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.
```

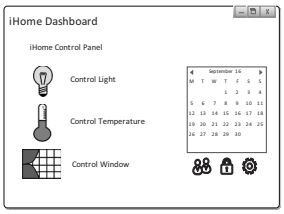

Abb. 1-1 *Eine Anforderung – so viele Fragen*

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.

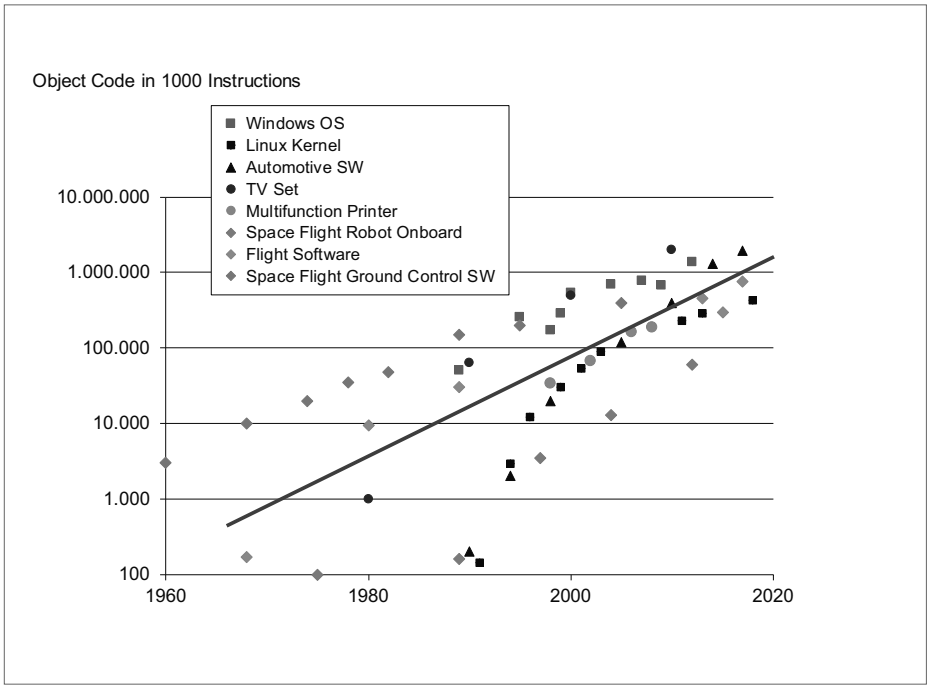

Abb. 1-2 *Komplexität von Softwaresystemen über die Zeit*

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.)

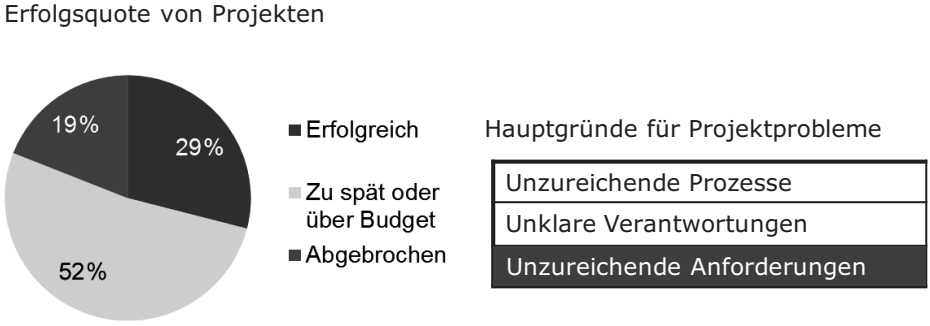

Abb. 1-3 *Unzureichendes Requirements Engineering reduziert den Projekterfolg.*

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.

