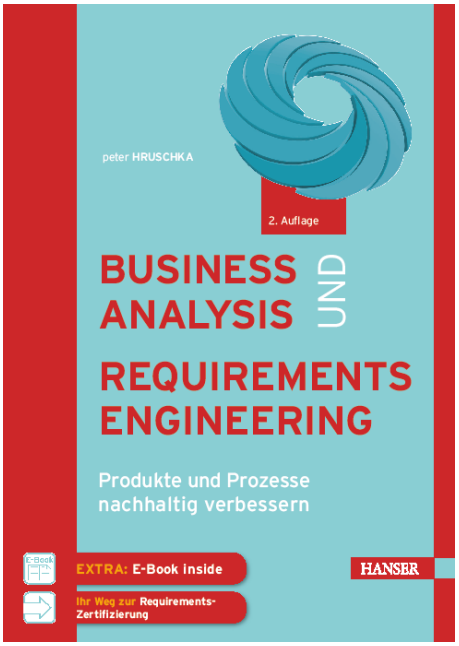
HANSER



Leseprobe

zu

Business Analysis und Requirements Engineering

von Peter Hruschka

ISBN (Buch): 978-3-446-45589-4
ISBN (E-Book): 978-3-446-45734-8

Weitere Informationen und Bestellungen unter
https://www.hanser-fachbuch.de/buch/Business+Analysis+und+Requirements+Engineering/9783446455894

sowie im Buchhandel

Inhalt

Vorwort

Als ich 1976 direkt nach dem Studium meinen ersten Arbeitsvertrag in Händen hielt, war ich überrascht. Als Berufsbezeichnung war „Systemanalytiker“ eingetragen. Ich war deshalb überrascht, weil ich „Programmierer“ erwartet hatte. Denn was wir an der Technischen Universität Wien im Informatikstudium gelernt hatten, war Programmieren, formale Sprachen, Mathematik und Logik, Hardware- und Elektrotechnikgrundlagen und vieles andere, aber keine Systemanalyse. Die Berufsbezeichnung „Systemanalytiker“ klang jedoch in meinen Ohren gut und ich fühlte mich aufgewertet. Ich habe mir jedoch vorgenommen, im Lauf meines Berufslebens herauszubekommen, was ein Systemanalytiker so tut.

In der Zwischenzeit haben wir die Berufsbezeichnung „Systemanalytiker“ oft durch „Business Analyst“ oder durch „Requirements Engineer“ ersetzt. Im Kern geht es immer noch um das gleiche Thema: herauszubekommen, wo unsere heutigen Produkte und Prozesse Schwachstellen haben, was Kunden und Nutzer wirklich brauchen, wie Prozesse durch neue Ideen effektiver gemacht werden können oder was IT-Systeme wirklich leisten sollten, um das Business besser zu unterstützen. Und das, was wir herausbekommen haben, wollen wir effektiv miteinander besprechen und verhandeln können, bevor wir es in Lösungen umsetzen (lassen), Produkte erstellen oder IT-Systeme bauen.

Jedem Vorhaben oder Projekt, d. h. jeder Korrektur, Verbesserung, Erweiterung, Änderung oder Diversifizierung Ihrer Produkte oder Ihrer betrieblichen Prozesse sollte ein klares Verständnis des Ist-Zustands, eine gründliche Analyse von Schwachstellen und Risiken und die Identifizierung von Verbesserungsoptionen vorausgehen.

Das ist das Thema dieses Buchs. Sie können es gerne Business Analysis, Requirements Engineering oder auch Systemanalyse nennen. Das sind nur verschiedene „Jahrgänge“ von Worten, die das gleiche Thema behandeln: analysieren und artikulieren, welche Anforderungen wir haben, um Produkte und Prozesse innovativ und nachhaltig zu verbessern.

Wie Sie alle bin auch ich beruflich und privat Nutzer von sehr vielen Softwaresystemen und Produkten. Das beginnt beim Mobiltelefon, bei den Programmen auf meinem Rechner, mit denen ich Kursfolien gestalte oder Bücher und Artikel schreibe; die Systeme, die die Abrechnungen erstellen, die ich jeden Monat erhalte und bezahlen muss, aber auch viele andere technische Systeme im Auto, mit denen ich täglich unterwegs bin, oder in meinem Kaffeevollautomat, um mich mit verschiedenen Spezialitäten bei Laune zu halten. Mit diesem Buch möchte ich ganz einfach meinen Beitrag dazu leisten, dass diese Systeme und Produkte den Benutzern mehr Freude als Ärger bereiten.

Aachen, Januar 2014 *Peter Hruschka*

Vorwort zur zweiten Auflage

Vier Jahre sind vergangen seit der ersten Auflage. Business Analysis und Requirements Engineering hat sich durch die Verbreitung agiler Ansätze geändert – und doch auch nicht. Einiges hat neue Namen erhalten, einiges hat sich an den bevorzugten Notationen geändert und es sind auch einige neue Tools populär geworden, um Anforderungen zu erfassen und zu verwalten. Deshalb habe ich in dieser zweiten Auflage die Begriffe Notationen und Tools mit aufgenommen, über die man heute spricht, wenn man die Rolle Business Analyst, Product Owner oder Requirements Engineer im Rahmen des Unternehmens ausfüllen soll.

Vieles ist aber seit vielen Jahrzehnten einfach immer noch wahr und wichtig. Noch immer wollen wir als Analytiker die Probleme und Geschäftsideen verstehen und kommunizieren, bevor wir an Lösungen gehen. Nur ist unsere Welt immer schnelllebiger geworden: Wir wollen die Lösungen schneller und öfter. Daran müssen sich auch die Analyseprozesse anpassen.

Eine meiner Hauptbotschaften, die sich im Titel „Business Analysis und Requirements Engineering“ ausdrückt, ist immer noch nicht ganz in der täglichen Praxis angekommen: die möglichst reibungslose Zusammenarbeit zwischen Fachabteilungen und IT-Abteilungen, zwischen Auftraggebern und Auftragnehmern, zwischen denen, die das Business verbessern wollen, und denen, die IT-Systeme dafür entwickeln und liefern können.

Aber ich gebe die Hoffnung nicht auf, die Grenze zwischen Firmen oder Abteilungen weiter abzubauen: Denn, egal, auf welcher Seite Sie angesiedelt sind, wir ziehen doch (hoffentlich) am gleichen Strang (und auch am gleichen Ende des Strangs) und wollen, dass IT-Systeme unser Leben immer weiter erleichtern.

Aachen, Februar 2019 *Peter Hruschka*

1 Probleme, Ziele, Ideen und Visionen

Wenn man ein Projekt aufsetzt, dann hat man entweder im heutigen geschäftlichen Alltag ein Problem oder eine Schwachstelle identifiziert, die man beseitigen möchte, oder man hat eine Idee oder eine Vision, wie ein Produkt oder ein Prozess etwas besser, schneller, effizienter, billiger etc. werden könnte. Nun ja, manchmal hat man vielleicht auch einfach noch Geld übrig und möchte es für irgendetwas ausgeben. Das alles sind Ausgangspunkte für Business Analysis und Requirements Engineering.

Im ersten Teil wollen wir uns mit einigen Grundbegriffen und einer Einführung in das Thema Business Analysis und Requirements Engineering auseinandersetzen. Um Sie zu motivieren, betrachten wir einige quantitative und qualitative Gründe für den Einsatz derartiger Methoden. Wir erwähnen aber auch Ursachen, warum es trotz aller guten Gründe manchmal nicht ernsthaft betrieben wird.

Danach werden wir die drei Säulen erfolgreicher Projekte betrachten, von denen Business Analysis und Requirements Engineering eine ist.

Wir besprechen Definitionen: Was sind überhaupt Requirements oder auf Deutsch, was sind Anforderungen? Und was bedeutet Business Analysis und was Requirements Engineering? Was gehört dazu, was gehört nicht dazu?

Dann diskutieren wir die Berufsbilder Business Analyst, Requirements Engineer und Product Owner. Was haben diese Personen zu tun? Was müssen sie machen? Aber auch: Welche Fähigkeiten sollten sie mitbringen, welche Soft-Skills und welche methodischen Fähigkeiten?

Anschließend besprechen wir die drei wichtigen Arten von Anforderungen. Und zum Schluss des Kapitels betrachten wir verschiedene Vorgehensweisen, wie man unter unterschiedlichen Randbedingungen zu Anforderungen kommen kann – vom Wasserfallmodell bis zu sehr agilen Vorgehensweisen.

1.1 Wovon sprechen wir?

In immer stärkerem Maß unterstützen heute technische Produkte oder IT-Systeme unser Leben. Sie helfen uns, Ziele schneller und effektiver zu erreichen, erleichtern oft vormals manuelle Tätigkeiten oder ermöglichen Dinge, von denen wir vor einiger Zeit nicht einmal zu träumen gewagt haben. Wir nehmen Geldtransaktionen von zuhause aus vor, wir überlassen das Schalten der Gänge und die Traktionskontrolle unseres Autos, wir telefonieren und

surfen mobil fast überall, wir nutzen medizinische Geräte zur Verbesserung der Diagnostik, wir gestalten Kundenprozesse benutzerfreundlicher und effektiver ...

Dazu ist es natürlich notwendig, unsere Probleme, Wünsche, Träume und Visionen so zu durchleuchten und aufzubereiten, dass die IT-Entwickler und Produktentwickler dafür geeignete Lösungen erstellen können oder bestehende Produkte und Prozesse nachhaltig verbessern können.

Ohne klare Kenntnis des Ist-Zustands und ohne klare Ideen, was wir erreichen wollen, verzetteln wir uns sonst in Lösungen, die an den Marktbedürfnissen oder den Bedürfnissen einer Organisation vorbeigehen und mühevoll und aufwendig nachgebessert werden müssen. Deshalb brauchen wir Methoden und Techniken, um herauszubekommen, was das Geschäft wirklich braucht oder was Produkte leisten sollen. Das nennen wir Business Analysis oder Requirements Engineering.

1.2 Quantitative Gründe

Wir können sicherlich sehr viele quantitative und qualitative Argumente anführen, warum wir uns mit dem Thema auseinandersetzen sollen. Sprechen wir zuerst mal über die Manager. Die wollen immer Zahlen hören. Sehr bekannt ist die Studie der Standish-Group über Fehler in Softwareprojekten. Software, die nie fertig wird, die viel zu teuer ist, die zu spät ausgeliefert wird und die weit überteuert ist. Oder sogar total scheitert. In dieser Studie, die jährlich erneuert wird, hat man die Top-10-Risiken, die zehn größten Risiken, in Projekten festgehalten. In Bild 1.1 können Sie sehen, dass sich vier von diesen Top-10-Risiken mit dem Thema unseres Buchs beschäftigen, mit den Anforderungen.

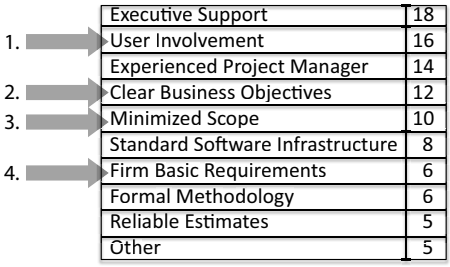


Bild 1.1
Die Top-10-Risiken in Projekten gemäß Standish-Studie

Erstens die wirkliche Beteiligung der Benutzer, die das System haben wollen, an dem Prozess (heute eher Stakeholder Involvement genannt – mehr dazu in Abschnitt 2.4); zweitens klare geschäftliche Zielsetzungen – heute oft auch Vision genannt; drittens eine eindeutige Festlegung des Scope und viertens eine vernünftige Formulierung der Anforderungen. Die Zahlen hinten den Punkten sind Gewichtungen. Zusammengenommen machen diese vier Themen 44 von 100 Punkten aus, fast den halben Projekterfolg!

Aber ich glaube, noch deutlicher hat es der Großmeister der Zahlen, Capers Jones, ausgedrückt. Er hat auch untersucht, wie viele Fehler Teams machen, die keine guten Analysemethoden,

keine guten Requirements-Methoden oder andere Qualitätsmethoden für Anforderungen haben [Jon08]. Sein Ergebnis: Teams machen 0,23 Fehler pro Function Point. Sie brauchen jetzt nicht genau zu verstehen, was ein Function Point ist. Vergleichen Sie den Wert nur mit dem Ergebnis beim Einsatz guter Analysemethoden.

Mit guten Requirements-Methoden kann man die Fehlerrate von 0,23 auf 0,08 pro Function Point reduzieren. Das ist ein Drittel davon. Dreimal weniger Fehler. Und sicherlich kennen die meisten von Ihnen die Statistiken, was es uns kostet, Fehler erst im Betrieb eines Produkts oder eines Systems zu korrigieren. Es ist aufwendiger, es ist viel teurer, als wenn wir Fehler zu dem Zeitpunkt finden, wo sie gemacht werden, nämlich beim Beschreiben des Problems. Manche Chefs fangen an zuzuhören, wenn man sagt, dass drei Mal weniger Fehler gemacht werden und drei Mal weniger Kosten hinterher für die Korrektur dieser Fehler aufgewendet werden müssen. Das sind schlagende betriebswirtschaftliche Argumente für den Einsatz guter Analysemethoden.

1.3 Qualitative Gründe

Aber es gibt auch qualitative Argumente, warum wir Business Analysis und Requirements Engineering betreiben sollten.

Falsche Anforderungen → falsche Systeme

Der Hauptgrund ist: Falsch verstandene oder nicht korrekte Anforderungen führen zu falschen Systemen. Wenn also die Anforderungen ungenügend klar sind, erhalten wir die falsche Software und Produkte und meistens merken wir das erst im Betrieb. Also, ich brauche gute Anforderungen, um zu guten Systemen zu kommen.

Implizite Anforderungen

Ein zweiter Punkt, warum das Thema so wichtig ist, sind die vielen impliziten Annahmen. Das ist ohnehin klar, da brauchen wir nicht drüber sprechen, das weiß doch jeder. Naja, scheinbar doch nicht. Denn wenn es nicht vorher klar gesagt wurde, kommen Designer und Programmierer auf ihre eigenen Ideen und entwickeln wieder vorbei an den Bedürfnissen des Markts.

Sehr viele Anforderungen sind impliziter Natur und es ist gar nicht leicht, diese aus den entsprechenden Personen herauszulocken. Wenn man es aber nicht tut, bleibt es eine implizite Annahme und jemand trifft Entscheidungen.

Interpretierbare Anforderungen

Selbst wenn Anforderungen explizit gemacht werden, ist der Projekterfolg noch nicht gewährleistet. Denn viele Anforderungen sind interpretierbar. Der Auftraggeber hält eine Aussage für absolut klar und verständlich; der Auftragnehmer interpretiert die Aussage jedoch ganz anders – und schon wieder haben wir unter Umständen eine falsche Lösung.

Requirements Creep

Zuletzt wollen wir noch auf ein Problem hinweisen, das uns oft betrifft. Anforderungen ändern sich im laufenden Projekt. Wir verfolgen ein bewegliches Ziel. Saubere Methoden und Vorgehensweisen dieses „Requirements Creep“ sollten Ihnen daher einiges wert sein.

1.4 Warum macht es nicht jeder richtig?

Man könnte sich jetzt die Frage stellen, warum es nicht jeder richtig macht, wenn wir doch wissen, dass schlampige Analyse sehr viele Fehler produziert und sehr teuer ist. Ich glaube, der Hauptgrund ist: Wir haben Kommunikationsprobleme miteinander. Es ist nicht leicht, Anforderungen in einer Weise auszudrücken, die alle Beteiligten verstehen. Das wird umso schwieriger, je weltweit verteilter unsere Projekte sind, wenn wir nicht mehr die gleiche Sprache sprechen oder in anderen Kulturen groß geworden sind – dann ist auch das gegenseitige Verständnis nicht so gut.

Das ist aber nur ein Argument. Ein weiteres Argument, warum es nicht jeder richtig macht, ist: Wir wollen einfach Geld sparen. Gute Analyse kostet Zeit und Aufwand. Zeit und Aufwand, den man vielleicht erst ersetzt bekommt, indem man keine Nachbesserungen machen muss. Das ist aber *nach* Abwicklung des Projekts. Mancher Projektleiter ist heilfroh, wenn er seinen Abnahmetest bestanden hat und fertig ist, und nach ihm die Sintflut. Was es dann noch kostet, die Fehler auszubessern, ist unter Umständen nicht mehr in der Verantwortung eines Projektleiters. Und deshalb versucht man, möglichst wenig Analyse zu machen, um möglichst schnell zu Lösungen zu kommen, was aber vielleicht dann teure Nachbesserungen erfordert.

Vielleicht noch ein weiteres Argument: Es geht mitunter nicht nur um das Sparen von Geld, sondern manchmal auch von Zeit. Wir wollen einfach rasche Lösungen und nehmen uns nicht die Zeit, vorher festzulegen, was es sein soll, das wir hier produzieren. Das ist aber ein falsch verstandenes Zeit sparen, denn die Zeit brauchen Sie hinterher mehrfach wieder.

Aus vielen dieser Gründe werden Business Analysis und Requirements Engineering nicht von allen Leuten ernst genommen, aber in den letzten 20 Jahren haben viele Unternehmen erkannt, dass es ein sehr wichtiges Thema ist, haben viel in dieses Thema investiert und es ist deutlich besser geworden.

Natürlich sind auch die Methoden in den letzten 35 Jahren deutlich besser geworden. Als ich angefangen habe zu arbeiten, gegen Ende der 1970er-Jahre, gab es nicht viel an Analysemethodik. In den letzten 35 Jahren ist hier eine Menge dazugekommen an Verständnis, wie man systematisch Geschäftsprozesse untersucht und Anforderungen erheben kann, wie man sie dokumentieren, prüfen und im Lauf der Zeit verwalten kann.

1.5 Standardisierung und Zertifizierung

Business Analyst und Requirements Engineer sind heute keine geschützten Berufsbezeichnungen. Jeder, der möchte, kann sich diese Titel selbst verleihen, kann sich Systemanalytiker nennen oder Business Analyst oder auch Requirements Engineer, ohne dafür einen bestimmten Nachweis zu erbringen.

Zur Verbesserung dieser Situation hat sich 2006 eine Gruppe von Fachleuten zusammengeschlossen, um das IREB zu gründen, das International Requirements Engineering Board (*www.ireb.org*).

