→ 4., überarbeitete Auflage

Christof Ebert

Systematisches Requirements Engineering

Anforderungen ermitteln, spezifizieren, analysieren und verwalten

dpunkt.verlag

Systematisches Requirements Engineering

Dr. Christof Ebert ist Geschäftsführer der Vector Consulting Services GmbH. Er unterstützt Unternehmen weltweit bei der Verbesserung ihrer Produktentwicklung und Produktstrategie sowie im Veränderungsmanagement. Zuvor war er fünfzehn Jahre in internationalen Führungsfunktionen für Alcatel-Lucent tätig. Dr. Ebert sitzt in verschiedenen Aufsichts- und Expertengremien. Er ist ein international renommierter Redner, lehrt an der Universität Stuttgart und ist Autor mehrerer Bücher, ein IREB »Certified Professional for Requirements Engineering« und vom SEI als CMMI-Trainer zertifiziert. Viele Unternehmen haben bereits seine Erfahrungen in Requirements Engineering und Produktmanagement genutzt, um ihre Marktposition zu verbessern. Auf der internationalen Requirements-Engineering-Konferenz 2005 wurde er für den besten Praxisbeitrag ausgezeichnet.

Christof Ebert

Systematisches Requirements Engineering

Anforderungen ermitteln, spezifizieren, analysieren und verwalten

4., überarbeitete Auflage

dpunkt.verlag

Christof Ebert
christof.ebert@vector.com

Lektorat: Christa Preisendanz
Copy-Editing: Ursula Zimpfer, Herrenberg
Herstellung: Birgit Bäuerlein
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: Media-Print Informationstechnologie, Paderborn

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

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 978-3-89864-812-7
PDF 978-3-86491-112-5
ePub 978-3-86491-113-2

4., überarbeitete Auflage 2012

Ringstraße 19 B
69115 Heidelberg

5 4 3 2 1 0

Meinen Eltern Elfriede und Otto
und meinem Lehrer Prof. Rudolf Lauber.
Sie haben mir gezeigt,
dass ein *Werk* nur dann zu einem Wert wird,
wenn die *Anforderungen richtig* verstanden und umgesetzt sind.

Vorwort zur 4. Auflage

It isn't that they can't see the solution.
It is that they can't see the problem.
Gilbert Keith Chesterton

Die Saturn-V-Rakete ist die größte jemals gebaute Rakete. Mit ihr wurden die ersten Menschen zum Mond gebracht. Der deutsche Wernher von Braun wusste als ihr Konstrukteur, dass der Schlüssel zum Erfolg in abgestimmten Anforderungen lag. Eine wichtige Anforderung war, wie viel Gewicht die Saturn V transportieren muss. Das war eine fundamentale Anforderung im gesamten Raumfahrtprogramm, denn nur mit ausreichend Schub konnte die Raumkapsel die Erde verlassen. Doch verschiedene kritische Anforderungen hingen voneinander ab. Startgewicht, Manövrierfähigkeit und auch die große Menge Treibstoff waren bei starkem Schub kaum beherrschbar. Was tun? Von Braun analysierte die Anforderungen und Randbedingungen und erhielt vom NASA-Management nach einiger Zeit die Antwort, dass die Raumkapsel maximal 34 Tonnen wiegen würde. Das wäre verlässlich, enthielte genug Sicherheit, und er solle nun die Rakete dazu konstruieren. Von Braun glaubte dies aufgrund seiner eigenen Anforderungsanalyse nicht und kommunizierte seinen Entwicklungsleitern, dass sie eine Zuladung von 40 Tonnen transportieren müssen. Das erforderte substanzielle Änderungen des ursprünglichen Designs, beispielsweise fünf statt vier Triebwerke. Einige Jahre später kam dann die Apollo-11-Kapsel zum Start – und wog 50 Tonnen! Der Start der Saturn mit dieser Zuladung gelang zwar gerade noch, wäre aber mit den ursprünglichen Anforderungen gescheitert. Der Rest ist Geschichte – und zeigt die Bedeutung der Anforderungsermittlung und Analyse.

Software- und IT-Projekte sind allzu oft in einer ähnlichen Situation. Die Anforderungen sind unsicher und demzufolge die Schätzungen von Dauern, Produktivität und Kosten falsch. Zusätzliches Budget und einen Zeitpuffer hinzuzufügen ergibt aber auch keinen Sinn, denn man weiß ja gar nicht, auf welcher Basis geschätzt werden soll. Und jeder Personentag mehr macht das Projekt teurer. Der Software-Guru Tom DeMarco, der inzwischen auf fünfzig Jahre Erfahrungen mit verkorksten IT-Projekten zurückblickt, fasst seine Erfahrungen lapidar wie folgt

zusammen: »Früher dachte ich, Termin- und Budgetüberschreitungen kommen von schlechter Schätzung und Planung. Dann dachte ich, sie kommen von der hohen Komplexität. Heute weiß ich, dass sie davon kommen, dass praktisch jedes Projekt zu spät begonnen wird. Dann fehlt die Zeit für Anforderungsanalyse und Planung. Und damit ist das Scheitern vorprogrammiert.«

Requirements Engineering ist schwierig in der Umsetzung, da es zwar viele Interessengruppen gibt, die sich gerne überall einmischen, sich aber nicht festlegen wollen. Projekte scheitern vor allem wegen eines unzureichenden Requirements Engineering!

Grund genug, sich intensiver mit Anforderungen zu befassen. Anforderungen kommunizieren Bedürfnisse und Randbedingungen, und Requirements Engineering ist die Disziplin, die die Behandlung von Anforderungen über den gesamten Lebenszyklus der Software hinweg umfasst.

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.

Entstanden ist das Buch aus meinen weltweiten Seminaren, Vorträgen und Beratungsprojekten. Ziel und Inhalt des Buchs ist es zu zeigen, wie man Requirements Engineering systematisch – und damit erfolgreich – in die Praxis umsetzt. Hier geht es um das »Machen«: Wie muss ich mein eigenes Requirements Engineering aufstellen, um erfolgreich zu sein?

