit jedem Release vervielfachen.

Prüfen Sie Anforderungen mit Reviews (siehe dazu auch Kap. 6). Nutzen Sie dazu Checklisten und Szenarien wichtiger Abläufe. Spielen Sie vor allem die Szenarien durch, mit denen Ihr Kunde Geld verdient oder die dem Benutzer später Schwierigkeiten machen, wenn sie nicht optimal funktionieren. Prüfen Sie, ob Abhängigkeiten oder Fehlerszenarien übersehen wurden. Wer macht diese Reviews? Im Bestfall ein Tester. Tester haben einen Blick für Fehlermöglichkeiten

und entdecken in Reviews sehr viel mehr Fehler als Designer oder Projektmanager. Achten Sie auch darauf, dass die Anforderungen kundenseitig geprüft und formal freigegeben wurden.

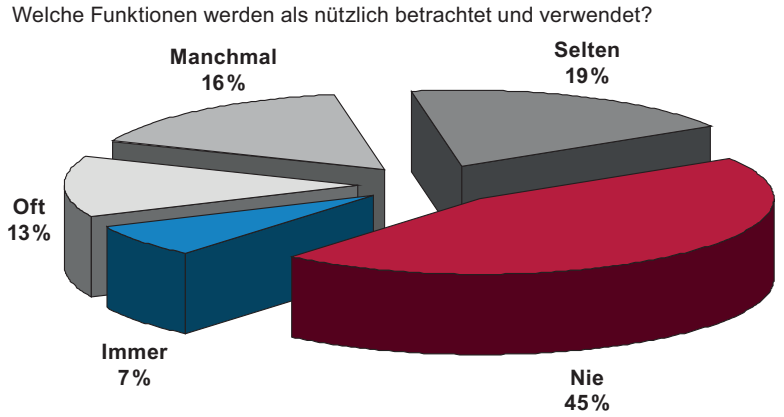


Abb. 1-4 *Falsche Anforderungen erhöhen die Kosten.*

Risiko 3: Sich ändernde Anforderungen

Kontrollieren Sie Änderungen. Anforderungen, deren Änderungen nicht beherrscht werden, führen zu Kosten- und Terminüberschreitungen und reduzieren die Qualität. Anforderungen ändern sich in beinahe jedem Projekt. Die Änderungsrate hängt von verschiedenen Faktoren ab, beispielsweise vom Neuigkeitsgrad von Produkt und Technologie. Häufig existiert eine gewisse Basis von Anforderungen, mit denen ein Projekt gestartet wird. Einige Punkte sind noch offen und klären sich im Laufe der Zeit. Auftraggeber haben allerdings oftmals gar kein großes Interesse daran, diese Punkte zu klären. Erstens ist es Zusatzaufwand und zweitens könnte der Auftraggeber bei der Abnahme davon profitieren, wenn nicht alles so läuft wie abgesprochen, denn das ist die Chance, komplett neue Anforderungen als Kompensation für diesen Projektfehler kostenlos zu erhalten.

Kontrollieren Sie die Änderungsrate im Projekt. Zu bestimmten Meilensteinen muss die Änderungsmenge reduziert werden, um die nächste Phase erfolgreich durchlaufen zu können. Üblich ist es, mit einem Puffer zu arbeiten, der sowohl Schätzungenauigkeiten als auch Anforderungsänderungen abfangen kann. Eine weitere Maßnahme ist die sogenannte Rückwärtsplanung von der Übergabe aus zurück ins Projekt, um zu erkennen, ab wann der kritische Pfad keine Parallelarbeit als Puffer mehr zulässt. Als Regel gilt, dass in der zweiten Projekthälfte nur noch sehr wichtige Änderungen ohne große Auswirkungen zugelassen werden. Eine dritte Maßnahme ist schließlich, Änderungen grundsätzlich nur zu diskutieren, wenn eine Analyse der Auswirkungen stattgefunden hat.

Andernfalls verschwenden beide Seiten ihre Zeit. In diesem »Spiel« wird gern gepokert, nur um zu sehen, wie sich der Lieferant verhält. Oftmals genügt der Verweis auf die Einflüsse im Projektplan, um zu zeigen, dass die vorgeschlagene Änderung Kosten und Projektdauer unzulässig erhöhen wird.

Klären Sie frühzeitig und verbindlich Verantwortungen und Rollen der Kunden bei der Projektarbeit. Unzureichende Einbeziehung der Benutzer schafft viele Risiken. Oft werden Anforderungen interpretiert, ohne den Kunden direkt einzubeziehen. Dann entwickelt sich das Projekt in zwei getrennte Richtungen, denn sowohl die Projektmitarbeiter als auch der Kunde lernen ständig dazu. Da der Kunde allerdings nicht weiß, wie er damit umgehen soll, wartet er ab. So wachsen die Divergenzen an, statt aufgelöst zu werden. Daraus hat sich das agile Prinzip »Kunde an Bord« entwickelt. Das kann jedoch auch schwierig werden, wenn kundenseitig nicht abgestimmte Anforderungen als Versuchsballons lanciert werden. Und die haben die Tendenz zu platzen. Viele Kunden haben sich nie zu Verantwortungen Gedanken gemacht. Abgestimmte Inhalte werden von uninformierten Kundenvertretern ständig neu hinterfragt und geändert.

Projekte scheitern wegen unzureichender Anforderungen. Abbildung 1-5 zeigt diesen Effekt, wie wir ihn bei einem Kunden beobachtet haben. Unzureichende Anforderungsqualität brachte dort nicht nur das Projekt in Schieflage, sondern reduzierte auch die Mitarbeitermotivation dramatisch, denn viele wussten nicht mehr, wie sie die sich ständig ändernden Vorgaben meistern sollten.

