Chris Rupp

„Siehst du den Horizont? Direkt über´m Boden fängt der Himmel an. Und wäre ich dort, würde ich wetten, dass ich ihn erreichen kann. Doch hier hat es den Anschein, bin ich dafür zu klein."

Thomas D. im Lied „Rückenwind"

1

Anforderungsqualität – Der Maßstab Ihres Projekterfolges

Fragen, die dieses Kapitel beantwortet:

- Warum benötige ich Requirements Engineering?
- Welche Argumente sprechen für Requirements Management?
- Welche typischen Probleme treten heute in der Softwareentwicklung auf und welchen Bezug haben Sie zu den Anforderungen?
- Wie unterscheide ich exzellente von schlechten Anforderungen?
- Was sind die Kriterien für ein exzellentes Anforderungsdokument?
- Welche Standards fordern welche Aspekte von Requirements Engineering?

1.1 Motivation für eine erfolgreiche Systemanalyse

Erfolgsindikator Anforderungsqualität

Die Analyse als erster Schritt der Systementwicklung entscheidet maßgeblich über den Erfolg oder Misserfolg eines Projektes. Verschiedene Untersuchungen zeigen, dass die Mehrzahl aller Fehler in der Softwareproduktion in der Analyse entstehen. Fehler, die in einer frühen Phase der Softwareentwicklung gemacht und erst in späteren Phasen behoben werden (wenn es dafür eigentlich schon zu spät ist), sorgen für einen Summationseffekt, bei dem sich die Fehler fortpflanzen und potenzieren. Tatsächlich lassen sich ca. 65% der schwerwiegenden Fehler in Programmen auf Unzulänglichkeiten in der Analyse zurückführen.

Um die Explosion der beiden kritischen Faktoren Kosten und Entwicklungszeit zu vermeiden, müssen die Ursachen von Analysefehlern beseitigt werden. Verbesserungen greifen während der Analyse wesentlich effektiver als während des Design oder gar während der Implementierung. Je früher ein Fehler gefunden und behoben wird, desto weniger Folgefehler werden verursacht, desto eher können die relevanten Entwicklungsparameter Kosten und Zeit in Grenzen gehalten werden.

1.1.1 Definition und Aufgaben einer Anforderung

Eine Anforderung ist eine Aussage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen.

Die Aufgaben von Anforderungen lassen sich in *primäre* und *sekundäre* Aufgaben unterteilen. Die Unterscheidung drückt aus, ob sich die Anforderungen unmittelbar oder nur mittelbar auf das Projekt auswirken.

1.1.2 Primäre Aufgaben einer Anforderung

Die primären Aufgaben einer Anforderung wirken unmittelbar im Softwareentwicklungsprozess. Eine Anforderung dient in der Softwareentwicklung als:

Kommunikationsgrundlage

Kommunikationsgrundlage: Anforderungen dienen allen an der Softwareentwicklung Beteiligten als Kommunikations-, Diskussions- und Argumentati-

onsgrundlage. Ihre Aufgabe ist es, das gemeinsame Verständnis und Wissen der Teammitglieder widerzuspiegeln.

Vertragsgrundlage

Grundlage für die Ausschreibung und Vertragsgestaltung: Das Anforderungsdokument (die Menge beschriebener Anforderungen) stellt eine Grundlage für die Ausschreibung und die anschließende Vertragsgestaltung dar. Ferner wird bei der späteren Abnahme basierend auf dem Anforderungsdokument das Ergebnis getestet. Als zu beanstandender Fehler gilt dann nur noch eine Abweichung des Ist-Zustandes (wie reagiert das System tatsächlich?) gegenüber dem festgeschriebenen Sollzustand (den definierten Anforderungen) im Anforderungsdokument.

Grundlage für Integration, Wartung

Grundlage für Systemintegration, Wartung und Pflege: Die Bedeutung der Anforderungen beschränkt sich nicht ausschließlich auf den Zeitraum zwischen Anforderungserhebung und Abnahme des Systems. Vielmehr müssen Systemänderungen und Erweiterungen auf Basis der dokumentierten Anforderungen erfolgen.

Grundlage für die Systemarchitektur

Grundlage für die Systemarchitektur: Anforderungen haben einen wesentlichen Einfluss auf die Systemarchitektur des zu erstellenden Systems. Eine genaue Definition aller Anforderungen ist von daher unumgänglich, wenn man zum Beispiel spätere „Performance-Überraschungen“ vermeiden möchte.

1.1.3 Sekundäre Aufgaben einer Anforderung

Die sekundären Aufgaben einer Anforderung wirken sich auch außerhalb des Entwicklungsprozesses aus, das heißt, sie wirken schon in der Planungsphase, sowie nach der Fertigstellung des zu entwickelnden Systems. Allerdings variieren sie je nach Branche, Einsatzgebiet und Verwendung des zu entwickelnden Produkts.

Rationalisierungspotenziale

Eröffnung von Rationalisierungspotenzialen: Im Zuge der Definition der Anforderungen lassen sich schon im Vorfeld der Einführung eines neuen Systems Rationalisierungspotenziale, zum Beispiel in der Unternehmensorganisation oder in den Arbeitsabläufen erkennen.

Kundennutzen

Optimierung des Kundennutzens: Im Rahmen der Anforderungserhebung werden auch Kundenwünsche oder Optimierungspotenziale des Kunden erhoben.

Mitarbeiterzufriedenheit

Erhöhung der Mitarbeiterzufriedenheit: Werden die vom Mitarbeiter explizit geforderten und definierten Anforderungen umgesetzt, kann das eine erhöhte Mitarbeiterzufriedenheit und Akzeptanz des neuen Systems zur Folge haben. Ein Mitarbeiter ist zufriedener, wenn er im neuen System die zuvor bemängelten Unzulänglichkeiten behoben sieht.

1.1.4 Warum benötigen Sie ein professionelles Requirements Engineering?

Requirements Engineering befasst sich mit dem Auffinden, der Dokumentation und der Überprüfung von Anforderungen an ein System. Requirements Engineering beschreibt den systematischen Weg von der Idee über die Ziele zur vollständigen Dokumentation der Anforderungen.

Klassischerweise finden sich in der Projektrealität allerdings häufig die im Folgenden beschriebenen Varianten eines Vorgehens:

Zu vage Anforderungsdokumente

Variante: Ein nicht besonders ausführliches Anforderungsdokument, das zu großen Teilen aus vagen Formulierungen in natürlicher Sprache (Prosa) besteht, wird dem Analytiker vorgelegt. Dieses Dokument umfasst angeblich alle Anwenderanforderungen. Der Analytiker überführt dieses Prosawerk soweit möglich in eine semiformale Darstellung, zum Beispiel in ein objektorientiertes Analysemodell. Anhand der objektorientierten Darstellung forscht der Analytiker nach Aspekten, die er zum Beispiel durch Befragung der Anwender ermittelt. Danach erweitert und ergänzt er das lückenhafte Modell. Häufig fließen in das Modell aber auch die Interpretationen und Phantasien des Modellierers mit ein, die von der Realität des Endanwenders weit entfernt sind. Das ursprüngliche Anforderungsdokument verliert jegliche Relevanz, da alle neuen Anforderungen lediglich im Modell stecken und das Pflichtenheft nicht „nachgeführt“ wird.

Zu formale Anforderungsdokumente

Variante: Das Team, welches das Anforderungsdokument erstellt, besteht aus sehr technisch orientierten Mitarbeitern, die die Wünsche der Anwender gleich in Form von Tabellen, Zustandsautomaten oder ähnlich formalisierten Darstellungstechniken niederschreiben. Die Anwender als eigentliche Wissensträger werden selbstverständlich in die Analyse des Systems mit einbezogen und mit den seitenlangen sehr technisch angehauchten Spezifikationen konfrontiert. Die Techniker ernten daraufhin wissendes, etwas verschüchtertes Nicken der Anwender, die sich keine Blöße geben wollen und nicht zugeben, dass sie sehr wenig verstanden haben. Zufrieden beginnen die Techniker mit der Umsetzung der unverstandenen Spezifikation.

Es ließen sich hier noch andere Varianten dieses Spiels aufzählen, doch kennen Sie vermutlich Ihre ganz persönliche Variante, die Sie bereits in Projekten erlebt haben.

Doch nun zur Beantwortung der Frage: „Warum benötigen Sie ein professionelles Requirements Engineering?“

Requirements Engineering beschreibt einen systematischen Weg von der Projektidee über die Ziele zu einem vollständigen Satz von Anforderungen. Neben dem Vorgehen werden auch die Qualitätsmerkmale definiert, die jede ein-

zelne Anforderung, aber auch das gesamte Anforderungsdokument erfüllen müssen. Eine Beschreibung des Vorgehens und der geforderten Qualität ist nötig, weil es nicht allgemein bekannt und selbstverständlich ist und die an der Systemanalyse beteiligten Mitarbeiter eine klare Beschreibung der Methodik und der Zielqualität benötigen.

1.2 Warum benötigen Sie ein funktionierendes Requirements Management?

Richtlinien und Verfahren

Durch Requirements Management werden dem Analytiker Techniken und Richtlinien an die Hand gegeben, mit denen Anforderungen und die dazugehörigen Informationen verwaltet werden können. Durch Requirements Management wird ein definiertes Verfahren etabliert, das es ermöglicht, die oftmals unüberschaubare Anzahl an Anforderungen komplexer Projekte zu beherrschen. Anforderungsänderungen werden erst durch die Einführung von Requirements-Management-Techniken systematisch handhabbar.

Versions-Wirrwarr

Auch hier existieren in der Realität derzeit Vorgehensvarianten, die dem Projekterfolg nicht gerade dienlich sind. Nicht selten erleben wir, dass Anforderungen in einem Word-Dokument verwaltet werden, welches auf der lokalen Platte eines Projektbeteiligten liegt. Änderungen werden dann häufig zentral von einer Person durchgeführt. Dies führt dazu, dass das Dokument zwar konsistent bleibt, Änderungen sich aber lange hinziehen und nur zu gewissen Zeitpunkten eingepflegt werden (das heißt, man arbeitet fast immer auf der Basis eines veralteten Dokumentes). Bei einer anderen Variante werden die Änderungen fortlaufend eingepflegt und damit ständig neue Versionen des Dokumentes generiert und versandt. Dies führt kurzfristig zu einer Versionsvielfalt, bei der man sich nach einigen Wochen darauf verlassen kann, dass jeder Projektmitarbeiter auf der Basis einer anderen Version arbeitet. Sollten die neuen Versionen des Dokumentes nicht versandt werden, sondern auf einem zentralen, jedem zugänglichen Laufwerk zur Verfügung stehen, so hat dies lediglich den Effekt, dass jeder Zugriff auf die neueste Version des Dokumentes hätte – die ausgedruckte Version des Dokumentes, die auf dem Schreibtisch liegt und nach der wirklich gearbeitet wird, repräsentiert allerdings meist nicht den aktuellsten Stand.

Paralleles, verteiltes Arbeiten

Requirements Management versucht, die genannten Probleme zu lösen. Es ermöglicht ein paralleles und weltweit verteiltes Arbeiten an den Anforderungen eines Projekts und führt dadurch zu einer signifikanten Beschleunigung der Projektabwicklung in allen Phasen.

1.3 Typische Probleme in der Anforderungsanalyse

Unzulänglichkeiten und Mängel im Umgang mit Anforderungen werfen eine Vielzahl an Risiken für den Projekterfolg auf. Unter Projekterfolg verstehen wir hier die Lieferung eines Produktes oder definierten Projektergebnisses, das die Benutzererwartungen bezüglich Funktionalität und Qualität befriedigt – unter Einhaltung vereinbarter Kosten und Zeitvorgaben.

Fehler, die im engen Zusammenhang mit Anforderungen stehen, stellen mengenmäßig die häufigsten dar. Genau diese Analysefehler verursachen aber auch die höchsten finanziellen Aufwände innerhalb eines Projektes. Im gesamten Software-Entwicklungsprozess gibt es dementsprechend genau eine Stelle, an der zuerst optimiert werden sollte: der Bereich der Systemanalyse. Die Abbildung 1.1 Fehler und deren Bezug zu Anforderungen zeigt eine Studie mit 3800 Teilnehmern der European Software Process Improvement Training Initiative (ESPITI) von 1995, die diese Tatsache verdeutlicht.

