Karlsruhe Institute of Technology

Bewertung und Optimierung von Anforderungsprozessen und Neueinführung eines Anforderungsverwaltungssystems

Bachelorarbeit
von

Florian Salah El Din

An der Fakultät für Informatik
Institut für Programm und Datenstrukturen (IPD)

Erstgutachter:	Walter F. Tichy
Betreuender Mitarbeiter:	Sven J. Körner

Bearbeitungszeit: 01. Dezember 2009 – 28. Februar 2010

Hiermit erkläre ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Alle nicht von mir stammenden Inhalte sind durch Angabe der Quelle kenntlich gemacht.

Karlsruhe, den 28. Februar 2010

Abstract
Im Rahmen dieser Bachelorarbeit werden die Prozesse zur Anforderungserhebung und -verwaltung der Softwareentwicklung eines Messinstrumenteherstellers optimiert und parallel dazu Anforderungsverwaltungssysteme im Unternehmenskontext evaluiert. Ziel der Prozessoptimierung sind Aufwandseinsparungen und eine Qualitätsverbesserung der Softwareprodukte des Unternehmens. Dafür werden die Anforderungsprozesse des Unternehmens untersucht und verschiedene, aus der Literatur empfohlene Vorgehensweisen gemäß der vorliegenden Situation geprüft. Die Anforderungsprozesse werden daraufhin mit den empfohlenen Vorgehensweisen verglichen und gegebenenfalls angepasst. Um den daraus resultierenden Vorschlag zur Prozessoptimierung einzuführen, werden dem Unternehmen Handlungsempfehlungen für das Veränderungsmanagement ausgesprochen. Grundlage für die Anpassung der Anforderungsprozesse bilden der HOOD Requirements Development Process [HW05] und der Volere Requirements Process[RR06]. Die wichtigsten Veränderungen des Optimierungsvorschlags umfassen einen inkrementellen Aufbau der Anforderungserhebung und eine Qualitätserhöhung der Anforderungen durch den Einsatz von Prototypen und Qualitätsschranken. Zur Evaluation der Anforderungsverwaltungssysteme werden relevante Systeme des Markts ausgewählt und anhand ermittelter Anforderungen miteinander verglichen. Das beste Anforderungsverwaltungssystem für die Bedürfnisse des Unternehmens ist CaliberRM von Borland.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Diese Bachelorarbeit untersucht Probleme im Bereich der Anforderungserhebung im Praxiseinsatz. Ziel ist es, einen Vorschlag zur Optimierung der Anforderungserhebungsprozesse zu erstellen und eine Umsetzungsplanung vorzubereiten.

Ausgangspunkt dieser Arbeit sind bereits bestehende Anforderungserhebungsprozesse der Firma [Name des Unternehmens entfernt][1]. Ziel des Unternehmens ist es, ihre Anforderungserhebungsprozesse nachhaltig zu verbessern, sowie ein neues Anforderungsverwaltungssystem einzuführen.

1.1 Motivation

Die Erhebung von Anforderungen ist grundlegend für eine erfolgreiche Umsetzung von Kundenwünschen. Dass heutzutage immer noch viele Softwareprojekte nicht erfolgreich verlaufen, zeigt der CHAOS Report der Standish Group aus dem Jahr 2009 [SG09]: 24% aller Projekte scheiterten und lediglich 44% konnten als teilweise erfolgreich eingestuft werden. Auch Jan Stafford sieht unklare Anforderungen als vermeidbaren Hauptgrund für das Scheitern von Softwareprojekten [Sta09]. Bestätigt wird das durch die Statistik in Abbildung 1, in der zu sehen ist, dass 41% aller Fehler eines IT-Projekts ihre Ursache in Problemen mit Anforderungen haben.

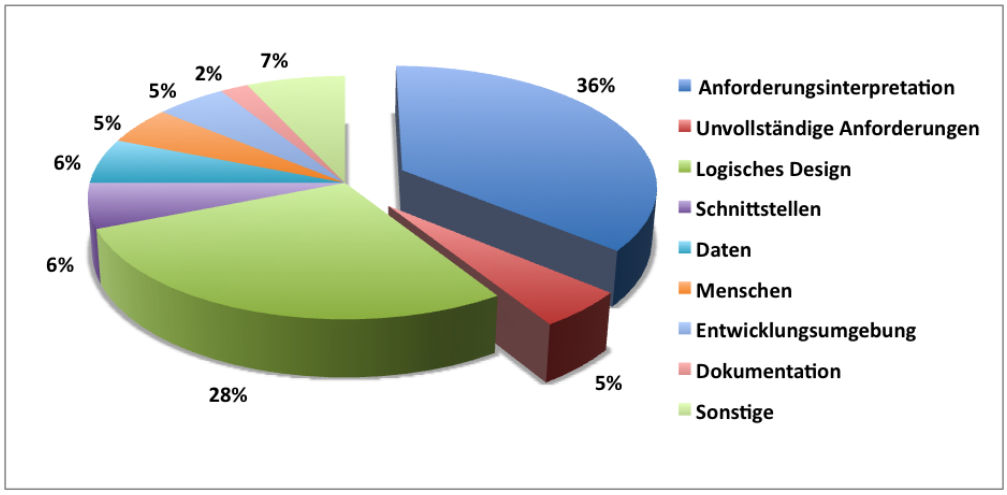


Abbildung 1: Anteil der Fehler eines Luftfahrt IT-Projekts, die nach dem Problemtyp klassifiziert sind [SKT+92, S. 19]

Die Motivation zur Anforderungserhebung ist die Vermeidung hoher Kosten, die durch Missverständnisse zwischen dem Auftraggeber (Nutzer) und dem Auftragnehmer (Produzent) entstehen [Sut02, S. 4]. Während der Projektlaufzeit werden in verschiedenen Iterationen weitere Missverständnisse aufgedeckt und gelöst, weshalb die Änderungen während eines Projekts verwaltet werden müssen.

[1] Im Folgenden „Unternehmen“ genannt.

Anforderungsverwaltung[2] ist zum einen ein systematischer Ansatz um Anforderungen an ein Softwareprodukt aufzunehmen und zu organisieren. Zum anderen ist es ein Prozess um die Akzeptanz und das Verständnis von Änderungen dieser Anforderungen sowohl beim Kunden als auch beim Projektteam sicherzustellen [LW00, S. 16].

An der Anforderungsverwaltung sind im Regelfall unterschiedliche Abteilungen des Auftragnehmers beteiligt. Mangelhafte Kommunikation zwischen diesen Abteilungen ist ein Hauptgrund für das Scheitern von Softwareprojekten [Hau08, S. 2]. Um dies zu verbessern, können Anforderungsverwaltungssysteme eingesetzt werden[3].

1.2 Problemstellung

Beim Unternehmen sind die Abteilungen Marketing, Research & Development (Forschung & Entwicklung) und Quality Assurance (Qualitätssicherung) an Anforderungserhebungs- und Anforderungsänderungsprozessen der Softwareabteilung beteiligt. Aufgrund fehlender Werkzeuge sind die Kommunikationswege zwischen den Abteilungen lang bzw. umständlich. Des Weiteren genügen die Prozesse zur Anforderungserhebung weder den Bedürfnissen der beteiligten Angestellten noch dem Qualitätsanspruch des Unternehmens.

Die Prozesse zur Anforderungserhebung beim Unternehmen sind stark projektgetrieben. Für übliche Auftragsprojekte ist dies eine typische Vorgehensweise. Da das Unternehmen in den meisten Fällen Produkte für den anonymen Markt[4] herstellt, kommt es allerdings oft dazu, dass einzelne Projekte nur einen Teil des marktfertigen Produkts liefern. Diese Projekte werden unabhängig voneinander entwickelt, obwohl meist Abhängigkeiten im Form von Anforderungen zwischen diesen Projekten bzw. Teilprodukten bestehen.

Ein anderes Problem stellen Anforderungserhebungen für Produktüberarbeitungen dar. Nur die neuen oder geänderten Anforderungen werden in ein Lastenheft aufgenommen[5]. Abgeänderte Anforderungen einer Vorgängerrevision können Auswirkungen auf andere Produktbestandteile haben. Aufgrund der Nicht-Erfassung gleichgebliebener Anforderungen werden diese eventuell nicht getestet. Solche Fehler werden dann spät bei Integrations- oder Systemtests gefunden und sind teuer zu beheben [HW05, S. 24ff.].

Zusätzlich dazu ist eine Rückverfolgbarkeit von Anforderungen für den Kundenstamm des Unternehmens, der hauptsächlich aus der Pharmaindustrie stammt, wegen beiden vorhergehend genannten Problemen oft nur schwer möglich. Mit der Rückverfolgung eines beim Kunden aufgetretenen Fehlers kann nachgewiesen werden, ob der Produzent alle nötigen Schritte getätigt hatte, um das Fehlerrisiko auf ein vereinbartes Niveau einzuschränken.

1.3 Gliederung der Bachelorarbeit

Die Bachelorarbeit ist wie folgt gegliedert. In Kapitel 2 werden die Zielsetzung und der Beitrag der Arbeit erläutert. Kapitel 3 gibt eine Einführung in die Grundlagen der Anforderungserhe-

[2] Für eine genauere Diskussion dieses Begriffs siehe Unterkapitel 3.1.

[3] Siehe z.B. `http://www.ibm.com/software/awdtools/doors/`.

[4] Das bedeutet, dass das Unternehmen aus der Sicht des Markts einen Projektauftrag an sich selbst vergibt.

[5] Dieser Ausschnitt der Anforderungen wird Delta-Anforderungen genannt.

bungsprozesse und deren Werkzeuge. Die aktuelle Situation bezüglich der Anforderungserhebung beim Unternehmen wird im darauf folgenden Kapitel behandelt. Der Vergleich einer Auswahl von Anforderungsverwaltungssystemen wird anschließend in Kapitel 5 beschrieben. Der letztendliche Vorschlag zur Optimierung der Anforderungsprozesse des Unternehmens wird in Kapitel 6 gezeigt und anschließend werden im Kapitel 7 Hinweise auf eine mögliche Evaluierung der Arbeit gegeben. Diese Ausarbeitung schließt mit der Beschreibung des weiteren Vorgehens zur Umsetzung des Prozessvorschlags und deren Planung in Kapitel 2 ab.

2 Zielsetzung

2.1 Vorschlag zur Prozessoptimierung

Ziel dieser Arbeit ist die Erstellung eines Vorschlags zur Optimierung der Anforderungsprozesse. Eine vollständige Umsetzung dieses Vorschlags kann aus zeitlichen Gründen nicht mitbegleitet werden.

Der Vorschlag für eine Optimierung basiert auf aus der Literatur empfohlenen Prozessen wie z.B. Colin Hoods „Requirements Development Process“ [HFPW08, S. 55f.], Suzanne und James Robertsons „Volere Requirements Process“ [RR06, S. 17ff.] oder Karl Wiegers „Change Request Process“ [Wie03, S. 335]. Durch einen Vergleich des Ist-Zustands der Prozesse beim Unternehmen mit den empfohlenen Prozessen werden Probleme aufgedeckt, Potentiale erkannt und bei Bedarf Änderungen vorgeschlagen.

2.2 Planung der Umsetzung

Das zweite Ziel dieser Arbeit ist eine Umsetzungsplanung zu erstellen.

