Christof Ebert

Systematisches Requirements Engineering

Anforderungen ermitteln, dokumentieren, analysieren und verwalten

dpunkt.verlag

Christof Ebert ist Gründer und Geschäftsführer von Vector Consulting Services. Er unterstützt Unternehmen bei Strategie, Technologie und agiler Transformation. Zuvor war er zwölf Jahre bei einem IT-Konzern in weltweiten Führungsaufgaben tätig. Er arbeitet in Aufsichtsgremien von Unternehmen und stimuliert Start-ups als Business Angel. An der Universität Stuttgart und der Sorbonne in Paris ist er Professor und hält Patente in Softwaretechnik und AI. Er wirkt in den Herausgeber-Komitees von IEEE Software, Software Quality Journal und dem Journal of Systems and Software. In seiner Freizeit spielt er als Musiker Keyboard und Kirchenorgel und engagiert sich im sozialen Bereich.

Folgen Sie ihm auf Twitter und LinkedIn: *@ChristofEbert*.

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

Homepage des Buches: *www.requirements-excellence.com*

Christof Ebert

Systematisches Requirements Engineering

Anforderungen ermitteln, dokumentieren, analysieren und verwalten

7., überarbeitete und aktualisierte Auflage

dpunkt.verlag

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

Lektorat: Christa Preisendanz
Lektoratsassistenz: Julia Griebel
Copy-Editing: Ursula Zimpfer, Herrenberg
Layout & Satz: Birgit Bäuerlein
Herstellung: Stefanie Weidner
Umschlaggestaltung: Helmut Kraus, *www.exclam.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:
Print 978-3-86490-919-1
PDF 978-3-96910-768-3
ePub 978-3-96910-769-0
mobi 978-3-96910-770-6

7., überarbeitete und aktualisierte Auflage 2022

Wieblinger Weg 17
69123 Heidelberg

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *hallo@dpunkt.de*.

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 7. Auflage

Wer immer tut, was er schon kann, bleibt immer das, was er schon ist.

Henry Ford

Wie kaum ein anderer reüssierte Henry Ford als Ingenieur und Unternehmer. Er rüttelt uns mit seinem Weckruf auf – gerade jetzt im neuen Normal. Viele haben die Pandemie als willkommene Entschuldigung benutzt, sich zu Hause einzuigeln und zu stagnieren. Manche hoffen selbst heute noch auf die Rückkehr in frühere Zeiten. Damit sind wir gerade in einem Hochlohnland nicht wettbewerbsfähig. In den Worten des großen Aufklärers Immanuel Kant müssen wir heraus aus der selbstverschuldeten Unmündigkeit. Das Buch zeigt, wie Sie Anforderungen setzen und verfolgen. Denn im neuen Normal ist Neues normal. Anforderungen sind dafür die Basis.

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. 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.

Mit diesem Buch bekommen Sie alles für gutes Requirements Engineering. Branchenübergreifend – von der Konzeption bis zur Evolution. Es ist aus der Praxis für die Praxis geschrieben und zeigt Ihnen, wie Sie Wert für sich und Ihr Unternehmen schaffen – und erhalten. Es adressiert Requirements Engineering für unterschiedliche Anwendungsbereiche, sei es agile Entwicklung versus formale Vorgehensweise für komplexe Projekte.

Diese 7. Auflage unterstreicht agiles Requirements Engineering, testorientierte Anforderungen, Systems Engineering und aktuelle Trends. Sie enthält viele neue Abbildungen und ein komplett überarbeitetes Glossar. Mit aktualisierten Beispielen wird die Rolle des Requirements Engineering im Projekt gezeigt. Das Buch berücksichtigt den aktuellen Lehrplan des IREB-Zertifizierungsprogramms. Die URL-Verweise sind aktualisiert. Alle Templates sind online verfügbar. So bleibt diese Neuauflage für alle Lesenden eine wichtige Referenz auf dem Schreibtisch.