Die Hauptaufgabe dieses Boards war, das Wissen, das man zur Ausübung dieses Berufsbilds benötigt, zusammenzuschreiben und einen Lehrplan zu formulieren, zunächst für Basiskenntnisse, die man unbedingt nachweisen sollte, um den Beruf gut ausüben zu können. Neben diesem Lehrplan über das Basiswissen wurden Fragenkataloge erstellt, mit denen man nachweisen kann, das Wissen wirklich erworben zu haben. Sie können sich dieses Wissen bei unabhängigen Stellen zertifizieren lassen.

Dieses Buch ist IREB-konform. Es bietet alle Grundlagen, die man kennen muss, um die Prüfung zum CPRE (Certified Professional for Requirements Engineering) auch erfolgreich bestehen zu können. Bis 2018 wurden mehr als 50 000 Personen nach diesem Programm weltweit zertifiziert, in den letzten Jahren mehr als 7 500 pro Jahr. Der Verein bietet seit einigen Jahren auch entsprechende Aufbaumodule an, um das Basiswissen zu vertiefen. Das Buch deckt auch große Teile von Advanced Level Requirements Modeling und RE@Agile ab (Bild 1.2).

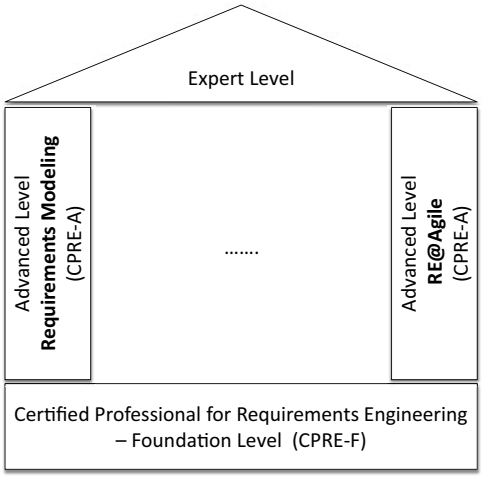


Bild 1.2
Die drei Ebenen des IREB-Zertifizierungsprogramms

Unabhängig davon ist auch in Kanada ein Programm entstanden: das IIBA (International Institute of Business Analysis, *www.iiba.org*), das ähnliche Ziele wie das IREB verfolgt. In diesem Buch lernen Sie auch alle relevanten Teile des BABOK (Business Analysis Book of Knowledge) kennen. Auch IIBA bietet ein mehrstufiges Zertifizierungsprogramm an.

1.6 Drei Säulen erfolgreicher Projekte

Beginnen wir mit einer Gesamtbetrachtung von Projekten oder Produktentwicklungsvorhaben. Wie passt unser Thema „Business Analysis & Requirements Engineering“ in den gesamten Lebenszyklus hinein? Vielleicht hat Ihre Firma, Ihr Unternehmen, Ihre Organisation ein vorgeschriebenes Vorgehensmodell. Besonders populär im Behördenumfeld ist zum Beispiel das V-Modell, das von einigen Ministerien für staatliche Projekte vorgeschrieben wird. Ein anderes populäres Modell war der Rational Unified Process, heutzutage von IBM Rational vertreten. Wenn Sie nach einem dieser Vorgehensmodelle arbeiten müssen, haben Sie sicherlich festgestellt, dass Sie sehr, sehr viele Tätigkeiten machen müssen, viele Dokumente erzeugen und verschiedenste Rollen besetzen. Die Zahl der Dokumente bei diesen Vorgehensmodellen ist dreistellig; es sind auch zirka 150 – 170 getrennte Tätigkeiten.

Ich möchte versuchen, Ihnen ein einfacheres Weltbild nahezulegen, das leichter zu merken ist und aus nur drei Grundpfeilern besteht (vgl. Bild 1.3). Einer von diesen dreien ist, herauszufinden, welches Problem der Auftraggeber, ein Kunde oder eine Organisation hat; das Problem zu definieren, das Problem zu durchleuchten und zu verstehen. Das ist unser Thema Business Analysis und Requirements Engineering. Vielleicht sagen Sie auch Geschäftsprozessanalyse dazu; vielleicht verwenden Sie auch das Wort Systemanalyse. Wir meinen auf jeden Fall: herauszubekommen, was unser Geschäft oder unsere Kunden brauchen bzw. was unsere Anwender von IT-Systemen erwarten oder mit den Produkten machen wollen.

Bild 1.3
Drei Säulen für erfolgreiche Projekte

Die zweite Säule meines einfachen Weltbilds ist es, die Anforderungen in eine Lösung umzusetzen. Wenn diese Lösung eine IT-Lösung ist, dann gehört dazu der Aufbau einer Softwarearchitektur, vielleicht auch einer Hardwarearchitektur und die Entwicklung, der Test und die Inbetriebnahme von Software. Wenn die Lösungen organisatorisch bewerkstelligt werden sollen, dann umfasst dieser Block die detaillierten Festlegungen und die Umsetzung in Aufbau- und Ablauforganisation. Kurz gesagt: eine komplette Lösung für ein vorgegebenes Problem erstellen.

Und die dritte Säule in meinem einfachen Weltbild ist Projektmanagement: das richtige Team zu haben; diesem Team gute Ziele zu geben, vernünftige Pläne aufzustellen, Zwischenziele festzulegen, deren Erreichung man prüfen kann, und das Team erfolgreich über diese Zwischenziele zum Ziel zu führen.

Die Tätigkeiten in den drei Säulen werden heutzutage hochgradig iterativ durchgeführt. Ständig werden Sie gleichzeitig Probleme analysieren, Lösungen designen und Ihre Vorhaben oder Produktentwicklungen managen.

Mit diesen drei Säulen können Sie Projekte erfolgreich abwickeln, Ihr Business effektiver gestalten und Ihre Produkte erfolgreicher machen. Dieses Buch handelt hauptsächlich von der ersten dieser drei Säulen: von Business Analysis und Requirements Engineering. Wir werden aber auch die Schnittstellen zwischen diesen beiden Themen und der Umsetzung in Lösungen sowie die Schnittstellen zwischen Business Analysis/Requirements Engineering und Projektmanagement ausführlich behandeln.

1.7 Definition: Business Analysis und Requirements Engineering

Bevor wir uns mit den Inhalten von Business Analysis und Requirements Engineering näher beschäftigen, sollten wir uns auf Definitionen einigen. Was also verbirgt sich hinter den beiden Begriffen?

Das International Requirements Engineering Board [IREB], dessen Programm diesem Buch zugrunde liegt, hat die Gesamtheit des Themas als Requirements Engineering bezeichnet. Die beiden Hauptteile, die dazugehören, sind einerseits die Requirements-Analyse – von vielen auch als Systemanalyse bezeichnet – und andererseits Requirements-Management, d. h., wenn ich einmal Anforderungen habe, diese auch zu managen, zu verwalten und zu pflegen. Zu den Unterpunkten der Requirements-Analyse gehört einerseits, Requirements aus Kunden herauszulocken, und andererseits, diese auch zu Papier zu bringen – also Requirements ermitteln und spezifizieren.

Sie sehen an dem einfachen Mengendiagramm in Bild 1.4 die drei Teile, aus denen dieses Gebiet besteht:

- Anforderungen ermitteln (oder herauslocken),
- Anforderungen spezifizieren (auf irgendeine Art skizzieren, niederschreiben oder zeichnen) und
- Anforderungen managen (verwalten, pflegen, versionieren, priorisieren).

Die ersten beiden davon werden oft auch als „Requirements-Analyse“ oder „Systemanalyse“ zusammengefasst. Welche Wörter Sie für die Summe aller dieser Tätigkeiten verwenden, überlasse ich Ihrer Entscheidung. Das IREB hat sich für „Requirements Engineering“ als Oberbegriff entschieden; ob Sie das in Ihrem Umfeld als Requirements Engineering bezeichnen oder Requirements-Management als Oberbegriff verwenden oder vielleicht Problemanalyse oder Business-Analyse, überlasse ich Ihnen.

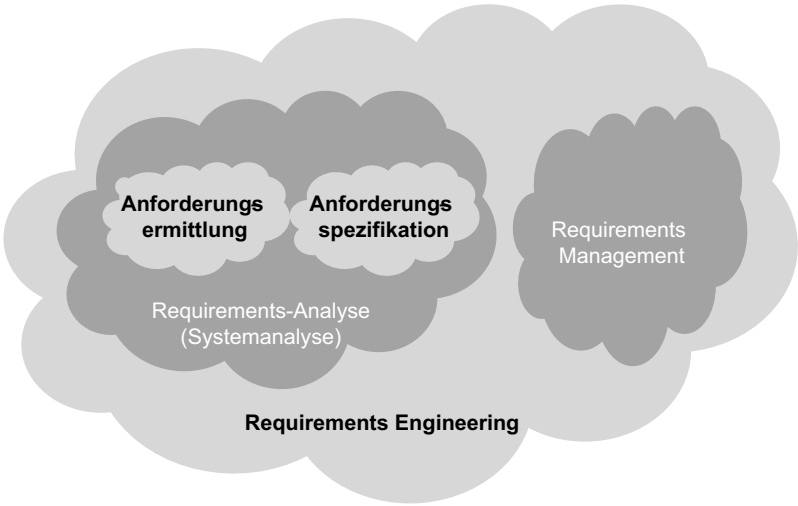


Bild 1.4 Die Hauptthemen im Requirements Engineering

Definition: Requirements Engineering

Ich möchte Ihnen aber neben diesem Mengendiagramm doch noch eine ausführliche Definition geben. Requirements Engineering im heutigen Sinne ist ein kooperativer, iterativer und inkrementeller Prozess mit folgenden drei Zielen:

1. Alle relevanten Anforderungen sollen bekannt und im erforderlichen Detaillierungsgrad verstanden sein.
2. Die involvierten Personen und Organisationen (Stakeholder) sollen ausreichende Übereinstimmung über die bekannten Anforderungen erzielen.
3. Die Anforderungen sollen konform zu den Dokumentationsvorschriften der Organisation spezifiziert sein.

Alleine in der Einleitung dieser Definition stecken drei Aussagen drin. Ein kooperativer Prozess heißt: Wir arbeiten zusammen und nicht gegeneinander. Wir wollen nicht Anforderungen über den Zaun werfen zu anderen Leuten, sondern wir wollen sie gemeinsam entwickeln zwischen Auftraggeber und Auftragnehmer oder in Scrum: zwischen Product Owner und Development Team.

Die Wörter „iterativ“ und „inkrementell“ richten sich gegen das alte Wasserfallmodell. Ich muss nicht alle Anforderungen am Anfang perfekt kennen. Ich kann in Iterationen (Releases oder Sprints) – schrittweise – Teile davon, weitere Teile davon, noch mehr Teile davon zeitgerecht entwickeln.

Danach lernen wir durch die Definition die drei wesentlichen Ziele dieses kooperativen Prozesses kennen.

Das erste Ziel ist: Ich möchte alle **relevanten** Anforderungen so gut und im **erforderlichen Detaillierungsgrad** kennen, dass ich eine Lösung darauf basieren lassen kann. Sehen Sie die Weichmacher in dieser Definition? Alle relevanten Anforderungen in hinreichendem Detaillierungsgrad? Was ist relevant? Was bedeutet hinreichender Detaillierungsgrad? Wir werden uns ausführlich mit beiden Fragen auseinandersetzen.

Das zweite Ziel besagt, dass die beteiligten und betroffenen Personen, die Stakeholder, die Organisationen, die mitspielen müssen, hinreichende Übereinstimmungen über diese Anforderungen erzielen sollten. Nun ja, in der Diktatur reicht es, wenn der Diktator ja sagt. Wenn Sie etwas demokratischer aufgestellt sind, brauchen Sie vielleicht die Zustimmung von ein paar mehr Leuten. Aber selbst in einer Demokratie sind 50,01 % eine Mehrheit. Es wird also immer wieder Personen geben, die dagegen sind, die andere Wünsche haben. Wir wollen aber, dass der Prozess dafür sorgt, dass wir eine hinreichende Übereinstimmung über die bekannten Anforderungen erzielt haben.

Und das dritte Ziel für diesen Prozess Requirements Engineering ist noch einfacher. Wir wollen, dass die Anforderungen nach den Regeln des Hauses, nach Ihren Spielregeln, nach Ihren Dokumentationsvorgaben aufgeschrieben werden. Wenn Sie in Ihrem Haus gar keine Vorschriften haben, wie man ein Pflichtenheft oder ein Lastenheft erstellt oder den Product Backlog erfasst, sind Sie auf jeden Fall konform dazu ☺. Aber die meisten Firmen haben natürlich Vorgaben, was alles festzuhalten ist und wie es festgehalten werden sollte.

Requirements Engineering ist also ein kooperativer Prozess, der in Zyklen durchgeführt wird. In jedem Zyklus muss ich genügend herausfinden für das, was ich demnächst umsetzen möchte. Ich brauche genügend Übereinstimmung und ich muss es hausgerecht festhalten, nach unseren Dokumentationsvorschriften.

Wenden wir uns nun dem etwas weiter gefassten Begriff „Business Analysis“ zu. Das International Institute of Business Analysis (IIBA) definiert in seinem Business Analysis Body of Knowledge (BABOK) Folgendes:

Definition: Business Analysis

Business Analysis ist eine Menge von Aufgaben und Techniken, die genutzt werden, als Liaison zwischen Stakeholdern zu arbeiten, um die Struktur, die Direktiven (Policies) und die Arbeitsweise einer Organisation zu verstehen und (fachliche) Lösungen vorzuschlagen, die es der Organisation ermöglichen, Ihre Ziele zu erreichen.

Das entsprechende Überblickbild über Business-Analyse identifiziert acht Kernbereiche, die dort als Wissensgebiete (Knowledge Areas) bezeichnet werden (vgl. Bild 1.5).

Sicherlich sehen Sie die starken Überlappungen zu der Definition von Requirements Engineering. Auch hier stehen die Erhebung (von Anforderungen) und das Managen von Anforderungen rechts und links im Fokus. Analysieren wurde hier dreigeteilt: Enterprise-Analyse und Requirements-Analyse, natürlich nicht ohne beides zu bewerten und zu validieren. Eingerahmt werden diese Kerntätigkeiten durch die Planung und Überwachung des Gesamtvorgehens sowie die Forderungen nach grundlegenden Kompetenzen der Mitwirkenden, die wir später in diesem Kapitel genauer betrachten werden.

Was haben die beiden Definitionen gemeinsam? Sowohl bei Business Analysis wie auch beim Requirements Engineering geht es um das Verstehen eines Systems (d. h. eines Geschäfts, einer Organisation, eines Produkts oder eines Systems), mit dem Ziel, Schwachstellen zu identifizieren, Ideen für Verbesserungen zu entwickeln, damit zu besseren oder neuen Produkten bzw. zu optimierten Ablauf- und Aufbauorganisationen zu kommen. Bild 1.6 fasst die wesentlichen Aspekte von Business Analysis und Requirements Engineering zusammen:

- Wir gehen davon aus, dass es jemanden gibt, der mit dem Ist-Zustand von Produkten oder Organisationen nicht zufrieden ist. Jemand, der ehrgeizige Ziele hat, Wünsche und Ideen und Visionen, oder aber durch äußere Zwänge wie Gesetze und andere Auflagen gezwungen wird, am Status-quo etwas zu verändern.

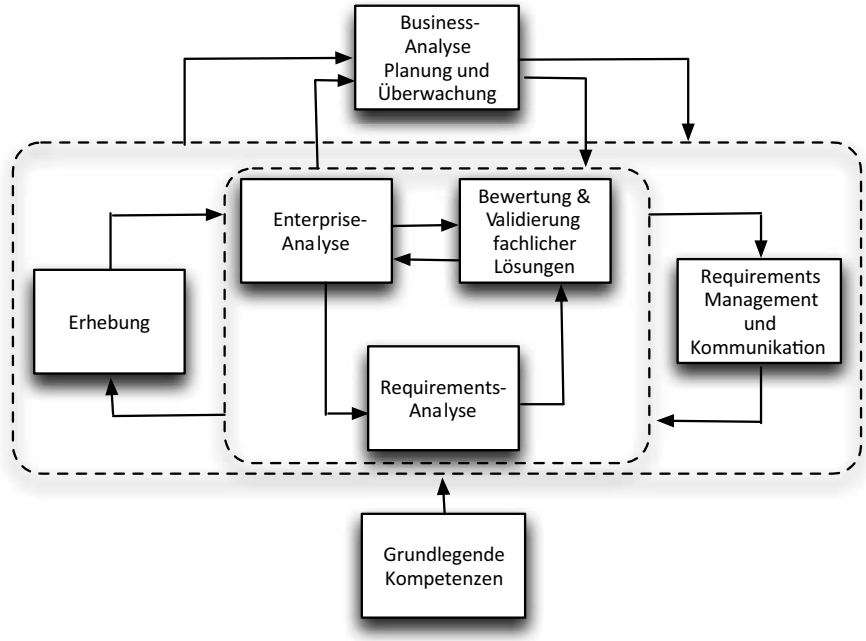


Bild 1.5 Wissensgebiete der Business-Analyse

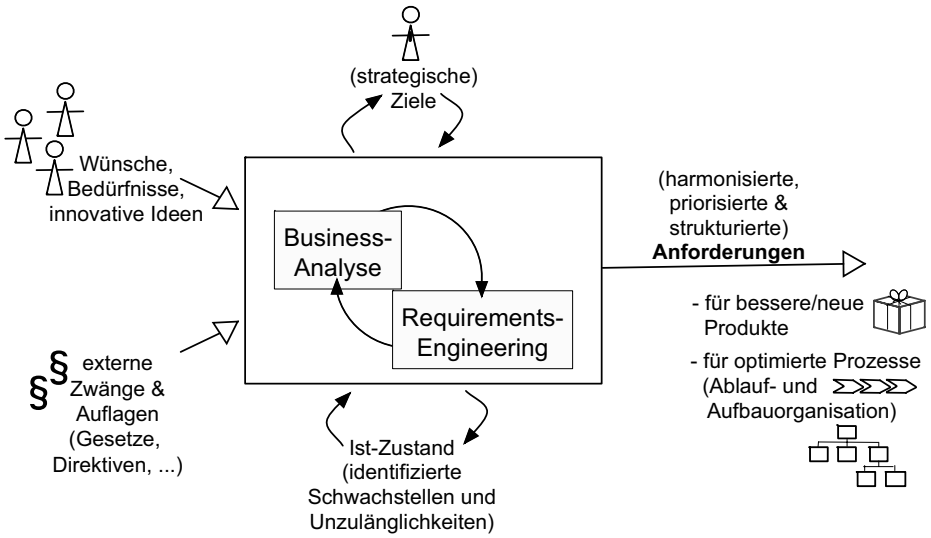


Bild 1.6 Business-Analyse und Requirements Engineering

- Durch Analyse des heutigen Zustands von Organisationen und Produkten identifiziert man Schwachstellen und entwickelt Lösungsideen und Verbesserungsvorschläge, die man in Form von möglichst präzisen Anforderungen an jemanden übergeben kann, der in der Lage ist, die Lösung in die Praxis umzusetzen. Entweder mittels IT-Systemen oder aber auch durch organisatorische Maßnahmen.
- Diese Tätigkeiten werden gemeinsam von allen Stakeholdern durchgeführt, die ein Interesse an der Sache haben, in einem kooperativen Prozess, wobei wir ausgewogen über kurz-, mittel- und langfristige Lösungen nachdenken. Je kurzfristiger wir etwas ändern wollen, desto genauer müssen wir Anforderungen spezifizieren. Je langfristiger Themen sind, desto vager können wir im Ergebnis bleiben. Durch iteratives und inkrementelles Vorgehen sichern wir jederzeit die Konzentration und Bündelung unserer Kräfte auf das zurzeit Notwendige.