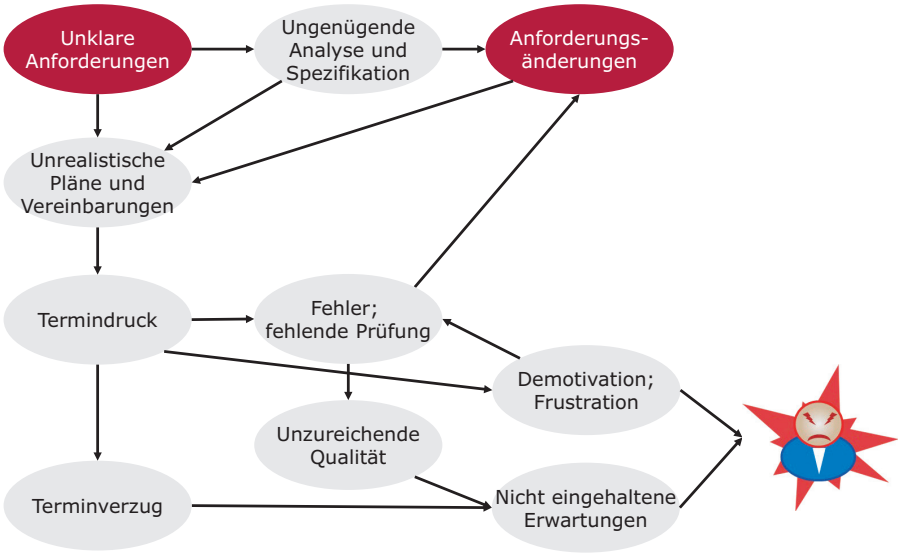


Abb. 1-5 *Unzureichende Anforderungen und die Konsequenzen*

Diese drei Risiken sind nicht softwarespezifisch und unterstreichen die breite Anwendbarkeit des Requirements Engineering – egal ob IT, Medizin oder Maschinenbau. Branchenübergreifend ähneln sich die Herausforderungen: Anforderungen fehlen, sind falsch und ändern sich während des Projekts.

Beispiel:
Die Qualität der Anforderungen ist ein wesentlicher Erfolgsfaktor beim Projekterfolg. Eines der ersten kommerziellen Düsenflugzeuge, die Comet 1, hatte keine Anforderungen zu dynamischen Spannungen der Fenster – und stürzte sehr häufig ab. Die Tacoma-Narrows-Brücke hatte unzureichende Anforderungen und Lösungsmodelle bei der Berücksichtigung der Windlast – und stürzte ein. Beim Therac-25-Bestrahlungsgerät war die Benutzerschnittstelle fehlerhaft spezifiziert – und verursachte mehrere Todesfälle. Beim Bau der Ariane-5-Rakete wurden Anforderungen außerhalb des erlaubten Kontexts von Ariane 4 wiederverwendet, und die Rakete stürzte auf dem Jungfernflug ab. Die Liste könnte beliebig verlängert werden. Sie selbst haben bestimmt eigene Beispiele unzureichender Anforderungen. Achten Sie auf hinreichend gute Anforderungen!

1.3 Wirtschaftlicher Nutzen und Return on Investment (ROI)

Die systematische Umsetzung von RE erfordert Aufwand in der Entwicklung und an den Schnittstellen im Produktmanagement, Produktmarketing und Vertrieb. Häufig wird dieser Aufwand als zu hoch und zu zeitraubend angesehen. Aus unserer Beratungspraxis kennen wir das Dilemma: Verbesserungen in Methodik, Ausbildung und Werkzeugen werden nicht angegangen, da der Anfangsaufwand, um diese Verbesserungen anzustoßen, als zu hoch betrachtet wird.

Systematisches RE hat einen klaren Nutzen und ROI, den wir hier zeigen. Einige dieser Faktoren, wie Termintreue oder weniger Nacharbeiten, schaffen einen unmittelbar greifbaren Nutzen. Andere, wie beispielsweise die Kundenzufriedenheit, werden in Form nachhaltig guter Kundenbeziehungen und weiterer Projekte greifbar Die folgenden Daten stammen aus der Vector Benchmark Datenbank sowie verschiedenen Metastudien zum Effekt von RE [Hussain2016, Standish2018, Gartner2018, Ebert2007, Standish2003, Terzakis2013]:

- **Wertorientierung**
  50 % aller Funktionen werden nie verwendet. Diese Zahl gilt branchenübergreifend als Richtwert. Wenn einige dieser sowieso eher unnötigen Funktionen frühzeitig erkannt und nicht entwickelt werden, reduziert das die Komplexität und Kosten. RE richtet den Fokus auf das Wesentliche, sorgt für eine klare Positionierung des Produkts und seiner Alleinstellungsmerkmale und schafft damit auch die Gewissheit, dass alle Anspruchsträger am gleichen Strang ziehen.
- **Qualität**
  80 % der Fehler im Test und 43 % der Feldfehler resultieren aus unzureichendem RE. Diese Fehlerkosten werden durch ein besseres RE direkt reduziert – der Umfang hängt natürlich davon ab, wie Sie die Schwerpunkte setzen.

- **Kostenreduzierung**
  Typischerweise werden 2–5 % des Aufwands in das RE investiert. Eine Verdoppelung reduziert die Lebenszykluskosten um typischerweise 20–40 %. Die Gründe dafür sind frühe Fehlerentdeckung, frühe Korrektur unzureichender Anforderungen, Fokus auf Erweiterbarkeit etc. Werden die Anforderungen so umgesetzt, wie sie vom Kunden oder Benutzer beschrieben werden, reduziert das die Nacharbeiten, die im Schnitt 45 % des Projektaufwands ausmachen.
- **Produktivitätsverbesserung**
  Typischerweise werden 30–50 % des Entwicklungsaufwands für die Fehlerbehebung und nicht entwicklungsbezogene Aktivitäten eingesetzt. Die Hälfte der Fehler ist das direkte Ergebnis unzureichender Anforderungen und unkontrollierter Änderungen. Im Systemtest sind es 80 % der Fehler, die aus unvollständigen (31 %) oder falschen (49 %) Anforderungen resultieren. Die Wiederverwendung von Anforderungen und davon abgeleiteten Arbeitsergebnissen (z. B. Mechanismen zur Systemsicherheit) schafft weitere Produktivitätsgewinne, insbesondere bei Produktvarianten.
- **Kürzere Durchlaufzeiten**
  Verbesserte Projektplanung und Ressourceneinteilung, weniger Verzögerungen vor dem Projektstart, eine schnellere Anlaufphase sowie Termintreue aufgrund von bekannten Anforderungen und klaren Verantwortungen im Projektteam und im Vertrieb reduzieren Wartezeiten sowie Terminverzug. Mehr Aufwand für die Entwicklung und die konsequente Umsetzung und das konsequente Testen der Anforderungen schaffen eine Verbesserung der Termintreue und reduzieren Verzögerungen auf unter 20 %.

Insgesamt zeigen unsere Erfahrungen, dass eine Verdoppelung des Aufwands für RE hin zu 10 % des Projektaufwands in den Bereichen Methodik, Prozesse, Training und Werkzeugunterstützung einen konkret realisierbaren Projektnutzen von über 20 % schafft. Das ist ein ROI von mehr als 4, und bei diesem Wert sind nur die direkt messbaren Vorteile berücksichtigt.