Die meisten Projekte umfassen Änderungen existierender Systeme. Das Buch adressiert diese Situation und bewegt sich nicht akademisch auf der »grünen Wiese«. Ein Selbsttest hilft bei der Bewertung Ihrer Fähigkeiten im Requirements Engineering. Der Nutzen und ROI von Requirements Engineering wird an verschiedenen Stellen herausgestellt. Damit können Sie zielorientiert eigene Herausforderungen adressieren. 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 Mitarbeitende und Kundenunternehmen von Vector Consulting, mit denen wir die genannten Praktiken umsetzen und ständig verbessern. Hervorheben möchte ich Frank Kirschke-Biller und Arnold Rudorfer, die mich in den Fallstudien bei Ford und Siemens unterstützten. Immer wieder erhalte ich Verbesserungsvorschläge von Lesenden. Danke dafür! Für bessere Lesbarkeit verwende ich dudengerechte Formen. Danke schließlich an Christa Preisendanz und den dpunkt.verlag für die gewohnte Professionalität.

Mein Freund Al Davis hatte mich als damaliger Chefredakteur von IEEE Software dazu angeregt, Requirements Engineering systematisch und dennoch agil (!) in der Industrie zu verankern. Dank an Al und die vielen, die dieses Thema in der Praxis antreiben, wie Ian Alexander, Anthony Finkelstein, Don Gause, Michael Goedicke, Martin Glinz, Matthias Jarke, Neil Maiden, Barbara Paech, Klaus Pohl, Mary Popendieck, Charles Symons, Suzanne Robertson, Ian Sommerville und Karl Wiegers.

Ihnen, liebe Leserinnen und Leser, wünsche ich viel Erfolg bei allem, was Sie neu anpacken! Dieses Buch gibt Ihnen mit lösungsorientiertem Denken die nötigen Impulse, um Ziele richtig offensiv anzugehen.

Christof Ebert
Stuttgart, 22.2.22

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

⁂

»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 Technology Officer, Vodafone

⁂

»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.

⁂

»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

Inhaltsübersicht

Anhang

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?

Das Problem vieler Unternehmen ist, dass zu viele Funktionen verkauft werden, und zu wenig Wert. Oft zerbrechen wir uns 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 und können deren Wert nicht darstellen. Wir optimieren und bemühen uns ständig, bessere Produkte zu entwickeln – und fühlen uns doch 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.

Zeig mir deine Anforderungen und ich sage dir, wie dein Projekt endet. So lautet eine alte Weisheit, und wir wollen sie hier gleich mal anwenden.

Abbildung 1–1 zeigt ein typisches Beispiel einer Anforderung. Sie ist in Prosa, weil das vermeintlich verständlicher 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 umgesetzt werden?
- Wie ist der Status der Anforderung?
- Wer ist für die Anforderung verantwortlich?
- Wie wird die Anforderung validiert?
- Wie wird die Anforderung umgesetzt?

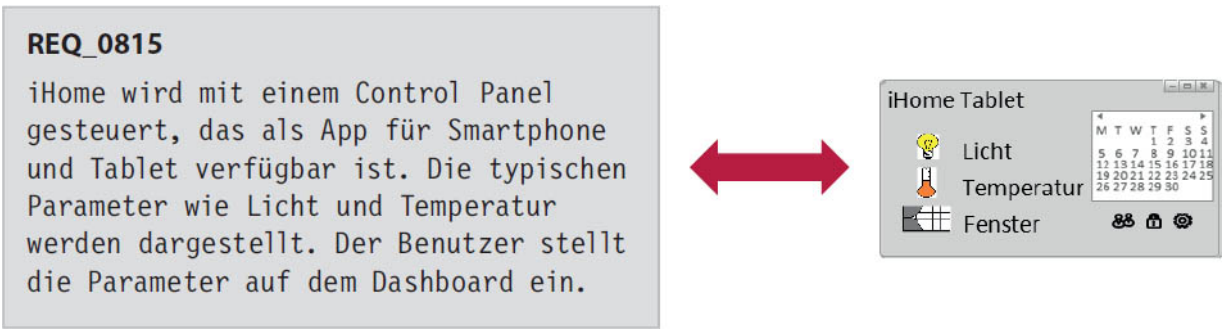

Abb. 1–1 *Eine Anforderung – so viele Fragen*

Software wird 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 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 Beispiel für hausgemachte Komplexität sind IT-Migrationsprojekte. Die Stakeholder, manchmal eingedeutscht als »Interesseneigner«, sind sich oft schnell darin einig, dass alle Funktionen des existierenden Altsystems übertragen werden müssen. Das ist ein großer 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. Komplexität ist der Feind jeder Innovation. Neues entsteht nur durch Weglassen von Ballast, wie wir von Unternehmen wie Apple oder Tesla lernen können. 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.