Da neue Prozesse erst dann wirken, wenn sie von den Mitarbeitern gelebt werden, wird eine Strategie bzw. ein Konzept entwickelt, mit der das Unternehmen dies erreichen kann. Dabei wird neben den Grundlagen des Veränderungsmanagements das Umsetzungsmodell „HOOD Capability Model“ [HFPW08, S. 243ff.] im Unternehmenskontext betrachtet, um darauf aufbauend Handlungsempfehlungen für das Unternehmen auszusprechen.

3 Grundlagen

In diesem Kapitel werden die Grundlagen zum Thema Anforderungen erklärt. Zunächst werden in Unterkapitel 3.1 die in dieser Arbeit verwendeten Fachbegriffe erläutert. Danach wird in Unterkapitel 3.2 ein kurzer Blick auf die Historie der Anforderungsverwaltung bis heute geworfen. Abschließend erfolgt in Unterkapitel 3.3 eine kurze Beschreibung der aus der Literatur empfohlenen Anforderungserhebungsprozesse. Zum Thema „Verwandte Arbeiten“ gibt es darüber hinaus kein eigenes Kapitel, da andere Arbeiten oft nur Randgebiete dieser Arbeit behandeln oder auf die hier eingeführten Grundlagen referenzieren.

3.1 Wichtige Begriffe

Anforderung Für den Begriff „Anforderung“ gibt es in der Literatur viele verschiedene Definitionen. Lawrence beschreibt eine Anforderung als etwas, das Entwicklungsentscheidungen steuert [Law98, S. 31]. Der IEEE-Standard [IEE90, S. 62] definiert den Begriff spezifischer als

1. eine Bedingung oder Eigenschaft, die ein System oder eine Person benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen,
2. eine Bedingung oder Eigenschaft, die ein System oder eine Systemkomponente aufweisen muss, um einen Vertrag zu erfüllen oder einem Standard, einer Spezifikation oder einem anderen formell auferlegten Dokument zu genügen oder
3. eine dokumentierte Repräsentation einer Bedingung oder Eigenschaft wie in 1. oder 2. definiert.

Nach dieser Definition sind wegen 3. also sowohl dokumentierte als auch nicht dokumentierte Elemente als Anforderungen zu bezeichnen. Die SOPHISTen[6] fassen dies kompakt zusammen und beschreiben Anforderungen als „eine Aussage über Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen“ [Rup09, S. 14]. Sommerville und Sawyer beschränken sich auf die dokumentenbasierte Bedeutung und beschreiben Anforderungen als eine Spezifikation, die implementiert werden soll und als Beschreibungen von Systemeigenschaften, Systemattributen oder wie sich das System verhalten sollte [SS97, S. 4]. Im Rahmen dieser Arbeit gelten die Definitionen des IEEE Standards und der SOPHISTen, da sie nicht nur dokumentierte Anforderungen umfassen.

Software Requirements Specification Der Begriff „Software Requirements Specification“ oder kurz „SRS“ wird regelmäßig im Unternehmen verwendet und soll daher an dieser Stelle erläutert werden. Grundsätzlich versteht man darunter ein Dokument von funktionalen Anforderungen, das das erwartete Verhalten des Softwaresystems so umfassend wie möglich beschreiben soll [Wie03, S. 10]. Im Deutschen wird solch ein Dokument oft als Lastenheft bezeichnet. Generell werden auch Repräsentationen der funktionalen Anforderungen in Form von Datenbanken oder Tabellen SRS genannt. Im speziellen Fall des Unternehmens liegen die SRS immer als Dokumente vor.

[6] SOPHISTen sind Experten im Bereich der Anforderungen und haben u.a. [Rup09] veröffentlicht. Siehe `http://www.sophist.de/` für weitere Informationen.

Darüber hinaus gibt es das Dokument „External Reference Specification“, das im Deutschen dem Pflichtenheft entspricht. Der Oberbegriff beider Dokumente ist Anforderungsspezifikation.

Delta-Anforderung Eine Delta-Anforderung ist eine Anforderung, die „einen Unterschied zu einer bereits existierenden Anforderung beschreibt“ [Ebe08, S. 162].

Anforderungserhebung Der Begriff „Anforderungserhebung“ wird in der Literatur häufig als „Requirements Engineering“ oder „Requirements Development“ bezeichnet. Ebert definiert „Requirements Engineering“ als das „disziplinierte und systematische Vorgehen zur Ermittlung, Strukturierung, Spezifikation, Verifikation, Analyse, Evolution und Verwaltung von Anforderungen unter kundenorientierten, technischen, wirtschaftlichen und erfolgsorientierten Vorgaben“ [Ebe08, S. 30]. In Anlehnung an die drei Dimensionen des Requirements Engineering definiert es Pohl als einen „kooperativen, iterativen, inkrementellen Prozess, dessen Ziel es ist zu gewährleisten, dass

1. alle relevanten Anforderungen bekannt und in dem erforderlichen Detaillierungsgrad verstanden sind,
2. die Projektbeteiligten eine ausreichende Übereinstimmung über die bekannten Anforderungen erzielen und
3. alle Anforderungen konform zu den Dokumentationsvorschriften dokumentiert bzw. konform zu den Spezifikationsvorschriften spezifiziert sind.“ [Poh07, S. 43].

Die Anforderungserhebung bzw. die englischen Übersetzungen sind ein Sammelbegriff für verschiedene Aktivitäten im Umgang mit Anforderungen. Nicht zu verwechseln ist der Begriff „Anforderungserhebung“ mit den Begriffen „Anforderungsaufnahme“ oder „Anforderungsermittlung“ (auf englisch „Requirements Elicitation“), die lediglich Teildisziplinen der Erhebung sind.

Anforderungsverwaltung Genauso wie für den Begriff „Anforderungserhebung“ existieren für den Begriff „Anforderungsverwaltung“ unterschiedliche Definitionen. In der Literatur stößt man häufiger auf den englischen Begriff „Requirements Management“. Die herrschende Meinung sieht Requirements Management als eine Teildisziplin des Requirements Engineering. So zum Beispiel Ebert: „Requirements Management ist ein Teil des Requirements Engineering, der sich mit Pflege, Verwaltung und Weiterentwicklung von Anforderungen im Lebenszyklus befasst.“ [Ebe08, S. 30]. Eine andere Definition von Stevens und Martin beschreibt wiederum, dass Requirements Engineering ein Teil des Requirements Management sei. Demnach sei Requirements Management die Identifikation, Ableitung, Zuordnung und Kontrolle von Systemfunktionen und -attributen [SM95]. Aufgrund dieser teilweise widersprüchlichen Definitionen der Anforderungserhebung und Anforderungsverwaltung hat die HOOD Group beides zu dem Begriff „Requirements Management & Engineering“, kurz „RM&E“ zusammengefasst. „RM&E stellt methodische Vorgehensmodelle zur Verfügung, um innerhalb des Produktentstehungsprozesses die Entwicklung und die Evolution von Anforderungen zu unterstützen. Es stellt auch die Verwaltung von Anforderungen und deren Eigenschaften innerhalb des Entwicklungsprozesses dar.“ [HW05, S. 4f.]. Im weiteren Verlauf dieser

Arbeit wird unter Anforderungsverwaltung die Teildisziplin der Anforderungserhebung im Gesamten verstanden.

Anforderungsverwaltungssystem Bei der Nutzung üblicher Bürosoftware stößt man bereits bei der Vergabe eindeutiger Kennzeichen für einzelne Anforderungen an die Grenzen der Werkzeugunterstützung [Rup09, S. 361]. Anforderungsverwaltungssysteme, die auch „Requirements Management Tools“ oder kurz „RM Tools“ genannt werden, können die Informationen der Anforderungen in Mehrbenutzer-Datenbanken speichern und diese je nach Bedarf in unterschiedlichen Formen darstellen [Wie03, S.368]. Genauere Informationen zu diesem Thema gibt es in Kapitel 5.

Anforderungsänderung Eine Anforderungsänderung ist eine Modifizierung des Anforderungswunschs [HHMS04, S. 5]. Nach Einführung eines Produkts kann es beim Kunden zu Problemen kommen, z.B. wegen Bedienungsschwierigkeiten. Der Umgang mit solchen Änderungsanfragen nennt sich „Incident-Management“ [Rup09, S. 424]. Neue Ansätze oder Innovationen von Fachbereichen können auch zu Änderungsanfragen führen [Rup09, S. 424]. Weitere Quellen für Änderungen sind die Tester, da sie unvollständige Anforderungen aufdecken können oder Entwickler, da sie Optimierungsansätze für die Architektur liefern können [Rup09, S. 425]. Wenn neue Anforderungen ein sehr hohes Ausmaß annehmen, sodass sie den Umfang des Projekts deutlich vergrößern, spricht man von einem „Scope Creep“, der oft zum Scheitern von Projekten führt [Wie03, S. 329].

3.2 Historische Entwicklung der Anforderungserhebung

Die Entwicklung der Anforderungserhebung unterteilt sich grob in drei Phasen: Die traditionelle Systemanalyse, die phasenbezogene Anforderungserhebung und die kontinuierliche Anforderungserhebung. Die folgenden Erläuterungen beruhen größtenteils auf Pohls Nachschlagewerk rund um das Thema „Requirements Engineering“ [Poh07, S. 25ff.].

Der bekannteste Vertreter der **traditionellen Systemanalyse** ist die „Strukturierte Analyse“ nach [DeM79]. Grundlegend ist die Vorgehensweise, bei der nach einer ausführlichen Analyse basierend auf dem daraus resultierenden Ist-Modell Anforderungen an ein Soll-Modell gestellt werden. Vorteil dieser Vorgehensweise ist, dass bei der Ist-Analyse wichtige Informationen über den Kontext (z.B. Schnittstellen) des vorherigen Systems erhoben werden. Somit wird sichergestellt, dass es bei der Integration des neuen Systems keine Schwierigkeiten gibt.

Eine Weiterentwicklung der „Strukturierten Analyse“ ist die „**Essenzielle Systemanalyse**“ nach [MP84]. Die Grundidee ist, dass man ein Modell bildet und dieses von den physikalischen Rahmenbedingungen abstrahiert und somit ein logisches (essentielles) Ist-Modell erhält. Auf Basis dessen werden die Anforderungen im Sinne eines logischen Soll-Modells entwickelt. Dieses wird dann wieder in die Rahmenbedingungen gesetzt um somit ein physikalisches Soll-Modell zu erhalten. Der größte Vorteil dieser Vorgehensweise liegt darin, dass Entwurfsentscheidungen aufgrund von Rahmenbedingungen nicht vorweggenommen werden, da eine perfekte Umgebung imitiert wird.

In den 70er und 80er Jahren, in denen die vorhergehend genannten Vorgehensweisen zur Anforderungserhebung eingesetzt wurden, waren die zu entwickelnden Softwaresysteme oft leicht zu verstehen. Danach gab es einen Wandel in der Systementwicklung, da zunehmend neue Technologien hinzukamen. Ist-Analysen wurden zu aufwändig.

Der erste Schritt, um diesem Wandel gerecht zu werden, war die **phasenbezogene Anforderungserhebung** [Poh07, S. 30ff.]. Bei dieser Vorgehensweise ist eine explizite Phase für die Anforderungserhebung vorgesehen. Der Unterschied zu den vorher beschriebenen Vorgehensweisen ist, dass hierbei unabhängig von anderen Projekten die Anforderungen erhoben werden. Unterscheiden kann man dabei zwei Abwandlungen: Die sequenzielle und die parallele Entwicklung. Die Abwandlungen unterscheiden sich darin, dass bei der sequenziellen Methode Entwicklungsprojekte immer streng nacheinander stattfinden; bei der parallelen Entwicklung dürfen sich die Entwicklungstätigkeiten überschneiden.