Diese vierte Auflage wurde komplett überarbeitet. Sie vertieft Themen, die aktuell an Bedeutung gewinnen. Dazu gehören agiles Requirements Engineering, Lean Development und verteilte Projektteams sowie Werkzeuge zur durchgängigen Unterstützung im Lebenszyklus. Neue Fallstudien zeigen die Umsetzung in die Praxis. Die meisten Projekte bestehen aus Änderungen von Bestandssoftware. Das Buch adressiert diese Situation und nicht nur Projekte auf der »grünen Wiese«. Die Zertifizierung zum Certified Professional Requirements Engineer hat inzwischen Fuß gefasst, und wir decken den aktuellen Kanon mit diesem Buch ab. 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 sind nun im Download frei verfügbar[1].

Ich bedanke mich bei den vielen Personen und Unternehmen, ohne deren Unterstützung ein solches Werk nicht möglich gewesen wäre. Dies gilt insbesondere für die Kunden und Mitarbeiter von Vector Consulting Services, mit denen

1. *http://consulting.vector.com/RE-templates*

wir die genannten Praktiken umsetzen und verbessern. Besonders danken möchte ich Felix Gutbrodt, Daniel Kanth und Stephan Pech für viele gute Tipps.

Requirements Engineering als Disziplin wird vor allem durch die »IEEE International Requirements Engineering Conference« angetrieben. Seit vielen Jahren arbeite ich im Programmkomitee der Konferenz. Mein Dank geht an Al Davis, der zu den ganz Großen des Requirements Engineering gehört. Al hatte mich als Chefredakteur von IEEE Software stimuliert, Requirements Engineering als Disziplin in der Industrie zu verankern. Die enge Zusammenarbeit mit Personen 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 führte zu einer Verzahnung von Theorie und Praxis, wie sie in vielen anderen Disziplinen der Softwaretechnik leider fehlt. Erfolgreicher Transfer ist das Ergebnis enger Zusammenarbeit von engagierten Praktikern und Forschern.

Ein Buch für die Praxis braucht ein praktisches Beispiel. Danken möchte ich dafür dem Institut für Automatisierungs- und Softwaretechnik der Universität Stuttgart. Ein spezieller Dank geht an IBM, MKS und Vector, deren Werkzeuge ich seit vielen Jahren nutze, einführe und verbessere. Viele Leser der früheren Auflagen haben mir wertvolle Tipps für diese Überarbeitung gegeben. Weiter so! Mein Dank geht schließlich an den dpunkt.verlag und insbesondere an Christa Preisendanz, die mich auf vielfältige Weise immer wieder stimuliert, dieses Buch zu verbessern.

Verbessern heißt inkrementell aufzubauen, Bestehendes zu optimieren und Komplexität zu kontrollieren. Der Hauptunterschied zwischen dem amerikanischen und dem sowjetischen Raumfahrtprogramm ist, dass in den USA ständig neue und noch komplexere Lösungen entwickelt werden, während das russische Sojus-Programm auf eine Rakete aus den fünfziger Jahren zurückgeht. Mit Erfolg. Aktuell ist es die einzige Rakete, die nach wie vor erfolgreich zur ISS fliegen kann. Beachten wir gerade in unseren Systemen, dass wir nicht nur Anforderungen entwickeln, sondern damit immer auch Komplexität beherrschen.

Ich stehe Ihnen, verehrte Leser, während und nach der Lektüre des Buchs für Fragen und Unterstützung gerne zur Verfügung. Das hilft Ihnen als Leser, und es hilft dem Requirements Engineering, sich weiterzuentwickeln.

Nun wünsche ich Ihnen und Ihren Projekten Erfolg mit diesem Buch und mit einem lösungsorientierten Requirements Engineering, das auf die richtigen Probleme eingeht!

Christof Ebert
Stuttgart, März 2012

Inhaltsverzeichnis

1 Motivation

Wer sein Ziel nicht kennt,
kann jeden Weg nehmen.
Alice im Wunderland

1.1 Warum ein Buch über Requirements Engineering?

»Könntest du mir bitte sagen, welchen Weg ich von hier aus nehmen soll?«, fragt Alice im wunderbaren Roman »Alice im Wunderland«. »Das hängt vor allem davon ab, wohin du gehen willst«, sprach die Katze. »Ich weiß es nicht ...«, sagte Alice. »Dann ist es egal, wohin du gehst«, antwortete die Katze.

Dieser kurze Dialog beschreibt, warum Anforderungen und Ziele eine Rolle spielen. Viel zu oft zerbrechen wir uns vorschnell den Kopf über eine Lösung – ohne verstanden zu haben, welches Problem wir konkret lösen müssen. Wir laden zu einer Besprechung ein, ohne zu hinterfragen, was sie eigentlich bringen soll. Wir entwickeln Funktionen für ein Softwaresystem und wissen nicht, welchen Wert sie für die Käufer und Benutzer darstellen. Wir optimieren und bemühen uns ständig, bessere Produkte zu entwickeln – ohne uns klarzumachen, was diese Produkte erreichen sollen, wenn sie auf den Markt kommen. 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.

Prüfen Sie sich einfach einmal selbst, und beantworten Sie die beiden folgenden Fragen spontan und ehrlich. Sind die Anforderungen einzeln verständlich und testbar dokumentiert? Hat Ihr derzeitiges Projekt einen expliziten Business Case, der immer wieder geprüft wird? Gibt es für jede einzelne Anforderung eine Begründung, die aus Benutzersicht beschreibt, was durch diese Anforderung besser wird? Falls nicht, ist das Buch das richtige für Sie. Falls ja, lesen Sie die Fragen nochmals und gehen aufrichtig in sich.

Beispielhaft sind Migrationsprojekte, die als wichtigste Vorgabe immer angeben, dass »alle Funktionen des existierenden Altsystems übertragen werden müssen«. Ein Fehler. Erstens kann sowieso keiner mehr alle existierenden Altfunktionen im Zusammenhang beschreiben (und Archäologie gehört zu den wenigen

Disziplinen, die nicht explizit im Software Engineering verankert sind). Und zweitens ist gerade ein neues System die einzige Chance, gleichzeitig auch alte Prozesse und Workflows über Bord zu werfen.