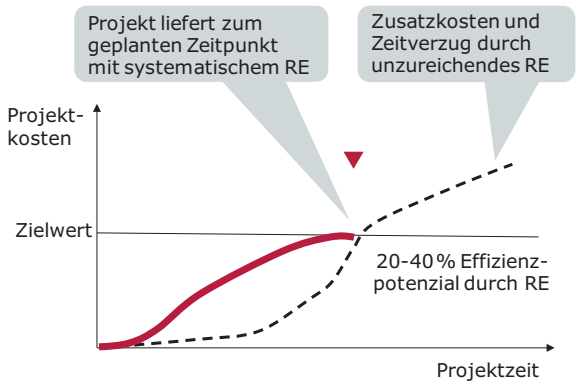


Abb. 1-6 *Effizienzpotenzial mit systematischem Requirements Engineering*

Die umfassendste Untersuchung zum Nutzen von RE stammt von Alcatel (heute Nokia). Über mehrere Jahre hinweg wurden in einer longitudinalen Feldstudie unterschiedliche Projektdaten systematisch erfasst und analysiert (Abb. 1-7) [Ebert2006].

Gute Termintreue wird nur dann erreicht, wenn die vier folgenden Techniken gleichzeitig umgesetzt werden. Wurde nur einer dieser Faktoren vernachlässigt, führte das sofort zu Terminverzug:

- Ein verantwortliches Kernteam, bestehend aus Produktmanagement, Marketing, Entwicklung und Produktion, das das gesamte Projekt (oder das Produktrelease) steuert
- Konsequente Nutzung eines definierten Lebenszyklus mit Meilensteinen, Checklisten etc.
- Transparenz aller Projektvereinbarungen (z.B. Anforderungen) im Intranet
- Gemeinsame Anforderungsanalyse durch das Kernteam mit Produktmanagement, Marketing, Entwicklung und Produktion

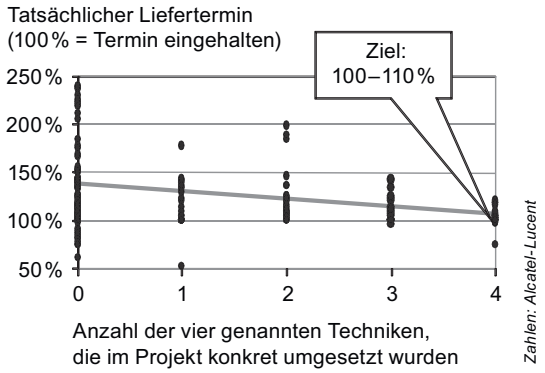


Abb. 1-7 *Der gleichzeitige Einsatz von vier Requirements-Techniken verbessert den Projekterfolg.*

Eine weitere umfassende Studie zum Nutzen von RE in Entwicklungsprojekten kommt von der NASA (Abb. 1-8) [Hooks2001]. Im Unterschied zu den beiden vorigen Studien wurde hier die Kosteneinhaltung in Abhängigkeit vom Aufwand für RE untersucht. Projekte mit 5 % Aufwand für RE führen zu einer Kostenüberschreitung von 80 % bis knapp 200 %. Wird dieser Aufwand in Richtung 8–14 % verdoppelt, liegt die Kostenüberschreitung bei unter 60 %. Offensichtlich sind IT-Projekte sehr anfällig für eine unzureichende Anforderungsanalyse und -spezifikation, denn die Anforderungen werden sich im Projektverlauf zunehmend ändern und zu beträchtlichen Zusatzaufwänden führen. Auch hier gilt, dass die absoluten Zahlen für Überschreitungen von Kosten natürlich durch viele Faktoren bestimmt werden. Aber ein unzureichendes RE hat einen starken Anteil an überbordenden Kosten. Intel hat in einer aktuellen Langzeitstudie gezeigt, dass systematisches RE die Zahl der Fehler im Produkt um 30–50 % senkt [Terzakis2013].

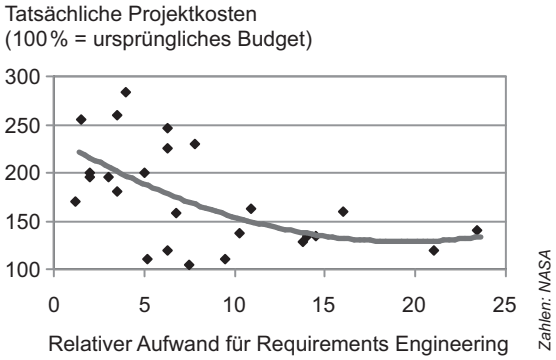


Abb. 1-8 *Kosteneinhaltung in Abhängigkeit vom Aufwand für das Requirements Engineering*

Viele Projekte geraten in Schwierigkeiten, da anfangs zu wenig Energie investiert wird und später in der »heißen« Phase die Zeit fehlt, um nachzudenken. Aufwendige Nacharbeit ist nun nötig, um Fehler und Entscheidungen zu korrigieren, die in der Startphase des Projekts viel leichter aufzulösen gewesen wären. Andere müssen Kapazität abgeben, um die jetzt dringend nötige Überlast abzudecken. Dieser Teufelskreis lässt sich nur durch frühzeitige Planung und kontinuierliches Requirements Engineering durchbrechen. Das wird als »Frontloading« bezeichnet und bedeutet, dass Projektaufgaben frühestmöglich erledigt werden, um Nacharbeiten und damit Terminverzug zu vermeiden. Abbildung 1-9 zeigt in der oberen Hälfte ein typisches Projekt. Anfangs wird zu wenig Energie in Requirements Engineering und Planung investiert.