Allerdings wurden auch in dieser Vorgehensweise Schwächen aufgedeckt. Da sich innerhalb des Entwicklungszeitraums Änderungen an den Anforderungen nicht verhindern ließen, waren nach Projektabschluss die Dokumente der Anforderungserhebungsphase unbrauchbar. Eine Wiederverwendung von Anforderungen war dadurch nahezu ausgeschlossen.

Als Folge dieser Schwächen entwickelten Pohl und Jarke bereits 1994 die **kontinuierliche Anforderungserhebung** [JP93]. Unterschieden werden hierbei die phasenübergreifende Anforderungserhebung und die projektübergreifende Anforderungserhebung. Bei ersterer werden kontinuierlich während des gesamten Entwicklungszeitraums hinweg Anforderungen dieses Entwicklungsprojekts erhoben. Bei projektübergreifender Anforderungserhebung geht man einen Schritt weiter und löst die Anforderungserhebung komplett von den Entwicklungsprojekten. Die Erhebung von Anforderungen wird global betrieben und verschiedene Projekte können gemeinsame Anforderungen verwenden. Ein Nachteil dieser Vorgehensweise ist der steigende Koordinationsbedarf.

Die kontinuierliche Anforderungserhebung setzt sich gemeinsam mit Prozessmodellen wie dem Rational Unified Process [Kru07] oder den agilen Methoden wie Extreme Programming [Bec00] oder Scrum [SB02] immer mehr durch. Trotzdem ist die phasenbezogene Anforderungserhebung immer noch in vielen Unternehmen anzutreffen. So auch im untersuchten Unternehmen.

3.3 Empfohlene Prozesse zur Anforderungserhebung

In den folgenden Unterkapiteln werden empfohlene Prozesse zur Anforderungserhebung aus der Literatur vorgestellt. Im Unterkapitel 3.3.5 werden weitere Prozessbestandteile, die für die Anforderungserhebung von Nutzen sein könnten, vorgestellt. In Kapitel 6 werden die hier erwähnten Prozesse wieder aufgegriffen und auf eine mögliche Nutzung zur Optimierung der bestehenden Anforderungsprozesse hin überprüft.

3.3.1 V-Modell

Das V-Modell ist eines der ersten Vorgehensmodelle und wurde 1992 zum ersten Mal im Auftrag des Verteidigungsministeriums und des Innenministeriums der Bundesrepublik Deutschland veröffentlicht. 1997 wurde das V-Modell überarbeitet [HHMS04, S. 21]. Aufgaben zur Anforderungserhebung sind lediglich im Submodell SE wiederzufinden (siehe Abbildung 2).

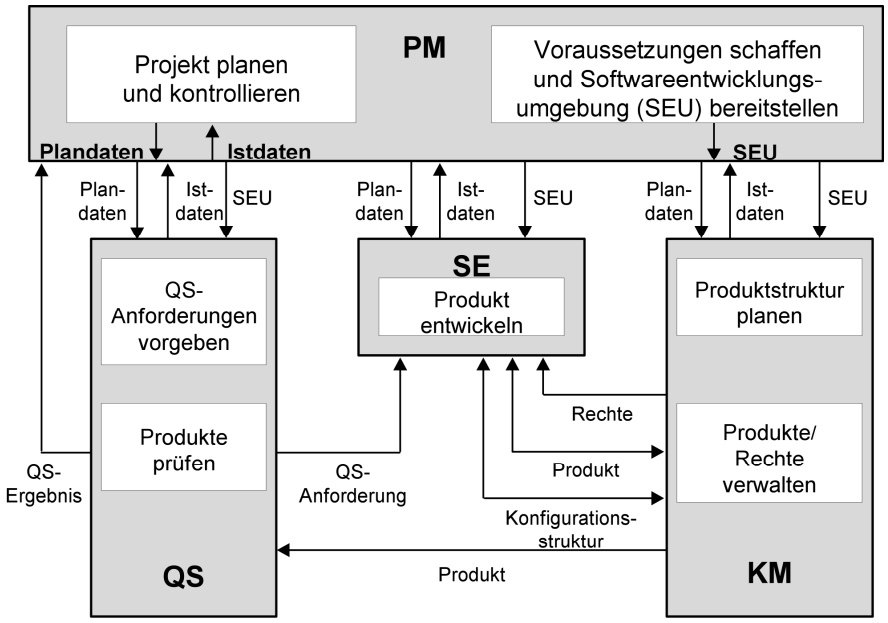


Abbildung 2: Das Zusammenspiel der Submodelle des V-Modells [Drö00]

Gemäß des V-Modell Handbuchs unterteilt sich SE 1 „System-Anforderungsanalyse“ in folgende Aktivitäten:

1. Ist-Aufnahme/-Analyse durchführen
2. Anwendungssystem beschreiben
3. Kritikalität und Anforderungen an die Qualität definieren
4. Randbedingungen definieren
5. System fachlich strukturieren
6. Bedrohung und Risiko analysieren

Das Endprodukt der System-Anforderungsanalyse stellen die so genannten „Anwenderforderungen“ dar. Die Anforderungserhebung ist im V-Modell nur schwach ausgeprägt bzw. beschränkt sich auf die Anforderungsanalyse.

3.3.2 Rational Unified Process

Im Jahr 1997 hat die Firma Rational Software nach erfolgreicher Akquisition mehrerer Softwareunternehmen den Rational Unified Process (kurz „RUP“) entwickelt. Im Februar 2003 wurde Rational Software von IBM aufgekauft [Rei03].

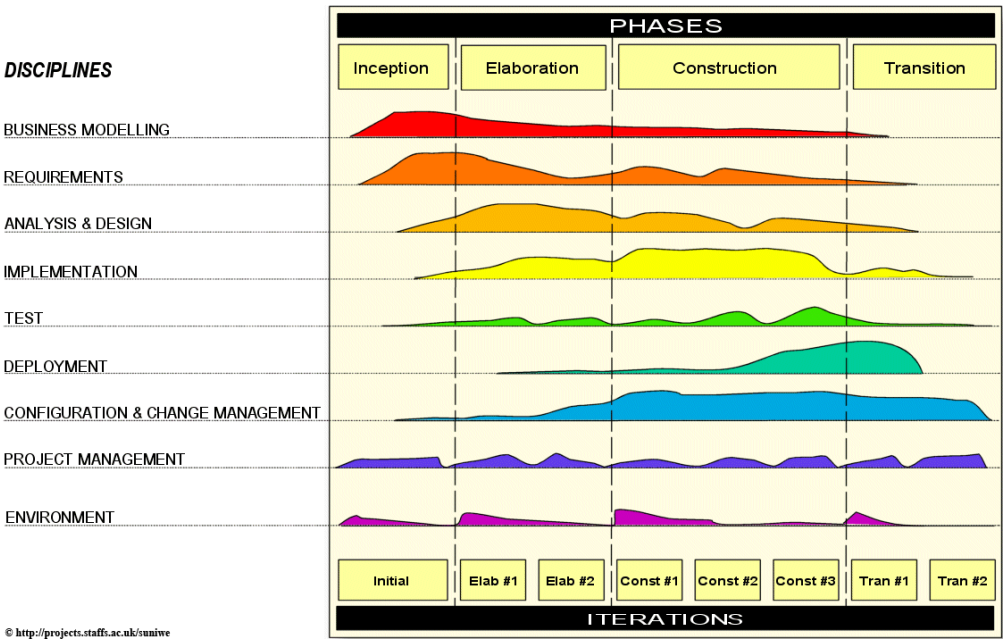


Abbildung 3: Übersicht über die einzelnen Disziplinen im Rational Unified Process [Kru07, S. 45]

Der Rational Unified Process besteht insgesamt aus neun Disziplinen, die in vier Phasen unterteilt sind (siehe Abbildung 3). Die Anforderungserhebung findet größtenteils in der Konzeptualisierungsphase statt. Sie ist in der Literatur zum Rational Unified Process noch detaillierter in sechs Hauptaktivitäten unterteilt, die in einem logischen und nicht sequenziellen Zusammenhang zueinander stehen. Als Rollen in der Anforderungserhebung sind der Systemanalyst, der Anforderungserheber und in manchen Fällen der Benutzerschnittstellenentwickler vorgesehen.

Der Rational Unified Process ist auf die kostenpflichtigen Produkte von IBM Rational abgestimmt. Daher ist es eher nachteilig, den Rational Unified Process anzuwenden, wenn keine Produkte der Produktgruppe genutzt werden [HHMS04, S. 23]. Eine nähere Betrachtung des Rational Unified Process hängt daher davon ab, ob ein Anforderungsverwaltungssystem von IBM Rational in Kapitel 5.4 empfohlen wird.

3.3.3 HOOD Requirements Development Process

Der HOOD Requirements Development Process ist ein Prozess zur Anforderungserhebung, der von Colin Hood entwickelt wurde. Er basiert auf zwei Konzepten: Dem Informationmodell

und dem Requirements Definition Process, der auch als Aktionswirbel bekannt ist.

Informationsmodell Wie in Abbildung 4 zu sehen ist, ist das Informationsmodell in einen Problembereich und einen Lösungsbereich unterteilt.

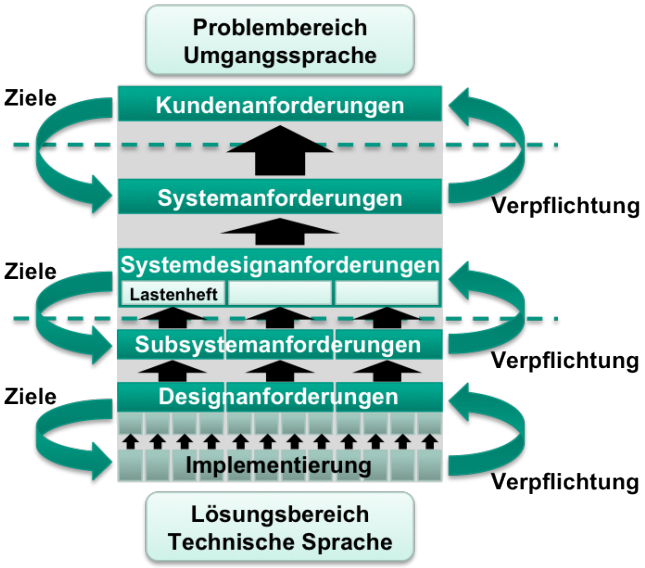


Abbildung 4: Das Informationsmodell mit graduellem Übergang vom Problem- in den Lösungsbereich [HW05, S. 66]

Im Problembereich sind die Anforderungen in der Sprache des Kunden verfasst, damit dieser sie versteht. Im Lösungsbereich sind die Anforderungen in einer technischen Sprache verfasst, damit die Schwächen der natürlichen Sprache (z.B. Uneindeutigkeit) umgangen werden.