Die Anforderungen an Software werden zunehmend komplexer. Abbildung 1–1 zeigt das Komplexitätswachstum von verschiedenen Softwaresystemen, 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 Betriebssysteme, Vermittlungssysteme 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 neunziger Jahre 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. Mit dieser zunehmenden Komplexität sind Funktionen korreliert und in unterschiedlichen Hardwaresystemen vernetzt, was zusätzliche Komplexität durch Qualitätsanforderungen mit sich bringt. Diese schnell wachsende Komplexität fordert systematisches Requirements Engineering (RE), um die Qualität und Kosten nachhaltig kontrollieren zu können.

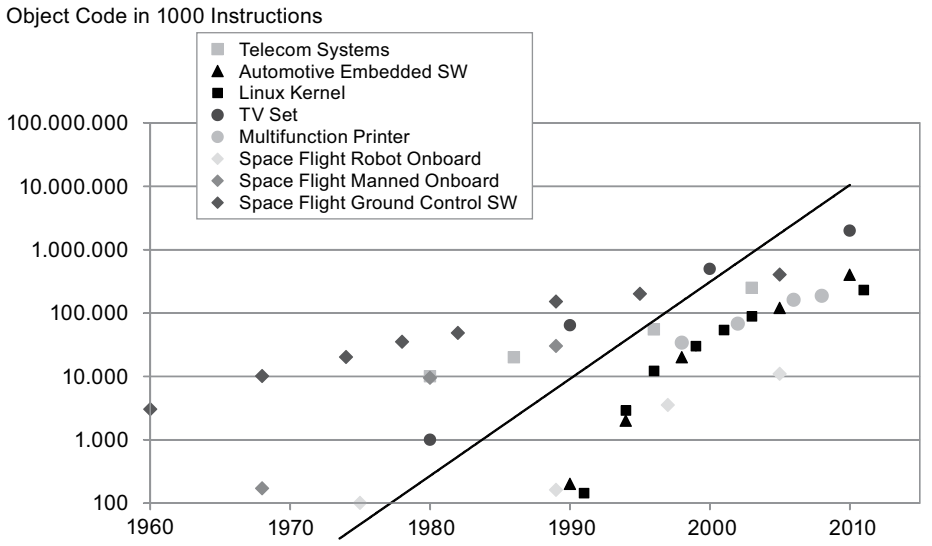


Abb. 1–1 *Komplexität von Softwaresystemen*

1. Quellen der Daten: Eigene Studien des Autors zu Telekommunikationssystemen von Alcatel-Lucent und Siemens, NASA, Studien der HIS sowie Codeanalysen bei Windows und Linux. Konversionen soweit nötig gemäß den Korrekturfaktoren von Les Hatton (*http://www.leshatton.org/Documents/LOC2005.pdf*).

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 und liefert Theorie und Praxis des Requirements Engineering, sodass die Konzepte direkt umgesetzt werden können. 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 Anforderungen

Wir alles wissen: Zu viele Projekte scheitern und Produkte erreichen die Marktziele nicht. Was wir nicht wissen (oder nicht wahrhaben wollen): **Unzureichendes Requirements Engineering ist ein Hauptgrund.** Die neueste Studie der Standish Group von 2010 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. 1–2) [Standish2011].

Die meisten Projekte, die abgebrochen wurden, hatten nur ungenügend geklärte Anforderungen und konnten Änderungen der Anforderungen nicht beherrschen [Standish2011, Ebert2007a, Charette2005, Tan2011]. Hier einige Beispiele aus Kundenprojekten, bei denen wir zur Unterstützung und Moderation gerufen wurden:

- Implizite Anforderungen (z.B. Kunde erwähnt Funktionen nicht, da sie für ihn selbstverständlich sind, nur Lieferant weiß es 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?)
- Anforderungen, die erst spät klar werden (z.B. Festpreisangebot; Anforderung im Grobkonzept klar, später im Feinkonzept werden weitere Details deutlich, die zu zusätzlichem Aufwand führen)
- Anforderungs-Baseline von vornherein fehlerhaft oder unzureichend (d.h., Kunde hat sich nicht die Zeit genommen, die nötigen Anforderungen zu präzisieren)

- Qualität der Ausschreibungen (z.B. oberflächlich und missverständlich beschriebene Anforderungen)
- Unsicherheiten und Unklarheiten (z.B. Schätzungen und Pläne basieren auf nicht verstandenen Risiken und oberflächlich dokumentierten Anforderungen)
- Unzureichendes Change Management (z.B. Kunde meldet sich beim Projektmanager oder Entwickler: »Wir brauchen das und das noch.«)
- Fehlende Dokumentation der Basis und Änderungen (z.B. 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)
- Rotation von Mitarbeitern in ein neue Feld (z.B. Projektmanager übernimmt neues Kundenprojekt und hat nicht das implizite Wissen zum Kunden und dessen Hintergrund)

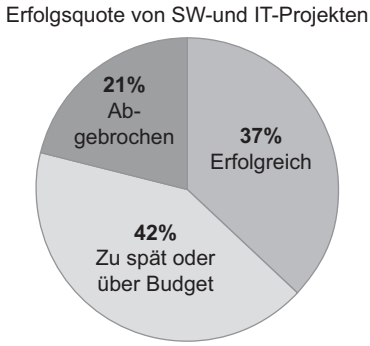
Ein wichtiger Grund dafür, dass Projekte ihre Ziele nicht erreichen, liegt in nicht sauber formulierten Zielen. Abbildung 1–2 zeigt im unteren Teil diesen Zusammenhang und unterstreicht schon aufgrund der Datenlage die immense – und wachsende – Bedeutung eines guten Requirements Engineering. Bei 87% aller abgebrochenen Projekte war unzureichendes RE ein wesentlicher Grund für das Scheitern. Häufiger wurden nur noch »Prozessfähigkeit« und »Organisationsmanagement« genannt, aber das sind auch offensichtliche Allgemeinplätze, die man sich in jedem verkorksten Projekt gut vorstellen kann.

Interessant ist übrigens die Beobachtung, dass in Krisenzeiten, wie 2001 und 2008, die Erfolgsquote zurückgeht. Dann werden vermeintlich unnötige Ausgaben, wie für Requirements Engineering und Reviews, reduziert – mit durchschlagenden Ergebnissen. **Technologische Herausforderungen sind keine gravierenden Projektrisiken. Schlechtes Management dagegen schon.**

Doch es gibt auch genügend Projekte, die ihre Ziele erreichen. Durch den systematischen Einsatz von besten Praktiken im Projektmanagement, in der Entwicklung und natürlich auch im RE hat sich die Zahl der erfolgreichen Projekte seit Mitte der neunziger Jahre verdoppelt. Abbildung 1–3 zeigt diesen Zusammenhang, in dem die Ergebnisse der ursprünglichen Studie der Standish Group aus 1994 mit jenen aus der Studie von 2010 verglichen werden.