1.8 Definition: Requirement

Anforderungen sind identifizierbar und halbwegs formalisiert

Jetzt haben wir so häufig das Wort „Requirements“ erwähnt. Was überhaupt ist eine einzelne Anforderung, ein Requirement? Nicht alles, was uns Projektbeteiligte erzählen, ist automatisch eine Anforderung. Sie sprechen über viele Bedürfnisse, über Wünsche. Sie geben uns zahlreiche Hintergrundinformationen. Sie erklären uns vielleicht sogar, warum sie das Ganze haben wollen, und geben uns Erläuterungen aus dem bisherigen Leben, aus Altsystemen oder was in Zukunft sein soll. Zur Anforderung wird es erst dann, wenn Sie es halbwegs formalisieren und identifizierbar machen. Identifizierbar, denn im Zweifelsfall wollen wir sagen, das war eine Anforderung oder das war keine Anforderung. Es nur gesagt zu haben oder nur erwähnt zu haben, reicht oft nicht aus. Wir wollen eine identifizierbare Aussage haben und sie sollte auch halbwegs formalisiert sein.

Sie brauchen aber keine Angst zu haben. Sie müssen nicht unbedingt komplexe Diagramme lernen. Ein sauber geschriebener deutscher Satz mit Subjekt, Prädikat und Objekt oder ein sauber geschriebener englischer Satz oder eine gut formulierte User-Story erfüllt auch das Kriterium, halbwegs formalisiert zu sein. Aber die beiden Punkte brauchen wir, denn wir brauchen eine Einigung, ob irgendwas eine Anforderung war oder nicht, denn wir schreiben Anforderungen hauptsächlich deshalb, damit wir beim Übergeben einer Lösung, beim Abnahmetest, beim Sprintende feststellen können, ob die Anforderung auch erfüllt ist. Wenn wir zu diesem Zeitpunkt nichts hätten, wogegen wir prüfen können, nichts, was wir identifizieren können, nichts, was halbwegs eindeutig formuliert ist, dann hätten wir Probleme; wir könnten nicht feststellen, ob das geliefert wurde, was wir bestellt haben oder haben wollten.

Für Formalisten hat IEEE, die amerikanische Ingenieursvereinigung, natürlich formale Definitionen festgehalten, was Anforderungen sind. Gemäß IEEE 610.12-1990 ist eine Anforderung

1. eine Bedingung oder Fähigkeit, die von einem Benutzer (Person oder System) zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.

2 Erfolgreich starten

Im zweiten Kapitel betrachten wir, wie wir Vorhaben oder Projekte erfolgreich starten. Aus Sicht des Business Analyst, Requirements Engineer oder Product Owner sind es drei Punkte oder drei Zutaten, die zu einem erfolgreichen Projektstart unbedingt notwendig sind.

1. Wir brauchen eine Einigung über die Projektziele; eine Vision von dem, was wir wirklich erreichen wollen. Ziele oder Visionen sind die höchsten und abstraktesten Anforderungen. Oder kurz gesagt, Ziele sind auch Anforderungen und müssen genauso prüfbar sein wie alle Anforderungen im System.
2. Wir müssen die Stakeholder ermitteln. Damit gemeint sind alle relevanten Personen, die im Projekt mitwirken sollen, wollen und können.
3. Wir müssen den Scope eingrenzen, den Umfang unseres Projekts. Wir müssen wissen, was uns gehört und was uns nicht gehört. Unser Thema abgrenzen von der Umwelt, von Nachbarsystemen und die Schnittstellen zu diesen Nachbarsystemen festlegen.

2.1 Drei Zutaten zu einem erfolgreichen Projektstart

Die Grundlage für den Erfolg oder den Misserfolg von Projekten wird schon sehr früh, ganz am Anfang gelegt. Deshalb wollen wir uns in diesem Kapitel die Ingredienzen ansehen, die dazu notwendig sind, ein Projekt erfolgreich zu beginnen. Drei Punkte sind es, auf die wir am Anfang achten sollten: die Festlegung der Projektziele, die Festlegung der Mitspieler (im Englischen „Stakeholder“ genannt) und die Festlegung des Umfangs des Projekts (im Englischen als „Scope“ bezeichnet). Wie Bild 2.1 verdeutlicht, haben diese drei Punkte Wechselwirkungen untereinander. Es ist klar: Andere Mitspieler haben unter Umständen andere Ziele. Umgekehrt, wenn Ziele vorgegeben werden, kann man die beteiligten Personen so auswählen, dass diese Ziele gut erreichbar sind.

Die Ziele geben aber auch den Umfang des Projekts vor und der Umfang bestimmt wiederum, was innerhalb des Zielbereichs liegt und was nicht. Und natürlich haben auch die Stakeholder und der Scope ihre Wechselwirkungen. Wir werden in den folgenden Abschnitten alle drei Ingredienzen ausführlich behandeln.

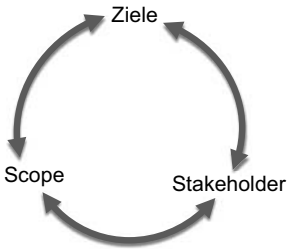


Bild 2.1
Drei Zutaten ausbalancieren

2.2 Ziele

Beginnen wir mit den Zielen. Natürlich sollte jedes Projekt Projektziele haben. Aber ist das wirklich Ihre Aufgabe als Systemanalytiker? Sind Sie dafür verantwortlich, dass es Projektziele gibt? Das ist doch der Job eines Projektleiters. Oder vielleicht noch besser, der Job eines Auftraggebers, denn als Projektleiter würde ich meine Auftraggeber fragen, was sie eigentlich anstreben. Warum sollten Sie sich als Systemanalytiker also um Ziele kümmern? Nun ja. Ziele sind die höchste und abstrakteste Form von Anforderungen. Alle weiteren (detaillierteren) Anforderungen, die Sie erheben, sollten sich aus diesen Zielen ableiten. Deshalb sollten Ziele Ihre Ausgangsbasis sein. Und wenn der Projektleiter oder der Auftraggeber sie noch nicht formuliert haben, dann müssen Sie ran und diese Ziele formulieren.

Ich habe vor ein paar Jahren einmal im Oktober ein Team kennengelernt, das mit fünfzehn Personen an einem großen Thema arbeitete. Wir haben dann für Anfang Januar vereinbart, einen Einführungskurs zu machen und auch ein bisschen Projektberatung. Nach den beiden Tagen des Einführungskurses, mit ähnlichen Inhalten wie in diesem Buch, habe ich am Mittwochmorgen die Fragen gestellt: Wo sind Eure Projektziele? Ihr arbeitet schließlich schon mit fünfzehn Leuten seit drei Monaten zusammen. Kann mir jemand die Ziele nennen? Alle sahen sich ratlos um. Insbesondere der deutsche Softwareprojektleiter und der italienische Hardwareprojektleiter sahen sich tief in ihre blauen Augen und sagten: „Wir haben keine expliziten Projektziele." Daraufhin habe ich die beiden Projektleiter in ein Kämmerchen verbannt, mit der Aufgabenstellung, diese Ziele doch schriftlich festzulegen, und habe mit dem Rest des Teams am Vormittag weitergearbeitet. Kurz vor dem Mittagessen wollten die beiden wieder heraus aus ihrem Kämmerchen. Und sie kamen ganz stolz mit einer PowerPoint-Folie, mit zwei dicken Bullet-Punkten, großer Schrift und haben die beiden Ziele dem Team vorgestellt. Das Erstaunliche war nicht, dass die beiden Projektleiter es in zwei Stunden geschafft hatten, sich auf diese beiden Sätze zu einigen, sondern das Erstaunliche waren die Gesichter der anderen dreizehn Projektmitglieder, die die Sätze gelesen haben und sagten: „Ach, das sind unsere Hauptziele in dem Projekt!" Natürlich hatte jeder im Team seine Meinung zu dem Thema, aber die Meinungen waren sehr unterschiedlich.

Sie sehen also, Ziele sind unbedingt notwendig, um ein Projekt in die gleiche Richtung zu treiben, sonst arbeitet jeder nach seinen eigenen Zielen. Wie würden wir also Ziele formulieren? Wir versuchen es zweiteilig, wie in Bild 2.2 vorgeschlagen: Teil 1 ist die Ausgangssituation, die uns veranlasst, etwas zu tun. Und Teil 2 sind die eigentlichen Ziele.

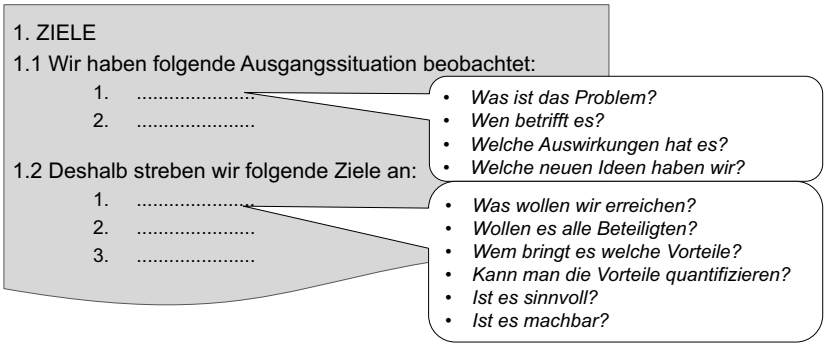


Bild 2.2 Ziele aus Ausgangssituationen motivieren

Skizzieren Sie zuerst Ihre Ausgangssituation. Es gibt sinnvollerweise nur zwei Möglichkeiten, warum Sie ein neues Projekt aufsetzen: Entweder drückt Sie irgendwo der Schuh oder Sie haben eine grandiose Idee, wie man das Geschäft verbessern kann. Fassen Sie das in Worte. Und danach versuchen Sie, die Ziele oder Visionen daraus abzuleiten. In großen Projekten ist es für den zweiten Teil sinnvoll, zwischen den Zeithorizonten für die Ziele noch weiter zu unterscheiden, wie es in Bild 2.3 dargestellt ist.

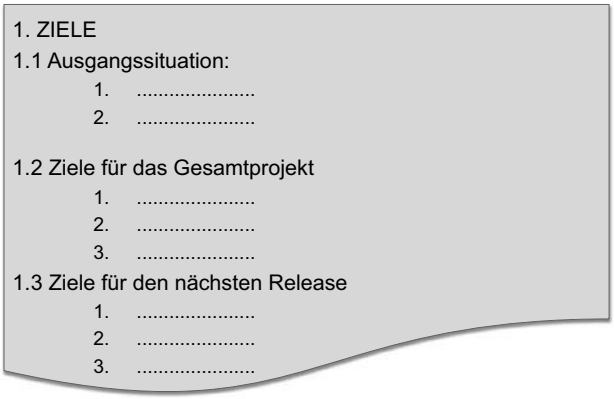


Bild 2.3
Mehrstufige Ziele

Nach der Ausgangssituation kommen die Langfristziele, die Ziele für die nächsten zwei bis fünf Jahre vielleicht, und darunter die Ziele für das jeweils nächste Release. Egal, wie weit es noch weg ist; drei Monate oder sechs Monate. Formulieren Sie als Release-Ziele die Zwischenziele, die Sie in drei bis sechs Monaten erreichen wollen. Oder, wenn Sie nach SCRUM arbeiten, die Sprintziele für die nächsten 30 Tage.

Ziele sind auch Anforderungen. Oder noch härter ausgedrückt: Ziele sind (eine Art von) Anforderungen. Das wird sehr oft missverstanden. Sehr oft finden Sie in der Literatur Aussagen wie: Ziele sind intentionale Aussagen über das Projekt. Was sind intentionale Aussagen anderes als etwas Gewünschtes, also eine Anforderung?

Wir haben in der Einleitung besprochen, dass Anforderungen sich mit 1 – 3 % pro Monat ändern. Wir schärfen daher unsere Definition von Zielen noch, wie in Bild 2.4 dargestellt.

Bild 2.4 Ziele sind (hoffentlich) stabil.

Ziele sollten (für den gewählten Zeitraum des Gesamtprojekts, des Release, eines Scrum-Sprints) stabile Anforderungen sein. Denn die detaillierten Anforderungen werden sich mit hoher Wahrscheinlichkeit mit 1 – 3 % pro Monat ändern.

2.3 Ziele spezifizieren

Wie können wir diese Ziele spezifizieren? Einige von Ihnen haben bestimmt Projektmanagementkurse gemacht und eine entsprechende Abkürzung gelernt. Die populärste Abkürzung ist SMART. Ziele sollten smart sein. S wie spezifisch, wir wollen definitiv spezifische Projektziele. M wie messbar, das gehört ebenfalls dazu: Jede Anforderung sollte messbar sein. Ziele sind auch Anforderungen. Also brauchen wir Messbarkeit auch für Ziele; A wie angemessen oder adäquat, R wie realistisch und T wie terminiert. Normalerweise wird Ihnen ein Zeitrahmen vorgegeben, in dem Sie diese Ziele erreichen sollten.

Die Abkürzung ist zwar sehr bekannt, aber wir arbeiten bei der Atlantic Systems Guild mit einem etwas kürzeren Akronym: PAM (vgl. Bild 2.5).

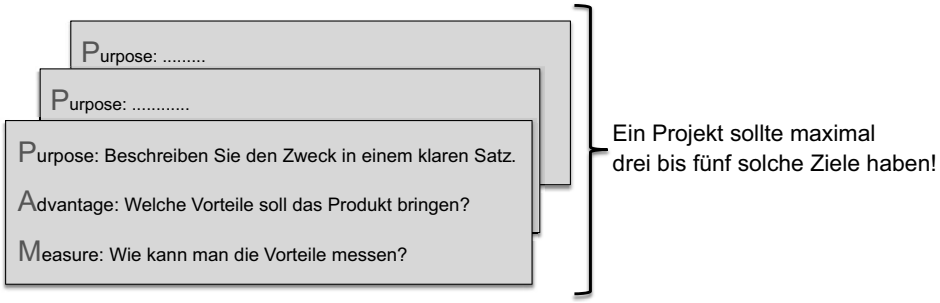


Bild 2.5 Ziele mit dem Akronym PAM spezifizieren

Zumindest die Baywatch-Fans können sich diese Abkürzung gut merken. Aber PAM steht nicht für Pamela Anderson, sondern für Purpose, Advantage, Measure. Purpose: Schreiben Sie einfach einen Satz, was das System tun soll. Das System soll das und das erreichen.

Überlegen Sie unter dem Buchstaben A: Wer hat etwas davon? Das Ziel sollte irgendjemandem Vorteile ermöglichen. Was ist der Vorteil und wer sind diejenigen, die den Vorteil haben? Vorteile könnten z. B. sein:

- Kostenreduktion/Geschäft vereinfachen
- Gewinne erhöhen

- Service verbessern
- Gesetze erfüllen

Wenn Sie niemanden finden, ist vielleicht das Ziel falsch gewählt.

Und wie bei SMART: Das M in PAM steht auch für Measure. Sie brauchen eine Metrik, woran Sie nach Ende des Projekts oder des Release feststellen können, ob Sie das Ziel erreicht haben oder nicht.

Verwenden Sie für jedes einzelne Ziel die Abkürzung PAM: Purpose – Advantage – Measure. Jetzt hat ein Projekt aber nicht beliebig viele Ziele. Wir versuchen im Normalfall, die Anzahl irgendwo zwischen ein, zwei, drei – bis maximal fünf zu begrenzen. Ein Projekt hat keine 97 Ziele. Sie können 97 Anforderungen haben. Sie können vielleicht 300 Anforderungen haben. Vielleicht auch 5 000. Aber wir haben garantiert nicht so viele Ziele. Beschränken Sie die Anzahl der Ziele auf maximal eine Handvoll, die Sie nach dieser Formel: Purpose – Advantage – Measure formulieren.

Ziele sind also die abstrakteste Form unserer Anforderungen. Deshalb sind sie oft vage und schwer prüfbar. Es lohnt sich aber trotzdem, sich über die Prüfbarkeit, über die Messbarkeit, Gedanken zu machen, auch wenn es schwierig ist. Manchmal kann man ein Ziel erst prüfen, wenn man die Anforderungen zerlegt hat und die Erreichung dieser Summe der zerlegten Anforderungen zusammengenommen die Zielerreichung auf oberer Ebene gewährleistet.