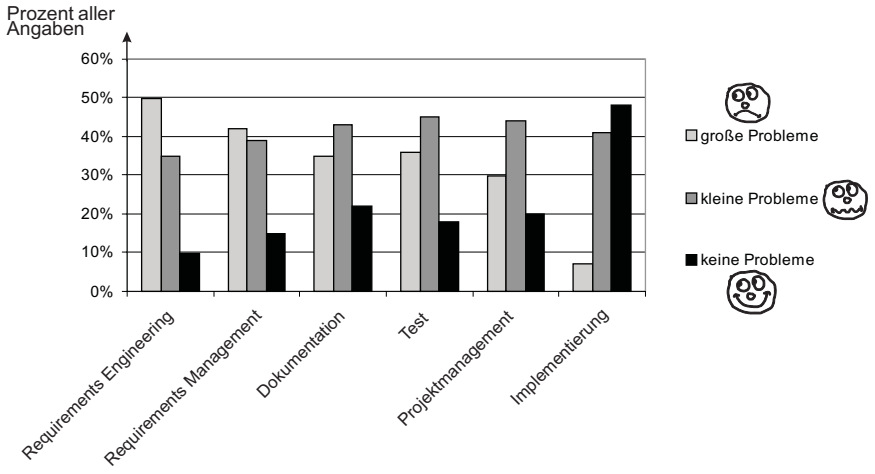


Abbildung 1.1: Fehler und deren Bezug zu Anforderungen

1.3.1 Herausforderungen und Projektrisiken

Die in der Systemanalyse entstehenden Probleme lassen sich systematisieren und auf sieben Hauptprobleme herunterbrechen.

Sieben Probleme der Systemanalyse

Die sieben Probleme der Systemanalyse:

- Unklare Zielvorstellung für das System
- Hohe Komplexität der zu lösenden Aufgaben
- Kommunikationsprobleme – Sprachbarrieren zwischen den Projektbeteiligten
- sich ständig verändernde Ziele und Anforderungen
- schlechte Qualität der Anforderungen
- Goldrandlösungen (Gold-plating) – das Produkt besitzt unnötige Merkmale
- ungenaue Planung und Verfolgung des Projektes basierend auf ungenauen Anforderungen

In diesem Kapitel werden wird die Probleme näher charakterisieren, Projektrisiken und Herausforderungen aufführen und die Lösungsmöglichkeiten durch Requirements Engineering aufzeigen.

Unklare Zielvorstellung für das System

Unklare Zielvorstellungen

Die meisten Produkte werden von verschiedenen Personengruppen benutzt, die möglicherweise unterschiedliche Teilmengen der Produktmerkmale verwenden, unterschiedliche Benutzungshäufigkeiten haben oder über ein unterschiedliches Bildungs- und Erfahrungsniveau verfügen. Wenn man alle diese Benutzeruntergruppen für sein Produkt nicht schon früh identifiziert, kann daraus resultieren, dass Anforderungen der vergessenen Benutzer nicht mit berücksichtigt werden, ein ungenügendes Produkt und dadurch Enttäuschung entsteht. Menügesteuerte Operationen sind zum Beispiel ineffizient für power user, während unklare Kommandos und Tastenkürzel Gelegenheitsnutzer überfordern. Aus diesem Grund ist es wichtig, von vorneherein alle Stakeholder zu ermitteln und in das Projekt zu integrieren. Erst wenn Repräsentanten aller Benutzergruppen ihre Ziele und Erwartungen an das System formulieren, gelangt man zu einer klaren und vollständigen Zielvorstellung.

Hohe Komplexität der zu lösenden Aufgabe

Hohe Komplexität

Die zu entwickelnden Systeme werden immer komplexer. Dabei stehen viele Anforderungen, die erhoben und verarbeitet werden, in komplexen wechselseitigen Abhängigkeiten. Die darauf aufsetzenden Funktionen sind ebenfalls

sehr anspruchsvoll. Die grundlegenden Anforderungen an die erstellten Systeme, wie zum Beispiel Anforderungen an die Bedienbarkeit, Reaktionszeiten (Echtzeitverarbeitung) und die vielfältigen Schnittstellen zu umgebenden Systemen, verschärfen die Problematik noch. In den letzten Jahren veränderten sich die Problemstellungen, die durch Software unterstützt oder realisiert werden sollten, sehr stark. In den Anfangsjahren der Softwareentwicklung wurden einfache mathematische Probleme durch die Rechnernutzung automatisiert. Heute hingegen unterstützen Rechner einzelne Personen, Teams oder ganze Unternehmen bei der Lösung komplexer Vorgänge und Denkprozesse.

Systeme bilden Denkprozesse ganzer Teams ab

Beispielsweise unterstützt ein Towersystem in der Flugsicherung einen Fluglotsen bei der Festlegung einer Start- und Landereihenfolge von Flugzeugen auf einem großen internationalen Flughafen. Die Prozesse, die dazu im Kopf der einzelnen Beteiligten abliefen (basierend auf einer langwierigen Ausbildung und jahrelanger Erfahrung), sollten nun durch das System vorweggedacht, optimiert und dadurch die Entscheidungsfindung des einzelnen Fluglotsen unterstützt werden.

Die Methoden der Systementwicklung sind mit dem Zuwachs an Komplexität nicht mitgewachsen und es ist fraglich, ob dieses Problem methodisch überhaupt sicher in den Griff zu bekommen ist.

Erkenntnistheorie, (Wirtschafts-) Informatik und Requirements Engineering

Von Prof. Dr. Alfred Holl

Erkenntnistheorie ist derjenige Zweig der Philosophie, der sich mit Gewinnung (kognitiven Methoden), Wesen und Grenzen von Erkenntnis befasst.
Was hat diese philosophische Disziplin mit Informatik zu tun, mit der Wissenschaft von der systematischen Informationsverarbeitung, insbesondere der Informationsverarbeitung mit Rechenanlagen? Computer (mit entsprechender Software) sind *formale Maschinen*, die Realität nicht in ihrer gesamten Komplexität erfassen können, sondern nur in einer reduzierten Form: ihnen sind nur die formalen Aspekte von Realität in Gestalt *formaler Modelle* zugänglich, weil sie nur *formale Sprachen* (speziell Programmiersprachen) verstehen. Die drei Termini *Modell*, *formales Modell* und *formale Sprache* sind nun in einem Exkurs zu erläutern.

Modelle sind Abbilder („Zerrbilder“) realer Gegenstände und Abläufe, denen mindestens eine wesentliche Eigenschaft genommen ist (Komplexitätsreduktion). Im Alltag verwendet man viele Modelle, ohne eigens darüber nachzudenken. *Kinderspielzeugen* fehlen zum Beispiel die Eigenschaften „Größe“ (Spielzeugautos) und „Leben“ (Stofftiere). *Landkarten* werden mit speziellen Projektionsverfahren erstellt, die von den Eigenschaften „Größe“, „Erd-

krümmung“, „Oberflächenrelief“ etc. abstrahieren. Letztere sind ein erstes Beispiel für formale Modelle: Sie sind in einer formalen Sprache formuliert. Formale Sprachen können gleichzeitig Wörter, Buchstaben, graphische Symbole und ggf. Farben verwenden (Landkarten, technische Zeichnungen; in der Informatik etwa Klassenmodelle und Geschäftsprozessmodelle) oder Wörter und mathematische Zeichen (Programmiersprachen) oder nur Buchstaben und mathematische Zeichen (mathematische, physikalische und chemische Formeln). Darstellungen in formaler Sprache sind eindeutig und lassen keinen Interpretationsspielraum zu wie etwa natürlichsprachliche Äußerungen: Der Satz „Das ist ein schönes Geschenk!“ kann je nach Tonfall und Begleitumständen Freude oder Missfallen ausdrücken, ist also in seiner geschriebenen Form nicht eindeutig.

Formale Modelle – insbesondere in der Informatik (zum Beispiel von betrieblichen Aufgabenbereichen) – sind spezielle *Formen von Erkenntnis*. Sie sind das Ergebnis kognitiver, empirischer Verfahren (Beobachtung, Beschreibung, Typisierung, Abstraktion, Formalisierung). Damit sind sie und ihre Erstellung Gegenstand erkenntnistheoretischer Überlegungen, womit die Eingangsfrage beantwortet und die nächste Frage zu stellen ist:

Gibt es eine einzige, die Erkenntnistheorie?

Nein, aus der geisteswissenschaftlichen Philosophie sind viele erkenntnistheoretische Schulen hervorgegangen, die in der Beurteilung des Erkenntniswerts von Modellen grundverschiedene, teilweise abstruse Positionen einnehmen. Erst mit der Entwicklung naturwissenschaftlicher Erkenntnistheorien in der Physik und Biologie des 20. Jahrhunderts lichtet sich dieses Dickicht. Die Streitfrage bleibt aber nach wie vor, wie gut Modelle die Realität beschreiben können.

Der naive Realismus vertritt die Ansicht, dass Modelle Eins-zu-eins-Abbilder (isomorphe Abbilder), also exakte, nicht verzerrte, nicht entstellte Abbilder der Realität sind. Dieser Objektivitätswahn findet sich häufig in der Mathematik und allgemeinen Informatik, nicht aber in den Naturwissenschaften. Zeigen doch schon einfache Selbstversuche, etwa mit optischen Täuschungen, erst recht aber Erfahrungen mit der Problematik des Einsatzes betrieblicher Informationssysteme in der Wirtschaftsinformatik, dass zwischen Modell und Realität immer eine gewisse Spannung herrscht. Diese entsteht unvermeidlich einerseits aus der grundsätzlichen Komplexitätsreduktion bei jeder Art von Modellierung, andererseits durch den Modellkonstrukteur selbst: Modelle sind nämlich immer Menschenwerk, menschliche Konstrukte. Sie entstehen nicht passiv mit dem Fotoapparat und fallen nicht vom Himmel, sondern sind Ergebnisse aktiver Realitätsinterpretationen durch einen Modellkonstrukteur, der je nach beschriebenem Realitätsausschnitt seine persönliche Subjektivität mehr oder weniger einbringt und einen bestimmten Modellierungszweck verfolgt. Der naive Realismus ist also wegen seiner Verleugnung der subjektiven

Komponente der Modellbildung bei der Modellierung komplexer Strukturen betrieblicher Informationsverarbeitung denkbar ungeeignet.

Diese Einsicht führt zum kritischen Realismus, der allerdings über die Größe der Spannung zwischen Modell und Realität im jeweiligen Einzelfall keine Aussage macht. Hier hilft erst die evolutionäre Erkenntnistheorie weiter, die aus der Biologie stammt. Sie sieht den menschlichen Erkenntnisapparat als Produkt der Evolution, d. h. als im Laufe von Jahrhunderttausenden an die Umwelt angepasst und erprobt an. Er kann sich keine existenzgefährdenden Irrtümer leisten, d. h. die Spannung zwischen Modell und Realität muss sich in Grenzen halten. Allerdings verläuft die technisch-kulturelle Evolution der letzten 5000 Jahre wesentlich schneller als die biologische Evolution, sodass der menschliche Erkenntnisapparat keine Chance hat, sich in vollem Umfang an mittlerweile völlig veränderte Erkenntnisgegenstände anzupassen. Resultat ist, dass heute weiterhin steinzeitliche (naivrealistische) kognitive Strategien verwendet werden mit dem Ergebnis, dass bei komplexen, kulturellen Erkenntnisgegenständen (Unternehmen, betrieblichen Abläufen) erhebliche Spannungen zwischen Modell und Realität entstehen können. Mit diesem Ansatz kann die evolutionäre Erkenntnistheorie problematische Aspekte in Modellierungsprozessen ziemlich detailliert erklären und damit einerseits den Grad der Abweichung von der Isomorphie einschätzbar machen und andererseits Wege zu Gegenmaßnahmen, das heißt zu einer Verringerung des Spannungsverhältnisses, zeigen.

Was gewinnt also die Informatik durch die Beschäftigung mit Erkenntnistheorie? Sie erhält kein Patentrezept zur Durchführung perfekter Modellierungen, denn die prinzipiellen erkenntnistheoretischen Probleme können durch keine Modellierungsmethode beseitigt werden. Jedoch kann Wissen um erkenntnistheoretische Zusammenhänge und bewusste Auseinandersetzung mit ihnen deren unerwünschte Folgen sehr wohl erheblich mindern. Es gibt sowohl allgemeine Gesetzmäßigkeiten als auch informatikspezifische, die weit über die Resultate aus der Beschäftigung mit dem *human factor* in der Informatik hinausgehen.

Requirements Engineering ist ein gutes Beispiel, wie man erkenntnistheoretisches Wissen in der Informatik hervorragend nutzen kann. Im Gegensatz zu naturwissenschaftlichen Erkenntnisgegenständen kann ein Unternehmensbereich in natürlicher Sprache über sich selbst Auskunft geben, weil er Menschen enthält. Der Informatiker beobachtet also nicht passiv betrachtend, sondern aktiv befragend und beeinflussend. Er bekommt die Anforderungen der künftigen IT-Anwender zunächst in natürlicher Sprache, also prämodellhaft. Sie sind inkonsistent, unvollständig, voll von firmeninternen Ausdrücken. Daraus soll der Informatiker nun vor dem Hintergrund seines subjektiven Vorwissens ein konsistentes, vollständiges, formales Modell der Geschäftsprozesse, Informationsflüsse und ihrer IT-Unterstützung ableiten. Allein mit

diesen Rahmenbedingungen der Modellierungssituation bewusst umzugehen, führt zu deutlich besseren Modellen.