Kovitz hat bereits 1999 ein ähnliches Konzept vorgestellt, das in Abbildung 5 zu sehen ist. Das Modell wird auch hier in zwei Bereich unterteilt: Hier sind es der Problembereich und der Maschinenbereich. Im Gegensatz zum graduellen, feineren Übergang zwischen diesen beiden „Welten“ in Hoods Modell beschränkt sich Kovitz auf nur drei Repräsentationen der Darstellung der Anforderungen. Im Laufe einer Anforderungserhebung müssen also Repräsentationen der Anforderungen in verschiedenen Sprachen erstellt werden, damit alle Akteure am Projekt angemessen teilnehmen können.

Requirements Definition Process Anders als in den üblichen Prozessbeispielen lässt sich die Definition von Anforderungen laut der HOOD Group nicht in einem typischen sequenziellen Modell darstellen [HFPW08, S. 43f.]. Anstelle eines sequenziellen Ablaufs tritt hier ein Aktionswirbel ein, der in Abbildung 6 dargestellt ist. Die zugeführten Daten in den Aktionswirbel bestehen aus dem Umfang, Eingangsanforderungen und einer Struktur. Nach iterativer Durchführung der Aktivitäten „Erheben“, „Spezifizieren“, „Analysieren“ und „Durchsicht“ entstehen freigegebene Anforderungen, die auf die Eingangsanforderungen rückverfolgbar sind.

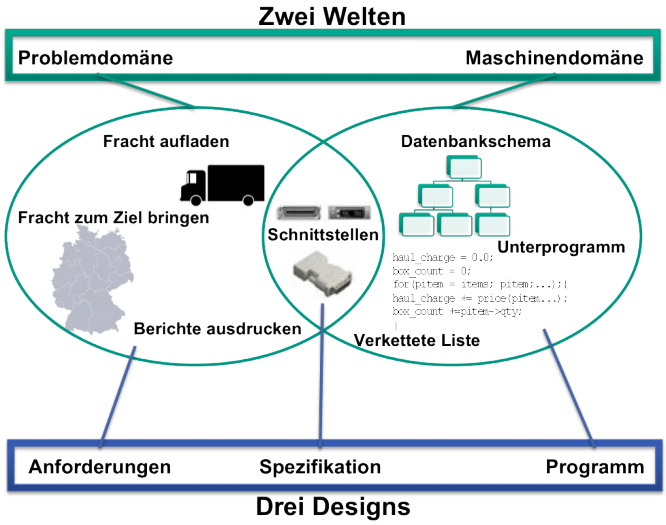


Abbildung 5: Zwei Welten und drei Designs: Der Zusammenhang zwischen Anforderungen, Spezifikationen und dem Programm [Kov99, S. 38]

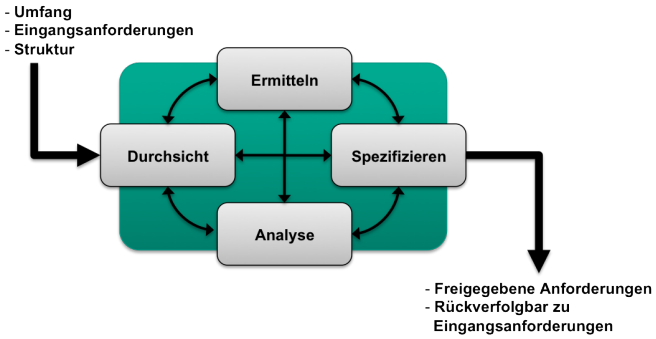


Abbildung 6: Der Requirements Definition Process von Hood [HFPW08, S. 51]

Durch Kombination beider Konzepte kommt man letztendlich zum HOOD Requirements Development Process, der in Abbildung 7 zu sehen ist. Auf jeder Informationsebene wird der Aktionswirbel durchgeführt, der als Eingangsanforderungen die jeweils vorher erstellten Anforderungen erhält.

Somit sind Änderungen der Anforderungen auf höheren Informationsebenen prinzipiell nicht möglich. Nur durch Änderungsanträge können nach einer durchgeführten Auswirkungsanalyse Anforderungen von höheren Informationsebenen nachträglich geändert werden. Der Grund für diesen umständlichen Weg liegt in den Kosten, die bei einer nachträglichen Änderung auftreten. Schätzungen zu Folge betragen die Kosten für eine nachträgliche Änderung auf einer höheren Informationsebene für jeden Ebenenübergang das Zehnfache der oberen Ebene [Boe81, S. 40].

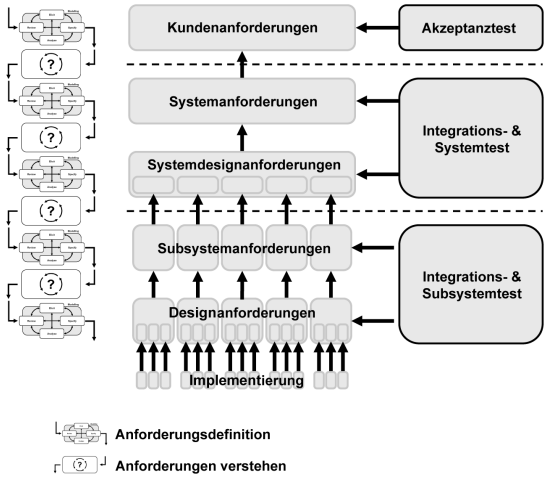


Abbildung 7: HOOD Requirements Development Process [HFPW08, S. 56]

3.3.4 Volere Requirements Process

„Volere“ ist ein italienisches Verb und steht für „wollen“ oder „wünschen“. Unter dem Namen Volere ist eine Sammlung von Anforderungsressourcen zusammengefasst. Sie beinhaltet Kurse, Vorlagen, Bücher und Prozesse, so wie z.B. den Volere Requirements Process von [RR06].

Die Besonderheit des Volere Requirements Process beruht darauf, dass es die agile Softwareentwicklung unterstützt [RR06, S. 4ff.]. Dies erscheint in sofern fraglich, da es einige extreme Beispiele gibt, in denen die Grundsätze des agilen Manifests [BBB+01] übertrieben interpretiert wurden. Einer der vier Grundsätze lautet:

> „Working software over comprehensive documentation.
>
> [...]
>
> That is, while there is value in the items on the right, we value the items on the left more.“

Robertson und Robertson betonen, dass dies nicht bedeutet, auf eine Dokumentation komplett verzichten zu müssen. Auf der anderen Seite bedeute ein Anforderungsprozess nicht gleichzeitig, dass unlesbare und ungelesene Spezifikationen erstellt werden müssten.

In Abbildung 8 ist der Volere Requirements Process grob dargestellt. Der Prozess hat keinen eindeutig identifizierbaren Anfangs- und Endpunkt, sondern bietet je nach Situation einen Einstiegspunkt in diesen Zyklus. Dokumentationen bieten demnach also unter anderem einen Mehrwert für kommende Projekte.

Einen wichtigen Unterschied im Vergleich zum HOOD Requirements Development Process stellt das im vorherigen Unterkapitel vorgestellte Informationsmodell dar. Hood stellt mit dem Modell den Zusammenhang zwischen Anforderungen und der Implementierung her, indem die Anforderungsdefinition kontinuierlich die Informationsebenen bis zum Lösungsbereich durchgeht. Im Volere Requirements Process ist dieser Zusammenhang nicht so stark wiedergegeben. Nach der Erstellung der Anforderungsspezifikation als ein Dokument folgt

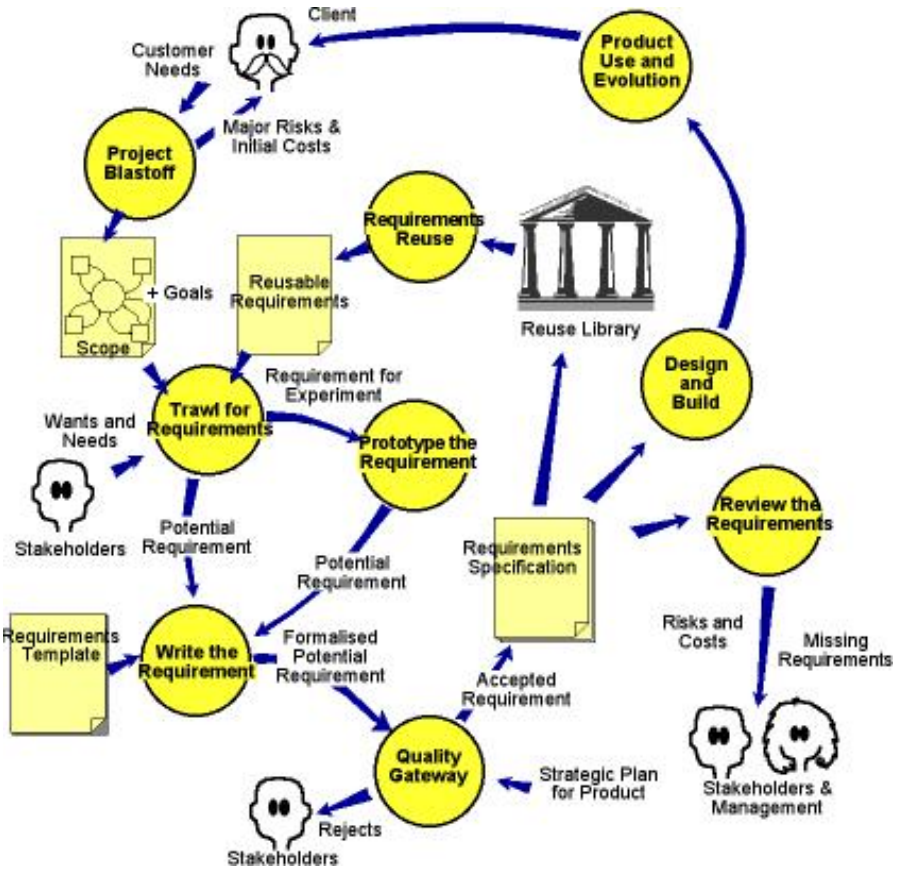


Abbildung 8: Volere Requirements Process [RR06, S. 18]

direkt der Prozess „Design and Build“. Der Volere Requirements Process grenzt sich damit durch einen geringeren Umfang deutlich vom HOOD Requirements Development Process ab. Es werden nur die Spezifikationen im Problembereich erstellt.

Neben einer sehr ausführlichen Prozessbeschreibung bietet Volere verschiedene Vorlagen an, die innerhalb des Prozesses verwendet werden. Die bekanntesten Vertreter dieser Vorlagen, sind zum einen die so genannte „Snow Card“ und zum anderen das „Volere Requirements Specification Template“.

Die „Snow Card“ ist eine Vorlage, die sehr früh im Projektzeitraum eingesetzt wird, um Anforderungen aufzunehmen. Durch ihre Kompaktheit ist sie in der Praxis sehr beliebt geworden [RR06, S. 243]. Nach Eintragung einer Anforderung in die „Snow Card“ kann diese zur Erstellung der Anforderung an einen Anforderungsanalysten weitergegeben werden.

Das „Volere Requirements Specification Template“ ist letztendlich das Enddokument des Prozesses. Die Vorlage beinhaltet eine Struktur, in die ein Anforderungsanalyst die aufgenommenen Anforderungen eintragen kann.

3.3.5 Weitere Prozesse

Change-Control Process Da in einem Softwareprojekt Änderungen auftreten (vgl. Seite 7), ist es notwendig, mit diesen Änderungen umgehen zu können. Wiegers hat eine Beschreibung für einen Änderungskontrollprozess entwickelt [Wie03, S. 333ff.], in dem beschrieben ist, welche verschiedenen Status ein Änderungsantrag durchläuft und welche Rollen dabei involviert sind. Es ist zu empfehlen, im Unternehmen eine Beschreibung über ihren Änderungskontrollprozess bereitzustellen, an dem sich die Angestellten orientieren können.