SMART und PAM sind zwei Möglichkeiten, die Ziele zu formulieren. In den letzten Jahren haben sich im Zug der agilen Methoden auch noch andere Techniken etabliert, um Ziele zu definieren (vgl. Bild 2.6). Eine Möglichkeit ist, einen Produktkoffer anzufertigen. Basteln Sie einen Würfel oder einen Quader. Schreiben Sie darauf den Produktnamen. Lassen Sie das Team (oder die Grafikabteilung) ein Logo entwickeln, das das Projekt oder Produkt charakterisiert. Und arbeiten Sie dann in zwei, drei dicken Punkten die Highlights heraus, die das Produkt erfüllen soll. Diesen Koffer stellen Sie sichtbar im Konferenzraum oder in Ihrem War Room auf, wo Sie sich für Projektbesprechungen treffen, sodass jeder Beteiligte diese Ziele immer vor Augen hat. In [DeM07] haben wir ein eigenes Kapitel über die Bedeutung von guten Zielen im Projekt. Meine Kollegin Susanne Robertson hat eine sehr nette Story aus einem ihrer Projekte erzählt. Ein Team hat die Ziele auf ein großes Flipchart geschrieben, dieses Flipchart-Blatt dann auf Pappe aufgeklebt und zu jeder Besprechung mitgenommen. Und die Ziele bekamen einen Platz am Konferenztisch. Sie waren also ständig präsent bei jeder Besprechung, sodass niemand, der bei der Besprechung anwesend war, vergessen konnte, worum es eigentlich ging. Das ist eine etwas andere Form des Zielkoffers.

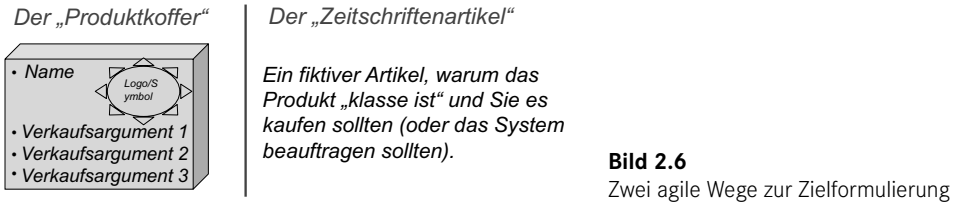


Bild 2.6
Zwei agile Wege zur Zielformulierung

In der agilen Welt hat sich noch eine zweite Technik bewährt: einen Zeitungsartikel zu schreiben; eine Kolumne, nur eine Spalte, ein Drittel einer Seite; nicht mehr als 20, 25 Zeilen. Stellen Sie sich dazu vor, das Projekt wäre gerade eben fertig geworden. Und Sie beschreiben am Tag nach der Freigabe als Zeitungsreporter, warum das System so toll geworden ist.

Das setzt natürlich voraus, dass Sie sich a) Gedanken über die Ziele gemacht haben und b) ein guter Redakteur oder ein guter Autor sind, um das auch in Form eines interessanten und spannenden Artikels wiedergeben zu können.

Für viele ist das Akronym „PAM“ (Purpose – Advantage – Measure) wesentlich leichter umzusetzen in der Praxis, als einen 25-zeiligen Zeitungsartikel zu schreiben. Versuchen Sie es aber einmal und versuchen Sie, dem Leser klar zu machen, warum er das System kaufen sollte, indem Sie in dem Artikel die Vorteile in ganz wenige Zeilen verpacken.

2.4 Stakeholder

Das zweite Thema, das wir am Projektanfang angehen müssen, sind die Stakeholder. Die deutsche Bezeichnung dafür ist schwierig zu finden. Wir haben selbst bei großen Konferenzen Preisausschreiben unter 120, 150 Teilnehmern gemacht: Gebt uns ein gutes deutsches Wort für das Wort Stakeholder. Üblicherweise sagt man Interessensvertreter oder Interessensträger. Wir haben auch sehr oft Projektbeteiligte und Projektbetroffene gehört. Das klingt so negativ: Projektbetroffene. Aber das ist genau das, was gemeint ist. Alle, die am Projekt Anteil haben. Alle, die von dem Projekt in irgendeiner Weise betroffen sind. Die originellste Lösung, die wir als Übersetzung bekommen haben, war „Platzhirsch“. Ja, die Platzhirsche sind Stakeholder. Das sind die Leute, die was zu sagen haben. Aber das sind nicht alle. Nicht jeder Stakeholder ist unbedingt Platzhirsch.

Sehen wir uns mal die formale Definition dazu an:

> *Stakeholder sind also alle Personen oder Organisationen, die direkt oder indirekt Einfluss auf die Anforderungen des Systems haben.*
>
> *Alternativ können wir festlegen: Stakeholder sind Personen oder Gruppen, die von dem Projekt oder dem Produkt in irgendeiner Weise betroffen sind.*

Sie sehen, es können Einzelpersonen sein, es können Organisationen sein. Sie sind vielleicht direkt am Projekt beteiligt oder sie sind nur die grauen Eminenzen im Hintergrund und haben sehr indirekten Einfluss; aber alle Personen und alle Gruppen, die Einfluss auf das Projekt haben, die in irgendeiner Weise von dem Projekt betroffen sind, gehören zu der Gruppe der Stakeholder. Unser amerikanischer Kollege Jerry Weinberg, der sehr, sehr viele Bücher geschrieben hat, hat mal gesagt: „Wenn in dieser Liste der Stakeholder nicht mindestens 200 Einträge drin sind, habt ihr bestimmt noch jemanden übersehen.“ Naja, 200 ist vielleicht übertrieben, aber 30 – 50 Rollen haben wir sehr schnell gefunden.

Warum sprechen wir überhaupt über Stakeholder? Der wichtigste Grund ist:

> *Verpasste Stakeholder sind verpasste Requirements!*

Wenn Sie irgendjemanden im Projekt nicht berücksichtigt haben, hat der vielleicht auch Wünsche und Anforderungen an das System, ist aber nicht gehört worden. Und jetzt hängt es ein bisschen von Ihrer Projektorganisation ab, ob Ihnen das weh tut oder nicht. Wenn Sie Personen oder Gruppen übersehen haben, der Abnahmetest trotzdem erfolgreich war und Sie Ihr Geld bekommen, dann könnten Sie eigentlich zufrieden sein. Aber es bleibt ein Bei-

geschmack. Da ist jemand, der ist nicht zufrieden. Der hätte noch Wünsche an das System gehabt, wurde aber nicht berücksichtigt.

Das Schlimmste, was Ihnen passieren kann, ist, dass diese übersehenen Personen Stimmung gegen Sie machen und schlecht über Ihr System sprechen, Meinungen vertreten, die Ihnen gar nicht angenehm sein könnten. Deshalb: Verpasste Stakeholder sind verpasste Requirements. Sie sollten niemanden übersehen.

Wenn Sie die Personen oder Gruppen identifiziert haben, können Sie immer noch darüber nachdenken, ob deren Anforderungen ernst genommen werden sollen oder nicht. Aber sie gar nicht zu kennen, ist schlecht. Deshalb wollen wir über Stakeholder nachdenken. Meine englischen Kollegen Susanne und James Robertson haben die Stakeholder in Form eines Zwiebelschemas aufgezeichnet [Rob12]. Sie sehen in Bild 2.7 ganz in der Mitte die Personen, die an dem Projekt mitarbeiten. Projektleiter, Systemanalytiker, Designer, Programmierer, Tester, Qualitätssicherer; alle, die direkt an der Entwicklung des Systems beteiligt sind. Natürlich ist auch jeder aus der Entwicklungsmannschaft ein Stakeholder in dem Projekt. Aber wahrscheinlich sind das nicht diejenigen, die die meisten Anforderungen vorgeben.

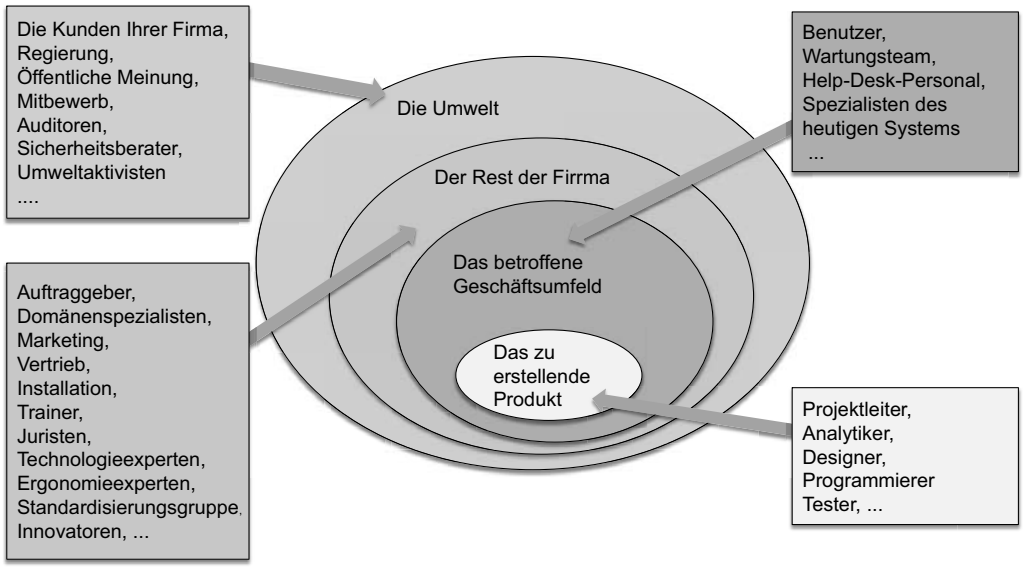


Bild 2.7 Projektnähere und -fernere Stakeholder

Die meisten Anforderungen kommen aus dem direkten Umfeld des Projekts. In dieser zweiten Ebene der Zwiebel sind die Personen enthalten, die das System heute nutzen: unsere momentanen Benutzer und Anwender; die Spezialisten des heutigen Systems, die Leute, die am Helpdesk sitzen, die Leute, die Auskunft geben, die Leute, die Trainings machen für das System. Aus dieser Richtung stammt wahrscheinlich die Majorität der Anforderungen für ein neues System.

Sie sehen, der Auftraggeber ist schon in der dritten Ebene dieses Zwiebelschemas enthalten. Das ist der Geldgeber. Als ich studierte, waren das die Einzigen, die wir ernst nehmen sollten. Man hat uns immer eingetrichtert: Achte auf diejenigen, die das Geld haben!

Aber diejenigen, die den Budgettopf besitzen, sind oftmals nicht diejenigen, die die detaillierten Anforderungen kennen und vorgeben können. Trotzdem sind sie natürlich wichtig in dem Projekt. Außer den Geldgebern hat man uns damals nur noch auf eine zweite Gruppe hingewiesen: Vielleicht sollte man auch an die künftigen Benutzer denken – obwohl die meist kein Geld für die Systementwicklung haben und daher aus Managementsicht sicherlich nicht so „wichtig" sind wie die Geldgeber. Aber: Wie Jerry Weinberg gesagt hat, es gibt viel mehr als nur die Geldgeber und die Benutzer.

In dieser dritten Ebene der Zwiebel sind neben dem Auftraggeber auch alle Domänenspezialisten enthalten; Leute, die die Branche gut kennen, die die Geschäftsregeln kennen. Dort sind auch Vertrieb und Marketing angesiedelt, da sind Juristen, da sind Trainer, Leute, die das Produkt trainieren, einschulen. Da sind aber auch Standardisierungsgruppen und Technologieexperten.

Und noch weiter außen haben wir die Umwelt. Sie sehen, je weiter wir nach außen kommen, desto indirekter wird der Einfluss. Da sind dann die Kunden Ihrer Firma. Da ist die Regierung, der Gesetzgeber, die öffentliche Meinung, der Mitbewerb. Da sitzen auch Auditoren, die sehen, ob Sie Projekte nach den Hausvorschriften abwickeln. Letztere interessieren sich nicht so sehr für Ihr einzelnes Projekt, sondern für Ihre Vorgehensweisen, weil Sie unter Umständen CMMI-zertifiziert sein wollen oder ISO 9000-zertifiziert. Da sind auch Umweltaktivisten und viele andere. Sie sollten alle diese Personen suchen und deshalb wollen wir uns im nächsten Abschnitt ein bisschen darüber unterhalten, wie wir Stakeholder finden können.

2.5 Stakeholder finden

Die einfachste Art, Stakeholder zu finden, ist, am Anfang des Projekts eine Kerngruppe zusammenzubringen und ein kleines Brainstorming zu veranstalten. Wenn Sie fünf bis sieben Personen am Tisch sitzen haben, wie lange dauert es, in einem Brainstorming eine Liste von potenziell betroffenen und beteiligten Personen aufzustellen? Ich bin sicher, nach 20, 30 Minuten haben Sie bereits 30 bis 40 Einträge in dieser Liste.

Was wollen wir wissen? Vier Punkte sollten Sie berücksichtigen:

1. Wer hat Interesse? (Name und Rolle)
2. Was können sie beitragen? Was wissen sie?
3. Welchen Einfluss haben sie? (Wichtigkeit für das Projekt)
4. Was sind ihre Interessen?

Nicht so sehr die Namen der Einzelpersonen, sondern in welcher Rolle die Personen unserem Projekt helfen können oder in welcher Rolle sie unter Umständen auch unser Projekt behindern können, sollte im Vordergrund stehen. Natürlich brauchen wir hinter jeder Rolle konkrete Personen, die wir ansprechen können. Aber wichtig ist, alle Rollen abgedeckt zu haben.

Wir brauchen nicht jeden Stakeholder zu jedem Thema. Deshalb ist es wichtig, herauszufinden, wer welchen Beitrag zu unserer Analyse leisten kann. Welches Wissen wollen wir bei den unterschiedlichen Stakeholdern anzapfen?

Und dann wollen wir natürlich festhalten, wie wichtig diese einzelnen Rollen und Personen für unser Projekt sind. Wenn wir hinterher Streit über Anforderungen bekommen, die unterschiedlich priorisiert werden, als wichtiger oder unwichtiger eingestuft, dann werden wir immer wieder zurückkehren in die Stakeholder-Liste und schauen, wer hat es eigentlich gefordert; wer hat es gesagt. Deshalb wollen wir die Wichtigkeit für das Projekt, den Einfluss der Stakeholder auf das Projekt auf jeden Fall festhalten.

Was haben die Leute für Interessen? Nicht immer spielen alle mit offenen Karten. Als Analytiker will ich aber die Motivation der Beteiligten kennen und auch verborgene Interessen (Hidden Agendas) kennenlernen.

Sie brauchen dazu keine besonderen Werkzeuge. Es reicht eine einfache Tabelle in Word oder in Excel mit ein paar Spalten (vgl. Bild 2.8), in der Sie die Rollen, die Personen, deren Wichtigkeit und ihren Einfluss und das Thema, zu dem sie was zu sagen haben, festhalten.

Name	*Rolle*	*Bedeutung/ Einfluss*	*Wo können sie helfen?*

Bild 2.8
Eine einfache Stakeholder-Tabelle

Ein kleiner Trick noch am Rande: Wenn Sie im Brainstorming in 20, 30 Minuten diese erste Liste erstellt haben, verschicken Sie sie an jeden in der Liste per E-Mail und fragen Sie: „Kennt ihr sonst noch Personen, die mitwirken sollten, die wir vergessen haben?“

Mit diesem Schneeballeffekt, alle Personen auf der Liste zu befragen, ob noch jemand fehlt, ist die Liste sehr schnell vollständig. Es sind nur ein paar Minuten oder höchstens – über die Zeit verteilt – ein paar Stunden, die sich aber lohnen.

Übertreiben Sie nicht. Sie brauchen kein Dossier über jede einzelne Person anzulegen. Sie sind ja nicht bei der CIA oder beim KGB. Wir wollen nur wissen, mit wem wir unter Umständen in dem Projekt sprechen müssen.

Wir werden diese Liste einmal am Projektanfang erstellen, aber wir werden immer wieder ein Auge darauf werfen, ob in der Zwischenzeit andere Personen dazugekommen sind, die wir am Anfang übersehen haben.

Auch dazu eine kleine Geschichte. Wir hatten in einem großen Konzern eine gute Stakeholder-Analyse in einem Projekt gemacht und trotzdem jemanden vergessen. Beim nächsten Meilenstein und beim nochmaligen Durchschauen stießen wir auf Stakeholder, nämlich andere Abteilungen, die schon mit der gleichen Technologie Erfahrungen gemacht hatten. Als wir die entdeckt hatten, änderte sich sofort unser Projektplan. Anstatt mühevoll aus dicken Standards Wissen herauszusaugen, vereinbarten wir ganz einfach Meetings mit denen, um sie zu fragen: „Könnt ihr uns eure Erfahrungen mit dieser Technologie weitergeben?“ Das war auf jeden Fall die schnellere Art, Wissen zu gewinnen, als dicke Standards auszuwerten. Sie sehen also, immer wieder einmal auf diese Liste zu schauen, macht sich im Projekt bezahlt und Sie sollten diese paar Minuten investieren.

Lassen Sie uns noch ein paar Punkte diskutieren. Muss ich wirklich jeden gefundenen Stakeholder berücksichtigen? Nun ja. Nicht unbedingt. Wenn Sie jemanden, von dem Sie wissen,

dass er Interesse an dem Projekt hat, bewusst ausschalten, dann gehen Sie ein Risiko ein. Das ist ganz normales Risikomanagement. Haben Sie schon einmal versucht, den Betriebsrat außen vor zu lassen? Der Betriebsrat ist unter Umständen mächtig. Der hat seine Rechte und Pflichten. Und wenn Sie etwas gegen die Interessen des Betriebsrats im Projekt beschließen, werden Sie früher oder später Einsprüche hören. Sie können natürlich versuchen, alles möglichst geheim zu halten. Früher oder später wird es dennoch auffallen.

Hat schon jemand von Ihnen die Datenschützer übersehen? Dazu gibt es Gesetze, die eingehalten werden müssen. Ich muss diese Gruppen nicht immer beteiligen, aber ich sollte. Oder den Security-Beauftragten der Firma? Aber Sie sehen schon, nicht jeder Stakeholder muss bei jedem Meeting anwesend sein.

Wir haben dafür den Satz von der „artgerechten Aufzucht und Haltung“ von Stakeholdern geprägt. Sorgen Sie dafür, dass jeder Stakeholder artgerecht behandelt wird. Wenn Sie von einer Person wissen, dass sie grundsätzlich jedes Meeting stört, dann werden Sie diese Person nicht mehr zu Meetings einladen. Sie können denen ja ein 150-seitiges Dokument drei Stunden vor dem Meeting schicken und genau zwei Stunden Zeit geben, um darauf zu reagieren ☺. Das ist in diesem Fall vielleicht die artgerechte Behandlung von Störenfrieden.