Weiterführende Literatur: Holl, Alfred: Empirische Wirtschaftsinformatik und Erkenntnistheorie. In: Becker, J. et al. (ed.): Wirtschaftsinformatik und Wissenschaftstheorie – Bestandsaufnahme und Perspektiven. Wiesbaden: Gabler 1999, S. 163–207.

*Alfred Holl, (Alfred.Holl@fh-nuernberg.de) *1956, Studium der Mathematik und Linguistik an der Universität Regensburg, Entwicklung betrieblicher Informationssysteme, Dozent für Verwaltungsinformatik, seit 1990 Professor für Wirtschaftsinformatik an der Georg-Simon-Ohm-Fachhochschule Nürnberg, Forschungsschwerpunkt Modellierungsmethoden und (evolutionäre) Erkenntnistheorie.*

Kommunikationsprobleme – Sprachbarrieren

Sprachbarrieren

Projektbeteiligte sprechen nicht alle die gleiche Sprache. Es werden unterschiedliche Formulierungen für den gleichen Sachverhalt (Synonyme) oder gleiche Formulierungen für unterschiedliche Sachverhalte (Homonyme) verwendet. Sprache ist kein genormtes Ausdrucksmedium. Jeder Mensch ist in seinem Sprachgebrauch von seiner Umwelt, seinem Fachgebiet, seinem Hintergrund, seinen Kenntnissen und seinen persönlichen Erfahrungen geprägt.

Präzision in einer Fremdsprache

Häufig werden Projekte auch nicht in der Muttersprache der Projektbeteiligten abgewickelt. In den meisten Projekten werden derzeit die Anforderungen in Englisch verfasst, da auf der Basis der Anforderungen das Projekt international ausgeschrieben werden soll. Einen Sachverhalt in der Muttersprache präzise zu beschreiben ist bereits sehr schwer – viel schwerer ist es allerdings, dies in einer nicht perfekt beherrschten Fremdsprache zu tun.

Sich ständig verändernde Ziele und Anforderungen (requirements creeping)

Sich ändernde Anforderungen

Kennen sie das Gefühl, wenn sich eine nicht unerhebliche Anzahl von Anforderungen während des Projektes kontinuierlich verändert? Es ist normal, dass alle Stakeholder während des Projektes dazulernen und sich dies auch in den Anforderungen niederschlägt. Zudem verändern sich oft auch die Rahmenbedingungen des Projektes, was ebenfalls Änderungen in den Anforderungen notwendig machen kann. Wenn Anforderungen, teilweise auch Ziele, sich verändern, wird es immer wahrscheinlicher, dass der alte Zeitplan und das Budget angepasst werden müssen.

Um den Bereich der schleichenden Anforderungen zu managen, benötigt man zunächst klare Projektziele. Damit können sinnvolle Veränderungen im Sinne des Projektziels von sinnlosen unterschieden werden. Alle Änderungen in den Anforderungen müssen gegenüber den Projektzielen geprüft werden. Ein gut definierter (und gelebter!) Änderungsprozess, der den Einfluss der Analyse

jeder vorgeschlagenen Änderung mit einschließt, wird den Stakeholdern helfen, fundierte Geschäftsentscheidungen zu treffen, welche Änderungen zu akzeptieren sind (mit Rücksicht auf Aufwand an Zeit und Mitteln).

Auswirkungen auf die Systemarchitektur

Da Änderungen sich sukzessive auf das ganze Projekt auswirken können, kann die Architektur des Systems langsam zerbröckeln. Das Flicken von Code macht Programme schwer wartbar und unverständlich.

Das Wegfallen von Anforderungen und den dahinter stehenden Funktionalitäten kann genauso problematisch werden, vor allem dann, wenn das Projekt-Konfigurationsmanagement nicht perfekt funktioniert.

Lassen sich in einem Projekt Bestandteile ausmachen, die im Laufe der Zeit wahrscheinlich stärker durch Änderungen betroffen sein werden, so lässt sich eine robuste Architektur etablieren, indem die Auswirkungen dieser Änderungen möglichst vom Rest des Systems abgekapselt werden. Häufiger von Änderungen betroffen sind zum Beispiel die Schnittstellen eines Systems zu Fremdsystemen.

Anforderungsänderungen müssen so früh wie möglich erkannt und im Entwurf berücksichtigt werden. Damit vermeiden Sie, dass sich die Änderungen als code patches in der Implementierung wiederfinden.

Schlechte Qualität der Anforderungen

Mehrdeutigkeiten

Mehrdeutigkeit ist ein Kernproblem bei der Dokumentation von Anforderungen. Sie ist immer dann im Spiel, wenn mehrere Leser einer Anforderung zu unterschiedlichen Auffassungen über deren Bedeutung kommen. Ein weiteres Zeichen für Mehrdeutigkeit ist, wenn ein einzelner Leser eine Anforderung mehr als auf eine Art und Weise logisch interpretieren kann.

Mehrdeutigkeiten haben zur Folge, dass einige der Anwender des Systems vermutlich überrascht sein werden über das, was geliefert wird. Mehrdeutige Anforderungen führen zu Zeitverschwendung, wenn Entwickler eine Lösung für das falsche Problem ausführen und wenn die Tester erwarten, dass sich das Produkt anders verhält, als es von den Entwicklern hergestellt wurde. Es müssen dann teilweise nicht nur das Design und die Implementierung überarbeitet werden, sondern auch alle betroffenen Testfälle neu geschrieben werden. Das Kapitel „Der lange Weg vom Satz zur Anforderung“ stellt Ihnen einen sehr effektiven Weg vor, Mehrdeutigkeiten aufzuspüren und zu beseitigen.

Redundanzen

Redundanzen erscheinen auf den ersten Blick nicht gefährlich – sind es aber auf die Dauer. Anforderungen sollten immer redundanzfrei beschrieben werden, da Redundanzen bei der Pflege, Weiterentwicklung und Änderung des Systems zu Problemen führen können. Sofern eine Funktionalität redundant beschrieben ist, muss sichergestellt werden, dass bei der Änderung einer Stelle in der Anforderungsspezifikation alle anderen Stellen, die hierzu redundant sind, ebenfalls identischen Änderungen unterzogen werden. Dies geschieht

jedoch selten. Dadurch entwickeln sich Redundanzen sehr schnell zu Widersprüchen.

Widersprüche

Widersprüche entstehen vor allem, wenn mehrere Benutzer mit unterschiedlichen Erwartungen an das System befragt werden (oft auch noch von unterschiedlichen Analytikern). Bei umfangreichen Anforderungsdokumenten sind diese Widersprüche nicht einfach zu finden. Hierbei hilft im natürlichsprachlichen Bereich lediglich eine sinnvolle Sortierung der Anforderungen (zum Beispiel anhand von Anwendungsfällen, englisch: Use Cases), wodurch die sich widersprechenden Anforderungen erwartungsgemäß räumlich sehr nahe zusammen stehen und beim Lesen entdeckt werden können. Eine weitere, sehr effektive Möglichkeit ist die Erstellung eines Objektmodells. Ein Objektmodell sortiert die Anforderungen nach fachlich-inhaltlichen Gesichtspunkten zusammen und ermöglicht es somit, Widersprüche aufzudecken.

Ungenaue Angaben

Ungenaue Angaben. In manchen Projekten ist es fast vermessen, von Anforderungen zu sprechen. Die Entwickler erhalten von ihrem Kunden (häufig: der Vertrieb) sehr vage Vorgaben für die Produkterstellung. Wir kennen Projekte, in denen die gesamte Spezifikation einer Waschmaschine (inklusive all der komplexen Elektronik und Software, die derartige Maschinen heutzutage besitzen) auf einer Seite Anforderungsdokument beschrieben war. Die Entwickler sind bei derartigen Vorgaben sehr stark auf sich gestellt, müssen das Produkt sozusagen selbst erfinden und laufen dabei Gefahr, an den unkommunizierten Wünschen der Auftraggeber vorbei zu entwickeln. Diesen Fall kann man daran erkennen, dass die Entwickler die Anforderungen erst schreiben, wenn die Produktherstellung schon läuft. Dieser Ansatz mag für hochentwickelte Forschungsprojekte oder für Projekte, bei denen die Anforderungen wahrhaft flexibel sind [McConnell96], geeignet sein. Dennoch führt es in den meisten Fällen zur Frustration der Entwickler (welche vielleicht unter unkorrekten Voraussetzungen und mit eingeschränkter Anleitung arbeiten) und zum Ärger der Kunden, die ein Produkt erhalten, welches nicht ihren Vorstellungen entspricht.

Goldrandlösungen (Gold-plating) – das Produkt besitzt unnötige Merkmale

Goldrand-lösungen

Die Kreativität und die oftmals vorhandene tiefgreifende Fachkenntnis von Entwicklern, die in vielen Fällen Projekte rettet, kann in anderen Fällen zu erhöhtem Aufwand und auch Fehlentwicklungen führen. Ausschmückungen und zusätzliche Funktionen (englisch: gold-plating) weisen auf die Tendenz einiger Entwickler hin, neue Funktionalitäten hinzuzufügen, die nicht im Anforderungsdokument standen. Manchmal treffen diese nicht geforderten Zusatzfunktionen genau den Geschmack der Benutzer. Allzu oft finden Benutzer diese Funktionalität aber auch nicht nützlich, und die aufgewendeten Bemühungen für die Spezifikation, das Design, die Implementierung und den Test dieser Funktionen waren überflüssig.

Statt ungefragt neue Merkmale einzusetzen, sollten Entwickler den Kunden Ideen, Alternativen und kreative Methoden für ihre Überlegungen anbieten. Entscheidungen bezüglich der Funktionalität sollten ein Gleichgewicht darstellen zwischen den Bedürfnissen der Kunden und dem, was die Entwickler innerhalb des verfügbaren Zeitrahmens für technisch durchführbar und erreichbar halten. Die Entwickler sollten dabei höchstens den Funktionsumfang in Absprache mit dem Kunden reduzieren, aber keineswegs den Funktionsumfang ohne Rücksprache mit dem Kunden beliebig erweitern.

Genauso mögen Kunden um Merkmale bitten, die cool aussehen, aber dem System keinen oder nur geringen funktionalen Nutzen hinzufügen. Jede zusätzliche Funktionalität kostet Zeit und Geld. Um die Gefahr von Ausschmückungen zu minimieren, sollten Sie sich in Ihrem Projekt vergewissern, dass jede einzelne Funktion auf ihren Ursprung zurück verfolgbar ist.

Ökonomie-Check durch Stakeholder

Bei der Erhebung von Anforderungen ist es wichtig, dass jede Anforderung der Erfüllung eines Projektziels dient, von diesem abgeleitet wurde und auch auf dieses wieder zurückverfolgt werden kann. Zudem benötigt jedes Systemziel und damit jede Anforderung einen Stakeholder, der hinter dieser Anforderung steht und an dessen Urteil die Ökonomie der Zielerfüllung immer wieder verifiziert werden kann. Nur so lässt sich ein unkontrolliertes Wachstum an Funktionalität und Komplexität des Systems verhindern, welches zudem nicht zur Erreichung der Systemziele beiträgt.

Ungenaue Planung und Verfolgung des Projektes basierend auf ungenauen Anforderungen

Ungenaue Planung und Verfolgung

Dürftig formulierte Anforderungen führen typischerweise zu falschen, häufig zu übertrieben-optimistischen Schätzungen, da die im Projekt verborgene Komplexität noch nicht erkannt wurde. Die unvermeidliche Folge sind Termin- und Budgetüberschreitungen.

Das Management eines Projektes, dessen Systemziele und Systemanforderungen nicht klar festgelegt sind, gleicht durchgehend einem Stochern im Nebel. Es ist immer wieder faszinierend, mit welcher Überzeugung Projektleiter Zeit- und Kostenpläne erstellen und aktualisieren, obwohl ihnen die Sinnlosigkeit ihrer Tätigkeit eigentlich bewusst sein sollte. Projektmanagement, welches ein nicht klar definiertes Systemziel mit unklaren Anforderungen plant und verfolgt, gleicht einer Reise ins Ungewisse, bei der man weder das Wie noch das Wohin kennt – es ist von vorneherein zum Misserfolg verurteilt.

Die fünf Hauptursachen falscher Kosteneinschätzung, die in Untersuchungen ermittelt wurden, haben alle einen direkten Bezug zu Anforderungen und lauten: häufige Anforderungsänderungen, fehlende Anforderungen, unzureichende Kommunikation mit den Benutzern, schlechte Formulierung der Anforderungen und unzureichende Anforderungsanalyse [Davis95].

Studien und unsere Projekterfahrung

Die veröffentlichten Studien geben das wieder, was wir auch in Projekten erleben. Sie stellen aus unserer Sicht keineswegs eine Übertreibung, sondern eher eine Untertreibung dar, da bei den Zahlen meist nur jene Projekte einbezogen werden, die zu Ende geführt wurden. Viel katastrophalere Ergebnisse würde man vermutlich unter Einbeziehung der Projekte bekommen, die auf Grund der Probleme mit den Anforderungen nicht berücksichtigt werden konnten.