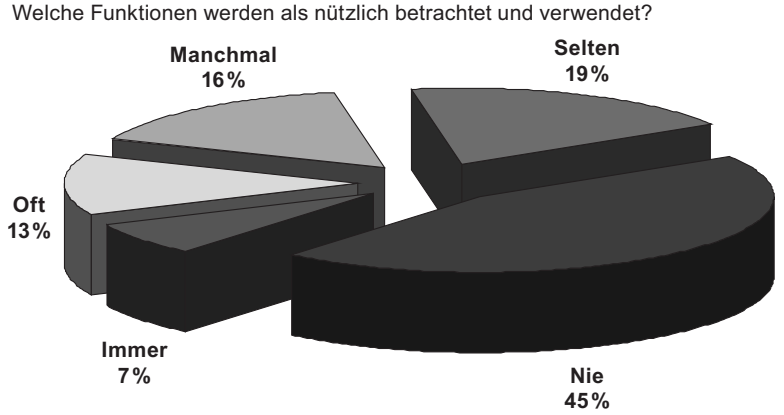

Abb. 1-4 *Falsche Anforderungen erhöhen die Kosten.*

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.

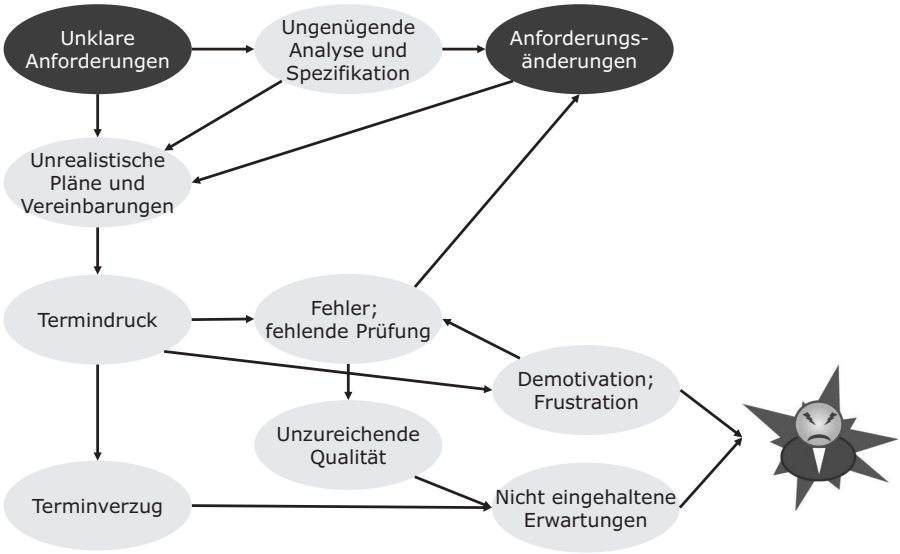

Abb. 1-5 *Unzureichende Anforderungen und die Konsequenzen*

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.

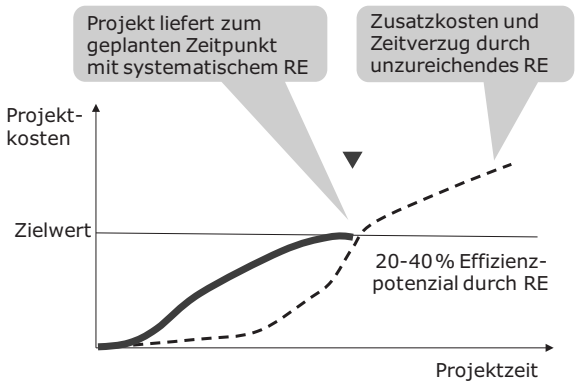

Abb. 1-6 *Effizienzpotenzial mit systematischem Requirements Engineering*

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

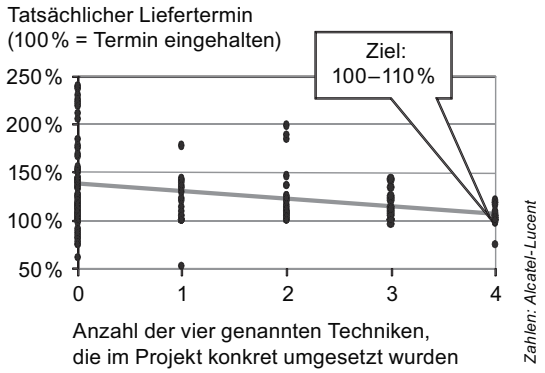

Abb. 1-7 *Der gleichzeitige Einsatz von vier Requirements-Techniken verbessert den Projekterfolg.*

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].