Erfolg ist machbar. Viele Unternehmen haben ihre Produktentwicklung bereits erfolgreich verbessert. Die Erfolgsfaktoren sind in der Regel die gleichen:

- Ergebnisorientierte Vorgaben
- Zielorientierte Prozesse
- Kompetentes Produkt- und Projektmanagement
- Standardisierte und optimierte Infrastruktur
- Fokus auf Anforderungen, Änderungen und Risiken im Projekt



Hauptgründe für Projektprobleme

Prozessfähigkeit	91%
Organisationsmanagement	87%
Requirements Engineering	87%

Abb. 1–2 *Unzureichendes Requirements Engineering reduziert Projekterfolge.*

Methodisch können diese Erfolgsfaktoren unterschiedlich erreicht werden, solange 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.

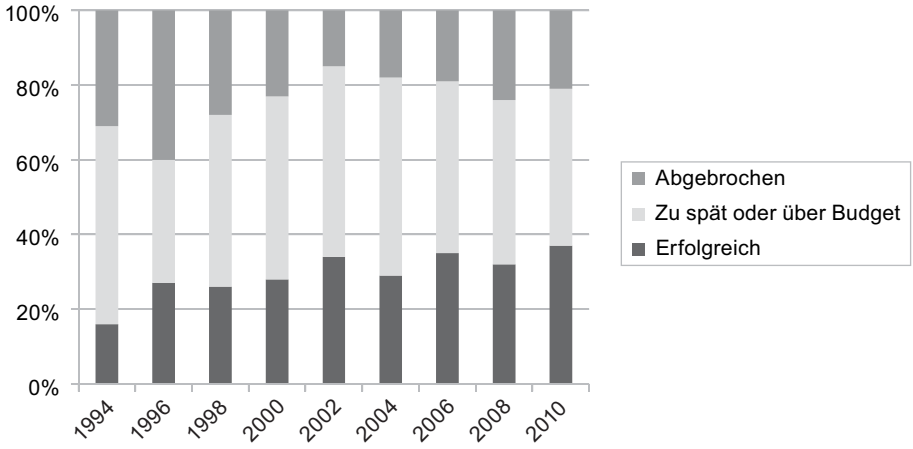


Abb. 1–3 *Projekte verbessern sich durch konsequente Nutzung der richtigen Techniken.*

Auf was muss man beim RE achten? Aus unterschiedlichen Praxiserfahrungen lassen sich die wichtigsten Risiken im RE ableiten. Die Risiken zu kennen, heißt, dass man sich darauf vorbereiten kann, um sie beim nächsten Mal zu vermeiden. Oder wie es Mao Zedong formulierte: »Die Niederlage zu verstehen ist der erste Schritt zum Sieg.« Die folgende Liste wurde ursprünglich von drei 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.

Eine der Schlüsselregeln im Requirements Engineering besagt, dass man dem Kunden das liefern muss, 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 Interessengruppen 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 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. Man versucht, aus der späteren Anwendung innerhalb der Geschäftsprozesse des Kunden heraus einzuschätzen, welche Anforderungen kritisch sind und ob vielleicht bestimmte Randbedingungen übersehen wurden.

Risiko 2: Falsche Anforderungen
Wie alle Arbeitsergebnisse sind auch Anforderungen fehlerhaft. 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 entdeckt 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. Abbildung 1–4 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.

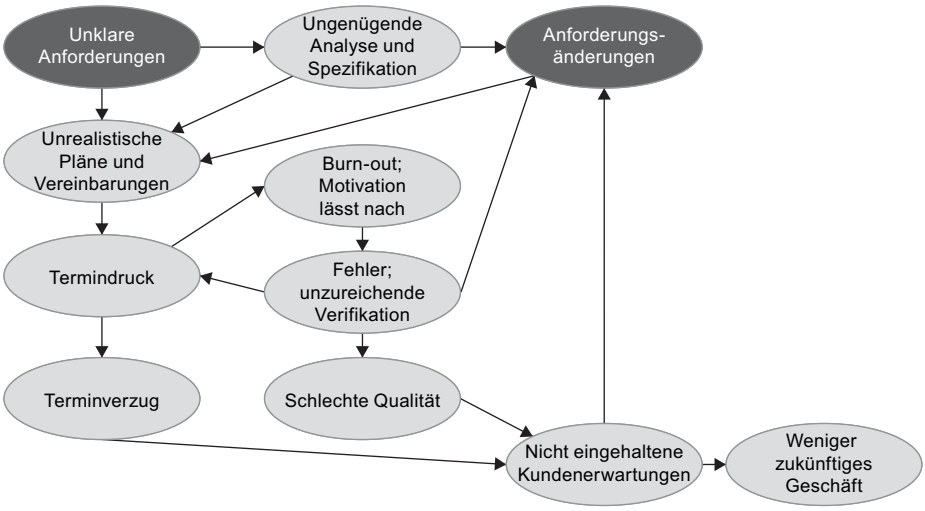


Abb. 1–4 *Fehlerhafte Anforderungen und die Konsequenzen*

Typische Fehler sind vage und ungenaue Beschreibungen, Widersprüche, Inkonsistenzen, Lücken und natürlich Denkfehler. Zu stark vereinfachte oder oberflächliche Spezifikationen führen später zu fehlender oder falscher Funktionalität. Aus Zeitgründen und zur Vereinfachung werden Anforderungen oft wortwörtlich von Kunden- oder Benutzerinterviews übernommen. Dies sind allerdings nur Rohfassungen, die erweitert und präzisiert werden müssen. Entdeckt werden diese Fehler nur mit frühzeitiger Verifikation und Validierung. 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? Klare Antwort: 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. Oft wurden schon kundenseitig falsche Versionen ins Projekt geschickt.