Unspezifizierte Features

Von Sven Biedermann

Bei Software-Produkten tauchen häufig zusätzliche Features auf, die in keiner Spezifikation und in keinem Handbuch stehen. Dem Projektmanagement fällt es schwer, mit diesen zusätzlichen Features umzugehen.

Auf der einen Seite existieren keine Anforderungen zu diesen Features und damit auch keine Abnahmekriterien. Niemand weiß, wie man diese Features testen soll. Außerdem verschlingen diese Features Ressourcen und müssen nach Projektabschluss auch gewartet werden.

Auf der anderen Seite will man diese Features nicht unterdrücken, um die Entwickler nicht zu demoralisieren. Diese haben sehr engagiert gearbeitet und sich mit ihrer Kreativität persönlich in das Produkt eingebracht.

Um aus diesem Dilemma herauszukommen, erscheint es ratsam, zusätzliche Features bei der Abnahme nicht zuzulassen. Ein zusätzliches Feature ist undefiniertes Verhalten, genauso wie ein Systemabsturz.[1]

Den Entwicklern sollte aber das Change-Request-Verfahren im Projekt zugänglich gemacht werden. Sie bringen ihre Ideen in Form von Änderungsanträgen ein. Dies bringt folgende Vorteile mit sich:

- Der Kunde kann entscheiden, ob ihm das Feature überhaupt angenehm ist.
- Es ist sichergestellt, dass Anforderungen und Abnahmekriterien zu den Features geschrieben werden, nach denen später auch getestet werden kann.
- Im Projekt können die Auswirkungen, Zeit- und Kostenfaktoren für die Realisierung der Features betrachtet werden. Und wer drückt da nicht gern mal ein Auge zu.

Dipl.-Inf. Sven Biedermann (Sven.Biedermann@Biedermann-Consulting.de) ist Baujahr 68. Er beschäftigt sich seit 1990 mit objektorientierter Softwareentwicklung. Heute führt er zusammen mit seiner Frau die Biedermann Consulting GmbH. Nach der Arbeit fotografiert er gern.

[1] Microsoft Excel fällt also wegen seines integrierten Flugsimulators durch!

1.4 Qualitätskriterien für jede einzelne Anforderung

Woran erkenne ich, dass die Anforderung, die ich gerade schreibe, wirklich gut ist? Das folgende Unterkapitel präsentiert Kriterien, die erfüllt sein müssen, damit eine Anforderung gut ist. Die genannten Charakteristiken lassen sich zum Beispiel im Rahmen eines Review bezüglich jeder Anforderung überprüfen. Ziel dieser Charakteristiken ist es aber nicht, Ihnen nur eine Checkliste für die Prüfung von Anforderungen zu geben, sondern Sie bereits beim Schreiben zu unterstützen.

Qualitätskriterien

Wenn von Anfang an bekannt ist, wie eine Anforderung aufgebaut sein soll, wird es realistischer, das Ziel auch zu erreichen. Wenn Sie diese Merkmale beim Schreiben und Überprüfen der Anforderungen im Kopf behalten und beherzigen, werden Sie von vorneherein bessere Anforderungen erstellen. Das genaue Vorgehen, wie die Charakteristiken sich methodisch berücksichtigen lassen, wird in den Kapiteln 6 „Der lange Weg vom Satz zur Anforderung“ und 7 „Der patternorientierte Weg zu perfekten Anforderungen“ erläutert.

Die Anforderungen sind nicht nur das Kommunikationsmittel zwischen allen Stakeholdern, sondern sie dienen auch als Basis für das Design und die Implementierung Aus diesem Grund gibt es eine Menge Kriterien, die jede einzelne Anforderung erfüllen sollte.

Qualitätskriterien für jede einzelne Anforderung

Jede einzelne Anforderung soll

- vollständig,
- korrekt,
- klassifizierbar bezüglich der juristischen Verbindlichkeit,
- konsistent gegenüber anderen Anforderungen und in sich konsistent,
- testbar,
- für alle Stakeholder verstehbar,
- umsetzbar – realisierbar,
- notwendig,
- verfolgbar und
- bewertet

sein.

Da diese Kriterien nicht leicht umzusetzen sind, ist ein Regelwerk zum Verfassen und Überprüfen von Prosaanforderungen zweckdienlich, das bei der Behebung oder zumindest bei der Einschränkung der Unzulänglichkeiten natürlicher Sprache behilflich ist. Ziel eines Analyseprozesses auf der Basis von Prosaanforderungen muss sein, dass alle genannten Kriterien erreicht werden. Dieses Regelwerk, das wir entwickelt und in vielen Projekten erfolgreich eingesetzt haben, stellen wir Ihnen im Kapitel 6 „Der lange Weg vom Satz zur Anforderung“ vor.

1.4.1 Die Merkmale exzellenter Anforderungen

Exzellente Anforderungen erfüllen eine ganze Menge von Kriterien, die im Folgenden aufgelistet und detailliert erläutert werden.

Vollständigkeit

Vollständigkeit

Jede einzelne Anforderung muss die geforderte und zu liefernde Funktionalität vollständig beschreiben. Das heißt, alle möglichen Eingaben oder Klassen von Eingabewerten und potenziell eintreffenden Ereignissen sind inklusive der möglichen Kombinationen zu betrachten. Zudem muss die gewünschte Reaktion des Systems detailliert beschrieben werden.

Kennzeichnung unvollständiger Anforderungen

Eine Anforderung enthält somit alle für den nachfolgenden Entwurf notwendigen Informationen (sie soll es dem Designer ermöglichen, das System zu entwerfen). Anforderungen, die noch unvollständig sind, sollten auch als solche gekennzeichnet werden, zum Beispiel durch Einfügen von tbd (to be determined) in den Text. Diese Marker können dann systematisch gesucht und durch die noch fehlenden Informationen ersetzt werden.

Korrektheit

Korrektheit

Beim Validieren einer Anforderung wird geprüft, ob sie an sich korrekt ist. Hierzu muss der Stakeholder (am besten derjenige, der die Anforderung gefordert hat) die erfasste/geschriebene Anforderung lesen und mit seiner Vorstellung abgleichen. Eine Anforderung ist dann korrekt, wenn sie vollständig die Vorstellung dieses Stakeholders wiedergibt. Hierzu ist es notwendig, dass die Stakeholder das Anforderungsdokument lesen und verstehen können, was einen signifikanten Einfluss auf die Art der Anforderungsdokumentation hat. So hängt die Anzahl verwendbarer Spezifikationstechniken sehr stark von den am Projekt beteiligten Personen ab.

Um korrekte Anforderungen schreiben zu können, ist es wichtig, dass der Schreiber alle erdenklichen Situationen sowie deren Einschränkungen, in denen das System verwendet werden soll, kennt. Dies umfasst die gewünschten Fähigkeiten sowohl bezüglich der Funktionalität als auch bezüglich der Darstellung der Funktionen auf der Benutzeroberfläche, auf dem Papier etc., sowie alle möglichen Interaktionen mit der umgebenden Welt (also auch mit den Schnittstellen der umgebenden Systeme). Dabei ist zu beachten, dass sich sowohl die Anforderungen an das System als auch die umgebende Welt immer wieder verändern.

Jede Anforderung beschreibt dabei ganz genau eine zu erstellende Funktionalität. Die Richtigkeit von Anforderungen kann auf der Ebene von Benutzeranforderungen (Level 2 Anforderung, siehe Kapitel 5 „Anforderungen oder Anforderungen – Der feine Unterschied“) nur vom Benutzer selbst geprüft und bestätigt werden. Detailliertere Anforderungen können nur validiert werden, indem geprüft wird, ob die Gesamtheit aller Verfeinerungen einer Benutzeranforderung immer noch die gleiche Funktionalität fordert und die zusätzlichen Details, die die Bereitstellung der Funktionalität ermöglichen.

Klassifizierbarkeit bezüglich der juristischen Verbindlichkeit

Klassifizierbarkeit

Für jede einzelne Anforderung muss deren rechtliche Relevanz festgelegt werden können. Dadurch wird die Einklagbarkeit als rechtlich verbindlicher Vertragsbestandteil klargestellt. Aus unserer Erfahrung wissen wir, dass dies in vielen Fällen kein einfaches Unterfangen ist. Häufig werden Grafiken oder Tabellen anstelle einer rein natürlichsprachlichen Anforderung verwendet. Dabei sind oft Bestandteile abgebildet, die nicht juristisch relevant sind, aber aufgrund der fehlenden Trennung von der verbindlichen Anforderung als solche klassifiziert werden.

Konsistenz gegenüber anderen Anforderungen und in sich

Konsistenz

Anforderungen müssen gegenüber allen anderen Anforderungen unabhängig vom Abstraktionsgrad oder der Art konsistent, sprich widerspruchsfrei sein. Zudem muss eine einzelne Anforderung so formuliert werden, dass sie in sich konsistent ist, das heißt nicht die Anforderung selbst schon Widersprüche aufwirft.

Um das Anforderungsdokument auf Konsistenz in sich zu testen, müssen alle Anforderungen mit anderen Anforderungen verglichen werden können. Da sich dies üblicherweise nicht auf ein Einzeldokument beschränkt, sind hier Traceability-Mechanismen zur Unterstützung notwendig.

Testbarkeit

Testbarkeit

Eine Anforderung muss so beschrieben sein, dass sich daraus (also rein aus der Anforderung und den dazu verfügbaren weiteren Informationen des Anforderungsdokuments) Testfälle ableiten lassen. Die Testfälle müssen zu dem konkreten Ergebnis führen, dass das System die Anforderungen erfüllt oder nicht – der Erfolg oder Misserfolg muss eindeutig messbar sein. Dies setzt voraus, dass Anforderungen präzise formuliert werden. Das heißt, eine Funktionalität, die durch eine Anforderung gefordert wird, muss sich durch eine Messung nachweisen lassen. Dazu ist es notwendig, konkrete Werte und Maßeinheiten in der Anforderung mit anzugeben. Insbesondere bei Anforderungen an die Dienstqualität wird dies häufig vergessen. Erfahrungsgemäß fällt es den Schreibern funktionaler Anforderungen leichter, sich in die Lage eines Testers hineinzuversetzen. Dieser kann und darf den Test der Anforderungen nur auf Basis von prüfbaren Werten durchführen. Wie soll er mit Anforderungen „Das System soll schnell reagieren“ oder „Die Eingabe muss leicht sein“ umgehen? Ohne messbare Anforderungen ist der Test des Systems nicht möglich.

8 Nicht funktionale Anforderungen

9 Abnahmekriterien

Verstehbarkeit für alle Stakeholder

Verstehbarkeit

Die Anforderungen müssen für alle Stakeholder verstehbar sein. Aus diesem Grund können sich die Dokumentationstechniken auch je nach Phase (und damit auch unterschiedlichen beteiligten Personen) stark unterscheiden. In der Analysephase, in der das breiteste Spektrum der Stakeholder teilnimmt, ist es wichtig, eine gemeinsam Sprache für alle zu schaffen. Dabei gibt es kaum einen anderen Weg, als natürlichsprachliche Anforderungen zu verwenden. In späteren Phasen (Design) eignen sich andere, formalere Beschreibungstechniken weit mehr, da der Kreis der Beteiligten dieser Phase kleiner und spezialisierter ist und die formaleren Darstellungstechniken (wie zum Beispiel objektorientierte Designmodelle) der Problemstellung angemessener sind.

Umsetzbarkeit – Realisierbarkeit

Realisierbarkeit

Es muss möglich sein, jede Anforderung innerhalb der bekannten Fähigkeiten, den Grenzen des Systems und seiner Umgebung in ein ablauffähiges System umzusetzen. Dies bedeutet, dass bei der Bewertung von Zielen und Anforderungen ein Mitarbeiter aus dem Entwicklungsteam beteiligt sein sollte, der die technologischen Grenzen der Umsetzung einzelner Anforderungen aufzeigen kann. Zudem sollten die Kosten für die Umsetzung in die Beurteilung der Umsetzbarkeit mit einbezogen werden. Manchmal tritt der Projektbetroffene von einzelnen Anforderungen zurück, wenn offensichtlich wird, welche Kosten für die Realisierung dieser einzelnen Funktionalitäten auftreten. Mit anderen Worten: es müssen alle Ziele und Anforderungen bezüglich der technischen Realisierbarkeit und der Kosten der Realisierung bewertet werden.

Notwendigkeit

Notwendigkeit

Jede Anforderung sollte eine Funktionalität fordern, die der Kunde tatsächlich benötigt oder die zur Anpassung an ein externes System benötigt wird. Eine Anforderung ist nur dann wirklich notwendig, wenn sie der Erfüllung eines Systemziels dient.[2] Dazu ist wiederum die traceability dieser Anforderung zurück zu den Zielen erforderlich, um deren Notwendigkeit für die Zielerfüllung wirklich prüfen zu können.