Stringente Anforderungserhebung Die von Ebert vorgestellte stringente Anforderungserhebung hat viele Ähnlichkeiten mit der in Unterkapitel 3.2 erwähnten phasenorientierten Anforderungserhebung. Charakteristisch für diesen Prozess sind klar definierte Phasen und Meilensteine [Ebe08, S. 70]. Allerdings legt Ebert hier Wert darauf, dass dies nicht bedeute, dass Anforderungen zu einem bestimmten Zeitpunkt eingefroren werden müssen. Die stringente Anforderungserhebung eignet sich vor allem für Projekte, deren Produkte für einen längerfristigen Einsatz vorgesehen sind oder wenn Anforderungen bereits bekannt sind oder als stabil gelten.

Iterative Anforderungserhebung Die iterative Anforderungserhebung zeigt Ähnlichkeiten mit der auf Seite 8 vorgestellten kontinuierlichen Anforderungserhebung. Entscheidend ist hierbei, dass ein Projekt in einzelne Bausteine unterteilt wird, die dann überlappend nacheinander bearbeitet werden [Ebe08, S. 72]. Diese Vorgehensweise wird für Projekte empfohlen, in denen Anforderungen zunächst unbekannt sind und erst während der Systementwicklung sichtbar werden.

Requirements Engineering Processes von Kotonya / Sommerville Kotonya und Sommerville behandeln in ihrem Buch „Requirements Engineering“ [KS00] verschiedene Anforderungsprozesse. Dabei gehen sie zunächst von einem sehr abstrakten Prozess aus und verfeinern diesen nach und nach in den folgenden Kapiteln. Die vier grundlegenden Aktivitäten [KS00, S. 32f.] sind dabei

1. die Aufnahme,
2. die Analyse und Verhandlung,
3. die Dokumentation und
4. die Validierung

von Anforderungen. Sie stellen diesen generischen Anforderungsprozess in verschiedenen Prozessmodellen vor und vertiefen anschließend jeweils die vier grundlegenden Aktivitäten. Auf diesem Abstraktionsniveau bieten sie dann meist eine Auswahl an Prozessen an, die aber immer noch sehr generisch sind. Neben den Prozessen gehen sie auch auf spezielle Techniken wie z.B. Prototyping ein.

Außerdem wird die parallele Aktivität der Anforderungsverwaltung in einem eigenen Kapitel erklärt. Neben sehr grundlegenden Themen wie die Volatilität von Anforderungen oder die eindeutige Kennzeichnung wird hier auch die Änderungsverwaltung

vertieft [KS00, S. 123ff.]. Diese könnte im weiteren Verlauf dieser Arbeit ähnlich wie der Change-Control Process von Bedeutung sein.

Architecture Tradeoff Analysis Method (ATAM) ATAM ist eine Methode zur Analyse von Systemarchitekturen. Es wird überprüft, ob eine Systemarchitektur den Anforderungen entspricht. Zusätzlich kann es mögliche Lücken in den Anforderungen aufdecken. Dies Analyse kann laut der Autoren innerhalb von vier Tagen durchgeführt werden [KKC00].

4 Ist-Analyse der Anforderungsprozesse

Ziel dieses Kapitels ist eine Dokumentation des Ist-Zustands der Anforderungsprozesse im untersuchten Unternehmen. Hierfür werden zunächst in Unterkapitel 4.1 die Erhebungsmethoden zur Ist-Analyse erläutert. Nach Durchführung der vorgestellten Vorgehensweise werden in Unterkapitel 4.2 die Befragungsergebnisse einzeln vorgestellt. In Unterkapitel 4.3 werden die einzelnen Ergebnisse in das Gesamtbild gebracht und zusammengefasst. In Kapitel 6 erfolgt dann letztendlich ein Vergleich der aktuellen Anforderungsprozesse mit den aus der Literatur empfohlenen Prozessen.

4.1 Erhebungsmethode der Ist-Analyse

Grundsätzlich kann man die Erhebungsmethoden zur Prozessaufnahme in die Primär- und die Sekundärerhebung aufteilen [LB08, S. 2].

4.1.1 Primärerhebung

Primärerhebungen sind Techniken, die Prozesse aus erster Hand betrachten. Dies kann entweder durch Berichte der Prozessbeteiligten oder durch Betrachtung der Prozessausführung erfolgen. Dazu gehören folgende Techniken [Pep04, S. 228ff.]:

- Persönliche Interviews
- Telefonische Interviews
- Schriftliche Fragebögen
- Online-Fragebögen
- Beobachtung

Fragebögen können sich je nach Fragestellungen (offene oder geschlossene Fragen) zum Einsatz der Befragung vieler Personen eignen. Für eine solche Art der Befragung muss der Ersteller des Fragebogens allerdings bereits im Voraus wissen, welche Antwortmöglichkeiten zu erwarten sind. Er muss sich also zunächst mit der zu befragenden Materie vertraut machen. Interviews dagegen sind gewöhnlich offener und benötigen somit weniger Vorwissen des Fragestellers.

Sowohl Fragebögen als auch Interviews unterliegen der Subjektivität der Befragten. Die einzige Methode, die nahezu neutral[7] und damit objektiv bleibt, ist die Beobachtung von Prozessen. Allerdings ist diese Methode sehr zeitaufwendig und unter Umständen können nicht alle Prozesse eines Unternehmens betrachtet werden, da sie möglicherweise nur bei bestimmten Rahmenbedingungen auftreten.

[7] Die Neuträlität ist natürlich vom Beobachter abhängig.

4.1.2 Sekundärerhebung

Sekundärerhebungen beschränken sich auf die Analyse von bereits vorhandenen Informationsquellen [Pep04, S. 222]. Informationquellen sind im Unternehmen in Form von Dokumenten vorhanden, die Handlungsanweisungen für die Mitarbeiter wiedergeben oder sogar Prozessbeschreibungen sind.

4.1.3 Vorgehensweise

Für diese Arbeit wurde folgende Vorgehensweise zur Erhebung des Ist-Zustands gewählt:

Schritt 1: Zunächst wird eine Sekundärerhebung durchgeführt, um sich mit der Materie vertraut zu machen. Dabei werden so genannte „Life Cycle"-Dokumente und Vorlagen für Spezifikationen des Unternehmens gelesen und untersucht. Unter der Annahme, dass die Mitarbeiter des Unternehmens die Prozesse des „Life Cycle" größtenteils umsetzen, werden die Inhalte zusammengefasst und als aktuelle Prozessdarstellung grob skizziert. Dies dient als Vorbereitung für Schritt 2.

Schritt 2: Mit Hilfe dieser groben Prozessdarstellung werden in persönlichen und telefonischen Interviews ausgewählte Mitarbeiter des Unternehmens befragt. Bei der Auswahl der Mitarbeiter wird darauf geachtet, einen möglichst repräsentativen Ausschnitt der Softwareabteilung des Unternehmens zu wählen. Die Interviews haben die folgende Struktur:

1. Selbstbeschreibung der Rolle des Befragten beim Unternehmen
2. Verifikation der bisher erarbeiteten Prozessdarstellung und gegebenenfalls Anpassung dieser
3. Detaillierung der Prozessdarstellung an Stellen, an denen der Befragte involviert ist
4. Identifikation möglicher Potentiale der Prozesse

Am Ende dieses Schrittes entsteht eine detaillierte Prozessdarstellung, die auf den Interviews und den Dokumentenanalysen basiert. Sie ist notwendig für Schritt 3.

Schritt 3: Die detaillierte Prozessdarstellung wird im 3. Schritt an alle zur Verfügung stehenden Mitarbeiter elektronisch versendet. Die Mitarbeiter des Unternehmens sollen anschließend die Prozessdarstellung auf ihre Korrektheit hin überprüfen und gegebenenfalls Rückmeldung geben. Nach Abschluss dieses Schrittes ist die Prozessaufnahme abgeschlossen.

Schritt 4: In einem letzten Schritt müssen alle aufgenommen Daten verarbeitet werden. Neben der Prozessaufnahme werden auch viele subjektive Eindrücke erfasst, die einen Hinweis darauf geben, an welchen Stellen der Prozesse Potentiale angesiedelt sind. Die subjektiven Eindrücke müssen daher den einzelnen Stellen der Prozesse zugeordnet werden um später bei der Optimierung beachtet zu werden.

Auf ausführliche Beobachtungen, die normalerweise bei Prozessaufnahmen durchgeführt werden, wird im Rahmen dieser Arbeit aus zeitlichen Gründen verzichtet. Trotzdem werden während der regelmäßigen Anwesenheiten am Standort des Unternehmens passive Beobachtungen durchgeführt, die einen Einfluss auf die Ist-Analyse der Prozesse haben werden. Auf eine Durchführung von Befragungen per Fragebögen wurde verzichtet, da bei der geringen Anzahl an verfügbaren Mitarbeitern die Rücklaufquote zu niedrig ist, um repräsentativ zu sein.

4.2 Ergebnisse der Ist-Analyse

Neben einer Prozessdarstellung des Ist-Zustand wurden bei den Interviews auch subjektive Eindrücke der Mitarbeiter aufgenommen. Diese werden zunächst in Unterkapitel 4.2.1 diskutiert. Danach werden in den folgenden Unterkapiteln detaillierte Auszüge aus der Prozessdokumentation dargestellt, die in Appendix A im Gesamten aufgeführt sind. Die selbst erstellte Prozessdokumentation und ihre Auszüge sind auf Englisch erstellt worden, damit im international tätigen Unternehmen alle Mitarbeiter mit den Ergebnissen weiterarbeiten können.

4.2.1 Allgemeines

Insgesamt verlief die Interviewphase sehr erfolgreich. Bis auf eine Person[8], die operativ stark eingebunden war, nahmen sich alle ausgewählten Mitarbeiter sehr kurzfristig[9] die Zeit für ein Interview. Aufgrund meiner nicht vollzeitigen Anwesenheit im Unternehmen und der dauerhaften örtlichen Abwesenheit eines Mitarbeiters, wurden die Hälfte der Interviews telefonisch durchgeführt. An der Struktur des Interviews änderte sich allerdings dabei nichts. Die hohe Bereitschaft der Mitarbeiter für Interviews bezüglich der Anforderungsprozesse ist nicht nur direkt für die Prozessaufnahme wertvoll. Dies zeigt auch eine wahrscheinlich hohe Akzeptanz für mögliche Prozessänderungen.

Wer und wieviele Mitarbeiter befragt wurden, ist in Tabelle 1 zu sehen.

Tabelle 1: Befragte Mitarbeiter des Unternehmens

Abteilung	**Manager**	**Mitarbeiter**
Qualitätssicherung	1	2
Forschung & Entwicklung	2	1
Marketing	2	1[1)]

[1)] Im Genaueren ist dies der Anforderungsanalyst.

Die Dauer der Interviews variierte zwischen 45 Minuten und zwei Stunden. Vor allem langjährige Mitarbeiter neigten dazu, in Details ihrer Arbeit abzuschweifen. Man kann schließen, dass eine hohe Routine in ihrer Arbeit vorherrscht, allerdings auch die Gefahren der „Betriebsblindheit“ vorhanden sein könnten. Eine Steuerung der Gespräche war hierbei nötig.