Sie sehen schon, ich möchte alle kennen, die am Projekt beteiligt und davon betroffen sind. Ich behalte mir aber vor, wie ich mit diesen einzelnen Gruppen umgehe; wie intensiv ich sie brauche, wie wenig intensiv ich sie brauche und wann ich sie zu Rate ziehe und in welcher Form ich mit ihnen kommuniziere.

Einen Punkt sollten wir noch erwähnen. Nicht immer ist es leicht, die Stakeholder kennenzulernen. Einer meiner Kunden hat auf einer unserer Requirements-Konferenzen einen fantastischen Vortrag über verborgene Stakeholder gehalten. In dem Projekt war sehr klar festgelegt, wer offiziell welche Rolle spielt. Aber die vom Auftraggeber benannten Personen hatten nicht die geringste Macht. Beim Auftraggeber spielten andere Leute im Hintergrund, die wir nicht kannten, die wichtigere Rolle und die wesentlichen Entscheidungen wurden von Leuten getroffen, die uns vollkommen unbekannt waren. In dem Fall war es also notwendig, als Projektleiter herauszufinden, wer wirklich das Heft in der Hand hat. Nicht die im Projektplan schriftlich festgelegten Ansprechpartner waren die wichtigen Personen, sondern die, die im Hintergrund die Stimmung gemacht haben.

In dem Fall war es so schlimm, dass die Fachabteilung sehr viele Wünsche hatte und der Einkauf kein Geld geben wollte. Daher passten Aufwand und Anforderung nie zusammen und dieser Krieg wurde im Projekt, als Stellvertreterkrieg, natürlich ausgetragen. Wir mussten also erst herausbekommen, wer wirklich das Sagen hatte und wo die wirklichen Wünsche lagen, bevor wir weiterarbeiten konnten.

Muss man diese Stakeholder-Liste im Requirements-Dokument niederschreiben? Naja. Sie werden, wenn Sie Negatives über Stakeholder festhalten wollen, das nicht gerade in diesem Dokument tun.

Einer meiner Kunden hat einmal gesagt: Doch, wir machen Stakeholder-Analyse. Alle, die wichtig sind, stehen bei uns auf dem Titelblatt und müssen auch unterschreiben. Ja, die, die da auf dem Titelblatt standen, waren bestimmt wichtige Stakeholder. Aber bei weitem nicht alle. Das waren nur die, die formal eine Unterschrift leisten mussten. Also sorgen Sie dafür, dass Sie alle kennen, aber nicht unbedingt alle im Dokument schriftlich festhalten. Aber das Entwicklungsteam sollte über alle betroffenen Personen Bescheid wissen.

2.6 Die wichtigsten Stakeholder: die Nutzer

Die allerwichtigsten Stakeholder sind unsere Nutzer, diejenigen, die das Softwareprodukt später anwenden wollen. Diesen sollten Sie deshalb ganz besonderes Augenmerk schenken. Die zukünftigen Benutzer sollten das Produkt bestimmen. Prüfen Sie, ob Sie an all die Personenkreise gedacht haben, die in Bild 2.9 aufgeführt sind.

Denken Sie auch an ...

- Personen mit Behinderungen
- Analphabeten
- Menschen, die eine andere Sprache sprechen
- Menschen mit Brillen
- Personen, die Fonts und Farben ändern müssen
- Personen, die Pakete oder Babies tragen
- Personen, die normalerweise nicht mit Computern arbeiten
- Personen, die wütend, frustriert oder in Eile sind

Bild 2.9
Haben Sie alle Nutzergruppen auf Ihrem Radar?

Das sind die, für die wir das Produkt entwickeln wollen. Sie sollten sie kennen, Sie sollten wissen, wer sie sind, Sie sollten vor allem wissen, welche Fähigkeiten diese Personen haben und wie sie so drauf sind. Wir machen Projekte unter Umständen auch für Analphabeten. Eine Tatsache, die mich sehr erstaunt hat. Deutschland hat etwa 4,5 % Analphabeten. Manche Zahlen sind sogar wesentlich höher. Kuba und auch Österreich haben eher 2 %. Also wenn Sie ein Projekt für die breite Masse entwickeln, müssen Sie damit rechnen, dass ein Teil nicht vernünftig lesen und schreiben kann.

Sie entwickeln ein Produkt unter Umständen auch für Leute wie mich, die schlecht sehen und eine Brille tragen müssen. Und eine 5-Punkt-Schrift, die man am Bildschirm nicht vergrößern kann, ist vielleicht nicht gerade die richtige Lösung für solche Personen. Oder denken Sie einmal daran, dass Leute auch bepackt sind. Wenn Sie also einen Parkscheinautomaten für Parkhäuser entwickeln, sollten Sie auch daran denken, dass man Pakete irgendwo abstellen will, um nach Münzen zu kramen und das Parkticket auszulösen. Eine kleine Ablagefläche wäre ein sehr willkommenes Feature, damit man seine Einkäufe nicht auf den Boden stellen muss. Das hat in diesem Fall nichts mit Software zu tun, aber das ist Produktdesign. Und ein Produkt, wie ein Parkscheinautomat, sollte solch eine Möglichkeit bieten.

Machen Sie sich Gedanken über die Fähigkeiten Ihrer Stakeholder (vgl. Bild 2.10) und berücksichtigen Sie dies in Ihrer Analyse.

Manche Branchen machen sich heftige Gedanken darüber, wer die späteren Benutzer sind. Im Bereich der Mobiltelefone haben wir inzwischen seniorengerechte Lösungen mit ganz wenigen, großen Tasten, die leicht zu bedienen sind.

In der Hotelbranche haben sich bestimmte Ketten darauf spezialisiert, Zimmer für geschäftsreisende Frauen anzubieten. Und Sie wissen, Frauen haben unter Umständen in Hotelzimmern andere Bedürfnisse als Männer. Wir Männer sind einfach: Fernsehapparat und Minibar reicht. Für Frauen dürfen es auch schöne Stoffe, schöne Farben, wie zum Beispiel Pastellfarben sein; ein Schminkspiegel im Bad und ähnliche Ausstattungsmerkmale gehören ebenfalls dazu.

- Domänenwissen
- Technologieerfahrung
- Geistige Fähigkeiten
- Einstellung zur Arbeit
- Einstellung zur Technologie
- Ausbildung
- Sprachliche Fähigkeiten
- Alter (ältere Menschen haben andere Erfahrungen als junge und können sich u.U. nicht mehr so leicht an ganz neue Konzepte anpassen)
- Geschlecht (Männer und Frauen denken und arbeiten vielleicht anders)

Bild 2.10 Welche Fähigkeiten haben Ihre geplanten Nutzer?

Sie sehen, wenn man sich Gedanken darüber macht, für wen wir ein Produkt entwickeln, dann kann man das Produkt auch zielgerechter viel besser gestalten. Als Entwickler von Videorekordern sollte man durchaus daran denken, dass auch eine Seniorin ihre Lieblingssendung aufnehmen möchte. Vielleicht könnte man die Bedienung dann so gestalten, dass sie nicht nur für Teenager und andere Technologiefreaks geeignet ist.

Denken Sie auch daran, welche Fähigkeiten die Benutzer haben; welche sprachlichen Fähigkeiten, welche geistigen Fähigkeiten. Wie ist überhaupt deren Einstellung zur Arbeit? Sie wissen sicherlich: Ältere und jüngere Leute gehen Themen jeweils anders an. Haben Sie schon mal einen Jugendlichen gesehen, der bei einem Videospiel eine Gebrauchsanweisung liest? Wenn das Videospiel nicht selbsterklärend ist oder Freunde erzählen können, wie's geht, wird es einfach weggelegt. In meinem Alter liest man vielleicht noch Gebrauchsanweisungen.

Diese Nutzer wollen wir sehr ernst nehmen und wir wollen Genaueres über diese Personengruppen wissen, damit wir unser Produkt in der richtigen Richtung gestalten können.

Einen Trick möchte ich Ihnen noch mitgeben, nämlich die Entwicklung von Personas.

Personas

Personas sind hypothetische Gestalten von unseren Zielkunden, also diejenigen, die typische Anwender unseres Produkts repräsentieren. Machen Sie's ruhig sehr konkret. Geben Sie ihnen einen Namen, ein Alter, eine berufliche Ausbildung, Hobbies, Neigungen. Machen Sie sie sehr transparent und stellen Sie sich immer wieder die Frage, ob Manfred diese Produktfeatures mögen würde? Würde Manfred Gefallen an bestimmten Produkteigenschaften finden? Diskutieren Sie bei jedem Requirement, ob es für diese Personas geeignet ist. Wir wenden diesen Trick übrigens auch beim Schreiben eines Buchs an. Wann immer wir ein Buch verfassten, haben wir solche Personas sogar gezeichnet und an die Wand gehängt und bei jedem Absatz, den wir formulieren, fragen wir uns immer wieder: Würde Jennifer daran Gefallen finden? Personas zu entwickeln, ist nicht schwierig, aber ein guter Trick, um sich ganz konkret in die späteren Benutzer hineinzuversetzen.

2.7 Weitere Quellen für Anforderungen

Wir haben nun lange Stakeholder behandelt, aber es gibt auch weitere Quellen für Anforderungen (vgl. Bild 2.11). Stakeholder sind garantiert die beste Quelle. Wenn Sie Personen haben, mit denen Sie sprechen können, lernen Sie am schnellsten und am effizientesten über die Anforderungen des Systems.

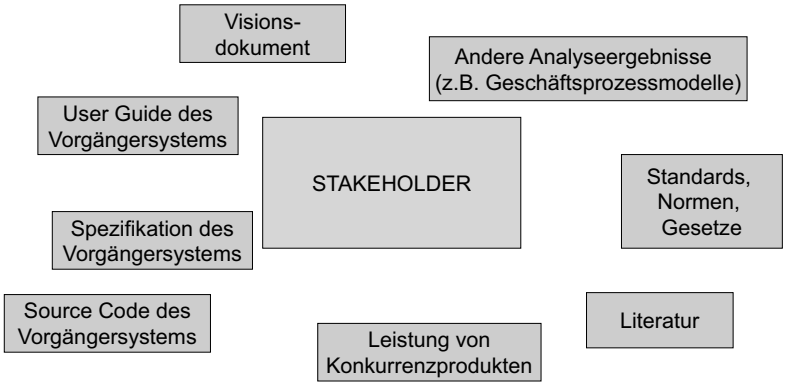


Bild 2.11 Stakeholder und andere Quellen für Anforderungen

Aber es gilt auch die alte Tatsache: 80 % der Anforderungen an ein neues System stecken bereits im alten System, im Vorgängersystem. Wenn Sie also User Guides vom Vorgängersystem haben oder alte Spezifikationen davon oder vielleicht noch schlimmer, Datenbankbeschreibungen und Source Code des Vorgängersystems, erfahren Sie eine ganze Menge über die Anforderungen. Es ist nicht leicht, aus diesen alten Beschreibungen, die zum Teil vielleicht gültig sind, zum Teil vielleicht nicht mehr gültig sind, die Anforderungen herauszukitzeln.

Im Kapitel über Erhebungsmethoden bezeichnen wir das als Softwarearchäologie – aus den alten Aufzeichnungen herausfinden, was das System eigentlich tut und was daher das neue System vielleicht auch wieder tun soll. Wenn Sie keine Ansprechpartner haben, ist das oftmals Ihre einzige Möglichkeit, etwas über die Anforderungen zu erfahren, denn das neue System soll sehr oft mindestens das Gleiche können wie das alte System.

Manchmal haben Sie Visionsdokumente in Firmen, wo irgendjemand Whitepapers geschrieben und darin Strategien vorgegeben hat. Wenn so etwas existiert, ist es ein hervorragender Ausgangspunkt, um konkrete Anforderungen herauszufinden, abgeleitet von Firmenzielen und den Firmenvisionen.

Manche Firmen beschäftigen eigene Abteilungen von Business Analysts für die Geschäftsprozessanalyse, die dann mit Mitteln wie ARIS-Diagrammen, Ereignis-Prozessketten oder BPMN (Business Process Model and Notation) bereits geschäftliche Abläufe formuliert haben, ohne direkten Bezug zur IT, hauptsächlich um das Geschäft besser zu verstehen. Wenn Sie eine derartige Ausgangsbasis haben, ist das schon die halbe Miete. Sie brauchen dann aus den geschäftlichen Vorgaben nur noch die IT-Anforderungen abzuleiten. Wenn diese Voraussetzung nicht gegeben ist, werden wir als Teil unserer Analyseaufgaben auch eine Geschäftsprozessanalyse betreiben.

Auch in Standards, Gesetzen, Normen, Erlassen sind zahlreiche Anforderungen versteckt. Oftmals sind es die Anforderungen, die in Branchen als selbstverständlich angesehen werden und daher nicht explizit von den Stakeholdern benannt werden. Wenn Sie lange Zeit im Bankenumfeld leben, dann ist Ihnen klar, dass Sie sich an Basel II und Basel III halten müssen. Wenn Sie in der Sicherheitstechnik tätig sind, dann gibt es zahlreiche nationale und internationale Normen und Standards, die einzuhalten sind, etwa beim Bau von Eisenbahnen oder Flugzeugen oder im Bereich der Medizintechnik.

Eine weitere Quelle für Anforderungen sind natürlich Konkurrenzprodukte. Sehr oft hört man, wir müssen mindestens das können, was unsere Konkurrenz kann. Nur zwei bis drei zusätzliche USPs („Unique Selling Points, also Punkte, die wir besser machen als die anderen) sollten noch dazukommen. Konkurrenzprodukte sind eine wertvolle Quelle für Anforderungen.

Und – last but not least – gibt es auch zu bestimmten Themen ziemlich viel an Literatur. Immer wieder wird ein Thema systematisch aufbereitet, als Buch oder in Artikelform, und Sie können auch dort sehr viel abschreiben.

Stakeholder sind somit unsere ertragreichste Quelle, um etwas über Anforderungen zu erfahren, aber es gibt viele weitere Quellen. Insbesondere unsere Altsysteme, die Sie vielleicht ablösen wollen, enthalten bereits 80 % der Anforderungen.

2.8 Scope und Kontext

Der dritte wichtige Bestandteil der Arbeit am Projektanfang ist die Festlegung des Scope, des Umfangs des Projekts. Der Scope ist unsere Spielwiese. Das ist der Bereich, in dem wir aktiv werden dürfen, wo wir gestalterisch tätig werden können, wo wir Vorschläge machen können, wie das System sich verhalten soll und was es tun soll.

Der Scope eines Systems ist der Bereich von Dingen, die bei der Entwicklung eines Systems bewusst gestaltet und beeinflusst werden können.

Rund um diesen Scope befindet sich der Kontext, die relevante Umgebung des zu analysierenden Systems (vgl. Bild 2.12). Relevant ist der Kontext deshalb, weil er unser System betreffen kann und weil er Auswirkungen auf das zu erstellende System haben kann; aber der Kontext ist nicht mehr Teil unserer Spielwiese. Er kann nicht ohne Verhandlung verändert werden. Er gehört uns nicht, aber es gibt Berührungspunkte. Wir bekommen Eingaben aus dem Kontext, wir müssen Gesetze beachten, die im Kontext gelten, und wir wollen Ausgaben dorthin liefern.

Noch weiter außen ist die irrelevante Umgebung. Dinge, die dort passieren, interessieren uns nicht; sie haben auch keinerlei Auswirkung auf unser System.

Warum wollen wir über Scope und Kontext sprechen? Der Hauptgrund ist: Wir müssen die Grenzen kennen. Wir haben es in [DeM07] verglichen mit dem Tennis, mit der weißen Linie im Tennis (vgl. Bild 2.13). Die weiße Linie gibt klipp und klar vor, was drin ist und was draußen ist. Früher konnte John McEnroe noch mit dem Schiedsrichter streiten, ob der Ball noch

innerhalb der Grenzen oder außen war. Heutzutage haben wir Kameras zur Überwachung und können durch sofortige Wiedergabe sehen, ob noch irgendein Härchen dieser gelben Filzkugel die Linie berührt hat oder nicht, ob der Ball gut oder nicht mehr gut ist. Diese weiße Linie sollten Sie auch in Ihrem Projekt kennen. Wir wollen sehr früh unseren Scope gegen den Kontext abgrenzen und eventuell auch den Kontext gegen den irrelevanten Bereich.

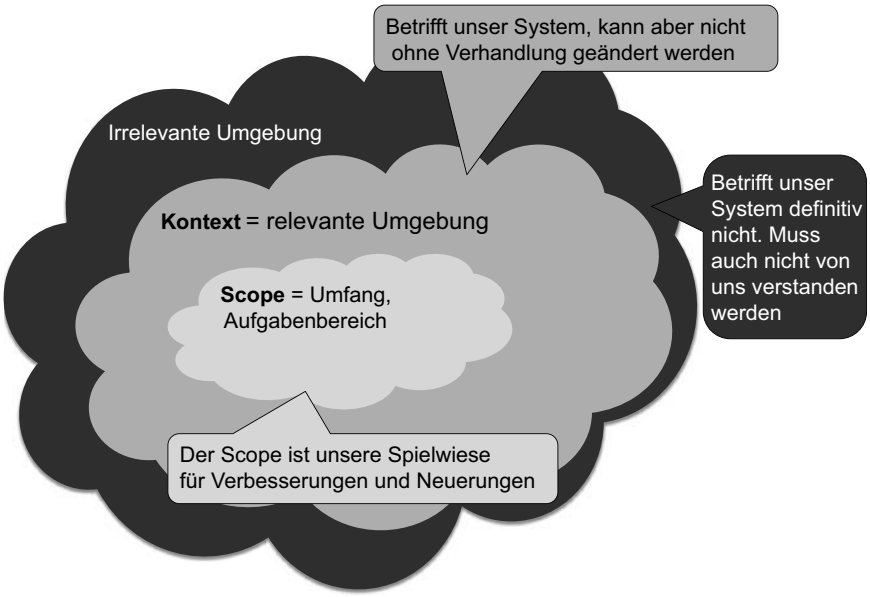


Bild 2.12 Scope und Kontext

Bild 2.13
Die „weiße Linie“ (Foto: David Lee, Agency: Dreamstime.com)

Das Wichtigere ist die Abgrenzung unseres Systems vom Kontext. Wir wollen diese weiße Linie auch im Projekt ziehen. Die weiße Linie rund um den Scope nennen wir Systemgrenze oder Produktgrenze. Die Systemgrenze trennt also Scope vom Kontext. Die weiße Linie rund um den Kontext nennen wir Kontextgrenze, sie trennt Relevantes von Irrelevantem (vgl. Bild 2.14).