Verfolgbarkeit

Verfolgbarkeit

Jede Anforderung muss für sich eindeutig zu identifizieren sein. Sichergestellt wird dies meist über eine eindeutige Anforderungsnummer, die während des gesamten Lebenszyklus der Anforderung unverändert bleibt. Über diese eindeutigen Identifikatoren können auch voneinander abgeleitete Anforderungen verschiedener Spezifikationsebenen verbunden werden, sodass ein Systemziel über alle Anforderungsebenen hindurch, vom Design bis zur Implementierung und Test, verfolgt werden kann. Details hierzu erläutert das Kapitel 10 „Ordnung im Chaos – Requirements Management“. Bei der traceability handelte es sich um eines der am meisten unterschätzten Themen. Sobald Sie sich verändernde Anforderungen konsistent halten müssen und sicherstellen wollen, dass Sie Ihre Systemziele immer noch erreichen, ist die Nachverfolgbarkeit (traceability) der Anforderungen durch den gesamten Projektlebenszyklus unabdingbar.

Bewertbarkeit

Bewertbarkeit

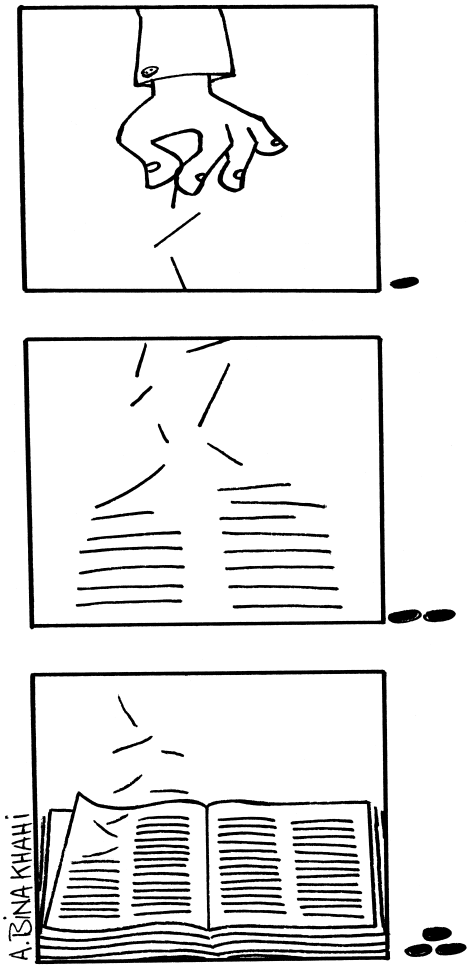
Ab einer gewissen Komplexität oder Größenordnung eines Systems ist es wichtig, die Anforderungen nach Wichtigkeit oder Priorität zu bewerten. Insbesondere dann, wenn nicht alle Funktionalitäten in einem Release eines Systems sofort umgesetzt werden können, muss durch die Stakeholder eine Bewertung der Anforderungen durchgeführt werden. Hierzu müssen die Hauptgeschäftsvorfälle, bei denen eine größtmögliche Entlastung der Benutzer des Systems erzielt werden kann, ausfindig gemacht und die daran hängenden Anforderungen als vorrangig bewertet werden. Die Bewertung der Anforderungen kann nach unterschiedlichen Kriterien zum Beispiel nach Dringlichkeit oder Wichtigkeit erfolgen. Sie sollten hier aber bereits beachten, dass eine rein fachliche Priorisierung auch negative Auswirkungen auf die Systemarchitektur haben kann. Auch die Systemdesigner sind Stakeholder, die bei der Priorisierung mitwirken sollten. Die Reihenfolge der Entwicklung sollte ein stabiles Design und eine sinnvolle Modulbildung erlauben.

[2] Natürlich können auch die Ziele noch hinterfragt und erweitert werden, wenn neue Anforderungen auftreten, für die es keine passenden Ziele gibt.

Eindeutigkeit

Eindeutigkeit

Eine eindeutige Anforderung kann nur auf eine Art und Weise verstanden werden. Es darf nicht möglich sein, andere Sachverhalte hineinzuinterpretieren. Alle Leser einer Anforderung sollten zu einer einzigen, konsequenten Interpretation der Anforderung gelangen. Die Eigenschaft der Eindeutigkeit ist bei der Verwendung natürlicher Sprache sehr schwer zu erreichen. Die natürlichsprachliche Methode lehrt allerdings eine wirksame Methode, Auslassungen, zweideutige oder interpretationsbedürftige Formulierungen aufzudecken und durch eindeutige Formulierungen zu ersetzen.



1.5 Qualitätskriterien für das Anforderungsdokument

Das Anforderungsdokument

Die Menge aller beschriebenen Anforderungen bildet das Anforderungsdokument. Dieses Dokument muss analog zu den einzelnen Anforderungen ebenfalls bestimmten Qualitätskriterien genügen. Die folgenden wichtigsten Kriterien werden wir im Laufe dieses Kapitels näher erläutern (angelehnt an [IEEE830-98]):

- angemessen bezüglich des Umfangs und klar strukturiert,
- qualitativ hochwertig bezüglich der Merkmale, die sich aus den Qualitätskriterien für Einzelanforderungen ableiten lassen,
- modifizierbar und erweiterbar,
- optimiert bezüglich des gewählten Vorgehens,
- sortierbar und
- gemeinsam zugreifbar.

Angemessener Umfang und klare Struktur des Anforderungsdokuments

Struktur und Umfang

Das Anforderungsdokument ist in seinem Umfang begrenzt. In Kapitel 10 „Ordnung im Chaos – Requirements Management“ können Sie nachlesen, was inhaltlich zu einem Anforderungsdokument gehört und wie dieses sinnvoll zu strukturieren ist. Wir stellen immer wieder fest, dass Entscheidungen, die erst in späteren Entwicklungsphasen, zum Beispiel im Design, gefällt werden (Aufteilung der Software auf Module oder Funktionen, eingesetzte Hardware ...), bereits in der Analyse in Anforderungsdokumente niedergeschrieben werden, obwohl sie dort noch völlig fehl am Platz sind.

Unserer Meinung nach ist es durchaus sinnvoll, auch Anforderungen, die an das Projekt an sich gestellt werden (wie Kosten- und Zeitvorgaben, Regeln bezüglich Berichte über den Projektstatus, ...), mit in das Anforderungsdokument aufzunehmen. Wir definieren daher ein Anforderungsdokument als eine Menge beschriebener (also nicht nur funktionaler) Anforderungen.

Leider lassen sich bezüglich eines angemessenen Umfangs keine klaren Messwerte angegeben. Bezüglich der Struktur des Anforderungsdokuments gibt Ihnen das Kapitel 10 „Ordnung im Chaos – Requirements Management“ einen detaillierteren Einblick. Dort werden auch die unterschiedlichen Gliederungsvarianten, mit denen sich der IEEE-Standard 830-1998 sehr ausführlich befasst, vorgestellt.

Merkmale, die sich aus den Qualitätskriterien für Einzelanforderungen ableiten

Qualität der Einzel-anforderungen

Ein exzellentes Anforderungsdokument sollte natürlich exzellente Anforderungen enthalten. Die in diesem Abschnitt aufgeführten Kriterien gleichen dementsprechend in großen Teilen den im letzten Unterkapitel aufgeführten Qualitätsmerkmalen exzellenter Einzelanforderungen. Insbesondere bei verteilten Projekten, bei denen viele Personen an einem Anforderungsdokument schreiben, ist es notwendig, die Qualitätskriterien für Einzelanforderungen zu überprüfen. Jeder einzelne Schreiber mag vielleicht exzellente Anforderungen geschrieben haben, aber ob diese sich nach dem Zusammenfügen zu einem exzellenten Anforderungsdokument komplettieren, bleibt fraglich und wird immer unwahrscheinlicher, je mehr Verfasser beteiligt sind.

Vollständigkeit

Vollständigkeit

Anforderungsdokumente müssen *vollständig* sein, das heißt, sie müssen alle Anforderungen beinhalten, die relevant sind. Dies bedeutet, dass für jede gewünschte Funktionalität des Systems alle möglichen Eingaben, eingehenden Ereignisse und die geforderten Reaktionen des Systems beschrieben werden müssen. Auch Anforderungen zum Beispiel bezüglich der Qualität, der Reaktionszeiten, der Verfügbarkeit und Bedienbarkeit des Systems müssen notiert werden. Anforderungen, die noch komplettiert werden müssen, müssen unter der Angabe von Gründen und den noch zu erledigenden Bearbeitungsschritten als solche gekennzeichnet werden.

Quellen- und Abkürzungs-verzeichnisse

Zur Vollständigkeit tragen aber auch formale Gesichtspunkte bei. Grafiken, Diagramme und Tabellen müssen beschriftet werden. Ein wichtiger Punkt ist das Vorhandensein konsistenter Quellen- und Abkürzungsverzeichnisse. Gerade im Bereich der Flugsicherung, in dem es Tausende von mehrdeutigen Abkürzungen, Akronymen oder Maßeinheiten gibt, sind (fachfremde) Stakeholder ohne Abkürzungsverzeichnis kaum in der Lage, Anforderungen korrekt und eindeutig zu interpretieren. Ebenso kennen wir zahlreiche Dokumente, in denen Fachtermini und wichtige Begriffe nicht definiert werden. Definitionen oder Normreferenzen, die Begriffe verbindlich werden lassen, sind notwendiger Bestandteil eines jeden Anforderungsdokuments. (Normen oder Standards müssen dann aber auch den Lesern des Dokuments zugänglich gemacht werden!)

Die Vollständigkeit des Anforderungsdokuments stellt mitunter die größte Herausforderung der Anforderungsanalyse dar. Häufig muss man einen Kompromiss zwischen verfügbaren Zeitressourcen und Vollständigkeit treffen. Das Kapitel 7 „Der patternorientierte Weg zu perfekten Anforderungen“ zeigt einen Weg auf, mittels objektorientierter Modellierung ein Anforderungsdokument auf Konsistenz und Vollständigkeit zu überprüfen.

Eindeutigkeit und Konsistenz

Eindeutigkeit, Konsistenz

Anforderungsdokumente sind nur dann *eindeutig* und *konsistent*, wenn alle einzelnen Anforderungen eindeutig und konsistent sind. Ist eine Eigenschaft (Qualitätskriterium) bei einer Anforderung des Dokuments nicht gegeben, so besitzt die gesamte Spezifikation diese Eigenschaft nicht. Die betroffenen Anforderungen müssen dementsprechend abgeändert oder notfalls entfernt werden. In Ausnahmefällen kann jedoch eine schlechte Anforderung besser als eine fehlende Anforderung sein. Insbesondere wenn dadurch die Lesbarkeit des Anforderungsdokuments erhöht wird.

Die Konsistenz des Anforderungsdokuments erstreckt sich zudem auch auf weitere Dokumente und ist nicht nur innerhalb einer einzelnen Anforderung oder des gleichen Anforderungsdokuments (interne Konsistenz) nötig.

Traceability

Traceability

Eines der wichtigsten Kriterien eines Anforderungsdokuments ist die *traceability*, also die Nachvollziehbarkeit von Zusammenhängen. Deutlich wird der Sachverhalt, wenn Sie sich ein mehrere hundert Seiten umfassendes Anforderungsdokument vorstellen. Viele Leser haben nicht die Zeit, sich stark in das Dokument einzuarbeiten, und wollen daher Zusammenhänge zwischen Anforderungen schnell erfassen können. Möglichkeiten und Techniken hierzu erläutert das Kapitel 10 „Ordnung im Chaos – Requirements Management“. Die traceability hört allerdings nicht beim Anforderungsdokument auf, sondern erstreckt sich auf weitere Dokumente (zum Beispiel Test- oder Entwurfspläne), die in vorangegangen oder späteren Entwicklungsphasen erstellt werden.

Modifizier- und Erweiterbarkeit

Modifizier- und Erweiterbarkeit

Anforderungsdokumente müssen erweiterbar sein. Es ist utopisch zu glauben, dass man einmalig sämtliche Anforderungen in einem oder mehreren Dokumenten festhalten kann. Der Analyseprozess ist sehr dynamisch. Es gibt Anforderungen, die nachträglich geändert oder aber neu hinzugefügt oder entfernt werden. Dies geschieht häufig in Form von Änderungsanträgen. Gleichzeitig bedeutet das aber auch, dass Struktur und Aufbau des Anforderungsdokuments leicht modifizierbar und erweiterbar sein müssen, da sonst die Gefahr von Inkonsistenzen und Strukturfehlern besteht. Die Modifizierbarkeit schließt einige Aspekte mit ein: Zum einen sollte die Struktur zum Beispiel durch Übersichtlichkeit, Inhaltsverzeichnisse und Indizes eine leichte Handhabbarkeit gewährleisten und zum anderen sollten Modifikationen immer nur an genau einer Stelle nötig sein (Ausschluss von Redundanzen). Zudem ist es notwendig, unterschiedliche Anforderungen auch getrennt zu verwalten, sodass Modifikationen leicht und schnell ohne die Verwicklung unbeteiligter Anforderun-

gen vonstatten gehen. Das ist einer der Gründe, warum wir jede Einzelanforderung als ein separates Formular sehen, welches völlig unabhängig von allen anderen Anforderungen oder dem gesamten Anforderungsdokument seinen eigenen Lebenszyklus durchläuft. Aus der Sicht des Konfigurationsmanagements ist somit jede einzelne Anforderung ein eigenes „configuration item".