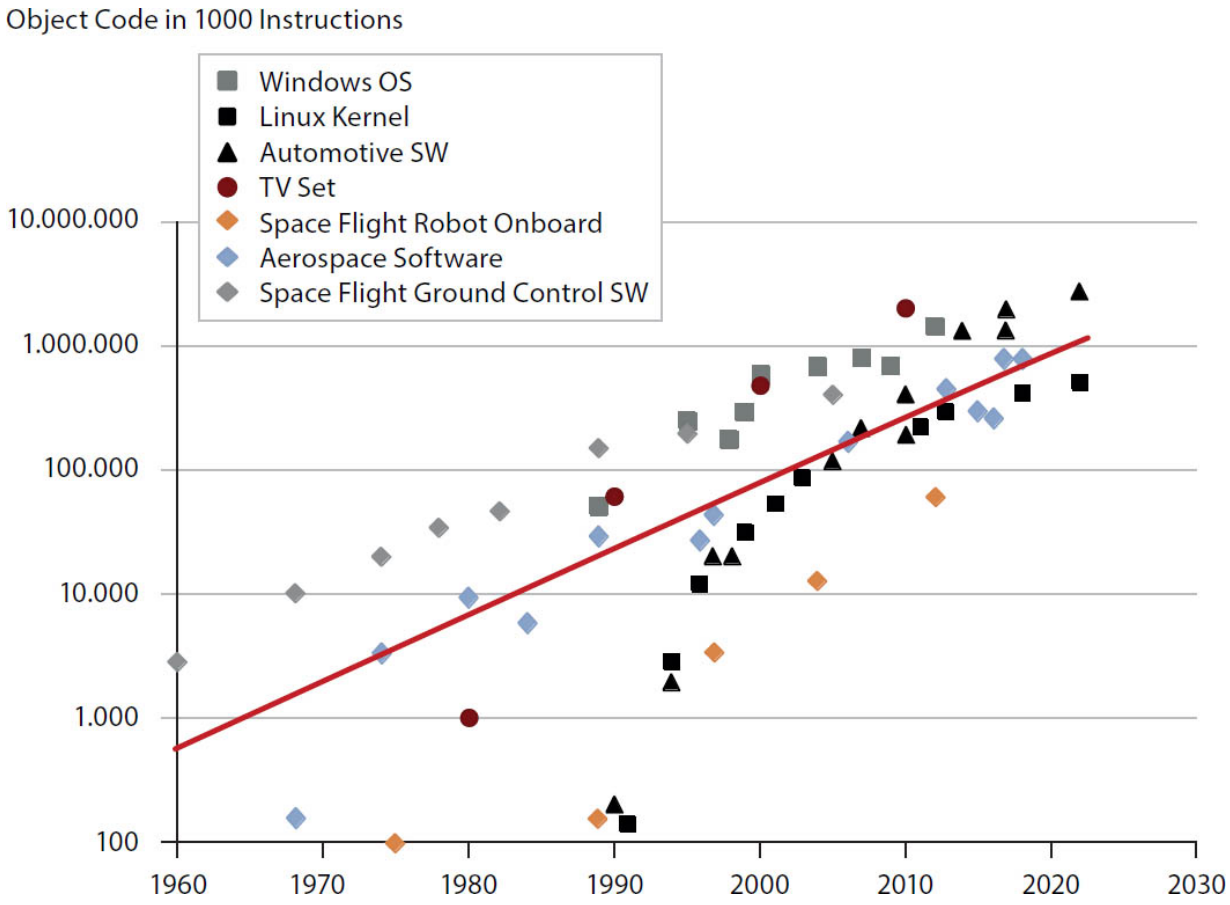

Abb. 1–2 *Komplexität von Softwaresystemen über die Zeit*

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. Die 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) [Standish2021]. RE erhält im Schnitt nur 2–5 % des Projektaufwands, aber Fehler in den Anforderungen haben die größten Effekte [Gartner2022, Lauesen2020, Standish2021, Charette2018, Garousi2019, Ebert2014a, Ebert2007].