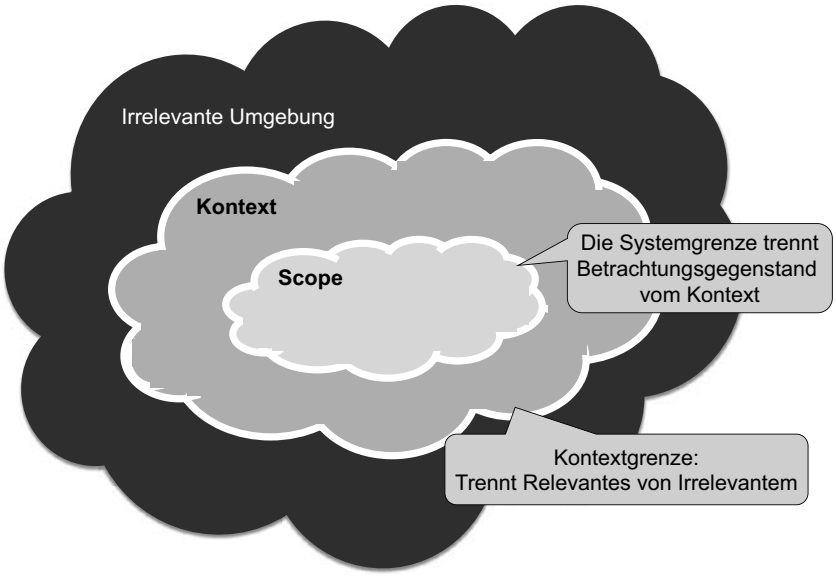


Bild 2.14 System- und Kontextgrenze

Wir gehen in vielen Fällen sogar noch eine Stufe weiter. Wir betrachten beim Scope zwei unterschiedliche Bereiche. Der allerinnerste Bereich ist unser Produkt, das wir bauen wollen, Scope of Product oder System Scope. Rundherum können wir aber auch noch den Scope unseres Arbeitsgebiets abgrenzen (Business-Scope, Scope of Business oder auch Scope of Work genannt.)

Der Produkt-Scope ist der potenziell zu automatisierende Teil unseres Geschäfts. Warum sollten wir nicht nur über unser Produkt nachdenken, sondern auch über unser Arbeitsgebiet? Lassen Sie mich dazu eine kleine Geschichte erzählen. James Robertson aus London hat einen fantastischen Service, um den ich ihn beneide. Wenn er von einer Dienstreise nach Hause kommt, ruft er bei einer Firma an und erzählt am Telefon alles, was er an Einkäufen benötigt: seine zwei Flaschen Whiskey, ein Kasten Mineralwasser, Toilettenpapier, Putzartikel und Ähnliches. Zwei Stunden später steht die Lieferung vor der Tür und er braucht sie nur entgegenzunehmen und zu bezahlen. James hat folgende Überlegung angestellt:

Warum muss ich jedes Mal, wenn ich von einer Dienstreise nach Hause komme, den Lieferanten sagen, was ich brauche. Ich bin seit fünf Jahren Kunde bei diesem Unternehmen. Die wissen, wie viel ich normalerweise nach vier Wochen Abwesenheit brauche. Die könnten doch ihrerseits mit einem Vorschlag kommen, mit einer Liste, die ich nur noch in den Mengen korrigieren oder leicht ergänzen muss. Dieses Mal brauche ich nur eine Flasche Whiskey; ich habe noch etwas übrig. Aber Toilettenpapier ist ausgegangen.

Oder gehen wir noch weiter: Warum installieren die Lieferanten nicht bei mir zuhause Sensoren in den diversen Schränken, die dann feststellen, wie viel von den diversen Artikeln noch vorrätig ist, um automatisch eine gute Vorschlagsliste präsentieren zu können. Naja, Sie sehen, man kann es auch übertreiben mit der Automatisierung, aber die Grenze des Systems, das wir bauen, ist nicht so offensichtlich. Muss ich anrufen und Wünsche durchgeben oder erhalte ich eine Vorschlagsliste, die ich nur noch korrigieren muss.

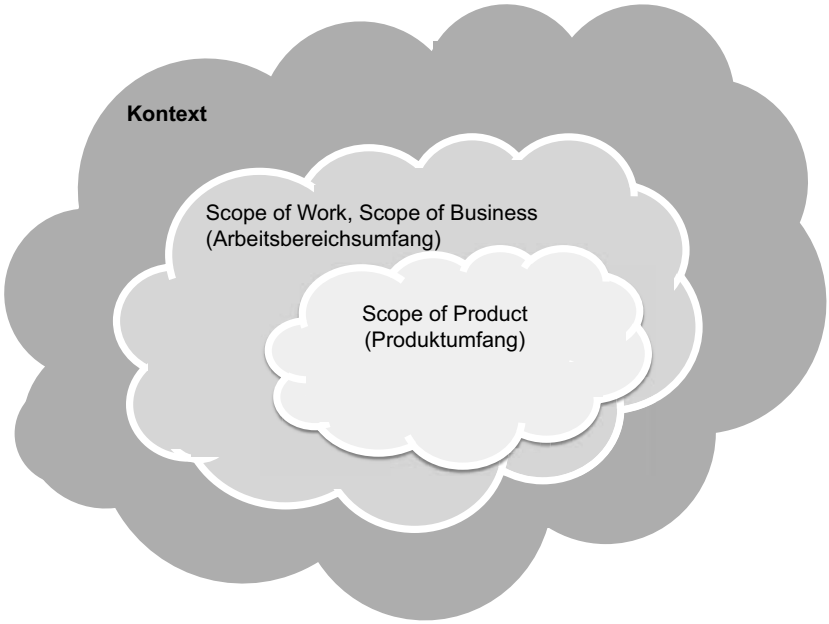


Bild 2.15 Was ist Ihr System? Ein ganzes Arbeitsgebiet oder ein Produkt?

Damit wir diese Entscheidung nicht zu früh treffen, wollen wir zunächst unser Arbeitsumfeld und unsere Geschäftsprozesse betrachten (vgl. Bild 2.15). Erst danach treffen wir eine Entscheidung, wie viel wir davon automatisieren werden und welche Teile weiterhin manuell bleiben sollen.

Ähnliche Überlegungen kann man auch am Beispiel des Buchhandels anstellen. Ich habe in Aachen eine der schöneren und größeren Buchhandlungen in Deutschland, die Mayersche Buchhandlung, wo ich auch regelmäßig Bücher kaufe. Dort kennt mich aber kaum jemand, obwohl auch meine eigenen Bücher im Regal stehen. Wenn ich mich hingegen bei Amazon anmelde, erhalte ich als Begrüßung schon die Meldung: „Herzlich willkommen, Herr Hruschka, wir haben Vorschläge für Sie“. Amazon ist doch im Wesentlichen auch nur eine elektronische Buchhandlung – natürlich inzwischen mit einem wesentlich breiteren Sortiment.

Warum können die einen anderen Service bieten als meine Mayersche Buchhandlung in Aachen. Ja, sie wissen viel mehr über mich. Sie speichern viel mehr Informationen. Wenn ich in meiner Buchhandlung nach dem Einkauf bar bezahle, haben die an der Kasse keine Ahnung, wer hier eingekauft hat. Ok, wenn ich mit Scheckkarte oder Kreditkarte zahle, sehen sie wenigstens meinen Namen. Amazon speichert über lange Zeit hinweg mein ganzes Einkaufsverhalten. Die können sogar noch mehr machen. Amazon bietet mir Musik an, obwohl ich normalerweise keine CDs oder DVDs kaufe, und die angebotene Musik entspricht noch dazu meinem Geschmack. Woher wissen die, welche Musik ich gut finde? Nun ja. Andere Leute, die so komische Bücher lesen, hören auch so schräge Musik und daraus können die Schlüsse ziehen und mir entsprechende Angebote machen, obwohl ich dort noch nie Musik gekauft habe.

Sie sehen also, wenn wir das Geschäftsfeld etwas ausdehnen können, kommen wir vielleicht auf ganz neue Ideen, die wir durch ein Produkt unterstützen könnten. Lassen Sie uns von dem Buchhandelsbeispiel wegkommen. Wir haben das ernsthaft einmal in einer Versicherung angewandt, als wir uns überlegten: Was weiß ein cleverer Versicherungsvertreter in einem kleinen Dorf? Er kennt natürlich alle Personen, er geht zum Stammtisch am Donnerstagabend, er liest die Aushänge am Rathaus und an der Kirche und kommt schon mal auf die Idee, Ausbildungsversicherungen rechtzeitig zu verkaufen oder andere Versicherungen, weil er das Dorfgeschehen kennt. Wir haben uns überlegt, wie viel von dem, was ein gewiefter Versicherungsvertreter im Dorf weiß, ist in der heutigen Versicherungsdatenbank enthalten und können wir Teile von dem, was der gewiefte Versicherungsvertreter an Informationen nutzt, unter Umständen in die Versicherungsdatenbank packen?

Sie sehen, nicht immer ist die Grenze des Produkts so eindeutig festgelegt. Manchmal hilft es, das Arbeitsgebiet zu betrachten, um daraus Schlüsse zu ziehen, wie viele Teile davon automatisiert werden sollten. Deshalb unser Vorschlag: Arbeiten Sie zweistufig im Scope. Legen Sie nicht nur den Produkt-Scope fest, also die Grenze des Produkts, das wir bauen wollen, sondern auch den Business-Scope, unser Arbeitsgebiet. Außerhalb befindet sich jeweils der Kontext. Außerhalb unseres Produkts ist der Produktkontext und außerhalb des Arbeitsgebiets befindet sich der Arbeitsgebietskontext.

2.9 Scope und Analytiker

Nachdem wir die Scope-Begriffe kennen- und unterscheiden gelernt haben, können wir uns nochmal den unterschiedlichen Berufsbildern im Bereich der Systemanalyse zuwenden. Oftmals finden wir in Unternehmen organisatorische Trennungen zwischen denen, die für den Business-Scope zuständig sind (normalerweise Business Analysts), und denen, die für den Produkt-Scope zuständig sind (oft in der IT-Abteilung angesiedelt, mit dem Titel „Requirements Engineer“ oder „(IT-)Systemanalytiker“). Leider sind die organisatorischen Trennungen manchmal sogar unüberwindliche Gräben für die Kommunikation untereinander. Es kann noch schlimmer kommen: Beide Gruppen bestehen darauf, die Ergebnisse Ihrer Analysearbeiten mit ihren Lieblingsnotationen zu dokumentieren, natürlich in ihren Lieblingstools. Oft sind weder die Notationen noch die Tools untereinander verträglich und automatisch aufeinander abbildbar. Die geschäftlichen Forderungen werden von der IT nicht verstanden und die IT-Anforderungen werden vom Geschäft nicht mehr gelesen und kommentiert, weil ja „ohnehin schon gute Business-Anforderungen vorliegen“.

Machen Sie sich das Leben doch nicht gegenseitig schwer. Arbeiten Sie miteinander, statt gegeneinander. Sicherlich liegt der Schwerpunkt in der Business-Analyse im Verstehen des heutigen Systems mit allen seinen Schwachstellen und Unzulänglichkeiten und im Erarbeiten von Anforderungen und Vorschlägen, was sich ändern soll. Ob diese Änderung durch IT-Systeme oder durch organisatorische Maßnahmen (neue Abläufe, neue Aufbauorganisation) erfolgen soll, ist zunächst aus Sicht der Business Analysts nebensächlich.

Bei konkreten Anforderungen an IT-Systeme – worauf das Gebiet „Requirements Engineering“ (leider) manchmal eingeschränkt wird – steht im Vordergrund eine möglichst präzise Vorgabe von Anforderungen als Ausgangsbasis für die Arbeit von IT-Entwicklungsteams. Nicht immer wird dabei (leider) zunächst einmal von dem heutigen Zustand abstrahiert und überlegt, ob man es nicht auch ganz anders machen könnte, um den gleichen – oder sogar einen besseren – geschäftlichen Nutzen zu erzielen. Das führt dann manchmal zu getrennten Arbeiten und getrennten Dokumenten (vgl. Bild 2.16).

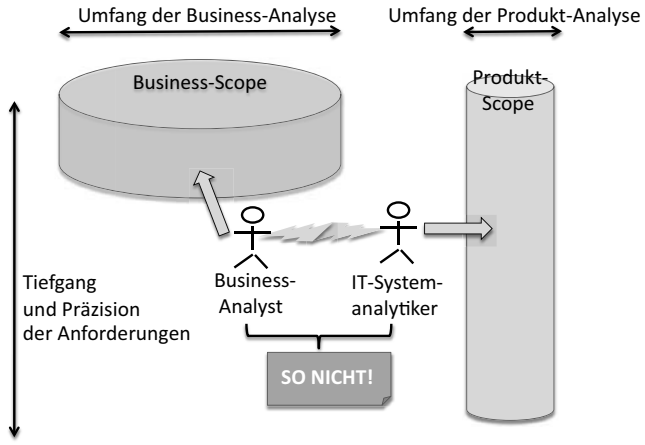


Bild 2.16
Miteinander, nicht gegeneinander

Business Analysts und Requirements Engineers sollten jedoch – ausgehend von Zielen – gemeinsam überlegen, was erreicht werden soll und welche Teile davon mit welchen Mitteln (organisatorisch, technisch, mit Hardware, mit Software, auf andere Weise, ...) entwickelt werden sollen. Dazu sollten Sie eine gemeinsame Sprache über Anforderungen verwenden. Dieser „gemeinsamen Sprache“ in Form von vielen Bildern mit erläuternden Texten oder von umgangssprachlich möglichst präzise ausgedrückten Anforderungen ist der Großteil des Buchs gewidmet (Kapitel 3 – 7). Sie wählen für Ihr Unternehmen dann die Notationen und Werkzeuge aus, die Vorgehensweisen, Methoden und Techniken, mit denen Sie in Ihrem Umfeld am schnellsten und effektivsten zu einem gemeinsamen Verständnis kommen.

Im Umfeld von kommerziellen IT-Projekten ist das eben geschilderte Problem ausgeprägter als bei technischen Systemen, weil es in vielen Fällen, in denen Geschäftsprozesse nicht zu hundert Prozent automatisiert werden, einen deutlicheren Unterschied zwischen den zu erledigenden Geschäftsprozessen und den Teilen davon, die durch IT-Systeme unterstützt werden, gibt.

In technischen Systemen (vor allem in Embedded-Real-Time-Systemen) liegen Business-Scope und Produkt-Scope oftmals durch äußere Randbedingungen sehr viel näher zusammen. Es ist häufig sehr deutlich, wo die Mensch-Maschine-Schnittstelle und wo die Schnittstellen zu anderen technischen Systemen liegen. Aber auch in diesen Fällen sollten Analytiker versuchen, „das größere Bild“ zu sehen. Was will der Arzt wirklich erreichen und wie können ihn diverse Systeme dabei möglichst reibungslos unterstützen? Was erwartet der Autofahrer an Unterstützung von den vielen heutzutage im Auto integrierten Hardware-/Softwaresystemen? Nicht die Details der einzelnen Produkte sollten zunächst im Mittelpunkt stehen, sondern deren Einbettung in den gesamten Prozess des Autofahrens.

Aus diesen Ideen ergibt sich mein „Manifest für die Systemanalyse“:

1. **Arbeiten Sie miteinander, nicht gegeneinander.**
   Ein gemeinsames Verständnis des nächstgrößeren Scope ermöglicht gezielte Entscheidungen über die Schnittstellen Ihres Systems und die präzise Festlegung Ihres System-/Produkt-Scope. Es öffnet die Augen für Chancen und Risiken.
2. **Machen Sie anfangs die Scheuklappen etwas weiter auf.**
   Was auch immer Sie verbessern, effektiver gestalten oder neu machen wollen: Betrachten Sie das Umfeld, in das Ihr System eingebettet ist. Ein besseres Verständnis der übergeordneten Ziele, Abläufe und Strukturen führt – auch bei nicht perfekt geäußerten Detailwünschen – dazu, dass sich ein konkret zu entwickelndes System oder Produkt reibungsloser in übergreifende Prozesse einfügt. Dies ist oft ein Problem in zu agilen Ansätzen: Man schränkt sich viel zu früh auf die Anforderungen an das Produkt ein, statt zunächst im größeren Umfeld nach Optimierungen Ausschau zu halten und dann gezielt die Produkteigenschaften daraus abzuleiten.
3. **Minimieren Sie die Anzahl der Notationen und Tools, mittels derer Sie kommunizieren.**
   Welche das sind, ist beliebig, aber es sollten möglichst wenige sein, die von allen verwendet und verstanden werden. Vermeiden Sie Notationsbrüche und die Notwendigkeit für Werkzeugbrücken. Dadurch werden Vorgehensmodelle nur komplexer und die Anzahl von unterschiedlichen – potenziell widersprüchlichen – Dokumenten vergrößert sich.

Kurzum: Business Analysis und Requirements Engineering sind zwei Seiten derselben Medaille (siehe Bild 2.17). Wenn wir sie überhaupt unterscheiden wollen, dann hauptsächlich bezüglich des Umfangs (Business-Scope gegen Produkt-Scope) und des Tiefgangs der Anforderungen (d. h. deren Detaillierung und Formulierungspräzision):

- Business Analysis konzentriert sich eher auf den Business-Scope, Requirements Engineering setzt den Fokus eher auf den Produkt-Scope.
- Business Analysis darf Anforderungen abstrakter, konzeptioneller, weniger scharf vorgeben; Requirements Engineering muss eine präzise Kommunikation mit den Personen, die eine (IT-)Lösung entwickeln sollen, sicherstellen.