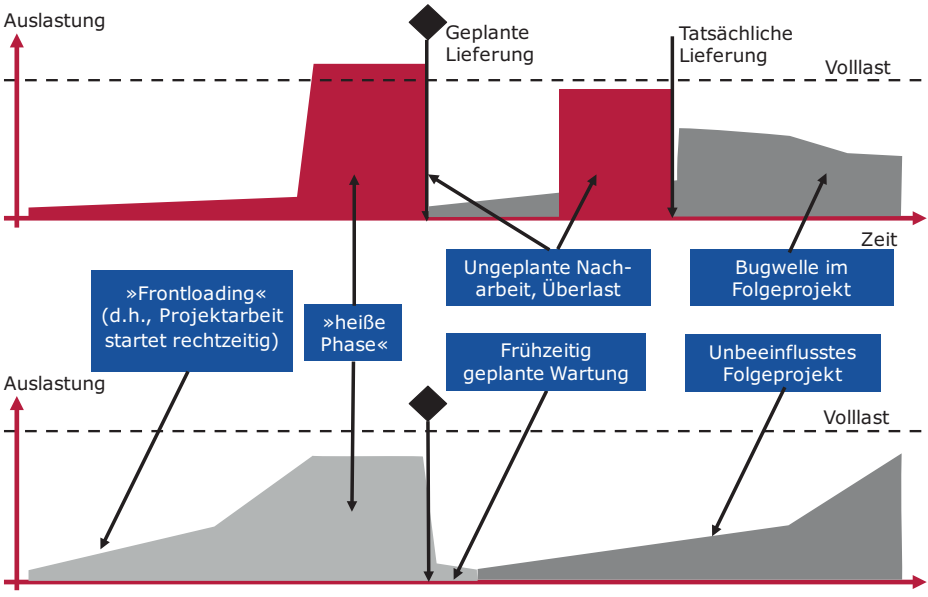


Abb. 1-9 *Frühzeitige Lastbalance durch Frontloading*

Daraus resultieren Nacharbeiten und die Kannibalisierung nachfolgender Projekte. Zudem führt die kontinuierliche Überlast in der »heißen Phase« zu Demotivation der Mitarbeiter. Die untere Hälfte in der Abbildung zeigt den Effekt des »Frontloading« mit systematischem Requirements Engineering. Anforderungen werden frühzeitig ermittelt und das Projekt wird realistisch geplant. Aufwände werden über einen größeren Zeitraum verteilt und erlauben ein agiles Arbeiten mit inkrementellen Lieferungen. Auswirkungen auf Folgeprojekte werden vermieden.

1.4 Wie Sie von diesem Buch profitieren

Dies ist ein Buch für Einsteiger und Profis. Nach der Lektüre

- haben Sie einen Überblick über Theorie und Praxis des Requirements Engineering;
- verstehen Sie, wie moderne Werkzeuge und Notationen Sie beim Requirements Engineering praktisch unterstützen können;
- können Sie die wichtigen Elemente des Requirements Engineering in Ihren Projektalltag übertragen und dort produktiv einsetzen.

Abbildung 1-10 zeigt die Struktur des Buches. Das Thema RE wird zunächst anhand verschiedener Übersichtskapitel eingeführt. Kapitel 2 führt kurz und knapp in das Requirements Engineering ein und zeigt den Nutzen in der Praxis, aber auch die Risiken, wenn es nicht ausreichend gelebt wird. Die Kapitel 3 bis 8 beleuchten die einzelnen Aktivitäten innerhalb des RE systematisch. Kapitel 3 beschreibt, wie Anforderungen ermittelt werden. Der Nutzen und Wert aus der Sicht desjenigen, der bezahlt, steht im Vordergrund, denn das ist die wesentliche Basis für jedes erfolgreiche Projekt. Kapitel 4 betrachtet die Dokumentation, also das Beschreiben von Anforderungen. Dabei geht es um die Verbesserung der Anforderungsqualität und verschiedene formale Arten der Beschreibung, die hinsichtlich ihrer Praktikabilität und Schwierigkeit in der Umsetzung diskutiert werden.

Die relevanten Analyse- und Modellierungstechniken werden in Kapitel 5 eingeführt. Ein durchgängiges Beispiel erlaubt eine gute Vergleichbarkeit der Techniken und ihres Nutzens. Kapitel 6 zeigt, wie qualitativ gute Anforderungen erreicht werden. Wir zeigen hier Techniken zu Reviews, Prüfungen und konkrete Checklisten, um die Anforderungsqualität zu verbessern. Kapitel 7 unterstreicht die Abstimmung von Anforderungen mit den verschiedenen Anspruchsträgern. Das Kapitel betrachtet auch rechtliche Zusammenhänge, beispielsweise Gewährleistungs- und Haftungsfragen. Schließlich beschreibt Kapitel 8 Techniken, um Anforderungen im Projekt zu kontrollieren und zu verwalten.

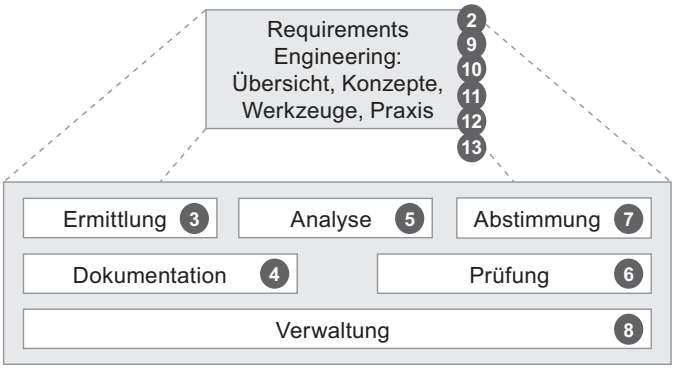


Abb. 1-10 *Das Buch im Kontext des Requirements Engineering (Schwarze Kreise sind Kapitelnummern.)*

Kapitel 9 ist komplett neu und beschreibt agiles Requirements Engineering. Wir gehen auf Methoden wie Design Thinking, Planning Poker und testorientiertes RE ein. Werkzeuge werden in Kapitel 10 charakterisiert und bewertet. Wir beschreiben einige populäre RE-Werkzeuge ganz praxisnah in unterschiedlichen Szenarien. Die Praxis des RE wird in Kapitel 11 dargestellt. Damit sehen Sie den praktischen Nutzen und die Abhängigkeiten der einzelnen Schritte im RE. In Kapitel 12 möchte ich Ihnen zeigen, wie Sie Ihre eigene Kompetenz ausbauen können. Das umfasst fachliche Kompetenzen und Soft Skills. Das abschließende Kapitel 13 fasst den Stand der Technik zusammen und beleuchtet die wichtigsten Trends des RE in den nächsten Jahren. Hier werden auch die empirischen »Gesetzmäßigkeiten« des RE nochmals an einer Stelle zusammengefasst.

Abgerundet wird das Buch durch eine Zusammenstellung von Internetressourcen, also den wichtigsten URLs zum Thema RE. Alle Begriffe, die im Buch definiert werden oder auf deren Definitionen zurückgegriffen wird, sind im Glossar am Ende des Buches nochmals zusammengefasst. Eine Zusammenstellung der Literaturquellen sowie ein Index beschließen das Buch.

Jedes der Kapitel besitzt am Ende einige Checklisten sowie konkrete Fragen an die Praxis. Damit können Sie das gerade Gelesene in Ihrem eigenen Kontext reflektieren und leichter umsetzen. Es handelt sich hier nicht um die Art von Verständnisfragen, die Sie aus Lehrbüchern kennen, sondern um Fragen, die Ihren eigenen Horizont erweitern, um das gerade konsumierte Wissen verdauen und umsetzen zu können. Damit erhalten Sie unmittelbar nutzbare Impulse für Ihre eigenen Projekte. Praxisbeispiele und Tipps dazu sind direkt im Text eingebettet und farblich hervorgehoben:

Beispiel:
Konkrete Beispiele aus meiner Arbeit in ganz verschiedenen Unternehmen und Branchen zeigen, wie die Techniken des Requirements Engineering in die Praxis umgesetzt werden. Die meisten Beispiele sind zum Nachmachen gedacht. Manche dienen zur Abschreckung und sind dann auch so charakterisiert. Das Buch hat eine eigene Webseite *www.requirements-excellence.com*, die auch auf Vorlagen, White Papers und Fallstudien verweist.