Fehlerhafte Anforderungen entstehen auch, wenn der Bedarf (Warum kauft der Kunde? Was will der Kunde?) und die Lösung (Wie werden die Bedürfnisse adressiert?) gemischt und verwechselt werden. Spezifikationen von Anforderungen und Lösungen sind zwei grundlegend unterschiedliche Dinge. Kundenanforderungen beschreiben, was geliefert werden muss. Die Lösungsspezifikation im Pflichtenheft beschreibt, wie die Lösung entwickelt wird. Trotz dieser klaren Unterscheidung werden die Was- und Wie-Perspektiven immer wieder vermischt. Häufig geschieht dies aus einer vermeintlichen Zeitersparnis heraus. Man beginnt, die Anforderungen zu notieren. Im Beschreiben der Anforderung kom-

men erste Ideen zu gegenseitigen Abhängigkeiten und zur möglichen Realisierung, die einfach mit der Anforderung zusammen notiert werden. Selbst wenn man dies in getrennten Teilen der Anforderung macht, sollte es klar sein, dass prinzipiell nie eine 1:1-Abbildung möglich ist. Die nächste Anforderung hat vielleicht überlappende Einflüsse und schon passt das Muster nicht mehr. Schlimm wird es vor allem, wenn sich Anforderungen später ändern oder gestrichen werden. Wohin mit der teilweisen Analyse, die vielleicht auch noch von anderen Anforderungen gebraucht wird? Hier gilt die klare Regel, immer zwei Spezifikationsdokumente zu führen, nämlich die Liste der Anforderungen (Lastenheft) und die Liste der Lösungsspezifikation (Pflichtenheft). Nachverfolgbarkeit durch eindeutige Bezeichner und eventuell eine angepasste Werkzeugunterstützung erlauben später, auch umfangreiche Änderungen schnell in trockene Tücher zu bekommen.

»Gold Plating«, also Verschnörkelungen durch Entwickler und Benutzer, bringt unnötige Funktionen, Verzögerungen und zu hohe Kosten ins Projekt. Entwickler versuchen oft, Anforderungen, die sie verstanden haben, mit Leben zu füllen, und entwickeln so Funktionen, die nie vereinbart wurden. Im besten Fall passiert gar nichts, denn der Kunde wird sich hüten, solche verzichtbaren Funktionen zu würdigen. Im Normalfall allerdings wird diese Komplexität unbeherrschbar, weil ja keine Ressourcen dafür vorgesehen wurden. Jede weitere Funktion bringt Abhängigkeiten zu anderen Funktionen, Sonderfälle, Ausnahmesituationen – und damit zusätzlichen Entwicklungs-, Korrektur- und Testaufwand. Anforderungen sollen widerspiegeln, was der Kunde als relevant betrachtet. Das Gleiche gilt für Spezifikationsdokumente. Dies sind Dokumente, die straff beschreiben, wie gearbeitet wird. Es sind keine detaillierten Entwurfsbeschreibungen. Hilfreich ist es, von vornherein mit verschiedenen Dokumenten zu arbeiten, sodass Entwurfsentscheidungen von Beginn an im richtigen Dokument – also in der Architektur- oder der Entwurfsbeschreibung – dokumentiert werden, ohne den Umweg über ein Spezifikationsdokument, wo solche Informationen nicht hingehören. Im Unterschied zu diesen Verschnörkelungen sollten allerdings Ausnahmebehandlungen der regulären Anforderungen durchaus mit dem Kunden geklärt und als Erweiterung der Anforderung spezifiziert werden. Use Cases beispielsweise haben bereits in ihrem Template eine solche Ausnahmebehandlung vorgesehen. Wird die Behandlung von Ausnahmen nicht vorab abgestimmt, kann dies zu sehr verwickelten und inkonsistenten Realisierungen führen.

Risiko 3: Sich ändernde Anforderungen
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 dem Neuigkeitsgrad von Projekt und Technologie bei Lieferant und Benutzer und der Anwendung beim Benutzer. 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 und vor allem der eigene Vertrieb haben allerdings oftmals gar kein großes Interesse, 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.

Unzureichende Einbeziehung der Benutzer führt zu nicht akzeptierten Produkten und zu unzufriedenen Kunden. 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. Das Gleiche gilt für den Projektmanager, der eine Liste führt, was er beim nächsten »offiziellen« Kundengespräch auf den Tisch bringen will. So wachsen die Divergenzen an, statt aufgelöst zu werden.

Eine andere Facette ist, dass es beim Kunden verschiedene Rollen gibt, aber nur dessen Einkauf repräsentiert ist. Auch das führt später zu einem bösen Erwachen, wenn man feststellen muss, dass der Einkauf primär auf formale Kriterien achtete, aber nicht auf Benutzbarkeit oder Effizienz des Produkts.

Zur Minderung dieses Risikos ist es zunächst wichtig, die Änderungsrate zu verfolgen. 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. Mancher Projektmanager und auch Kunde wird überrascht sein, wie früh im Projekt dies der Fall ist. 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.

Grundsätzlich sollten informierte und repräsentative Kundenvertreter regelmäßig konsultiert werden. Dann können offene Punkte zeitnah abgestimmt werden. Das ist nicht in allen Projekten möglich. Schließlich wollen viele Kunden ihre Zeit nicht unbedingt mit Projektarbeit verbringen. Daher müssen Anforderungsqualität, Mitarbeit des Kunden, Änderungsmanagement sowie Abstimmungsgespräche und Eskalationsmöglichkeiten vertraglich geregelt werden. Aber auch zu viel Kundenbeteiligung schafft Probleme. Viele Projekte scheitern, da sie wachsweich aufgesetzt werden. Man weiß, dass man nachher im Projekt sowieso eng zusammenarbeitet, und verschiebt Detailfragen auf später. Dann aber kann man kaum von Projekt sprechen, sondern eher von einem Versuchsballon. Und dass jene die Tendenz haben, zu platzen, das dürfte hinreichend bekannt sein. Machen Sie also im Vorfeld die Rollen der Kunden bei der Projektarbeit sehr deutlich und klären Sie die Erwartungen im Rahmen des Vertrags.