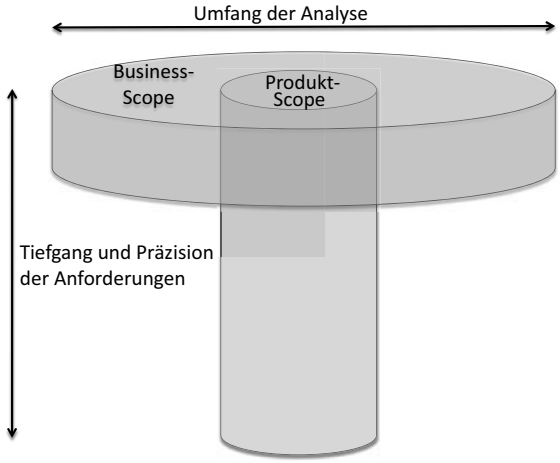


Bild 2.17
Zwei Seiten derselben Medaille

Unterscheiden Sie lieber nicht. Arbeiten Sie gemeinsam mit allen Stakeholdern daran, auf der Basis von heutiger Unzufriedenheit oder neuen Ideen und Randbedingungen – ausgerichtet auf Ziele – iterativ und inkrementell Anforderungen zu formulieren. Vergleichen Sie dazu Bild 1.4 und lesen Sie nochmals die Definitionen von Business Analysis und Requirements Engineering in Kapitel 1.

2.10 Umgang mit Grauzonen

Nicht immer ist diese weiße Linie, die wir in Abschnitt 2.8 beschrieben haben, so eindeutig. Egal, um welche Abgrenzung es sich handelt: Es existieren Grauzonen an all diesen Grenzen (vgl. Bild 2.18). Warum gibt es diese Grauzonen? Warum existiert oft eine Grauzone zwischen dem, was wir bauen wollen, und unserem Arbeitsgebiet. Ein Grund könnte sein: Wir haben noch nicht darüber entschieden, ob es dazugehört oder nicht. Das lässt sich als Systemanalytiker relativ leicht beheben. Sprechen Sie einfach darüber. Warum vermeiden es manche gerne, darüber ein klärendes Gespräch zu führen? Oh, ich kenne Ihre Antwort: Weil der Auftraggeber die falsche Antwort für Sie hat. „Selbstverständlich gehört das auch noch zum Produkt“ und das wollten Sie nicht hören. Sie waren ohnehin schon überlastet mit den bisherigen Anforderungen. Deshalb spricht man manchmal nicht über die Grauzonen. Tun Sie es bitte trotzdem. Spätestens beim Abnahmetest fällt es auf, wenn Teile fehlen. Und dann ist es oft zu spät, um noch Korrekturen anzubringen bzw. Nachbesserungen sind wesentlich teurer. Hören Sie die unangenehmen Nachrichten lieber früher als später.

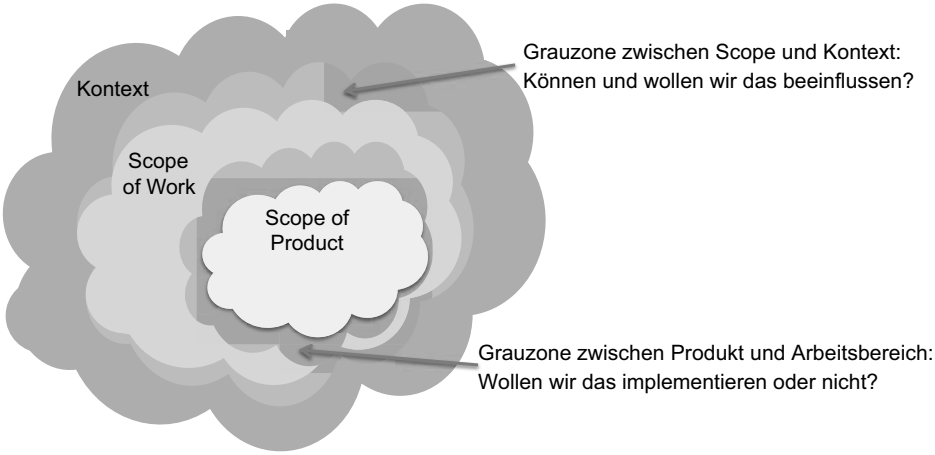


Bild 2.18 Grauzonen an den Grenzen

4 Funktionen genauer betrachtet

Auch im vierten Kapitel beschäftigen wir uns weiter mit den funktionalen Anforderungen. Wir betrachten, wie man einen Ablauf, einen Geschäftsprozess oder einen Produktablauf, den wir als Use Case oder als User Story gefunden haben, systematisch in kleine Teile zerlegen kann, so lange, bis alle Details hinreichend klar an ein Entwicklungsteam kommuniziert werden können und letztendlich auch Abnahmekriterien dafür festgelegt werden können.

4.1 Zerlegungskriterien

Das grundlegende Zerlegungskriterium im Großen haben wir im letzten Kapitel angesprochen: Wir wünschen uns von außen getriggerte oder zeitgetriggerte Prozesse. Das entspricht dem Mandarinenmodelle der IT. Bill Wake [Wake 03] hat das im agilen Umfeld in andere Worte gefasst: Requirements sollten drei Eigenschaften haben:

I *Independent (unabhängig voneinander)*
Wenn Requirements unabhängig voneinander sind, wenig gegenseitige Abhängigkeiten haben und konzeptionell nicht überlappen, dann haben wir die Möglichkeit, ihre Implementierung unabhängig voneinander zu planen und durchzuführen.

N *Negotiable (verhandelbar)*
Obwohl sie schon formuliert sind, bieten sie noch genügend Spielraum für Verhandlungen. Sie stellen am Anfang noch keinen festen, präzisen Vertrag dar, sondern ermöglichen die Diskussion darüber, wie viel oder wie wenig davon man implementieren möchte.

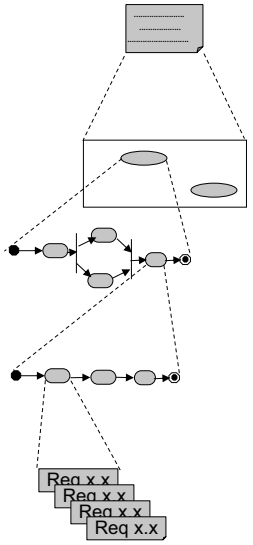
V *Valuable (wertbringend)*
Die Umsetzung einer Anforderung sollte für den Anforderer auf jeden Fall einen Mehrwert liefern, was auch immer das genau sein mag. Anforderungen, in denen niemand einen Mehrwert erblickt, sollten nicht gestellt werden.

Eine Zerlegung in Prozesse liefert diese Unabhängigkeit per Konstruktion, die Verhandelbarkeit, weil noch viele Details offen sind, und den Mehrwert, weil wir bei jedem Prozess jemanden suchen, der den Prozess auf jeden Fall haben möchte.

Experten erkennen an den drei Absätzen die ersten drei Buchstaben der INVEST-Regel von Bill Wake.

Im Laufe der Zeit müssen wir die Schwammigkeit der Verhandelbarkeit natürlich auflösen, indem wir präziser fordern, was genau implementiert werden soll.

Klassischerweise haben wir also solche Prozessketten in ihre einzelnen Schritte zerlegt, um den Prozess präziser zu verstehen. Und komplexe Prozessschritte haben wir in feinere Prozessschritte zerlegt (wie Bild 4.1 zeigt).



Ebene 1: Ziele

Ebene 2: Use Cases: die Hauptprozesse, die das System ausführen soll

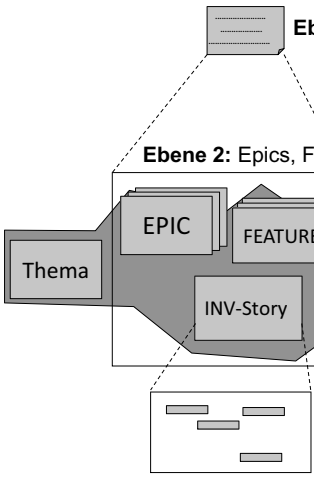
Ebene 3: Hauptaktivitäten in den Use Cases

Ebene 4 – **5** (wenn nötig): Subaktivitäten

Ebene n: atomare Anforderungen

Bild 4.1
Zerlegung von Prozessen in Schritte und Teilschritte

Diese Zerlegung fördert zwar das Verständnis eines komplexen Prozesses, gefährdet aber die Unabhängigkeit und den Wert der Teile. Auch die Teile sollten möglichst unabhängig voneinander sein und immer noch einen gewissen (wenn auch kleineren) Wert bringen, damit wir frühzeitig Teile der Anforderungen umsetzen können und andere auf die lange Bank schieben. Die agile Welt verwendet daher nicht nur eine andere Terminologie (Themes, Epics, Features, ...) und unterschiedliche Notationen für Anforderungen auf unterschiedlichem Granularitätsniveau, sondern auch zusätzliche Kriterien, die bei der Zerlegung berücksichtigt werden sollten (vgl. Bild 4.2).



Ebene 1: Vision, Ziele

Ebene 2: Epics, Features, Themen

Ebene 3 – **n:**
"INVEST" User Stories mit Akzeptanz-Kriterien

Bild 4.2
Agile Zerlegung grober Anforderungen

Neben dem Ablauf in Form von Schritten könnten Sie folgende Kriterien anwenden (vgl. dazu auch das gute Story-Splitting-Cheat-Sheet von [Law]):

- Die CRUD-Zerlegung (Create, Read, Update, Delete): Statt „Kunden verwalten“ könnten Sie vier Anforderungen schaffen: Kunde neu anlegen, Kundendaten lesen, Kundendaten modifizieren und Kunde löschen.
- Variationen in den Geschäftsregeln: Wenn es für einen Vorgang verschiedene Regeln im Geschäft gibt, kann man evtl. zuerst nur eine Regel implementieren und andere Regeln später nachziehen.
- Variationen der Daten: Kann man die Anforderung anhand der Daten aufteilen, z. B. zuerst nur Stammkunden behandeln und später auch Laufkundschaft nachziehen? Oder nur Versicherungsfälle mit Bagatellschäden zuerst bearbeiten und die schwierigen Fälle später.
- Variationen an der Schnittstelle: Kann man die Anforderung eventuell so aufteilen, dass zuerst nur Eingaben über die Tastatur möglich sind, später auch über Touchscreen oder per Sprache? Kann man anfangs eine einfache Schnittstelle zur Verfügung stellen und diese später komplexer machen?
- Einfach/komplex: Kann man die Anforderung so aufteilen, dass man einen einfachen Kern, der den größten Lerneffekt oder den größten Business-Wert liefert, zuerst implementieren kann und weitere Ergänzungen zeitlich nach hinten geschoben werden können?
- Performanz verzögern: Wenn der Hauptaufwand in der Implementierung daher rührt, mehr Performanz zu erreichen, kann man dann evtl. zuerst eine lauffähige Version erzeugen und sie anschließend auf Performanz optimieren? (Vorsicht bei dieser Regel: nachträgliches Hinzufügen wichtiger Qualitätseigenschaften ist manchmal extrem schwierig bis unmöglich!)
- Letzte Möglichkeit: Spikes, Durchstiche: Wenn keines der oben stehenden Kriterien zum Erfolg geführt hat, dann kann man sich fragen, ob es einen kleinen Teil der Anforderung gibt, der schon gut verstanden ist, sodass man damit in der Implementierung beginnen kann. Ansonsten ist die Frage: Können wir irgendetwas tun, um das Problem besser in den Griff zu bekommen, also uns durch Zwischenschritte Wissen einkaufen, wie das Problem zerlegt werden kann?

Bild 4.3 fasst das Grundprinzip dieser Zerlegungsmöglichkeiten grafisch zusammen. Eine Zerlegung in Prozessschritte birgt die Gefahr, dass die Implementierung eines einzelnen Schritts keinen Mehrwert erbringt. Deshalb sollten Sie lieber versuchen, die grobe Funktionalität so abzugrenzen, dass jeder Teil davon unabhängig implementiert werden kann UND Mehrwert erbringt.

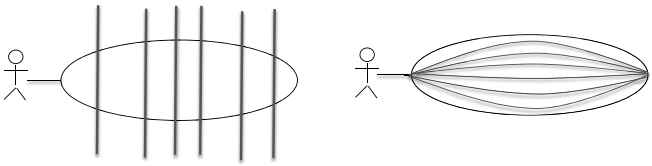


Bild 4.3
Slicing statt Prozessschritte

Auch Ivar Jacobson hat mit seinen Use Cases 2.0 diese Idee ins Spiel gebracht [Jac11] und statt User Stories vorgeschlagen, besser über Use Case Slices nachzudenken.

Auf den Punkt gebracht: Man kann einen groben Prozess immer dadurch zerlegen, dass man

1. Alternativen weglässt (z. B. Normalablauf zuerst, Sonderfälle später).
2. Optionen weglässt (Dinge, die man nicht unbedingt im ersten Anlauf implementieren muss).
3. Schritte weglässt, für die am Anfang noch die Möglichkeit besteht, sie weiterhin manuell zu erledigen und erst später zu automatisieren.

Achten Sie also beim Zerlegen nicht nur auf mehr Verständlichkeit einer ursprünglich komplexen Anforderungen, sondern auch darauf, dass man Teile bereits frühzeitig (wertbringend) umsetzen kann.

4.2 Wo hört man auf?

Die offensichtliche Antwort auf die Frage „Wie weit muss man gehen in der Zerlegung?“, ist wohl: Bis das Entwicklungsteam keine großen Rückfragen mehr hat und glaubt, alles verstanden zu haben. Formaler ausgedrückt hat die agile Welt dafür ein „Definition of Ready“ (DoR) eingeführt.

Sie ist im Wesentlichen durch die volle INVEST-Regel definiert. Die Teile sollten nicht nur I, N und V erfüllen – was wir für jegliche Art von Anforderung empfehlen, sondern auch E, S und T. Die drei Buchstaben stehen für:

E *Estimable (schätzbar)*
Wenn eine Anforderung noch nicht einmal grob schätzbar ist, ist sie definitiv noch zu vage und muss weiter präzisiert und zerlegt werden.

S *Small (kein genug für eine Iteration)*
Wir wollen die Anforderung einem Entwicklungsteam übergeben und am Ende der Iteration prüfen können, ob sie erfüllt ist oder nicht. Deshalb muss jede Anforderung klein genug sein, um in einen Sprint (oder einen Release) zu passen. In Wahrheit wollen wir dem Team für eine Iteration ja nicht nur eine Anforderung übergeben, sondern eine Menge an Stories. Man geht heute eher davon aus, dass jede Story so klein sein muss, dass zirka 6 – 10 davon in einer Iteration erledigt werden können.

T *Testable (testbar)*
Bei jeder Anforderung sollte dem Team klar sein, was der Auftraggeber oder Product Owner am Ende der Iteration prüfen möchte. Auf der Rückseite der Story-Kärtchen werden deshalb die Abnahmekriterien notiert.

Eine Anforderung muss diese sechs Kriterien erfüllen und vom Team verstanden werden. Dann ist sie „Ready“. Die DoR ist das Gegenstück zur Definition of Done (DoD), die im agilen Umfeld die Kriterien beschreibt, die das Team erfüllt haben muss, bevor es „fertig“ melden darf.

4.3 Top-down oder bottom-up?

Jetzt haben wir die ganze Zeit so getan, als würden wir ausgehend von Use Case oder EPICS solange zerlegen, bis wir präzise genug sind. Machen wir das wirklich top-down oder machen wir das Ganze vielleicht bottom-up?

Im Endeffekt hätten wir gerne die komplette Hierarchie, wie in Bild 4.1 oder in Bild 4.2 dargestellt.

Die Frage ist nur: Wie kommen wir dahin? Sie haben mindestens zwei große Alternativen (vgl. Bild 4.4 bzw. Bild 4.5). Entweder Sie gehen wirklich top-down vor. Das machen wir vor allem, wenn Sie Ansprechpartner haben, die sehr abstrakt denken können, die wissen, worauf sie hinauswollen, die Ihnen die Kernelemente des Prozesses zunächst auf einer Seite schildern können, mit denen man dann später über die Details spricht, und zwar so lange, bis es präzise genug ist.

Sie können das Ganze aber auch anders herum, nämlich bottom-up machen. Sie haben eine Vielzahl von Anforderungen erhalten. Und Sie erinnern sich an die Einleitung des letzten Kapitels. Wir haben festgestellt, dass Kunden gleichzeitig auf allen Abstraktionsebenen sprechen und fordern. Manche geben Ihnen grobe Anforderungen, manche geben Ihnen mittelfeine Anforderungen, andere liefern Ihnen sehr feine Anforderungen. Ihre Aufgabe ist es, das in eine derartige Hierarchie einzubauen. Das gilt zunächst nur für die funktionalen Anforderungen, die Anforderungen bezüglich gewünschter Abläufe; wir kommen später noch zu Qualitätsanforderungen, Datenanforderungen und vielen Randbedingungen. Ob Sie es eher top-down oder bottom-up machen, ist Geschmackssache.

Wenn Sie fertig sind, sollte es eine derartige baumartige Darstellung sein.

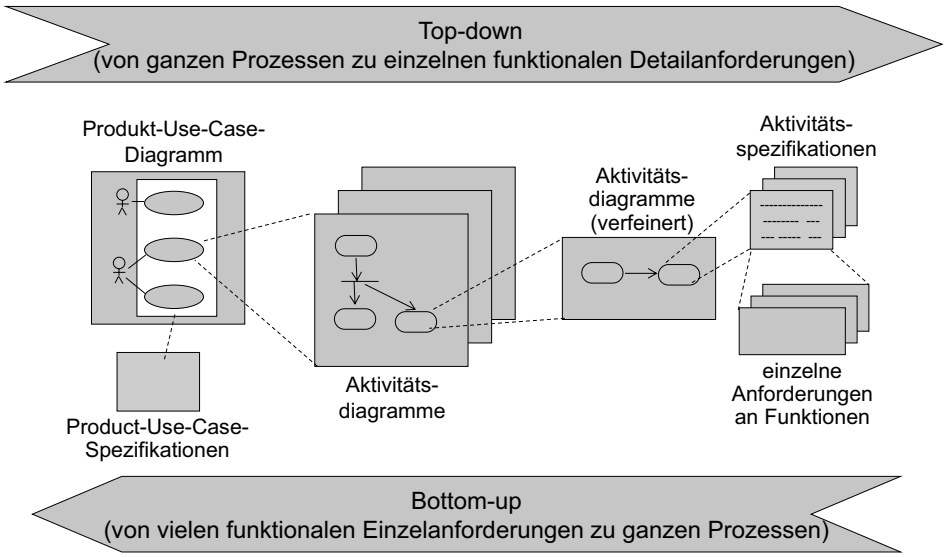


Bild 4.4 Top-down oder bottom-up (klassisch)?