Wenn Sie als *Entwickler* oder *Ingenieur* in einem Unternehmen arbeiten, gibt Ihnen dieses Buch einen guten Überblick zu den relevanten Fragestellungen und Lösungen des Requirements Engineering. Praktische Tipps am Ende jedes Kapitels helfen Ihnen dabei, essenzielle Vorgehensweisen schnell und »leicht verdaulich« zu extrahieren. Die Techniken sind praxiserprobt und leicht umsetzbar.

Selbstständige und *Freelancer* lernen praxisnah die wichtigsten Grundlagen, die gerade in kleinen Projekten eine große – oft überlebensnotwendige – Rolle spielen. Beispielsweise braucht auch ein Kleinstprojekt ein funktionierendes Änderungsmanagement, um zu verfolgen, welche Anforderungen im Moment akzeptiert sind und welche sich noch in der Pipeline befinden.

Als *Projektmanager* finden Sie eine Menge wertvoller Tipps und lernen, wie Requirements Engineering konkret implementiert wird und wie Risiken und typische Schwierigkeiten im Requirements Engineering gehandhabt werden können.

Sind Sie in der *Systementwicklung* tätig, ohne überhaupt Software zu betrachten? Das Buch hilft überall dort, wo Anforderungen systematisch ermittelt und umgesetzt werden sollen. Die hier vorgestellten Methoden und Prozesse sind nicht spezifisch für Software und IT, sondern universell für Produkte und Dienstleistungen einsetzbar.

Wenn Sie *Requirements-Ingenieur, Analyst, Systemanalyst* oder *Produktmanager* sind, zeigt Ihnen das Buch, wie Sie erfolgreich eine Brücke bauen zwischen den sehr verschiedenen Sichtweisen heterogener Anspruchsträger innerhalb und außerhalb des Projekts. Zielkonflikte tragen zu besseren Lösungen bei und sollten nicht sofort zu Konfrontationen führen. Wir unterstützen Sie mit Methoden und Tipps, um Ziele und Perspektiven herauszuarbeiten und auf Grundlage dieser Erkenntnisse die richtige Balance von Anforderungen umzusetzen.

Agile Coaches und *Berater* lernen aus dem Buch sowohl methodisch als auch in Praxisbeispielen und Fallstudien, wie agile Methoden im Requirements Engineering erfolgreich umgesetzt werden.

Sofern Sie als *Qualitätsverantwortlicher* oder im Bereich der Prozessverbesserung arbeiten, hilft Ihnen das Buch, Standards und Modelle (z.B. ISO 9001, CMMI, SPICE) praktisch einzusetzen. Ich habe selbst viele Jahre damit verbracht, unterschiedliche Unternehmen und Produktlinien im Anforderungs- und Produktmanagement zu verbessern.

Neulinge im Thema Requirements Engineering und *Studierende* werden von den ersten Kapiteln stark profitieren, denn dort verknüpfen wir das Requirements Engineering mit der Softwaretechnik. Fragen an die Praxis am Ende eines jeden Kapitels helfen dabei, sich selbst zu prüfen und zu erkennen, ob die relevanten Themen auch im Kontext verstanden wurden.

Beiden Gruppen, den Einsteigern und den Praktikern, gerecht zu werden, gelingt durch eine klare Struktur, die in jedem Kapitel wichtige Themen zusammenfasst. So wissen Einsteiger, was gemeint ist, und können sich orientieren, während Profis sofort in die Tiefe gehen und das finden können, was ihre derzeitige Situation gerade verlangt.

1.5 Selbsttest

Der folgende Test hilft Ihnen, Risiken im Requirements Engineering zu erkennen, zu bewerten und zu konkretisieren. Die Messlatte ist Ihr Erfolg mit Projekten, Produkten und Kunden. Das Ergebnis ist ein detailliertes Stärken- und Schwächenprofil, das anzeigt, auf was Sie sich im Requirements Engineering konzentrieren sollten. Damit haben Sie einen Startpunkt für eigene Verbesserungen. Sie müssen entscheiden, was für Sie wichtig ist, und dann mit den Änderungen beginnen.

Führen Sie den Test alleine und für Ihre eigene spezielle derzeitige Situation durch. Nehmen Sie relevante aktuelle Projekte, um eine repräsentative Antwort zu erhalten. Beantworten Sie alle Fragen aus der Perspektive des Projektmanagers. Bitte geben Sie auf alle Fragen die aktuelle Situation an, keinen Sollzustand oder Wunschdenken. Falls das Projekt gerade erst aufgesetzt wurde, beantworten Sie die Fragen basierend auf Ihrer Erfahrung und Intuition.

Der Test besteht aus verschiedenen Fragen, die Sie anhand einer Punkteskala bewerten:

- 3 Punkte: Ja, vollständig, da bin ich mir sicher.
- 2 Punkte: Davon gehe ich doch aus, und ich habe es bereits gesehen.
- 1 Punkt: Fragwürdig, das kann ich mir nicht richtig vorstellen.
- 0 Punkte: Nein, gar nicht!

#	Selbsttest Requirements Engineering	Antwort (nein = 0 ... ja = 3)
1	Hat das Projekt ein klares, eindeutiges und messbares Ziel?	
2	Ist sich das Projektteam einig, dass das Ziel realistisch ist?	
3	Gibt es für das Projekt einen schriftlich vereinbarten Business Case, der die erwarteten Kosten und Nutzen definiert?	
4	Existiert eine konfigurierte Anforderungsliste für das Projekt, die die zum aktuellen Zeitpunkt relevanten funktionalen Anforderungen und Qualitätsanforderungen mit testbaren Freigabekriterien umfasst?	