Diese drei Risiken sind nicht softwarespezifisch und unterstreichen die Anwendbarkeit des Requirements Engineering für jegliche Produktentwicklung – egal ob Software, IT, Medizin oder Maschinenbau. Jede Disziplin, in der Produkte oder Lösungen entwickelt werden, kämpft mit den immer wieder gleichen Herausforderungen. Anforderungen fehlen, sind falsch und ändern sich während des Projekts. Kein Wunder, dass die Entwicklung und Behandlung von Anforderungen branchenübergreifend adressiert wird. Abbildung 1–5 zeigt klassische Beispiele aus verschiedenen Projekten, die bei der Bearbeitung der Anforderungen kritische Fehler machten. Eines der ersten kommerziellen Düsenflugzeuge, die Comet 1, beispielsweise berücksichtigte bei den Fenstern die entstehenden Spannungen nicht – und stürzte sehr häufig ab. Die Tacoma-Narrows-Brücke hatte unzureichende Anforderungen und Lösungsmodelle bei der Berücksichti-

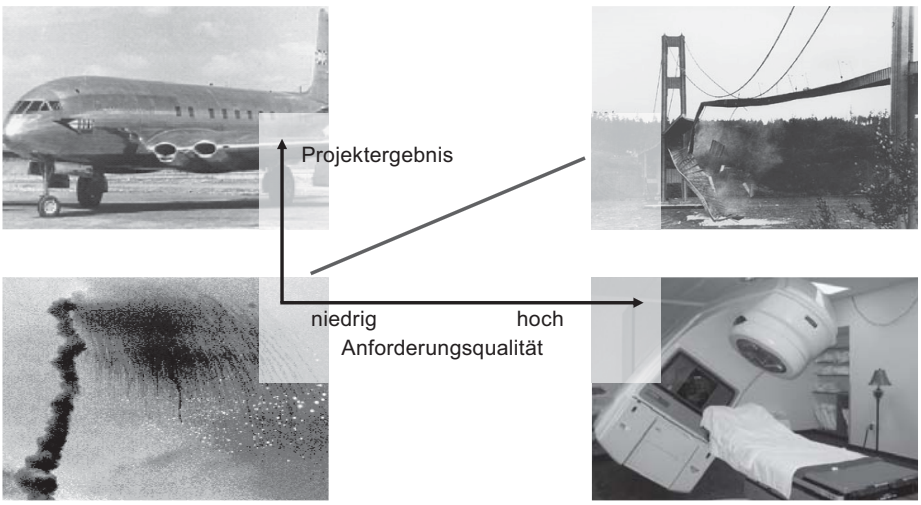


Abb. 1–5 *Erfolg hängt von der Qualität der Anforderungen ab.*

gung der Windlast – und stürzte ein. Das Therac-25-Bestrahlungsgerät spezifizierte die Benutzerschnittstelle fehlerhaft – und verursachte mehrere Todesfälle. Die Ariane 5 hatte Anforderungen außerhalb des Kontexts wiederverwendet – und stürzte auf dem Jungfernflug ab. Zahlen aus ganz unterschiedlichen Anwendungsbereichen unterstreichen es (Abb. 1–5, Mitte) [Griffith2001]: **Die Qualität der Anforderungen ist ein wesentlicher Erfolgsfaktor beim Projekterfolg.**

1.3 Wirtschaftlicher Nutzen und ROI

Die Einführung und systematische Umsetzung von RE erfordert Aufwand sowohl in der Entwicklung als auch an ihren Schnittstellen, also im Produktmanagement, Produktmarketing und Vertrieb. Häufig wird dieser Aufwand als zu hoch und zu zeitraubend gesehen, sodass die Anforderungen weiterhin ad hoc in das Projekt hineinpurzeln und dort bruchstückhaft und mit vielen Nacharbeiten umgesetzt werden, bis einmal mehr das Projekt seine Ziele verfehlt oder abgebrochen werden muss. Aus unserer Beratungspraxis kennen wir das Dilemma: Verbesserungen in Methodik, Prozessen, Ausbildung und Werkzeugen werden nicht angegangen, da der Anfangsaufwand, um diese Verbesserungen anzustoßen, als zu hoch betrachtet wird.

Daher wollen wir in diesem Abschnitt die **Nutzen eines systematischen RE quantitativ unterstreichen** und vor allem konkrete Hinweise geben, wie Sie in Ihrer eigenen Umgebung den Nachweis führen können, dass sich der Aufwand für das RE lohnt. Eine weitaus umfangreichere Darstellung der ROI-Konzepte und zugrunde liegenden Projektaufwandsdaten findet sich in [Ebert2007a].

Systematisches RE bringt klare Vorteile für die Softwareentwicklung (Abb. 1–6) [Ebert2007a, Standish2003, IAG2008, Standish2009]:

- **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.
- **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 3–6 % 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.

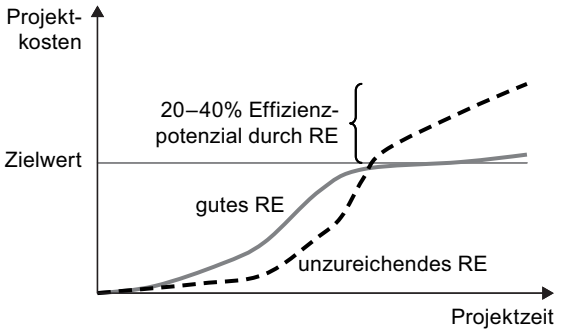


Abb. 1–6 *Effizienzpotenziale durch besseres Requirements Engineering*

Die zwei am häufigsten zitierten empirischen Studien untersuchten in Hunderten von Projekten den Zusammenhang von RE und Projekterfolg. Die Termintreue von Projekten wurde in beiden Studien als primärer Erfolgsfaktor betrachtet, denn in den meisten Branchen geht es heute aufgrund von Wettbewerb und Time-to-Profit darum, Softwarelösungen präzise zum gewünschten Termin zu liefern [Hamel2007].

Die umfassendste Untersuchung zum Nutzen von RE kommt von Alcatel-Lucent. 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 (Abb. 1–7, 4 Techniken eingesetzt). Wurde einer oder mehrere dieser Faktoren vernachlässigt, führte das sofort zu Terminverzug (Abb. 1–7, 0–3 Techniken eingesetzt).