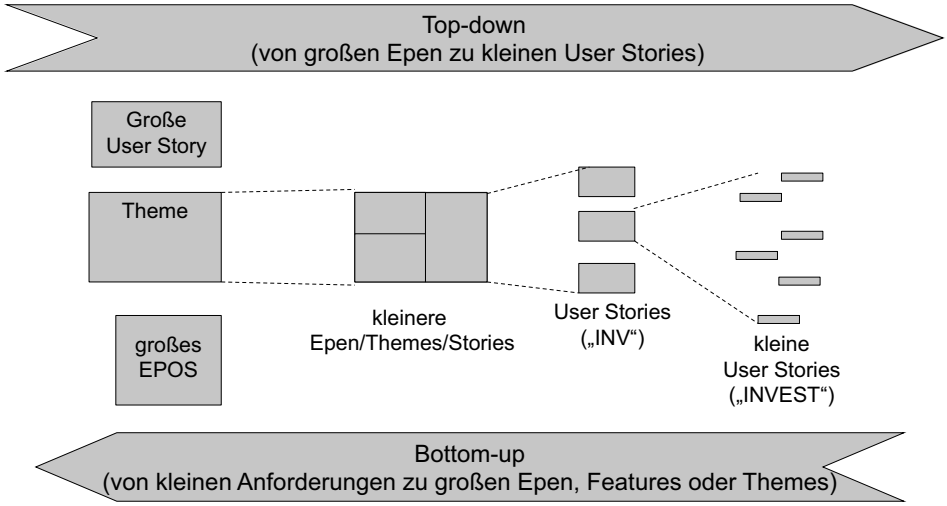


Bild 4.5 Top-down oder bottom-up (agil)

Das erinnert mich immer an den Mathematiker, der an die Tafel tritt und einen Beweis vorführt. Herr Professor tritt stolz an die Tafel und beginnt mit „Annahme 1, Annahme 2, Annahme 3“. Dann ein Strich darunter. Zwischenfolgerung abgeleitet aus 1 + 3, Zwischenfolgerung abgeleitet aus 2 + 3, Strich darunter. So trägt er den ganzen Beweis sehr folgelogisch vor. Ich garantiere Ihnen: So wie er ihn gerade erzählt, hat er den Beweis nicht entdeckt. Entdeckt hat er ihn mit sehr viel mehr Trial und Error, mit sehr viel mehr Versuchen. Aber wenn er die Lösung kennt, präsentiert er sie geordnet.

Das ist auch das Kunststück für funktionale Anforderungen. Wenn Sie die Anforderungen kennen, präsentieren und kommunizieren Sie diese in einer geordneten Reihenfolge von Zielen bis zu niedrigen detaillierten feinen Anforderungen.

Gefunden haben Sie sie unter Umständen ganz anders. Vor ein paar Jahren schrieb ich einen Artikel im Objektspektrum unter dem Titel „Bottom-up Use Cases“[Hru07]. Darin habe ich an einer Fallstudie gezeigt, wie man das Ganze von unten her aufrollen und nachträglich solche Prozesse bilden kann. Die Zusammenfassung und Bündelung von einzelnen Anforderungen zu Prozessen ist auf jeden Fall sinnvoll. Sie erinnern sich an das Mandarinenmodell aus Abschnitt 3.5, dass wir sehr disjunkte Prozesse finden wollen, aber wir haben sie vielleicht nicht von oben kommend gefunden.

Ich möchte Ihnen an der Stelle als Beispiel ein weiteres abstrahiertes Inhaltsverzeichnis einer ganzen Spezifikation zeigen; wiederum von dem FlexRay-System. Ignorieren Sie für den Moment die grauen Rechtecke und konzentrieren Sie sich nur auf die weißen Ellipsen und Rechtecke mit abgerundeten Ecken.

Sie sehen auf der obersten Zerlegungsebene sieben Use Cases. Und Sie sehen, dass bei dem zweiten, dem sechsten und dem siebten Use Case keine weiteren Zerlegungen notwendig waren. Die Use-Case-Beschreibung hat ausgereicht, diesen Prozess jeweils detailliert genug zu spezifizieren. Bei ein paar anderen, dem dritten, vierten und fünften Use Case, sehen Sie eine Zerlegung, zweimal in zwei Teilschritte, einmal in zehn Schritte.

Der allererste Use Case (links) ist über drei Ebenen zerlegt. FlexRay ist ein Protokoll, da geht's um Kommunikation, um Senden und Empfangen. Diese beiden Abläufe benötigten die komplexeste Zerlegung über drei Ebenen hinweg, bis alle notwendigen Details spezifiziert waren.

Das ist ein grafisches Inhaltsverzeichnis einer industriellen, hundertseitigen Spezifikation, die nach dieser Struktur aufgebaut wurde. Wir kommen in Abschnitt 6.6 noch zu den grauen Teilen von Bild 4.6. Das sind Zustandsautomaten, die die Ausführung der Funktionen einschränken. Denn diese Aktivitäten und diese Use Cases durften nicht zu beliebigen Zeitpunkten ablaufen, sondern wurden von Zustandsmaschinen kontrolliert. In einem bestimmten Zustand durften nur bestimmte von diesen Aktivitäten laufen, deshalb sind sie hier steuernd über die Aktivitäten darübergesetzt.

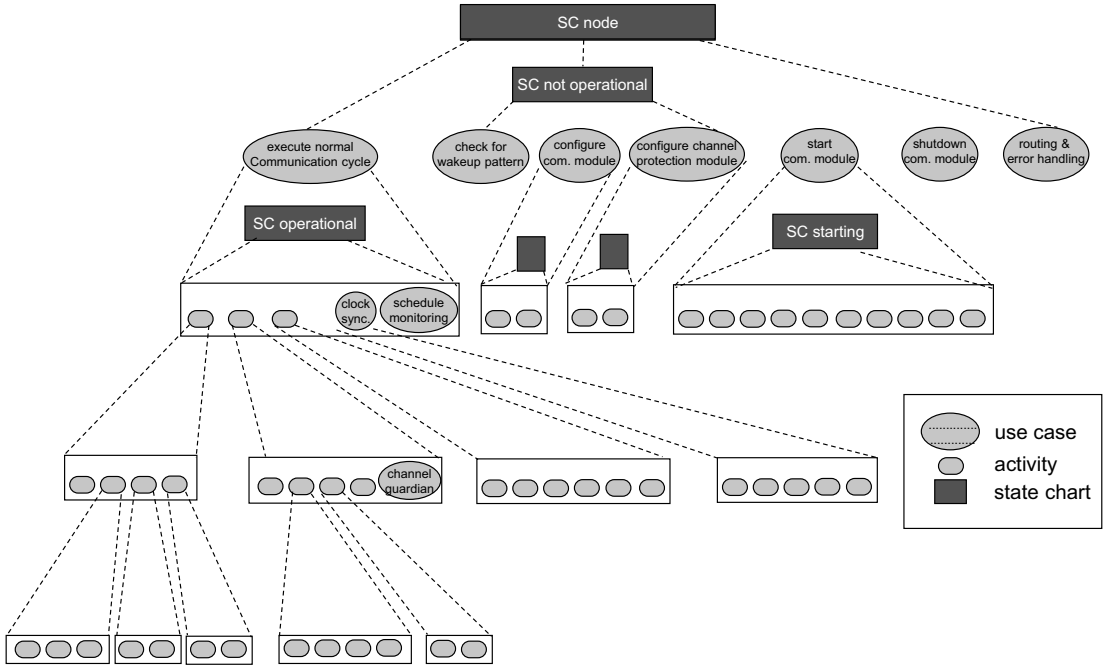


Bild 4.6 Die Struktur einer industriellen Spezifikation

Story Maps

Um einen derartigen Überblick über Anforderungen unterschiedlicher Granularität herzustellen, hat Jeff Patton vorgeschlagen, den Product Backlog nicht einfach linear zu ordnen, sondern ihn zweidimensional als Story Map darzustellen [Pat14] (vgl. Bild 4.7).

Dabei werden ein bis zwei Ebenen von abstrakten Anforderungen an ein Produkt (also z. B. Epics und Features) oberhalb einer Trennlinie so angeordnet, dass man jemanden anhand dieser groben Anforderungen die ganze Geschichte über sein Produkt erzählen kann. Die Reihenfolge oberhalb der Linie ist keine zeitliche Reihenfolge dieses Features und Epics. Sie stellt nur die Erzählrichtung dar. Man erläutert zuerst die Features weiter links und tastet sich nach rechts bei der Erzählung durch.

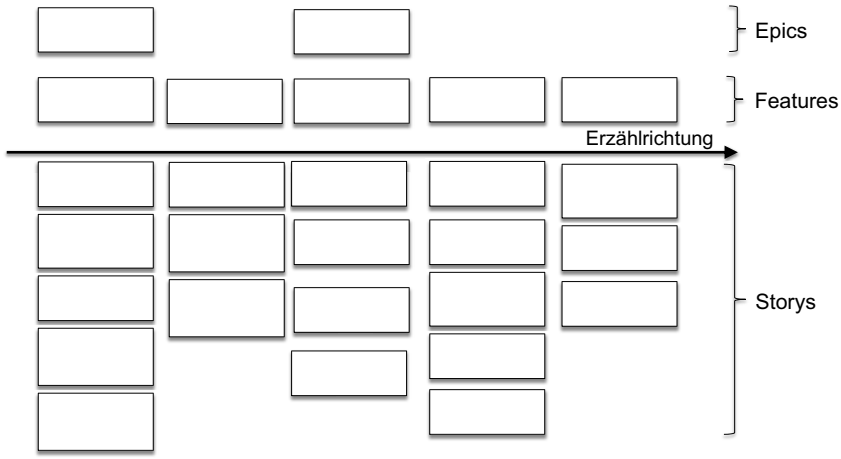


Bild 4.7 Story Map als Überblick im Großen

Unterhalb der Linie kann man dann zu jedem Epic oder Feature beliebig viele Storys unterbringen. Somit bleibt auch nach einer Zerlegung einer groben Anforderung in feine Storys der Zusammenhang zu dem Thema noch gewahrt.

Anmerkung: Story Maps können noch mehr, als nur den Überblick bewahren. In Kapitel 11 werden wir sie noch als Ausdrucksmittel zur Unterstützung der Releaseplanung kennenlernen.

4.4 Zusammenfassung

In Kapitel 4 haben wir gelernt, wie Sie einmal gefundene Abläufe in Ihrem Business oder in Ihrem geplanten IT-System durch Zerlegung so weit präzisieren können, dass diese von einem Implementierungsteam umgesetzt werden können. Je nach Ihrer Methode werden Sie Use-Cases in Prozessschritte zerlegen und diese weiter verfeinern. Oder aber Epics und Features in INVEST-fähige Stories zerlegen. Ob Sie dabei top-down oder bottom-up arbeiten, bleibt Ihnen überlassen. Sie sollten nur jederzeit in Ihrem Projekt einerseits genügend Überblick bewahren, damit Sie nicht den Wald vor lauter Bäumen nicht mehr sehen, andererseits natürlich so präzise werden, dass Teile der Anforderungen frühzeitig erledigt werden können.

Intermezzo

Bevor wir uns im Detail verlieren – oder nach so vielen Buchseiten schon verloren haben –, gehen wir nochmals zurück, ganz an den Anfang. Business Analysis und Requirements Engineering streben nach einem guten Verständnis heutiger Geschäftsprozesse oder Produktfunktionalität, damit wir gezielt Vorgaben für Verbesserungen, Erweiterungen, Modifikationen etc. machen können. Die Ergebnisse dieser Analysetätigkeiten und die Erkenntnisse wollen wir irgendwie kommunizieren, „zu Papier bringen" oder festhalten, damit sie uns als Diskussionsgrundlage dienen und auch als Vorgaben für diejenigen, die uns helfen, die erkannten Probleme zu lösen und unsere Produkte und Prozesse nachhaltig zu verbessern.

Um die Ergebnisse des Analyseprozesses festzuhalten, stehen uns heute eine Vielzahl unterschiedlicher Ausdrucksmittel zur Verfügung. Ich möchte Ihnen, bevor wir in Dutzende von Notationen absteigen, in einem Überblick die Alternativen aufzeigen, die Ihnen zum Festhalten zur Verfügung stehen.

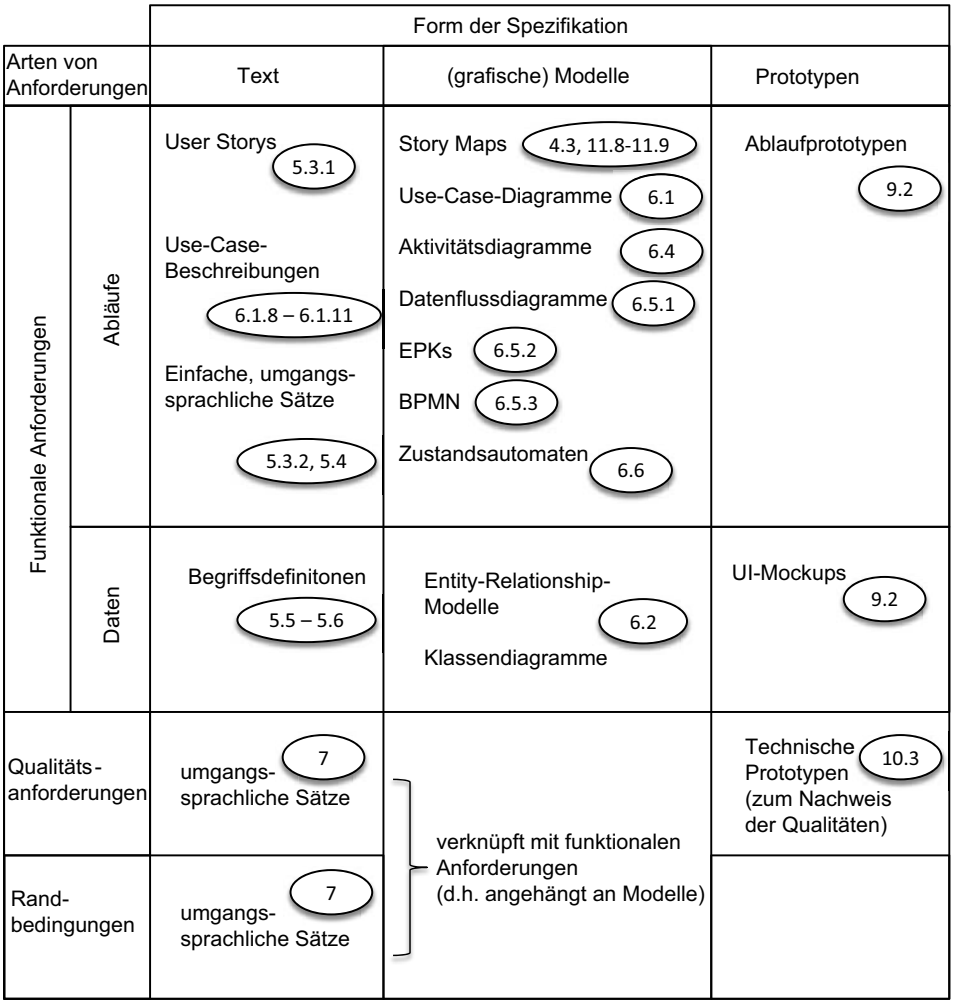
In der folgenden großen Matrix sehen Sie in der **vertikalen** Achse die unterschiedlichen Arten der Anforderungen: Wir studieren einerseits unsere Geschäftsprozesse oder Produktprozess oder Funktionalitäten, natürlich zusammen mit den Dingen, die dabei verbraucht, bearbeitet oder erzeugt werden, den Daten. Und wir interessieren uns für Qualitätseigenschaften und Randbedingungen.

Betrachten wir das mit etwas anderen Worten. Wir haben im Wesentlichen funktionale Anforderungen (das, was ein System oder ein Produkt tun soll; die Funktionen und die Daten und das Verhalten des Systems) und wir haben daneben noch die beiden anderen Kategorien Qualitätsanforderungen und Randbedingungen. Die funktionalen Anforderungen können wir in die geschäftlichen Abläufe oder Produktfunktionalität und in die Daten gliedern. Wir betreiben ja schließlich Informationsverarbeitung: Informationen sind unsere betrieblichen Daten oder Produktdaten; und die Verarbeitung sind die dazugehörigen Abläufe.

In der **horizontalen** Achse sehen Sie die drei prinzipiellen Möglichkeiten, die Ergebnisse zu erfassen. Alle diese Anforderungen, funktionale Anforderungen und nichtfunktionale Anforderungen, also Qualitätsanforderungen und Randbedingungen, kann ich auf drei unterschiedliche Arten darstellen.

1. Die Form der Spezifikation kann, wie in vielen Projekten heute noch häufig verwendet, einfach Text sein; umgangssprachliche, deutsche, englische oder andere natürlichsprachige Absätze und Sätze.
2. Wir haben seit 40 Jahren aber auch eine Vielzahl von grafischen Modellen, die wir nutzen können, um die unterschiedlichen Arten der Anforderungen zu Papier zu bringen.
3. Und last but not least stehen uns auch Prototypen zur Verfügung, um Anforderungen transparenter zu machen.

Neben den verschiedenen Notationen sehen Sie Zahlen in Ellipsen, die auf die Buchkapitel verweisen, wo Sie nähere Informationen zu diesen Notationen finden. Nutzen Sie diese Abbildung als Wegweiser durch den Rest des Buchs, wenn Sie an der einen oder anderen Notation mehr Interesse haben.



Unterschiedliche Spezifikation der Analyseergebnisse

Sie haben also die Qual der Wahl beim Kommunizieren und Dokumentieren von Anforderungen. Wollen wir schreiben, wollen wir malen oder wollen wir prototypen? Und das gilt für alle Arten von Anforderungen, für die beiden Arten von funktionalen Anforderungen (für Abläufe und Daten) wie auch für die beiden Arten von nichtfunktionalen Anforderungen, für Qualitätsanforderungen und für Randbedingungen.

Stichwortverzeichnis