[8] Eine Ersatzperson aus derselben Abteilung mit derselben Funktion wurde interviewt.

[9] Kurzfristig bedeutet in dem Fall, dass teilweise am selben Tag das Interview durchgeführt wurde, spätestens aber eine Woche nach Anfrage.

Weitere Eingriffe waren zu Beginn des Interviews notwendig, da Mitarbeiter beim Durchsprechen der Prozesse oft nicht ihre Erfahrungen schilderten, sondern den vorgeschriebenen (aber oft nicht ausgeführten) „Life Cycle“ erläuterten. Da sie das Gespräch mit einer extern einzuordnenden Person führten, ist dieses Verhalten verständlich. Hierfür wurden Interviewmethoden von [BD09] eingesetzt.

Es gab kein Interview, in dem der Punkt „Identifikation möglicher Potentiale der Prozesse“ (siehe Unterkapitel 4.1.3, Schritt 2) im Gesprächsverlauf nicht beantwortet werden konnte. Langjährige Mitarbeiter können sich an ihre Probleme gewöhnen und erkennen diese dann nicht mehr [Bra07, S. 160]. In diesem Fall hätte man erwartet, in manchen Teilen der Interviews keine Antworten zu bekommen. Dass trotzdem Potentiale angesprochen wurden, lässt zum einen darauf schließen, dass die Mitarbeiter objektiv und offen sind, zum anderen aber auch, dass großes Verbesserungspotential existiert.

Leichte Frustration wurde sowohl in manchen Interviews als auch in passiven Beobachtungen während meiner Anwesenheiten im Büro des Unternehmens und bei informellen Gesprächen bspw. beim Mittagessen festgestellt. Ein wichtiger Grund hierfür ist, dass die vorgeschriebenen „Life Cycle“-Dokumente, die ich bereits in der Dokumentenanalyse untersucht hatte, in der beschriebenen Form nicht umgesetzt werden. Mitarbeiter bemerken dies regelmäßig, da andere Mitarbeiter mit derselben Funktion bestimmte Arbeitsabläufe anders bearbeiten. Diese Abweichung vom vorgeschriebenen Standard resultiert in situativen (aufwendigen) Anpassungen anderer Mitarbeiter, die mit den Arbeitsergebnissen weiterarbeiten müssen. Eine Abweichung vom vorgeschriebenen „Life Cycle“ an manchen Stellen wurde oft damit begründet, dass das Wasserfallmodell[10], auf dem der „Life Cycle“ basiere, nicht mehr zeitgemäß sei. Auf der anderen Seite wurde das Wasserfallmodell aber auch von vielen Mitarbeiter gelobt, da Termine gut planbar und Audits damit leichter durchzuführen seien.

Eine Schwäche der Anforderungsprozesse, die fast in jedem Gespräch erwähnt wurde, stellt die Projektbezogenheit dar, mit denen Anforderungen erhoben werden. Nach Meinung vieler Mitarbeiter sollten Anforderungserhebungen von Produkten auch produktgetrieben erstellt werden. Da im Unternehmen allerdings in Projekten gearbeitet wird und über Projekte hinweg ein Austausch von Daten nur schwer möglich ist, ist dies schwierig zu erreichen. Die Problematik, die dies hervorruft, wird auch in Hoods Literatur diskutiert. Unternehmen müssten weg vom dokumentenorientierten und hin zum informationsorientierten Erheben von Anforderungen. Im dokumentenorientierten Ansatz werden Anforderungen eines Projekts in einem großen Dokument spezifiziert. Ein Austausch einzelner Elemente ist daher umständlich. Im informationsorientierten Ansatz hingegen wird jede Anforderung als eigenes Element behandelt, wodurch die Handhabung einzelner Anforderungen flexibler ist. Eine andere Mitarbeiterin erkannte dieses Problem auch und erwähnte, dass die Dokumentenform für Änderungen an Anforderungen sehr hinderlich und unübersichtlich sei. Ob und wie es in diesem Punkt Anpassungen an den Anforderungsprozessen geben könnte, soll allerdings nicht an dieser Stelle, sondern in Kapitel 6 diskutiert werden.

[10] Das Wasserfallmodell ist eines der ersten Vorgehensmodelle der Softwareentwicklung, in dem mehrere Phasen sequentiell nacheinander durchlaufen werden müssen [AM08, S. 301].

4.2.2 Kontrollpunkte

Während der Softwareentwicklung im Unternehmen gibt es nach Vollendung von Teilzielen Kontrollpunkte, bei denen die formelle Qualität geprüft wird. Die Anforderungsprozesse werden im Unternehmen grob in sechs Kontrollpunkte unterteilt.

1. Der **„Proposal-to-Investigation Checkpoint“** ist der Kontrollpunkt am Übergang vom Produktvorschlag zur Untersuchung. An diesem Kontrollpunkt wird ein Produktvorschlagsdokument erwartet, das in einer abstrakten Form wenige Anforderungen enthält.

2. Der **„Definition Checkpoint“** ist der Kontrollpunkt, an dem die Anforderungsspezifikation größtenteils abgeschlossen ist. An diesem Punkt muss die SRS[11] abgeschlossen sein.

3. Der **„Investigation-to-Design Checkpoint“** ist der Kontrollpunkt am Übergang von der Untersuchung zum Entwurf. An diesem werden mehrere Dokumente überprüft: Die ERS[12], eine Rückverfolgbarkeitsmatrix von der SRS zur ERS sowie ein erster Projektplan.

4. Der **„Schedule Resource Commit Checkpoint“** dient der Überprüfung des endgültigen Projektplans, der in dieser Phase verfeinert werden kann.

5. Der **„Test Release Checkpoint“** ist der Kontrollpunkt, an dem das Produkt an die Testabteilung weitergegeben wird. Zu diesem Zeitpunkt wurden alle Implementierungen inklusive der Modultests abgeschlossen.

6. Der **„Manufacturing Release Checkpoint“** ist der Kontrollpunkt, an dem entschieden wird, ob das komplett getestete Produkt in die Produktion und ausgeliefert werden kann. Zu diesem Zeitpunkt müssen alle Dokumentationen abgeschlossen sein.

Diese Kontrollpunkte werden im Vergleich zu anderen Richtlinien des „Life Cycle“ nahezu vollständig eingehalten. Bei manchen Projekten werden die Kontrollpunkte „Definition Checkpoint“,„Investigation-to-Definition Checkpoint“ und „Schedule Resource Commit Checkpoint“ zeitlich zu einem Kontrollpunkt zusammengefasst. Inhaltlich bleibt die obige Erläuterung dennoch bestehen. Der Grund für diese strenge Einhaltung liegt größtenteils darin, dass die FDA[13] vorschreibt, dass bestimmte Dokumente und Kontrollpunkte eingehalten werden müssen.

4.2.3 Rollen

An den Anforderungsprozessen sind im Unternehmen viele verschiedene Personen involviert. Die Hauptakteure werden im Folgenden beschrieben:

[11] Erläuterung siehe Tabelle 2.
[12] Erläuterung siehe Tabelle 2.
[13] FDA ist eine Abkürzung für die „Food and Drug Administration“, die Arzneimittelbehörde der Vereinigten Staaten.

1. Der **Produktmanager** ist verantwortlich für das Produkt, das erstellt werden soll. Er stammt aus der Marketingabteilung und ist der interne Kunde, der einen Softwareauftrag aufgibt. Damit kann er bestimmen, welche Anforderungen umgesetzt werden sollen.

2. Der **Projektmanager** der F&E-Abteilung ist der Leiter des betreuenden F&E-Teams. Er steht in einem engen Kontakt mit dem Produktmanager und definiert gemeinsam mit ihm, welche Anforderungen umgesetzt werden. Eine weitere maßgebliche Aufgabe für ihn ist das Erstellen des Projektplans.

3. Der **F&E-Ingenieur** ist derjenige, der die Anforderungen umsetzt. Er unterliegt den Anweisungen des Projektmanagers. Er führt außerdem Einheitentests an seinen Implementierungen selbständig durch.

4. Der **Anforderungsanalyst** ist dafür zuständig, die inhaltlich übermittelten Anforderungen niederzuschreiben. Diese Rolle ist bisher im Unternehmen mit nur einer Person besetzt, da die Rolle noch nicht lange eingeführt ist.

5. Die beiden Rollen des **Testmanagers** und des **Testers** werden im Folgenden nicht weiter unterschieden. Sie sind komplett für das Testen der Implementierung zuständig.

6. Die **Qualitätssicherungsrolle** wird in der Prozessdarstellung nicht angegeben, da ihre Funktion keinen wesentlichen inhaltlichen Einfluss auf Anforderungen hat. Ihre Aufgabe ist die formelle Überprüfung von Dokumenten an den jeweiligen Kontrollpunkten.

Tabelle 2: Überblick über benutzte Fachbegriffe und Abkürzungen

Abk.	**Fachbegriff**	**Erklärung**
ERS	External Reference Specification	Die entspricht im Deutschen dem Pflichtenheft.
SRS	Software Requirements Specification	Dies entspricht im Deutschen dem Lastenheft. Für eine detaillierte Erklärung siehe Unterkapitel 3.1 auf Seite 5.
MVS	Minimal Viable Solution	Dies beschreibt eine Auswahl aller Anforderungen, ohne die die Software keinesfalls akzeptabel wäre.

4.2.4 Phasen der Anforderungserhebung

Im Folgenden werden Auszüge der Anforderungsprozesse detailliert erläutert. Dabei können aufgrund des Umfangs nicht alle Bestandteile der Prozesse behandelt werden. Für eine detaillierte Darstellung der Anforderungsprozesse siehe Appendix A. Eine Übersicht der zeitlichen Abfolge aller Phasen und Kontrollpunkte ist in Abbildung 9 dargestellt.

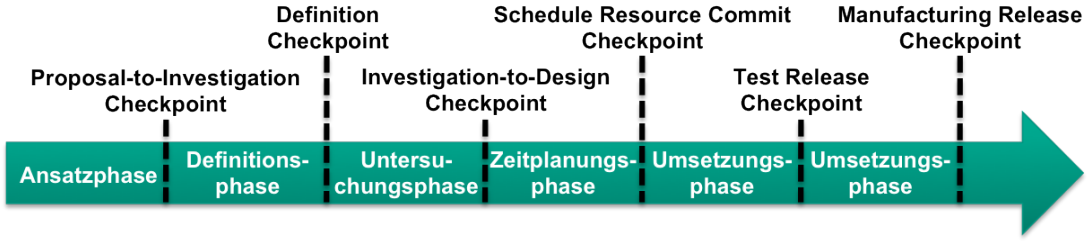


Abbildung 9: Phasen und Kontrollpunkte der Anforderungserhebung

Zum besseren Verständnis der folgenden Prozessausschnitte werden in Tabelle 3 die benutzten Abkürzungen erläutert.

Tabelle 3: Überblick über Rollen und englische Abkürzungen

Rolle (deutsch)	**Rolle (englisch)**	**Abkürzung**
Produktmanager	Product Manager	Product Mgr
Projektmanager	Project Manager	Project Mgr
F&E-Ingenieur	R&D Engineer	R&D Engr
Anforderungsanalyst	Requirements Analyst	Req. Analyst
Testmanager / Tester	Test Manager / Tester	Tester

Vorschlagsphase Als erstes erstellt der Produktmanager einen Produktvorschlag, der aus mehreren Textdokumenten besteht, die aber nur ansatzweise Anforderungen enthalten. Nachdem dieser Vorschlag überprüft wurde, erstellt der Produktmanager den Großteil der wichtigsten Anforderungen („MVS“[14]) in einer ersten SRS. Diese SRS inklusive des gesamten Ansatzes wird dann vom Produktmanager und vom Projektmanager überprüft um entweder über eine Anpassung oder ein Fortschreiten der Arbeit zu entscheiden.