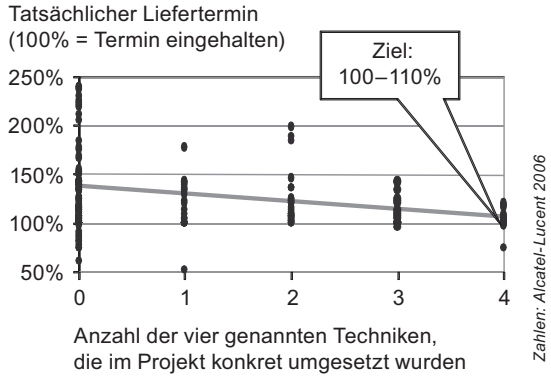


Abb. 1–7 *Der gleichzeitige Einsatz von vier wesentlichen Techniken des RE (Kernteam, Lebenszyklus, Transparenz, gemeinsame Analyse) verbessert den Projekterfolg.*

- 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 und Vereinbarungen
- Transparenz aller Projektvereinbarungen (z.B. Anforderungen) im Intranet
- Gemeinsame Anforderungsanalyse durch das Kernteam mit Produktmanagement, Marketing, Entwicklung und Produktion

Eine weitere umfassende Studie zum Nutzen von RE in Entwicklungsprojekten kommt von der NASA (Abb. 1–8) [Forsberg1997]. Der Zusammenhang ist auch in dieser Studie offensichtlich. Wenn die NASA in ihre Projekte weniger als 5 % Vorbereitungsaufwand (insbesondere auch Anforderungen) investiert, kommt es zu starken Verzögerungen. Bei einem Anteil von 10–20 % am gesamten Projektaufwand reduzieren sich die Terminverzüge auf unter 30 %.

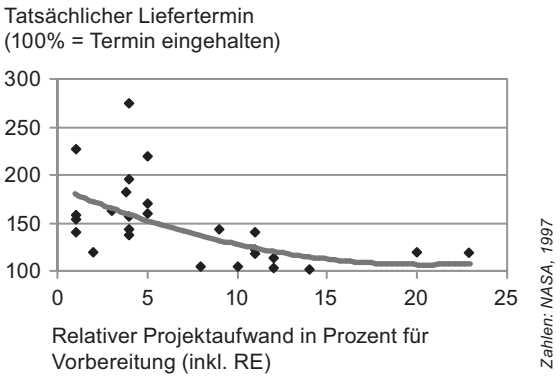


Abb. 1–8 *Termineinhaltung in Abhängigkeit vom Aufwand für die Projektvorbereitung*

Eine dritte longitudinale Feldstudie zum Nutzen von RE in IT-Projekten kommt ebenfalls von der NASA (Abb. 1–9) [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.

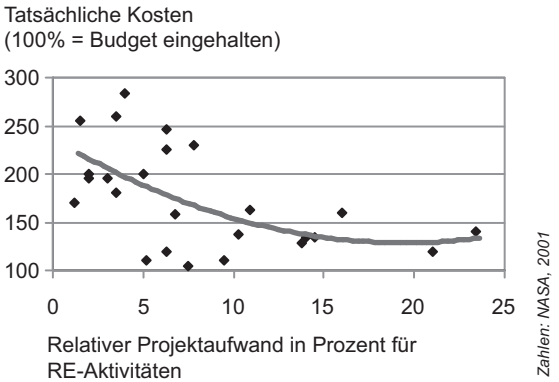


Abb. 1–9 *Kosteneinhaltung in Abhängigkeit vom Aufwand für RE*

Das deckt sich auch mit einer Studie von Borland mit 348 IT-Verantwortlichen in den USA [Borland2006]. Über 90 % der Antwortenden unterstrich, dass eine Verbesserung der RE-Prozesse einen klaren Wettbewerbsvorteil bringen würde. Mehr als die Hälfte der Antwortenden erklärte, dass mit einem besseren RE eine Senkung der Entwicklungskosten um 30 % möglich wäre.

Der Nutzen eines guten RE kann an verschiedenen Faktoren festgemacht werden. Im Folgenden geben wir Anhaltspunkte für die eigene Nutzenrechnung, die wir aus verschiedenen Kundenprojekten in unterschiedlichen Branchen gewonnen haben [Ebert2007a, Standish2003, Stevens1998, Leffingwell1997].

- Produktivitätsverbesserung. Typischerweise werden 30–50 % des Entwicklungsaufwands für Fehlerbehebung und nicht entwicklungsbezogene Aktivitäten eingesetzt. Die Hälfte der Fehler kommt direkt aus unzureichenden Anforderungen und unkontrollierten Änderungen. Im Systemtest sind es 80 % der Fehler, die aus unvollständigen (31 %) oder falschen (49 %) Anforderungen resultieren.
- Verbesserte Projektplanung und Ressourceneinteilung, weniger Verzögerungen vor Projektstart, eine schnellere Anlaufphase sowie Termintreue aufgrund von bekannten Anforderungen und klaren Verantwortungen im Projektteam und im Vertrieb. Mehr Aufwand für Entwicklung und konsequente Umsetzung und Test der Anforderungen schafft eine Verbesserung der Termintreue und reduziert Verzögerungen auf unter 20 %.
- Kürzere Durchlaufzeiten durch Fokus auf jene Aktivitäten und Inhalte, die Wert schaffen. Knapp die Hälfte aller Funktionen eines Softwaresystems werden nicht genutzt. Das gilt sowohl in der IT wie auch in technischen Produkten. Weniger Anforderungen reduzieren die Komplexität und machen damit die Projekte verlässlicher sowie die Qualität besser.
- Weniger Nacharbeiten von inkonsistenten Anforderungen durch Einflussanalyse bei sich ändernden Anforderungen und Nachverfolgbarkeit zu den

betroffenen Arbeitsergebnissen. Werden die Anforderungen so umgesetzt, wie sie vom Kunden oder Benutzer beschrieben werden, führt das zu Nacharbeiten, die im Schnitt 45 % des Projektaufwands ausmachen.
- Wiederverwendung von Anforderungen und davon abgeleiteten Arbeitsergebnissen (z. B. Mechanismen zur Systemsicherheit).
- Bessere Kundenzufriedenheit durch konsistentes Verständnis über die wirklichen Anforderungen.