Optimierung bezüglich des gewählten Vorgehens

Optimiert bzgl. Vorgehensmodell

Eng verbunden mit der Erweiterbarkeit des Anforderungsdokumentes ist auch die optimierte Weiterverwendbarkeit in den sich anschließenden Phasen. Bei einem iterativen Vorgehen wird zum Beispiel zuerst ein Anforderungsdokument mit geringem Detaillierungsniveau erstellt. Dieses wird anschließend modelliert und beispielsweise in einem Prototyp realisiert. Erfahrungen aus Modellierung und Prototyping fließen beim darauffolgenden Überarbeiten und Ergänzen der Dokumente dann wieder mit ein. Somit erreicht das Anforderungsdokument über das iterative Vorgehen einen immer höheren Detaillierungsgrad. Die Anforderungsdokumente sollten diese Vorgehensweise in ihrem Aufbau und in der Verwaltung der Anforderungen unterstützen. Wie dies konkret realisiert werden kann und welche Detailgrade sinnvoll sind, können Sie in Kapitel 5 „Anforderung oder Anforderung – Der feine Unterschied" nachlesen.

Sortierbarkeit

Sortierbarkeit

Anforderungsdokumente sollten es ermöglichen, die Anforderungen nach verschiedenen Kriterien zu sortieren. Als mögliche Kriterien kommen dabei zum Beispiel Wichtigkeit, Detaillierungsniveau, Teilsystemzugehörigkeit, Zugehörigkeit zu einem Use Case oder Kapitel in Betracht. Jedes Bewertungskriterium, welches bezüglich einer Anforderung erfasst wird, sollte auch ein Kriterium sein, nach dem die Anforderungen sortiert und ausgewertet werden können. Diese unterschiedlichen Sichten auf Anforderungsdokumente ermöglichen ein zielgerichtetes Lesen für alle erdenklichen Stakeholder, die häufig einen sehr unterschiedlichen Blickwinkel auf das System haben. Effizient lässt sich dies allerdings nur mit Anforderungsmanagement-Werkzeugen realisieren.

Gemeinsame Zugreifbarkeit

Gemeinsame Zugreifbarkeit

Wie bereits erwähnt, wirken in größeren Projekten mehrere Personen gleichzeitig an einem Anforderungsdokument mit. Dies bedingt, dass in dem Dokument der Autor eines jeden Eintrags vermerkt werden sollte und die Dokumente nur autorisierten Personen zugänglich gemacht werden. Zudem müssen Schutzmechanismen gegen das Überschreiben von Informationen schützen

und Dateninkonsistenzen vermeiden Wie dies in der Praxis umgesetzt wird, können Sie im Kapitel 10 „Ordnung im Chaos – Requirements Management“ nachlesen.

Anforderungsanalyse aus Sicht des Auftraggebers

Von Riko Pieper

Das Thema „Requirements Engineering“ wurde in den letzten Jahren allmählich als eigenständiges Thema erkannt, aber es gibt aus meiner Sicht einen Punkt, der bis heute extrem vernachlässigt wird:

Die systematische Erarbeitung der Kundenanforderungen ist Sache des Kunden.

Der Satz klingt trivial, steht aber leider im Widerspruch zur Realität und sogar zu den gängigen Vorgehensmodellen. Es scheint eine allgemein anerkannte Praxis zu sein, dass die Anforderungsdokumente des Kunden vom Auftragnehmer zwar bei der Erstellung eines „konkreten“ Anforderungsdokuments berücksichtigt werden, aber das vertraglich relevante Anforderungsdokument wird meistens vom Auftragnehmer erstellt und nicht vom Auftraggeber.

Es wird sicher auch in Zukunft Auftraggeber geben, denen diese Vorgehensweise am angenehmsten ist, aber der umgekehrte Fall, dass sich der Auftraggeber für die Klarheit, Korrektheit, Vollständigkeit, Widerspruchsfreiheit und Prüfbarkeit seiner Anforderungen selbst verantwortlich fühlt, sollte als Alternative in den Vorgehensmodellen mindestens als Möglichkeit erwähnt werden.

In der Version des V-Modells von 1997 gibt es immerhin einen deutlichen Hinweis auf die „Anwenderforderungen“, die in allen relevanten Phasen und Prozessen als Eingangsgröße angegeben sind. Das gibt den Anforderungen des Auftraggebers zwar schon die angemessene Priorität, hilft dem Auftraggeber bei seinem Prozess der Erstellung dieser Anforderungen aber nicht.

Bei genauerer Betrachtung stellt man dann außerdem fest, dass es sich bei diesen „Anwenderforderungen“ gar nicht um ein vom Anwender (Auftraggeber), sondern vom Auftragnehmer erstelltes Dokument handelt!

Ähnlich sieht es mit den englischsprachigen Standards aus. Hier gab es in der Vergangenheit die Standards „DoD 2167“, „DoD 2167a“, „MIL-STD-498“, „J-STD-016“. Immerhin: Erst im aktuell gültigen Standard „ISO/IEC 12207“ von 1995 gibt es in Kapitel „5.1 Acquisition process“ eine Beschreibung der Aufgaben des Auftraggebers, wie z.B. das Erstellen der „System requirements“ als Bestandteil der „Acquisition Documentation“. Hier wird zumindest die Möglichkeit erwähnt, dass die Systemanforderungen nicht vom Auftragnehmer erstellt werden.

Die Aufteilung all dieser Modelle, bei denen das Gesamtsystem in Subsysteme, Hard- und Software-Komponenten bis hin zu einzelnen Software-Modulen aufgeteilt wird, ist unter dem Gesichtspunkt des System Engineering sicher hilfreich. Bei dieser Gliederung wird aber die wichtigste Schnittstelle, eben jene zwischen Auftraggeber und Auftragnehmer, vollkommen vernachlässigt. Ein Auftraggeber interessiert sich hauptsächlich dafür, dass seine Kundenanforderungen erfüllt werden – egal ob es sich um Anforderungen auf Systemebene handelt, um Qualitäts-, Sicherheits- oder auch Softwareanforderungen.

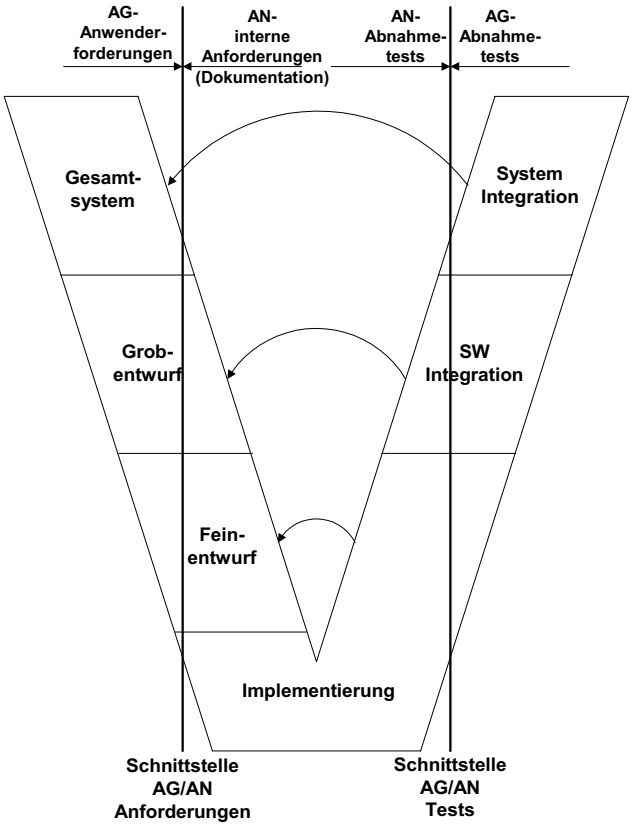


Abbildung 1.2: Vorschlag für ein Vorgehensmodell mit klarer Schnittstelle zwischen Anwenderforderungen und internen Anforderungen.

Das Konzept des V-Modells ist, dass das Gesamtsystem vom Groben ins Feine immer weiter zergliedert wird, bis es schließlich in realisierbare Komponenten aufgeteilt ist.

Die linke Seite des „V“ enthält auf jeder Ebene Anforderungen, die entweder von außen vorgegeben oder aus einer übergeordneten Ebene abgeleitet werden. Auf der rechten Seite des „V“ werden die Einzelkomponenten wieder zusammengeführt.

Auf jeder Ebene werden die integrierten Komponenten der rechten Seite des „V“ jeweils gegen die Anforderungen auf der gleichen Ebene der linken Seite des „V“ getestet.

Je weiter unten man sich im „V“ des V-Modells befindet, desto weniger Anforderungen wird man als Kunde hier haben. Wenn es keinen Grund für eine bestimmte Vorgabe gibt, dann sollte man dem Auftragnehmer hier auch alle Freiheiten lassen. Trotzdem kann es immer auch berechtigte Kundenanforderungen auf den unteren Ebenen geben. In Abbildung 1.2 wird eine (vereinfachte) Darstellung des V-Modells vorgeschlagen, in der die Schnittstelle zwischen Anwenderforderungen und der Dokumentation verdeutlicht wird. Auf der linken Seite des „V“ setzen sich die Anforderungen auf allen Ebenen aus zwei Teilen zusammen: Anwenderforderungen und interne Anforderungen, die aus Sicht des Auftraggebers keine Anforderungen, sondern nur Bestandteil der Dokumentation sind. Aus Sicht des Entwicklers wiederum ist es egal, ob eine Anforderung direkt vom Kunden kommt, oder ob sie aus einer oberen Ebene abgeleitet wurde – für ihn handelt es sich um Anforderungen, die sein zu realisierender Teil des Systems zu erfüllen hat.

Abschließend möchte ich noch einen weiteren Grund dafür nennen, dass der Auftraggeber die Kontrolle über die Anforderungen behalten sollte: Richtig ist, dass die Vorgehensmodelle vorsehen, dass die Anforderungen zu Projektbeginn auf Vollständigkeit etc. geprüft werden. Im Gegensatz zum Auftraggeber hat ein Auftragnehmer aber nicht nur Nachteile, wenn bei diesem Prozess einiges übersehen wird. Bei der Angebotserstellung müssen die Auftragnehmer oft sehr knapp kalkulieren, um einen Auftrag zu bekommen. Sie kalkulieren eventuell zum Selbstkostenpreis oder sogar darunter. Insgeheim aber rechnen sie eventuell mit einer schlechten Qualität der Anforderungen und hoffen von Anfang an auf Nachforderungen (Änderungen/Ergänzungen der Anwenderforderungen), die dann später zusätzlich berechnet werden können. Den Zuschlag erhält aber der Bieter mit dem (scheinbar) „günstigsten“ Angebot.

Fazit: Es lohnt sich für den Auftraggeber trotz des Aufwands auch in wirtschaftlicher Hinsicht, die Anwenderforderungen in Eigenverantwortung und nicht im Rahmen der Projektabwicklung des Auftragnehmers erstellen zu lassen.

Riko Pieper (riko.pieper@dfs.de), Dipl. Ing. der Technischen Informatik, 6 Jahre Entwicklungshilfe – Aufbau der Informatik an Forschungsinstituten in Argentinien. Seit 93 im Qualitätsmanagement der Deutschen Flugsicherung GmbH – Schwerpunkt „Requirements Engineering“. Co-Autor von IRE 9000.

1.6 Qualitätsstandards

In den neunziger Jahren kam der Anspruch auf, Entwicklungsprozesse zu standardisieren und mittels Qualitätsmanagementtechniken zu verbessern. Dabei bediente man sich Verfahren und Methoden aus Branchen und Bereichen wie der Automatisierungs- und Prozesstechnik, die jahrelange Erfahrung im Qualitätssicherheitsbereich vorweisen konnten. Ziel war es, die (Software-) Entwicklung zu beschleunigen, und bessere – vor allem fehlerfreie – Systeme zu liefern.

ISO 9000 Normenfamilie

Der bekannteste Standard in diesem Zusammenhang ist die ISO 9000 Normfamilie, die ein branchenunabhängiges, weltweit gültiges Regelwerk darstellt. Leider wird oft behauptet, dass die ISO-Norm (häufig 9001-3 oder 9003) Requirements Management vorschreibt. Das ist falsch und wird sich auch in naher Zukunft (mit der Überarbeitung des Regelwerks) nicht ändern. Die ISO 9000ff schreibt nur grundlegende Techniken und Mindestanforderungen an Qualitätsmanagementsystemen vor, die in den verschiedensten Geschäftsprozessen (und damit auch bei der Softwareentwicklung) angewendet werden sollen. Das abstrakte Niveau des ISO-Werks muss daher konkret bei der Anforderungsanalyse beziehungsweise dem Anforderungsmanagement angewandt werden. Die Techniken, die wir Ihnen dazu vorstellen, gliedern sich vollends in die ISO-Norm ein. Aussagen der ISO 9001-3 und deren Verwendung in unserem Vorgehen finden Sie im Detail in Kapitel 3 „Von der Idee zum System“.