Definitionsphase In der Definitionsphase (siehe Abbildung 10) werden die Anforderungen für das Projekt entwickelt. Dafür werden zunächst vom Produktmanager die wichtigsten Anforderungen („MVS“) fertiggestellt (1). Danach definiert der Anforderungsanalyst die weiteren Anforderungen und fügt sie der SRS hinzu (2). Die SRS wird dann von der F&E-Abteilung in einer Besprechung überprüft (3) und entweder mit Anpassungswünschen an den Produktmanager übergeben oder akzeptiert (4).

[14]Erläuterung siehe Tabelle 2.

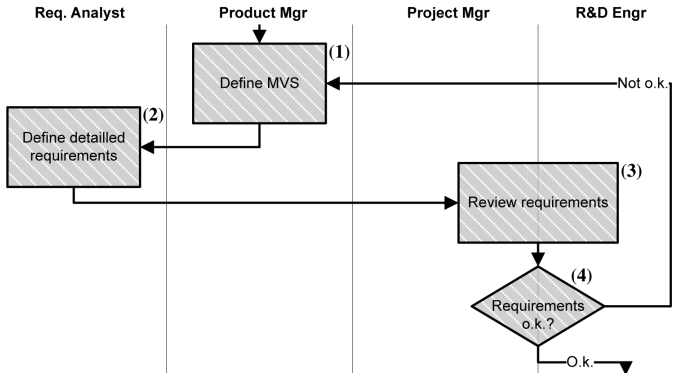


Abbildung 10: Die Definitionsphase der Softwarentwicklung

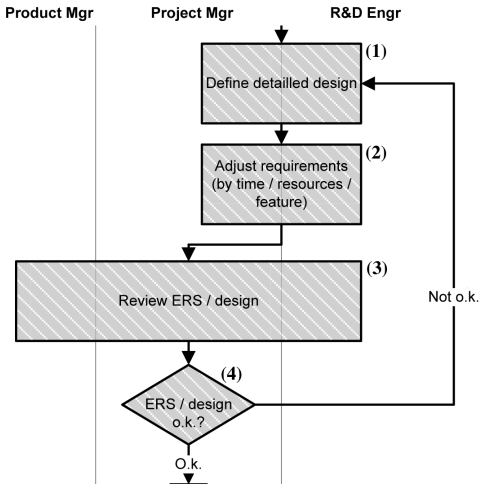


Abbildung 11: Der Softwareentwurf in der Untersuchungsphase

Untersuchungsphase Nach Akzeptieren der größtenteils fertiggestellten SRS folgen zwei parallele Arbeitsabläufe: Einer im F&E-Bereich und der andere im Testbereich.

Die Testabteilung erstellt zunächst einen initialen Testplan, der beinhaltet, welche Arten von Testfällen später entwickelt werden. Danach werden erste Schätzungen durchgeführt, wieviele Ressourcen für das Testen benötigt werden. Basierend auf der SRS und der in dieser Phase parallel erstellten ERS werden Testfälle in eigenen Testfalldokumenten erstellt.

Die F&E-Abteilung erstellt gleichzeitig den Softwareentwurf im ERS-Dokument (1) (siehe Abbildung 11). Dabei können, je nach Prioritätsstatus, Änderungen an der SRS vorgenommen werden (2). Ein Projekt kann entweder die Priorität „Zeit“, „Ressourcen“ oder „Feature“ besitzen. Je nach Wichtigkeit müssen dann zur Erfüllung dieses Prioritätsstatus Anforderungen gestrichen oder verändert werden. Nach diesem Schritt wird der Softwareentwurf in Form der ERS gemeinsam mit dem Produktmanager in einer Besprechung überprüft (3) und entweder freigegeben oder angepasst (4). Sofern die ERS freigegeben wurde, wird erstmalig eine Rückverfolgbarkeitsmatrix erstellt. Dies ist eine Tabelle, mit deren Hilfe rückverfolgbar

ist, weshalb bestimmte Entwurfsentscheidungen getroffen wurden.

Daraufhin beginnen erste zeitliche Projektplanungen (siehe Abbildung 12). Hierfür unter-

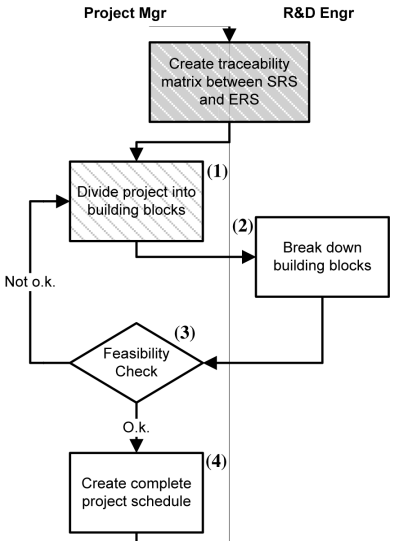


Abbildung 12: Die Projektzeitplanung in der Untersuchungsphase

teilt der Projektmanager das Projekt in mehrere Blöcke (1), die jeder F&E-Ingenieur selbständig implementieren kann. Die F&E-Ingenieure berechnen dann für ihre jeweiligen Blöcke den ungefähren zeitlichen Aufwand (2). Auf Basis dessen führt der Projektmanager eine Machbarkeitsstudie durch (3) und teilt bei negativem Ergebnis die Blöcke erneut auf oder erstellt einen Projektzeitplan bei positivem Ergebnis (4).

Zeitplanungsphase In dieser Phase verfeinert der Projektmanager den zuvor erstellten Projektzeitplan. Die Testabteilung kann in dieser Phase den Testplan finalisieren. Dies beinhaltet das Hinzufügen der Testkriterien für einen möglichen Regressionstest.

Umsetzungsphase In der Umsetzungsphase bleiben weiterhin die beiden vorher erwähnten Arbeitsabläufe parallel.

In der Testabteilung wird zunächst die Rückverfolgbarkeitsmatrix finalisiert, indem die ERS-Elemente mit den Testfällen verbunden werden. Danach folgt ein interner Abschluss bezüglich der Berechnung der benötigten Testressourcen. Mit diesem Abschluss ist der Arbeitsablauf beendet und wartet auf den zweiten, parallel ablaufenden Arbeitsablauf der F&E-Abteilung.

Hier wird zunächst der interne Softwareentwurf erstellt, der näher an der Implementierung ist, als der in Abbildung 11 erwähnte Softwareentwurf. Bei dieser Aktivität herrscht meist viel Kommunikation seitens der F&E-Abteilung mit dem Produktmanager, falls Änderungen am vorherigen Softwareentwurf notwendig sind. In seltenen Fällen, z.B. bei sehr großen Projekten, wird der interne Softwareentwurf formell dokumentiert. Danach werden Implementierungsmeilensteine definiert, sodass zeitlich nacheinander einzelne Module der Software implementiert werden können.

Die folgenden Aktivitäten werden für jeden Meilenstein jeweils ein Mal durchgeführt: Zunächst wird der Softwareentwurf für den Meilenstein verfeinert, damit danach der F&E-Ingenieur mit der Implementierung einzelner Einheiten im Meilenstein beginnen kann (1) (siehe Abbildung 13). Nach Implementierung einer Einheit wird diese vom F&E-Ingenieur

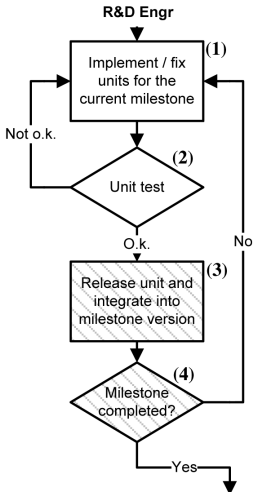


Abbildung 13: Einzelne Implementierungsschritte des F&E-Ingenieurs in der Umsetzungsphase

selbständig getestet (2). Hierfür existieren keine speziellen Testkriterien oder Anweisungen. Dieser Fakt wurde von vielen Mitarbeitern kritisiert, da der Einheitentest eine Grauzone sei, teilweise gar nicht durchgeführt werde und wenn, dann mache es jeder auf eine andere Weise. Wenn die Implementierung den Test erfüllt, wird die Einheit freigegeben und in die Meilenstein-Version integriert (3). Sofern weitere Einheiten zur Vollendung des Meilensteins existieren, die noch nicht implementiert wurden, werden die erwähnten Schritte wiederholt. Wenn der Meilenstein fertiggestellt wurde (4), wird mit der Implementierung des nächsten Meilensteins fortgesetzt, bis der letzte Meilenstein implementiert wurde. Parallel dazu wird von der Testabteilung ein interner Tester bereitgestellt, der nach der Implementierung des Meilensteins testet.

Wenn der letzte Meilenstein implementiert und getestet wurde, werden alle parallelen Arbeitsabläufe wieder zusammengeführt. Die Umsetzungsphase ist damit abgeschlossen und es erfolgt der Kontrollpunkt zum Übergang zur Testphase.

Testphase In der Testphase führen die Tester zunächst den Systemtest mit Hilfe der vorher definierten Testfälle durch. Sofern die Software einen Testfall nicht erfüllt, muss der Fehler von einem F&E-Ingenieur behoben werden. Wenn alle Testfälle erfüllt wurden, wird ein erster Veröffentlichungskandidat für einen weiteren Test freigegeben. Die Dokumentation, ob ein Testfall bestanden wurde oder nicht, erfolgt in einem anderen System als der Rückverfolgbarkeitsmatrix, obwohl dies dort thematisch gut passen würde. Das Problem ist, dass die Testfälle in einem System, deren Ergebnisse aber in einem anderen System abgelegt sind.

Dies führt dazu, dass es bei Änderungen sehr schwer ist, alle Systeme auf dem aktuellen Stand zu halten.