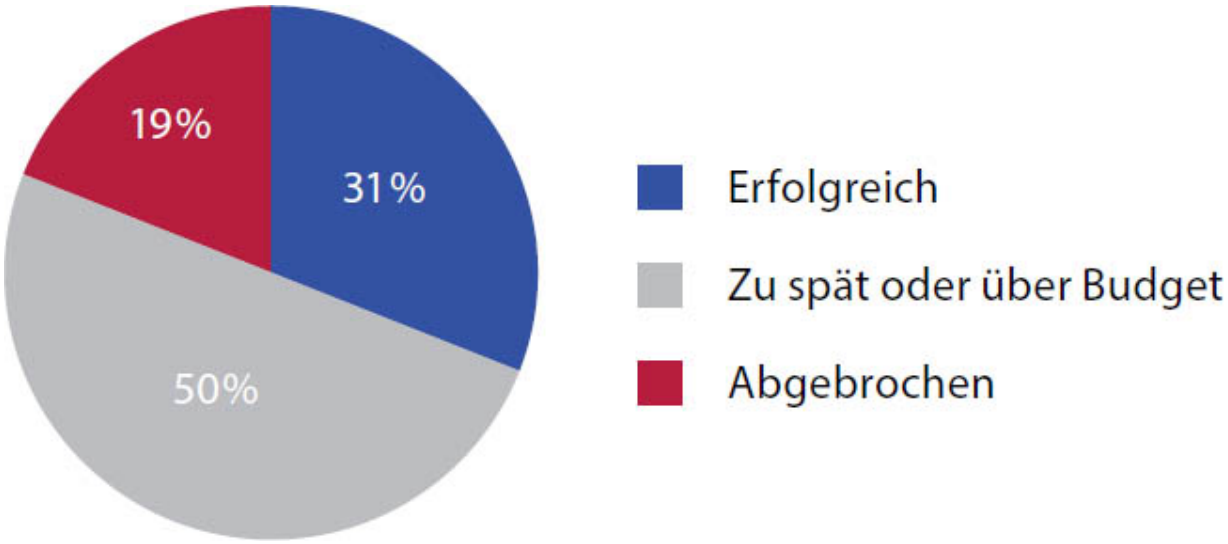

Abb. 1–3 *Unzureichendes Requirements Engineering reduziert den Projekterfolg.*

Ein wesentlicher Grund für Projektscheitern sind unklare Ziele. 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 offensichtliche Allgemeinplätze.

Beispiel:

Viele Projekte scheitern, weil man versucht, die Interessen aller Stakeholder zu berücksichtigen, statt klare Ziele und Führung einzusetzen [Lauesen2020]. Aktuelle Beispiele sind die elektronische Patientenakte in Deutschland (ePA) oder die amerikanische Gesundheitsversicherung. 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, regte aber politische Gegner an, das gesamte Programm zu hinterfragen. Die Mehrkosten der IT lagen im Bereich von knapp 500 Mrd. US$. Die Gründe für das Scheitern waren zu viele Stakeholder, die mitwirkten, sich starkändernde Anforderungen, eine unzureichende Validierungsstrategie und ein monolithisches System, das ungeeignet für Teillösungen und inkrementelle Änderungen war.

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 verfolgt werden müssen)
- Fehlendes Wissensmanagement (Beispiel: Projektmanager übernimmt neues Kundenprojekt und hat nicht das implizite Wissen zum Kunden und dessen Hintergrund.)

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

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 ist bis heute gültig und nur inhaltlich etwas 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 Stakeholder 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 [Pendo2019]. 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