Bootstrap, SPICE, CMM

Neben der ISO-Norm gibt es spezielle, auf den Softwareentwicklungsprozess zugeschnittene Qualitätsstandards, Vorgehensmodelle und Assessment-Methoden, von denen Bootstrap, Software Process Improvement and Capability Determination (SPICE) und das Capability Maturity Model (CMM®)[3] die bedeutendsten darstellen. Wir beschreiben wegen seiner Bedeutung für das Requirements Engineering und Requirements Management nur das Capability Maturity Model ausführlicher. Bootstrap wurde unter deutscher Beteiligung entwickelt und verbindet Ansätze aus SPICE und CMM. SPICE wurde zum Teil in ISO-Normen übernommen. Zu den genannten Ansätzen gibt es eine Menge sehr qualifizierter Informationen im Web. Links zu den entsprechenden Seiten finden Sie auf unserer Web-Seite www.sophist.de.

Sollten Sie sich in einem Qualitätsstandard Aussagen über die Durchführung des Anforderungsmanagements oder die Anforderungsanalyse erhoffen, so werden Sie enttäuscht sein. Bei den Standards steht das „Was“ und nicht das „Wie“ im Vordergrund. Dieses Buch konzentriert sich hingegen auf das „Wie“. Wir möchten Ihnen anhand des CMM konkret zeigen, wie ein abstrakter Standard, der im Wesentlichen nur Voraussetzungen und Ziele definiert, in

[3] ® CMM is registered in the U.S. Patent and Trademark Office
[SM] Capability Maturity Model is a service mark of Carnegie Mellon University

der Praxis Schritt für Schritt umgesetzt werden kann. Auch wenn Ihr Unternehmen oder Ihre Organisation den Softwareprozess nicht nach CMM strukturiert, ist es hilfreich, die Praktiken, die das CMM für das Anforderungsmanagement vorschreibt, zu nutzen.

1.6.1 Das Capability Maturity Model(SM) (CMM®)

Das Capability Maturity Model (in der Version 1.1.) wurde 1993 am Software Engineering Institute der Carnegie Mellon University in Pittsburgh entwickelt. Ziel war es, Kunden (in diesem Fall dem Verteidigungsministerium der USA) eine Infrastruktur an die Hand zu geben, mit der sie ihre Softwareprozesse evaluieren und verbessern können. Unzureichend organisierte Prozesse, die durch Abbruch, Kostenüberschreitung und Verzug charakterisiert sind, sollen dadurch in reife, organisierte, definierte und wiederholbare Prozesse überführt werden. Das Ergebnis der Prozesse soll ein qualitativ hochwertiges System sein, das die geforderte Funktionalität aufweist und die Wünsche des Kunden vollständig reflektiert.

Um die Eingliederung der Anforderungsanalyse in die Prozessstruktur des CMM zu verstehen, möchten wir Sie erst mit der Struktur des Modells vertraut machen:

Reifegrade nach CMM

Der gesamte Softwareentwicklungsprozess ist in fünf aufeinander aufbauende Reifegrade (Initial → Wiederholbar → Definiert → Gesteuert → Optimierend) eingeteilt (siehe Abbildung 1.3). Ein Reifegrad ist das Maß dafür, inwieweit der Softwareprozess definiert, effektiv, kontrolliert, messbar und organisiert ist.

Reifegrad Initial

Reifegrad Initial

Die erste Ebene *(Reifegrad Initial)* beschreibt den Softwareprozess als chaotisch. Organisationen und Firmen, deren Softwareentwicklung diesen Reifegrad besitzt, sind in höchstem Maße anfällig gegenüber Störungen und Änderungen, die sich im Projekt ergeben. Der Erfolg der Entwicklung hängt hier von den Erfahrungen und Kenntnissen der beteiligten Personen ab. Insbesondere die operativen Ziele sind weder plan- noch überprüfbar. Wenn überhaupt, gibt es nur wenige definierte Prozesse, die weder gesteuert noch kontrolliert ablaufen (reaktionsgetrieben).

Reifegrad Wiederholbar

Reifegrad Wiederholbar

Softwareentwicklungsprozesse der nächsten Ebene *(Reifegrad Wiederholbar)* besitzen dagegen bereits ein grundlegendes Management. Projekte werden geplant und sind dadurch wiederholbar, das heißt, Projekte basieren auf ähnlichen, bereits durchgeführten Projekten. Dazu müssen in den Unternehmen Standards eingeführt sein, die eine Zeit-, Kosten- und Funktionskontrolle ermöglichen. Bereits für die Erreichung dieses Reifegrads wird ein Anforderungsmanagement gefordert, da die Softwareprojekte nur auf Basis von ähnlichen Erfahrungen und neuen, definierten Anforderungen durchgeführt werden dürfen.

Reifegrad Definiert

Reifegrad Definiert

Der dritte Reifegrad *(Reifegrad Definiert*) weitet den Entwicklungsprozess auf weitere Unternehmens- oder Organisationsebenen aus. Abgeleitet von einem standardisierten und dokumentierten Entwicklungsprozess, der verifizierbar und nachvollziehbar ist, werden die einzelnen Prozesse projektspezifisch angepasst. Die für die Projektleitung und das Management relevanten Prozesse werden mit einbezogen, wodurch ein firmenweites Verständnis erzeugt und vermittelt wird.

Reifegrad Gesteuert

Reifegrad Gesteuert

Der vierte Grad *(Reifegrad Gesteuert)* ist geprägt durch die Messbarkeit der Prozess- und Produktqualität. Durch die Einführung verschiedenster Metriken ist es möglich, quantitative Abschätzungen zu treffen. Dies macht den Entwicklungsprozess überschaubar und vorhersehbar. Trends können erkannt und vorsorgendes Risikomanagement betrieben werden.

Reifegrad Optimierend

Reifegrad Optimierend

Die letzte Stufe *(Reifegrad Optimierend)*, die ein Softwareentwicklungsprozess im Capability Maturity Modell annehmen kann, wird durch die ständige Verbesserung des Prozesses geprägt. Dazu ist ein funktionierendes Änderungsmanagement zu etablieren, das das Feedback der Prozessbeteiligten und (externe) innovative Ansätze zur Prozessverbesserung heranzieht. Zudem ist die Fehlerprävention eine der wichtigen Säulen dieser Stufe.

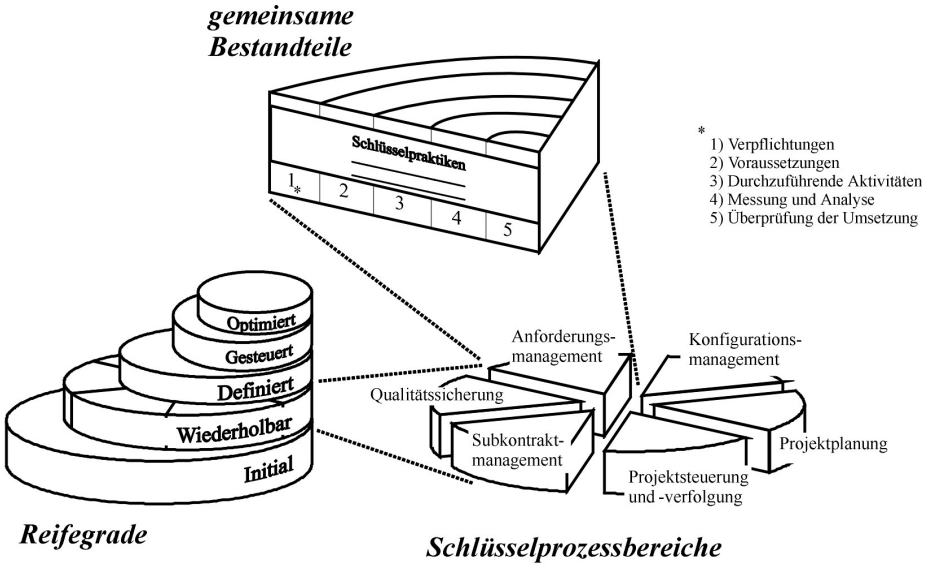


Abbildung 1.3: Das Capability Maturity Modell

Mit Ausnahme des ersten ist jeder Reifegrad in sogenannte Schlüsselprozessbereiche (englisch: Key Process Areas) eingeteilt (siehe Abbildung 1.3). Die Bereiche bestehen aus inhaltlich zusammengehörigen Aktivitäten, die ausgeführt werden müssen, damit ein Schlüsselprozessbereich abgedeckt wird. Die Schlüsselprozessbereiche wiederum bilden jeweils in ihrer Gesamtheit einen einzelnen Reifegrad ab. Exemplarisch gehen wir im Folgenden auf den Prozessbereich Anforderungsmanagement des *Reifegrads Wiederholbar* ein.

Requirements Management nach CMM

Sinn und Zweck des Anforderungsmanagements nach CMM ist es, zwischen dem Auftraggeber (Endkunde, interne Abteilungen, ...) und der Softwareentwicklungsgruppe ein gemeinsames Verständnis über die funktionalen Anforderungen und die Anforderungen an die Dienstqualität des Systems zu erlangen. Diese dienen dann als Grundlage für den weiteren Entwicklungsprozess und letztendlich auch für die produzierte Software. Das heißt aber auch, dass es durch Kontroll- und Sicherungsmechanismen eine ständige Rückkopplung zwischen Entwickler und Anforderer (Kunde) zur Gewährleistung der Konsistenz geben muss.

Ziele eines Schlüssel-prozessbereichs

Jeder Schlüsselprozessbereich besitzt Ziele, die erreicht werden müssen, damit der Bereich als Ganzes erfüllt ist. Die Ziele stellen eine Art Zusammenfassung der Schlüsselpraktiken dar und legen Umfang und Grenzen beziehungsweise den Grund für den Schlüsselprozessbereich fest. Folgende Ziele werden für das Anforderungsmanagement gefordert [Paulk 93]:

Ziele des Requirements Managements nach CMM

Ziele des Requirements Managements nach CMM

> *„Die Systemanforderungen an die Software werden kontrolliert verwaltet, um die Grundlage für Softwareentwicklung und das Softwaremanagement zu bilden."*
>
> *„Softwarepläne, Produkte und Tätigkeiten werden mit den Systemanforderungen an die Software konsistent gehalten."*

Um diese Ziele zu erreichen, schlägt das CMM® für jeden Schlüsselprozessbereich fünf gemeinsame Bestandteile (Verpflichtungen, Voraussetzungen, Durchzuführende Aktivitäten, Messung und Analyse, Überprüfung der Umsetzung) vor, die im Folgenden näher erläutert werden. Insbesondere diese Bestandteile (englisch: common features) sind nicht als unmittelbar umsetzbar anzusehen. Vielmehr muss der Weg zu Erreichung der oben genannten Ziele projektspezifisch angepasst werden. Jeder Schlüsselprozessbereich beschreibt zudem Schlüsselpraktiken (englisch: key practices), die die grundlegenden Tätigkeiten und Richtlinien darstellen. Schlüsselpraktiken sind jeweils einem gemeinsamen Bestandteil zugeordnet (siehe Abbildung 1.3). Bezüglich einer feineren Unterteilung verweisen wir Sie auf die Sekundärliteratur.

Im Folgenden erhalten Sie einen kurzen Überblick über die gemeinsamen Bestandteile und ihre Schlüsselpraktiken, die im CMM für das Anforderungsmanagement gefordert werden. Zahlreiche Querverweise zu anderen Buchkapiteln erläutern Ihnen, wo Sie genauer nachlesen können, wie die einzelnen Schlüsselpraktiken konkret umgesetzt werden. In den einzelnen Kapiteln ist beschrieben, „wie" das Vorgehensmodell mit Leben erfüllt wird.

1.6.2 Verpflichtungen

Verpflichtungen

Verpflichtungen (englisch: commitment to perform) stellen Tätigkeiten zur Etablierung und längerfristigen Verwendung von Prozessen dar. Konkret auf das Anforderungsmanagement bezogen, heißt das, dass Anforderungen dokumentiert und durch die fachlich verantwortlichen Personen stets kontrolliert und analysiert werden müssen.