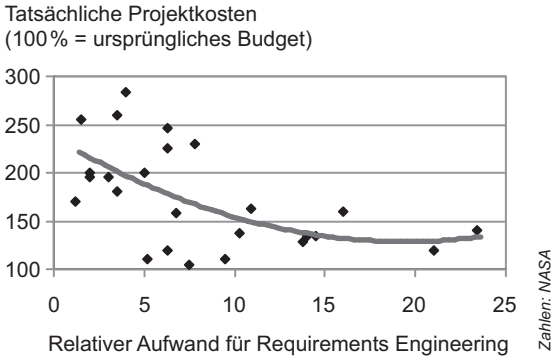

Abb. 1-8 *Kosteneinhaltung in Abhängigkeit vom Aufwand für das Requirements Engineering*

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.

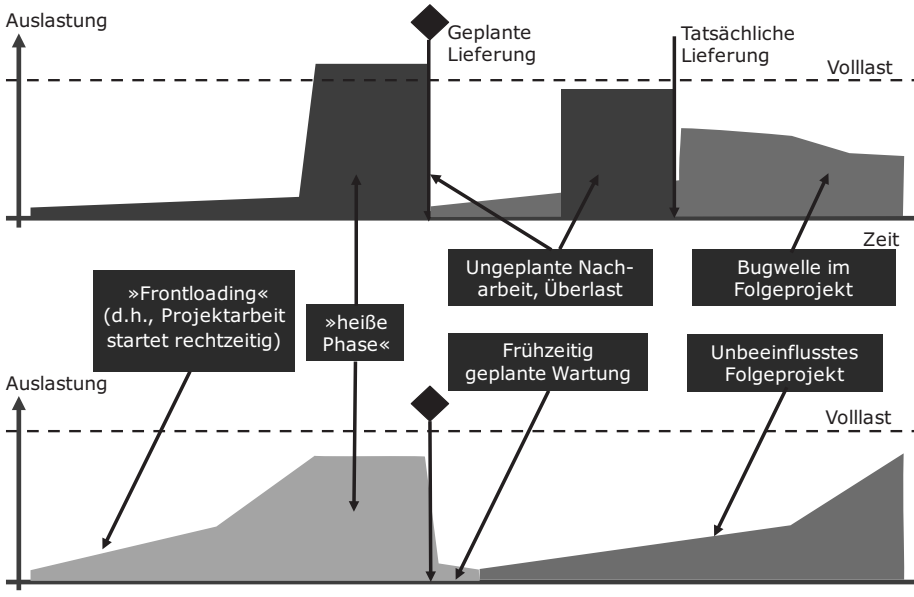

Abb. 1-9 *Frühzeitige Lastbalance durch Frontloading*

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.

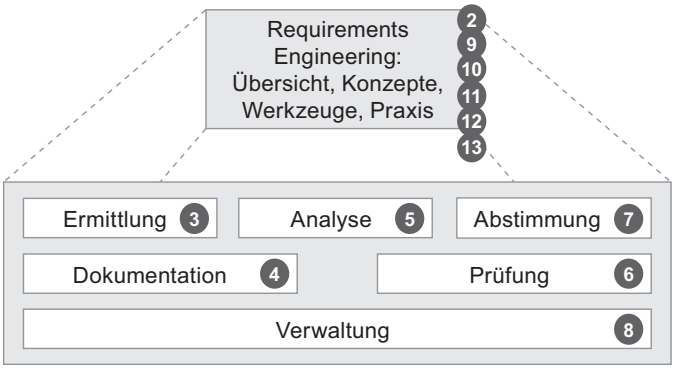

Abb. 1-10 *Das Buch im Kontext des Requirements Engineering (Schwarze Kreise sind Kapitelnummern.)*

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.