Einige dieser Faktoren, wie Termintreue oder weniger Nacharbeiten, schaffen einen unmittelbar greifbaren Nutzen. Andere, wie beispielsweise die Kundenzufriedenheit, sind eher opportunistisch und werden in Form nachhaltig guter Kundenbeziehungen und weiterer Projekte greifbar. Insgesamt zeigen unsere Erfahrungen, dass eine Verdoppelung des Aufwands für RE hin zu 10 % des Projektaufwands in den Bereichen Methodik, Prozesse, Training und Werkzeugunterstützung konkret realisierbare Projektnutzen von über 20 % schafft. Das ist ein ROI von mehr als 4, und damit sind nur die direkt messbaren Vorteile berücksichtigt.

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;
- haben Sie einen ersten Einblick, 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 Buchs. 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. Kapitel 5 beschreibt die relevanten Analyse- und Modellierungstechniken, wobei wir ein einheitliches Beispiel einsetzen, um die Unterschiede und Gemeinsamkeiten besser zeigen zu können. Kapitel 6 vertieft die Prüfung der Anforderungen. Häufig werden die falschen Anforderungen rea-

lisiert oder Fehler in der Umsetzung gemacht. Wir zeigen hier Techniken zu Reviews, Prüfungen und konkrete Checklisten, um die Anforderungsqualität zu verbessern. Kapitel 7 greift eine oft vernachlässigte Aktivität im Projekt auf, nämlich die Abstimmung und Vereinbarung von Anforderungen. Das Kapitel betrachtet auch rechtliche Zusammenhänge, beispielsweise Gewährleistungs- und Haftungsfragen. Schließlich beschreibt Kapitel 8 die Methoden der Anforderungskontrolle und -verwaltung.

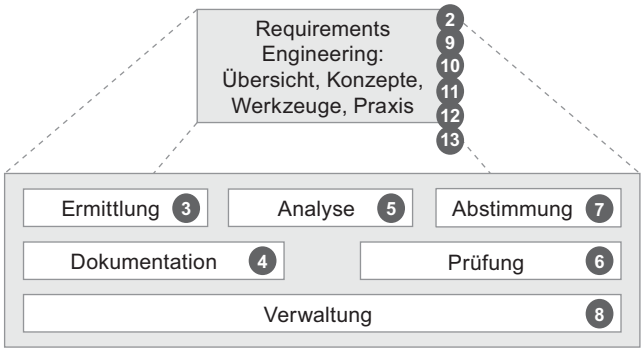


Abb. 1–10 *Zuordnung der Buchkapitel zu den Themen des RE (schwarze Kreise sind Kapitelnummern)*

Die Aufgaben und Verantwortungen und ihr Zusammenspiel werden in Kapitel 9 vertieft. Dort gehen wir auch auf die Zertifizierung nach IREB ein. Kapitel 10 gibt eine Übersicht zu Prozessen, Lebenszyklusmodellen, Standards und Normen, die zur Anwendung kommen. Requirements-Werkzeuge werden in Kapitel 11 charakterisiert und bewertet. Wir zeigen namhafte RE-Werkzeuge ganz praxisnah in unterschiedlichen Szenarien. Kapitel 12 zeigt beispielhaft, wie die Themen der Kapitel 3 bis 10 ineinandergreifen. Damit sehen Sie den praktischen Nutzen und die Abhängigkeiten der einzelnen Schritte im RE. Das abschließende Kapitel 13 fasst den Stand der Technik nochmals 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. Diese URLs können sich natürlich ändern, aber die beschriebene Auswahl hat sich in den vergangenen Jahren als recht stabil erwiesen. Alle Begriffe, die im Buch definiert werden oder auf deren Definitionen zurückgegriffen wird, sind im Glossar am Ende des Buchs 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 also nicht um die Art von Verständnisfragen, wie Sie es aus Lehrbüchern kennen, sondern eher um Fragen, die

Ihren eigenen Horizont öffnen, um das gerade konsumierte Wissen zu verdauen und genau das abzuleiten, was Ihre eigenen Projekte benötigen.

Wenn Sie als **Softwareentwickler** 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. Requirements Engineering klingt zwar nach Formalismus und wird häufig in einen Topf geworfen mit Aufwandschätzung, Modellierung und Projektmanagement. Das Buch versucht, die Themen sauber zu trennen und trotzdem einen Querbezug zu schaffen.

Selbstständige Entwickler und **Freelancer** lernen praxisnah die wichtigsten Grundlagen, die gerade in kleinen Projekten eine große – oft überlebensnotwendige – Rolle spielen. Da das Buch sehr klar zwischen Prozessen und Werkzeugen trennt, werden Sie lernen, wie Sie mit einfachsten Mitteln eine solide Basis Ihrer Anforderungen halten und verwalten. Die angesprochenen Prinzipien sind skalierbar und nicht nur für große Projekte relevant. 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. Wenn Sie einem Kunden einen Termin auf der Basis von gegenseitig vereinbarten Anforderungen versprochen haben, werden Sie es häufig erleben, dass kurze Zeit später die ersten Änderungen kommen. Dies ist normal, muss aber konsequent abgefangen werden. Wenn Sie einmal damit beginnen, solche Änderungen auf Zuruf zu akzeptieren, weil sie ja »sehr klein« sind, haben Sie eine Türe aufgemacht, die Sie nie mehr bei diesem Kunden schließen können. Auf welcher Basis würden Sie argumentieren, dass manche Änderungen »sehr klein« und andere »nicht mehr so klein« sind?

Besser ist es, von vornherein einen Prozess zu vereinbaren, der besagt, dass alle Anforderungen in einer Liste gepflegt werden. Falls es zu Änderungen kommt, werden diese analysiert und dann vereinbart. Änderungen können Auswirkungen auf Termine und Kosten haben. Falls Sie zu einem Festpreis arbeiten, wird es Ihnen manchmal sogar helfen, eine späte Änderung noch aufzunehmen, da Sie damit Termine und den Preis nochmals beeinflussen können. Es gibt keine Änderung ohne vorherige Abschätzung und formalisierte Vereinbarung. Danach finden sich die Anforderungen auf Ihrer Liste wieder. Das geht mit wenig Aufwand, und die Kunden werden Ihre Professionalität schätzen.

Das Buch empfiehlt wirksame Prinzipien und Vorgehensweisen, die sich an der Praxis orientieren.

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. Dieses Buch bietet eine Menge an konkreten Tipps aus dem Projektalltag, die der