1.6.3 Voraussetzungen

Voraussetzungen

Voraussetzungen (englisch: ability to perform) beschreiben organisatorische Maßnahmen, die zur Einführung der Prozesse notwendig sind. Folgende Schlüsselpraktiken sind dazu beim Anforderungsmanagement notwendig:

- Bestimmung von fachlich verantwortlichen Personen
  (siehe Kapitel 10 „Ordnung im Chaos – Requirements Management)
- Dokumentation von funktionalen Anforderungen und Anforderungen an die Dienstqualität
  (siehe Kapitel 5 „Anforderung und Anforderung – Der feine Unterschied“ und Kapitel 10 „Ordnung im Chaos – Requirements Management“)
- Festlegung und Verwaltung von Abnahmekriterien zur Verifikation des Systems
  (siehe Kapitel 9 „Abnahmekriterien – Der Prüfstein für Ihre Anforderungen“ und Kapitel 10 „Ordnung im Chaos – Requirements Management“)
- Einbeziehung von Personen und Beratern mit Erfahrungen in der Anforderungsanalyse
  (siehe Kapitel 11 „Und jetzt? – Strategien zur Einführung“)
- Unterstützung des Anforderungsmanagements durch Softwareprogramme (siehe Kapitel 10 „Ordnung im Chaos – Requirements Management)
- Ausbildung und Schulung der Projektmitarbeiter
  (siehe Kapitel 11 „Und jetzt? – Strategien zur Einführung“)

1.6.4 Durchzuführende Aktivitäten

Durchzuführende Aktivitäten

Durchzuführende Aktivitäten (englisch: activities performed) beschreiben die Tätigkeiten, die zur Erfüllung des Schlüsselprozessbereichs notwendig sind. Die folgenden, aufgeführten Schlüsselpraktiken stellen häufig die größte Herausforderung im Requirements Engineering und Requirements Management dar und sind deshalb mit besonderer Sorgfalt durchzuführen. Ohne sie ist die Zielerreichung für den Schlüsselprozessbereich Anforderungsmanagement nicht zu erreichen.

a) Anforderungen müssen kontrolliert und verbessert werden (Review)
   - Identifizierung von unvollständigen und fehlenden Anforderungen (siehe Kapitel 6 „Der lange Weg vom Satz zur Anforderung“)
   - Review und Verbesserung von Anforderungen, auf die die letztgenannten Kriterien nicht zutreffen, unter Einbeziehung der Stakeholder

- Analyse der Anforderungen auf Machbarkeit, Klarheit und Präzision, Konsistenz und Testbarkeit
  (siehe Kapitel 6 „Der lange Weg vom Satz zur Anforderung“, Kapitel 7 „Der patternorientierte Weg zu perfekten Anforderungen“ und Kapitel 9 „Abnahmekriterien – Der Prüfstein für Ihre Anforderungen)

b) Anforderungen bilden die Basis für den weiteren Entwicklungsprozess
   - Versions- und Änderungskontrolle muss für Anforderungen etabliert sein
     (siehe Kapitel 10 „Ordnung im Chaos – Requirements Management“)
   - Anforderungen bilden die Basis für die Analyse und den Entwurf
     (siehe Kapitel 7 „Der patternorientierte Weg zu perfekten Anforderungen“)

c) Änderungsmanagement
   - Änderungen müssen mit den fachlich verantwortlichen Personen abgeglichen, überprüft und verifiziert werden
     (siehe Kapitel 10 „Ordnung im Chaos – Requirements Management“)
   - Änderungen späterer Entwicklungsphasen beruhen auf Änderungen von Anforderungen
   - Die Änderungen müssen bewertet, dokumentiert und nachvollziehbar sein
     (siehe Kapitel 10 „Ordnung im Chaos – Requirements Management“)

1.6.5 Messung und Analyse

Messung und Analyse

Messung und Analyse (englisch: measurement and analysis) beschreibt die Maßnahmen, die getroffen werden müssen, damit der Prozess quantifiziert werden kann. Dazu werden im Requirements Management Mechanismen benötigt, die den Zustand und die Historie (Veränderungen) der Anforderungen festhalten und statistisch auswerten können (siehe Kapitel 10 „Ordnung im Chaos – Requirements Management“).

1.6.6 Überprüfung der Umsetzung

Überprüfung der Umsetzung

Überprüfung der Umsetzung (englisch: verifying implementation) beschreibt Schritte, die notwendig sind, damit die Tätigkeiten zielgerichtet durchgeführt werden können. Beim Anforderungsmanagement sind Reviews seitens der Projektleitung und des Qualitätsmanagements nötig. Dabei werden sämtliche organisatorische und strukturelle Maßnahmen, die für das Anforderungsmanagement benötigt werden, überprüft und notfalls angepasst.

1.7 Management – Zusammenfassung

Dieses Kapitel zeigt Gründe auf, warum Entwicklungsprozesse ohne ein funktionierendes und gutes Requirements Engineering zum Scheitern verurteilt sind.

Seit über 20 Jahren gibt es zahlreiche Untersuchungen, die belegen, dass die Beseitigung und Behebung von Fehlern umso kostengünstiger ist, je früher in der Entwicklung sie durchgeführt wird. Mit anderen Worten, das größte Optimierungspotenzial und die effektivste Möglichkeit, Fehler zu vermeiden, besteht in der Anforderungsanalyse.

Was aber trägt das Requirements Engineering dazu bei, dass die notwendigen Schritte bereits in der Analyse durchgeführt werden? Requirements Engineering bietet Techniken an, um die Hauptprobleme der Systemanalyse zu vermeiden.

Hauptprobleme der Systemanalyse

Die Schwierigkeiten in der Systemanalyse lassen sich durch sieben Hauptprobleme beschreiben:

- Unklare Zielvorstellung für das Projekt
- Hohe Komplexität der zu lösenden Aufgaben
- Kommunikationsprobleme – Sprachbarrieren zwischen den Projektbeteiligten
- Sich ständig verändernde Ziele und Anforderungen
- Schlechte Qualität der Anforderungen
- Goldrandlösungen (Gold-plating) – das Produkt besitzt unnötige Merkmale
- Ungenaue Planung und Verfolgung des Projektes basierend auf ungenauen Anforderungen

Diese Herausforderungen können Sie nur durch ein funktionierendes Requirements Engineering und Requirements Management meistern. Wichtigstes Ziel dabei ist es sicherlich, exzellente, qualitativ-hochwertige Anforderungen zu schreiben. Aber wie unterscheide ich gute und schlechte Anforderungen? Wichtig ist es, bei der Analyse also bereits während des Schreibens der Anforderungen darauf zu achten, dass Qualitätskriterien für Anforderungen beachtet werden. Dabei sind die nachfolgenden Merkmale relevant.

Anforderungen müssen:

Qualitätskriterien für Anforderungen

- vollständig (Abdeckung der gesamten durch die einzelne Anforderung geforderten Funktionalität),
- korrekt (korrekte Wiedergabe der vom Stakeholder geforderten Funktionalität),
- klassifizierbar bezüglich der juristischen Verbindlichkeit (Entscheidbarkeit über die rechtliche Einklagbarkeit),
- konsistent gegenüber anderen Anforderungen und in sich konsistent (Widerspruchsfreiheit),
- testbar (Überprüfbarkeit der realisierten Anforderung durch Testfälle),
- verstehbar (Verständlichkeit für alle Stakeholder),
- umsetzbar bzw. realisierbar (Umsetzbarkeit unter technologischen und systemspezifischen Gegebenheiten),
- notwendig (Nutzbarkeit der aus der Anforderung resultierenden Funktionalität),
- verfolgbar (eindeutige Identifizier- und Referenzierbarkeit der Anforderung),
- bewertet (Möglichkeit der Priorisierung von Anforderungen) und
- eindeutig (Ausschluss von mehrdeutigen Interpretationen einer Anforderung)

sein.

Diese einfachen Merkmale können in ein Regelwerk oder eine Checkliste, die sich unerfahrene Anforderer einfach und schnell einprägen können, überführt werden.
Insbesondere mit der steigenden Anzahl von Anforderungen an ein System ist es nicht nur wichtig, wenn jede einzelne Anforderung gut ist. Vielmehr muss das Gleiche auch für die Gesamtheit der Anforderungen, die im so genannten Anforderungsdokument festgehalten werden, gelten. Bei komplexen, verteilten Projekten werden Anforderungen naturgemäß in unterschiedlichen Abteilungen oder sogar an unterschiedlichen Orten von verschiedensten Personen verfasst. Für sich genommen mögen die Anforderungen exzellent sein, jedoch beim Zusammenfügen der Teildokumente kann dies zu erheblichen Problemen führen. Unsere Erfahrung zeigt, dass insbesondere die folgenden Punkte bei der Überprüfung des Anforderungsdokuments beachtet werden sollten.
Ist das Anforderungsdokument

Qualitätskriterien für Anforderungs-dokumente

- angemessen bezüglich des Umfangs und klar strukturiert,
- qualitativ hochwertig bezüglich der Merkmale, die sich aus den Qualitätskriterien für Einzelanforderungen ableiten lassen,
- modifizierbar und erweiterbar,

- optimiert bezüglich des gewählten Vorgehens,
- sortierbar und
- gemeinsam zugreifbar.

ISO 9000, SPICE, CMM

Die Bedeutung von Requirements Engineering wird durch verschiedenste Qualitätsstandards unterstrichen. Die bekanntesten Standards wie IS09000ff, CMM oder SPICE fordern mehr oder minder direkt die Analyse und Verwaltung von Anforderungen. Sie geben dazu keine konkreten Anweisungen, sondern beschreiben nur Ziele und Richtlinien, der die Anforderungsanalyse angeglichen werden muss. Dies kann auf unterschiedlichste Weise erreicht werden. Beispielsweise werden durch das Capability Maturity Model (CMM), einem Bewertungsverfahren für die Einteilung von Softwareentwicklungsprozessen in Reifegrade, zahlreiche Schlüsselpraktiken vorgeschlagen (siehe Abbildung 1.3). Ihre Aufgabe innerhalb Ihres Unternehmens ist es, diese Praktiken firmen- und projektspezifisch umzusetzen. Eine Gruppe von Schlüsselpraktiken bildet einen so genannten Schlüsselprozessbereich. Einer dieser Bereiche ist das Anforderungsmanagement und lässt sich nach [Paulk 93] wie folgt zusammenfassen:

Die Systemanforderungen an die Software werden kontrolliert verwaltet, um die Grundlage für die Softwareentwicklung und das Softwaremanagement zu bilden. Dabei sind Softwarepläne, Produkte und Tätigkeiten mit den Systemanforderungen an die Software konsistent zu halten.

Schlüsselpraktiken des CMM

Folgende Schlüsselpraktiken (Auszug) werden vom CMM für den Schlüsselprozessbereich Anforderungsmanagement gefordert:

- Bestimmung von fachlich verantwortlichen Personen.
- Dokumentation von funktionalen Anforderungen und Anforderungen an die Dienstqualität.
- Festlegung und Verwaltung von Abnahmekriterien zur Verifikation des Systems.
- Unterstützung des Requirements Management durch geeignete Werkzeuge.
- Analyse und Verbesserung von Anforderungen, zum Beispiel durch Reviews.
- Anforderungen bilden die alleinige Basis für den weiteren Entwicklungsprozess.
- Unterstützung des Softwareentwicklungsprozesses durch ein Änderungsmanagement.
- Festhalten von Veränderungen (Nachvollziehbarkeit).
- Etablierung eines Qualitätsmanagements für die Anforderungsanalyse.

Zusammenfassend lässt sich sagen, dass es zur Einführung der Anforderungsanalyse in einem Entwicklungsprozess sowohl aus formalen (Unterstützung von Qualitätsstandards) als auch aus handfesten praktischen Gründen (Erreichung von exzellenten Anforderungen und Minimierung von Projektrisiken) keine Alternative gibt. Jedes zeitgemäße Vorgehensmodell zur Entwicklung von Software beinhaltet Requirements Engineering-Praktiken.

1.8 Weiterführende Literatur

[Davis95]
Davis, A. M.: 201 Principles of Software Development. New York, McGraw-Hill 1995. ISBN 0-07-015840-1

[Gause93]
Gause, D.: Software requirements – Anforderungen erkennen, verstehen und erfüllen. München, Wien, Hanser 1993. ISBN 3-446-17113-4

[IEEE830-98]
IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications. Approved 25 June 1998. IEEE-SA Standards Board
Print: ISBN 0-7381-0332-2, SH94654; PDF: ISBN 0-7381-0448-5, SS94654

[Leffingwell99]
Leffingwell, D.; Widrig D.: Managing Software Requirements: A Unified Approach. Reading/MA, Addison Wesley 1999. ISBN 0-201-61593-2

[Lehner94]
Lehner, F.: Software-Dokumentation und Messung der Dokumentationsqualität. München, Wien, Hanser 1994. ISBN 3-446-17657-8

[McConnell96]
McConnell; S.: Rapid Development: Taming Wild Software Schedules Redmond/WA, Microsoft Press 1996. ISBN 1-556-15900-5

[Paulk93]
Paulk, M. C.; Weber, C. V.; Garcia, S. M.; Chrissis, M. B.; Bush, M.: Capability Maturity Model for Software. Carnegie Mellon University, Software Engineering Institute. Reading/MA, Addison Wesley 1994. ISBN 0-2015-4664-7

[Robertson99]
Robertson, S.; Robertson, J.; Foreword Weinberg, G.: Mastering the Requirements Process. Reading/MA, Addison Wesley 1999. ISBN 0-201-36046-2

[Wiegers99]
Wiegers, K. E.: Software requirements. Unterschleißheim, Microsoft Press 1999. ISBN 0-7356-0631-5