→

#	Selbsttest Requirements Engineering	Antwort (nein = 0 ... ja = 3)
5	Wurden die Anforderungen systematisch durch unabhängige Reviews geprüft und freigegeben?	
6	Wurden die Anforderungen mit den betroffenen externen und internen Anspruchsträgern explizit vereinbart?	
7	Wurden die Anforderungen methodisch abgeschätzt und wurde die Schätzung als Basis für den Projektplan übernommen?	
8	Sind die Anforderungen priorisiert und in einem inkrementellen Projektplan entsprechend der Priorität berücksichtigt?	
9	Existiert ein Projektplan, der auf den vereinbarten Anforderungen aufsetzt und den Fortschritt anhand des erreichten Werts kontrolliert?	
10	Werden Status und Planung während des Projekts regelmäßig kommuniziert und bei Änderungen aktualisiert?	
11	Sind die Anforderungen werkzeugunterstützt mit Status und Attributen dokumentiert?	
12	Sind die Marktanforderungen zu den Produkt- und Komponentenanforderungen sowie zu Projektplan, Liefergegenständen und Testfällen verfolgbar?	
13	Existiert ein formales Änderungsmanagement innerhalb des Projekts mit klaren Verantwortungen für die Handhabung von Änderungen?	
14	Ist ein hinreichend erfahrener Projektmanager mit voller Projektverantwortung für Budget, Inhalte und Lieferung benannt?	
15	Ist ein Sponsor für das Projekt benannt, der diese Aufgabe auch effektiv durchführt?	
16	Werden die Unsicherheiten und Risiken im Projekt abgestimmt und abgeschwächt?	
17	Kennt das Projektteam den Kunden und die Umgebung, in der das Produkt eingesetzt wird?	
18	Sind das Anforderungsmanagement mit Lieferanten und die Zusammenarbeit über die Schnittstellen und Unternehmensgrenzen hinweg klar geregelt?	
19	Haben Sie vergleichbare Projekte früher bereits erfolgreich umgesetzt?	
20	Haben Sie und die Projektbeteiligten ein gutes Gefühl, dass das Projekt erfolgreich abgeschlossen wird und seine Vorgaben einhält?	
	Zwischensumme	
	Multiplikationsfaktor: 1,5, falls die Anforderungen stabil sind und Sie an einem Standort ohne Lieferanten arbeiten; 1,25, falls sich die Anforderungen um maximal 5% pro Monat ändern. Falls die (zu erwartende) Änderungsrate größer ist oder Lieferanten im Projekt mitarbeiten, notieren Sie 1,0.	
	Summe (= Zwischensumme x Multiplikationsfaktor)	

Zählen Sie nun die Punkte zusammen. Falls Sie <30 Punkte haben, ist Ihre wesentliche Herausforderung, ein Projekt abschließen zu können und am Markt wettbewerbsfähig zu bleiben. Zwischen 30 und 50 Punkten liegen durchschnittliche Projekte mit durchschnittlichen Ergebnissen. Das heißt, Sie werden zu spät und über Budget liefern. Sie haben einiges Potenzial für Verbesserungen. Falls Sie über 50 Punkte haben, haben Sie das Requirements Engineering im Griff – oder vielleicht doch Wunsch und Realität etwas vermischt. Sie sollten nun auf Optimierung schauen, beispielsweise um Kosten zu reduzieren, Qualität zu verbessern und Komplexität zu beherrschen. Der Test gibt Ihnen Ansatzpunkte und Impulse, um sofort zielorientiert durchzustarten.

1.6 Ein Blick über den Tellerrand

Dieses Buch unterstützt die systematische und zielorientierte Umsetzung von RE in der Praxis. Es ist kein theoretisches Grundlagenwerk. Wir empfehlen daher zur Vertiefung einige weitere Bücher.

Die Softwaretechnik, deren Vorgehensweisen und Managementprinzipien sind im Buch von Helmut Balzert beschrieben [Balzert2008]. Das Buch stellt die relevanten Managementtechniken und Vorgehensmodelle vor und beschreibt, wie verschiedene Modelle in der Praxis eingesetzt werden. Verschiedene Schwerpunktthemen, wie strategisches Management in der IT und globale Softwareentwicklung, verdeutlichen die heutige Ausrichtung der Softwaretechnik.

Die theoretischen Grundlagen des RE sind im umfangreichen Werk von Klaus Pohl beschrieben [Pohl2008]. Viele Themen, beispielsweise Notationen und Methoden, die wir hier aus Platzgründen und aufgrund der umsetzungsorientierten Ausrichtung dieses Buches nicht vertiefen können, werden dort auf eine theoretische Basis gestellt. Für die Praxis des Requirements Engineering mit weiteren Beispielen und Fallstudien empfehle ich die Bücher von Suzanne Robertson und Karl Wiegers [Robertson2012, Wiegers2013]. Eine Vertiefung der Notationen UML und SysML ist in [Weilkiens2014] zu finden. Eine gute Ergänzung zur Erreichung von Win-win-Ergebnissen im RE ist das hervorragende Buch von Al Davis [Davis2005]. Er beschreibt eindrucksvoll, wie man Projekte und Produkte so anpackt, dass »hinreichend gute« Ergebnisse erzielt werden, ohne viel Overhead zu erzeugen.

An verschiedenen Stellen des Buches unterstreichen wir, dass RE als Disziplin so spannend und in der praktischen Software- und Systementwicklung so ungemein wichtig ist, weil man mit Menschen arbeitet und gemeinsam zielorientiert die Grundlagen für immer wieder neue Produkte und Geschäftserfolge schafft. Dazu braucht es sehr viel mehr als die Beherrschung von Notationen und Werkzeugen. Es geht um die Fähigkeit, mit anderen Menschen gemeinsam erfolgreich zu sein. Dazu gehören »Soft Skills für Softwareentwickler«, die im Buch von Vigenschow und Kollegen [Vigenschow2011] dargestellt sind. Ebenso empfehlen

möchte ich das Buch von Elisabeth Schick zum »Ich-Faktor«. Es ist flott und einprägsam geschrieben und zeigt, wie man auf andere wirkt und wie man diese Wirkung selbst verbessern kann [Schick2010]. Nutzen Sie das Buch, um sich selbst ganz gezielt weiterzuentwickeln.

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.

Anforderungen können mehrdeutig sein, sie können überspezifiziert sein, sie können unvollständig sein, sie können kontextspezifisch sein, sie können sich widersprechen, sie können unmöglich oder 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).

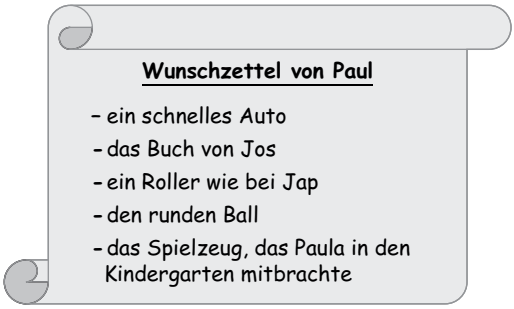


Abb. 2-1 *Anforderungen sind der »Wunschzettel« des Kunden.*

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 Features 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, dass man mit wertorientiertem Requirements Engineering aus Träumen 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 noch etwas weglassen? Ein Produkt war für ihn erst gut genug, wenn jede Funktion 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.

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 beziehungsweise 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 Bedürfnisse 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 Nachverfolgung 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 Anspruchsträger 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.

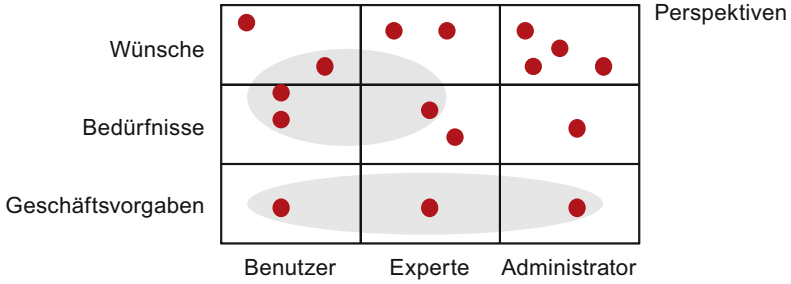


Abb. 2-3 *Aus verschiedenen Perspektiven belastbare Geschäftsanforderungen entwickeln*

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. ein Computerspiel), eine Rechnerumgebung mit verschiedenen Benutzern (z. B. ein Ressourcenplanungssystem in einem Unternehmen), eine interaktive Online-Umgebung mit sehr vielen unbekannten Benutzern (z. B. ein Online-Buchungssystem), ein gemischtes Hardware-Software-System (z. B. eine 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 Liefe-

ranten, 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.

Im Kontext von iHome sind Komponentenanforderungen beispielsweise die Anforderungen an das Steuergerät, an das Betriebssystem oder an die Rauchmelder. Solche Komponenten werden in der Regel nicht vom gleichen Hersteller wie das Produkt entwickelt, sondern zugekauft. Daher müssen diese Anforderungen 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 einem Auftraggeber (in B2B) beschrieben. Vertikal ist die Lösung als spätere Realisierung dargestellt. Verschie-

dene 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 Kap. 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 »Software-Krise« bekannt. Aufgelöst wurde diese Krise erst durch modernes Requirements Engineering, das es mit seinen Perspektiven und der Nachverfolgbarkeit 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.

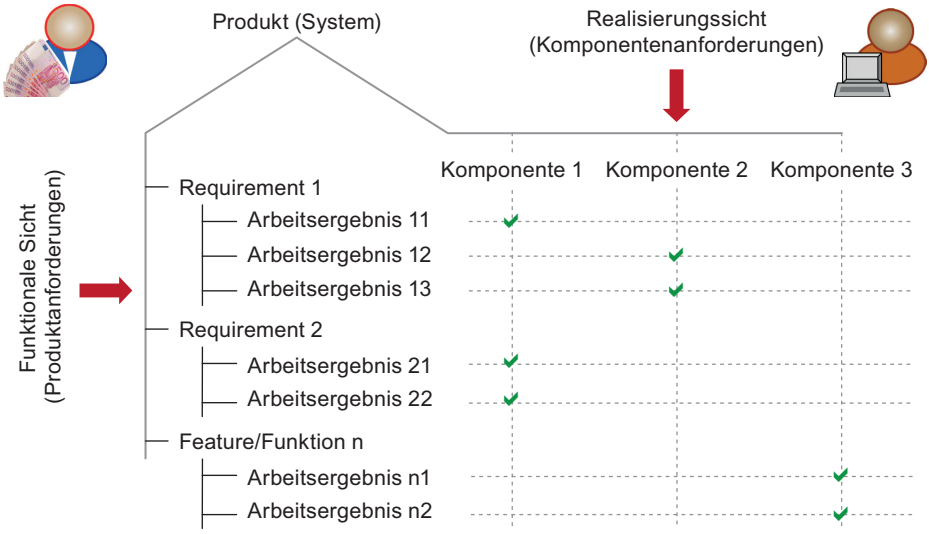


Abb. 2-4 *Komponentenanforderungen werden aus Produktanforderungen abgeleitet.*

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 Service-Orientierung 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: Service-Orientierung

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 Mobilitäts-Service, 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 diensteorientierten 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

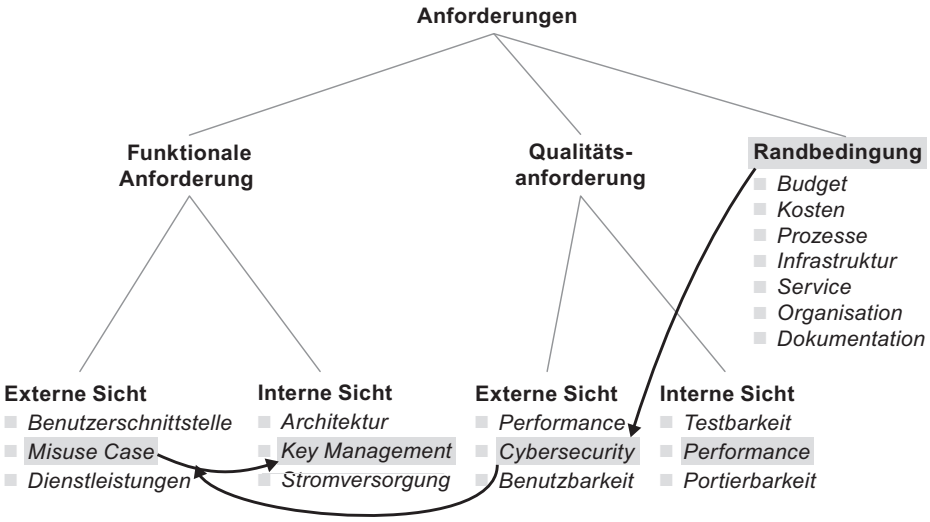


Abb. 2-6 *Unterschiedliche Typen von Produktanforderungen*

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.

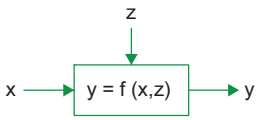
Beispiele für solche funktionalen Anforderungen sind Funktionen, Ablaufbeschreibungen und Szenarien, die angeben, wie ein System auf bestimmte Eingangsgrößen oder Eingaben zu reagieren hat (Abb. 2-7). Dazu gehören auch Daten- oder Schnittstellenanforderungen, die nötig sind, um das zu entwickelnde System 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 Geschäfts-

vorfälle 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: Abbildung von Eingangsgröße unter gegebenen Randbedingungen auf Ausgangsgröße z.B. berechnen, übertragen, wandeln
- Quantifizierung z.B. 10 V, 200 Nm/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 Informationssicherheit.[2]

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

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

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 Nachverfolgung sollten Qualitätsanforderungen in funktionale Anforderungen »übersetzt« werden. Das erleichtert die Nachverfolgbarkeit 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.

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, Unterauftragsnehmern 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 Startups, 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?

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 ist in Abbildung 2-8 dargestellt. Wir haben bewusst keine zeitlichen oder kausalen Abhängigkeiten dargestellt, denn das ist, wie Sie später sehen werden, gar nicht so einfach.

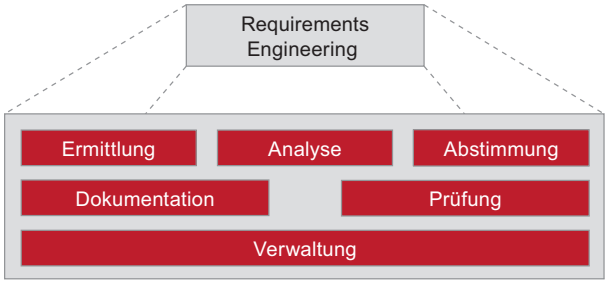


Abb. 2-8 *Aktivitäten des Requirements Engineering*

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 [INCOSE2015]. 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).

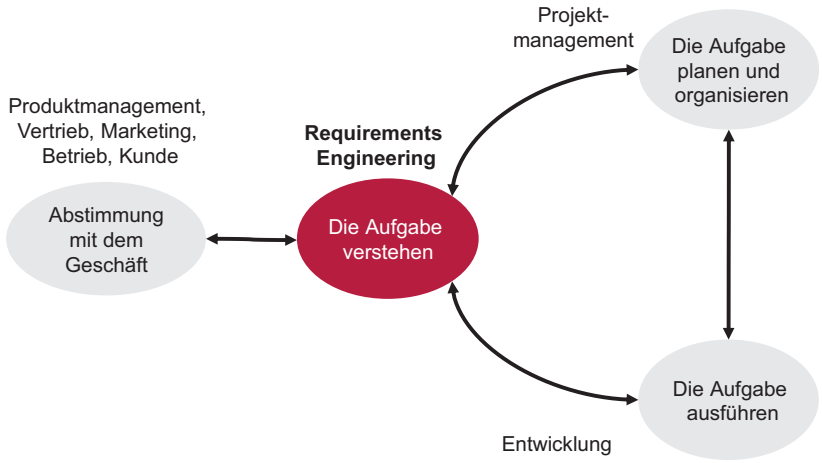


Abb. 2-9 *Requirements Engineering im Kontext der Produktentwicklung*

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.

Abb. 2-10 *Requirements Engineering und Wertschöpfung im Lebenszyklus*

RE begleitet ein Produkt durch seinen gesamten Lebenszyklus. RE wird sowohl bei neuen Produkten als auch bei Änderungen bestehender Produkte angewandt. RE wird vor dem Projektstart und während der gesamten Laufzeit eines Projekts eingesetzt.

Requirements Engineering muss als Geschäftsprozess bereichsübergreifend gelebt werden. Genauso wenig, wie es einen einheitlichen Entwicklungsprozess gibt, kann es einen standardisierten RE-Prozess geben. Produkte und Projekte sind zu unterschiedlich, um ein solches Ziel realistisch zu verfolgen. Wenn Sie beispielsweise eine Produktlinie haben, die sowohl eine Plattform als auch Varianten davon für Kunden produziert, dann ist es sicherlich sinnvoll, die Variantenproduktion durch einen definierten RE-Prozess zu steuern. Ein solcher Prozess

könnte dann Zugriff auf alle existierenden Funktionen der ganzen Produktlinie bieten, sodass Entscheidungen zu Durchführbarkeit und Aufwand vereinfacht und reproduzierbar werden.

Beispiel:

In einem aktuellen Fall hatte ein Zulieferer einige späte Änderungen erhalten, die allesamt angeblich ganz kritisch waren. Die Zeit für eine Einflussanalyse fehlte, und in der letzten Integrationsstufe gab es immense Verzögerungen.

Viele Hersteller, OEMs und Service-Provider bitten Vector daher um Unterstützung, um ihre Lieferantenprozesse zu verbessern. Unser Fazit: RE ist so gut wie immer die schwächste Stelle im Prozess – und diese Stelle befindet sich an der Schnittstelle zwischen Auftraggeber und Kunde. Fehler werden in der Regel auf beiden Seiten gemacht. Der häufigste Fehler auf beiden (!) Seiten ist, dass Änderungen unter Druck kurzfristig durchgewunken werden.

Abbildung 2-11 zeigt einen **Referenzprozess** für das Projektmanagement aus Sicht des Auftraggebers und des Auftragnehmers. Die wesentliche Schnittstelle ist das Requirements Engineering. RE betrachtet als Querschnittsprozess die Fragestellung der Ziele und Zielerreichung eines Projekts von den frühen Phasen des Marketings und Produktmanagements über die Angebotserstellung, die Projektplanung bis hin zu Projektausführung, Implementierung, Test und Wartung. Insbesondere unterstützt ein systematisches RE bei Projekten die Schnittstellen zu Lieferanten.

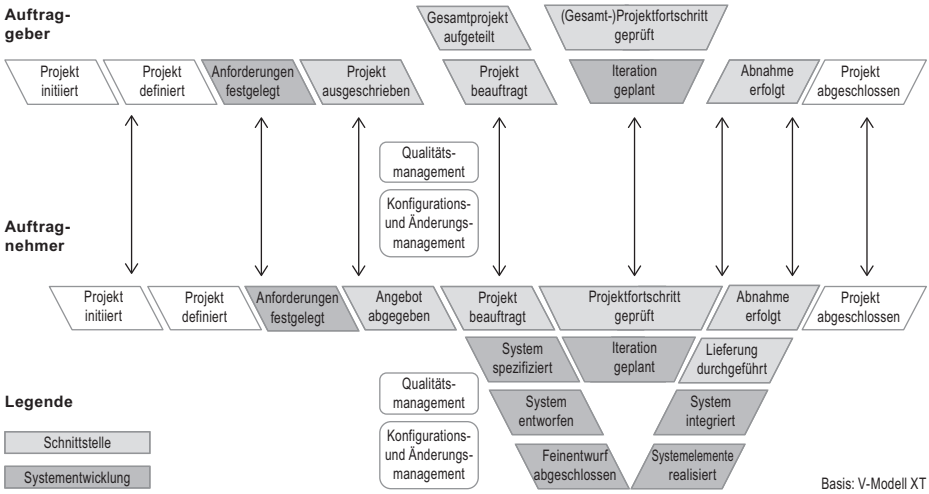


Abb. 2-11 *Requirements Engineering und Projektmanagement*