Der Veröffentlichungskandidat wird einem speziellen Test unterzogen (1), der eine willkürliche Auswahl der vorher definierten Testfälle beinhaltet (siehe Abbildung 14). Die Auswahl dieser Testfälle beruht auf subjektiven Erfahrungswerten des Testmanagers. Die Auswahl ist dennoch nachvollziehbar, da sie dokumentiert wird. Sofern bei diesem Test Fehler auftre-

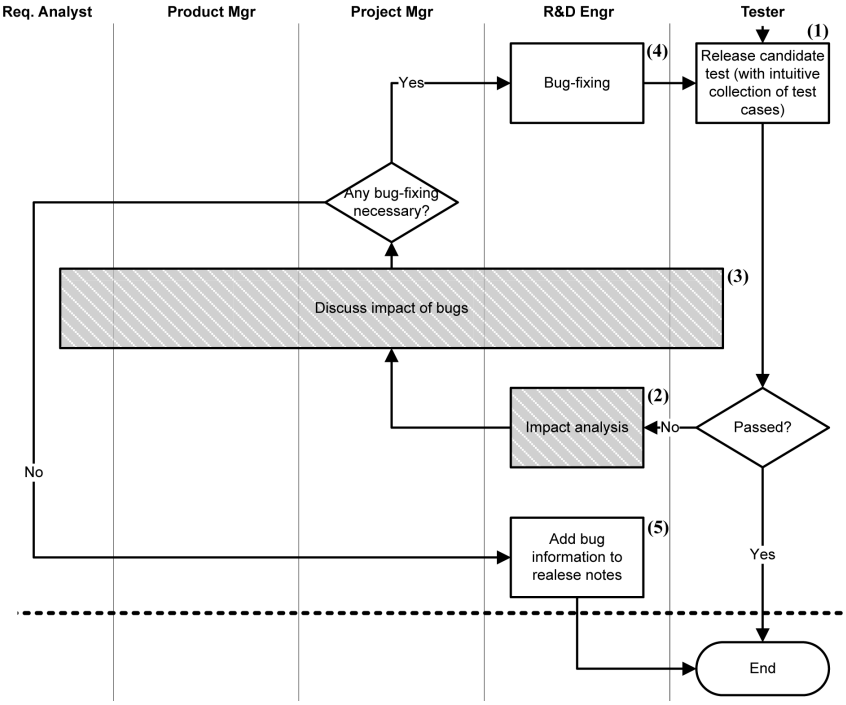


Abbildung 14: Die letzten Schritte in der Testphase

ten, wird eine Auswirkungsanalyse (2) bezüglich einer möglichen Fehlerbehebung vom F&E-Ingenieur durchgeführt. Ein Gremium bestimmt dann, ob der Fehler behoben werden muss, oder nicht (3). Dabei wird der Aufwand der Fehlerbehebung mit dem Nutzen verglichen. Sofern eine Fehlerbehebung als notwendig angesehen wird, muss der F&E-Ingenieur diese durchführen (4). Danach wird ein neuer Veröffentlichungskandidat freigegeben, der erneut getestet werden muss. Wurde entschieden, dass eine Fehlerbehebung nicht notwendig ist, muss der F&E-Ingenieur in die Veröffentlichungsnotizen einen Hinweis auf den gefundenen Fehler eintragen (5).

Wenn der endgültige Test bestanden wurde oder die nicht behebungswürdigen Fehler dokumentiert wurden, ist die Testphase abgeschlossen. Danach geht das Projekt in die Herstellungsphase, die im Rahmen dieser Arbeit nicht weiter betrachtet wird.

4.3 Zusammenfassung

In diesem Kapitel wurde erklärt, wie die aktuellen Anforderungsprozesse des Unternehmens erfasst und aufgenommen wurden. Daneben wurde an den jeweiligen Stellen Probleme bzw. Schwächen der Anforderungsprozesse identifiziert. Es existieren darüber hinaus allgemeine Schwächen, die sich nicht einzelnen Prozessschritten zuordnen ließen.

Ein generelles Problem ist die Änderungsverwaltung. Oft werden Änderungen an den Anforderungen nicht für alle bzw. nicht für die notwendigen Mitarbeiter sichtbar. Dies führt dazu, dass Dokumente, die nur von einzelnen Mitarbeitern gepflegt werden, nicht auf dem aktuellen Stand sind. Beispielsweise haben F&E-Ingenieure die Möglichkeit, nachträglich Änderungen an der ERS durchzuführen. Diese Änderungen werden aber oft nicht auf die SRS übertragen. Die Testabteilung erstellt dann allerdings Testfälle, die auf veralteten Anforderungen basieren. Dadurch kann es sein, dass bestimmte Teile nicht geprüft werden.

Der hier erwähnte Änderungsfall ist kein Sonderfall. In der Darstellung der Anforderungsprozesse des Unternehmens in Appendix A ließen sich die Änderungsfälle aufgrund der Übersichtlichkeit nicht darstellen. Es ist von fast jedem Punkt aus möglich, zur Definition der Anforderungen zurückzuspringen. Ein Verbot dieser Rücksprünge wäre aber nicht ohne Weiteres zu empfehlen. Rücksprünge machen die Softwareentwicklung flexibel.

Die in diesem Kapitel genannten Schwächen und Potentiale werden Ausgangspunkt für die Optimierung der Anforderungsprozesse in Kapitel 6 sein.

5 Vergleich von Anforderungsverwaltungssystemen

In diesem Kapitel wird ein Vergleich einer Auswahl der auf dem Markt verfügbaren Anforderungsverwaltungssysteme durchgeführt. Zunächst wird in Unterkapitel 5.1 ein kurzer Blick auf den Markt der Anforderungsverwaltungssysteme geworfen. Verwandte Arbeiten im Bereich des Vergleichs von Anforderungsverwaltungssystemen werden in Unterkapitel 5.2 behandelt. Danach werden zuerst in Unterkapitel 5.3 die Kriterien zum Vergleich der Systeme vorgestellt und danach in Unterkapitel 5.4 anhand der zuvor erläuterten Kriterien die Systeme miteinander verglichen. Abschließend werden die Ergebnisse dieses Kapitels in Unterkapitel 5.5 zusammengefasst.

5.1 Überblick der Anforderungsverwaltungssysteme

In den meisten Unternehmen werden heutzutage die Microsoft Produkte Word und Excel für die Anforderungsverwaltung verwendet [HW05, S. 217]. Neben diesen Produkten, die nicht für die Anforderungsverwaltung ausgelegt sind und sich daher nicht sonderlich dafür eignen, gibt es eine Vielzahl von Anforderungsverwaltungssystemen[15]. Die Softwareindustrie hat früh erkannt, dass die Anforderungsverwaltung ein potentiell großer Markt ist. Ein Vergleich aller Systeme würde den Rahmen dieser Arbeit sprengen. Daher werden im Folgenden nur die Systeme miteinander verglichen, die entweder ausdrücklich vom Unternehmen erwünscht sind oder aufgrund ihres hohen Marktanteils in eine nähere Untersuchung einbezogen wurden.

Borland CaliberRM ist ein System, das viele Freiheiten zur Anpassung bietet. Anpassungen z.B. der Anforderungstypen oder der Benachrichtigungen können mit einer Benutzeroberfläche vom Administrator durchgeführt werden. Der Preis für eine Einzellizenz beträgt $2400.

IBM Rational DOORS ist der klare Marktführer und wird daher oft von anderen Herstellern zumindest teilweise kopiert und preisgünstiger vertrieben. Vor allem Polarion wirbt damit, dass es sehr stark an das Design von DOORS angelehnt ist[16]. Der Preis für eine Einzellizenz beträgt $ 2200.

IBM Rational RequisitePro basiert auf der Textverarbeitungsanwendung Microsoft Word und synchronisiert diese Textdokumente während der Laufzeit mit einer Datenbank. Damit bietet es dem Nutzer eine bekannte Arbeitsumgebung. Dieses System wurde bereits vor einigen Jahren beim Unternehmen verwendet. Es setzte sich allerdings nicht durch, da es neben eventuellen Schwächen des Systems in einem zu großen Pilotprojekt eingesetzt wurde. Der Preis für eine Einzellizenz beträgt $2694.

Polarion Requirements ist ein web-basierendes Anforderungsverwaltungssystem. Es nutzt die üblichen Web 2.0 Kollaborationsmöglichkeiten und ist über XML-Dateien anpassbar. Das System basiert auf einem SVN-Repository zur Versionskontrolle und

[15] Eine Auflistung verschiedener Anforderungsverwaltungssysteme ist unter `http://www.volere.co.uk/tools.htmoderhttp://www.incose.org/ProductsPubs/products/rmsurvey.aspx` zu finden.
[16] siehe `http://www.polarion.com/products/requirements/index.php`.

ist vom Äußerlichen sehr stark an das Design von DOORS angepasst. Der Preis für eine Einzellizenz gehört mit $990 zu den günstigsten.

Serena Dimensions RM ist ein System, das im Gegensatz zu RequisitePro auf eine eigene Benutzeroberfläche setzt. Die dahinterliegende Datenbank stammt von Oracle. Der Preis für eine Einzellizenz beträgt ca. $2500.

5.2 Verwandte Arbeiten

Aufgrund der Vielzahl an Anforderungsverwaltungssystemen, die auf dem Markt existieren, gibt es mehrere Studien, die einen Vergleich bereits durchgeführt haben. Diese sollen zunächst betrachtet werden.

Eine in der Softwareentwicklung sehr bekannte Studie stammt von der Standish Group. Der CHAOS Report wurde bereits in Unterkapitel 1.1 (auf Seite 1) erwähnt. Neben der Untersuchung des Scheiterns der Projekte untersuchte die Standish Group außerdem, welche Anforderungsverwaltungssysteme verwendet wurden.

Dabei ergeben sich folgende Marktanteile (siehe Abbildung 15):

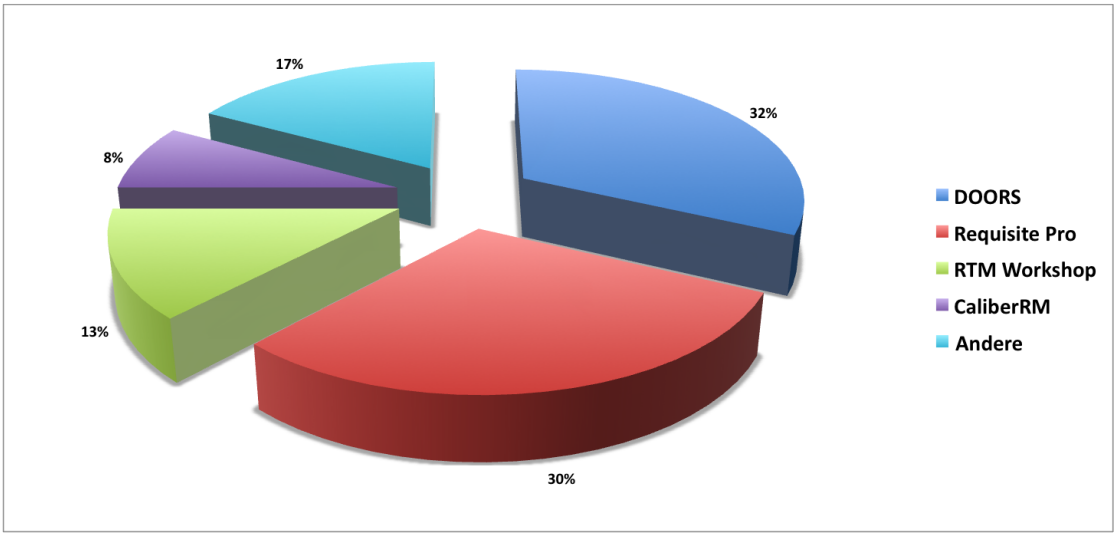


Abbildung 15: Marktanteile der einzelnen Werkzeuge für die Anforderungsverwaltung [SG09]

DOORS ist bereits zum sechsten Mal in Folge der Marktführer [HHMS04, S. 16]. RequisitePro und CaliberRM schneiden ebenfalls in der Spitzengruppe der Anforderungsverwaltungssysteme ab.

Marktanteile geben allerdings nicht immer einen Hinweis auf die Qualität und Güte eines Produkts. Zum Erreichen eines hohen Marktanteils spielen viele andere Faktoren eine Rolle. Daher haben die französischen Analysten von Yphise einen nach ISO 9001 zertifizierten Beurteilungsbericht veröffentlicht [Yph02]. Dabei haben sie fünf Anforderungsverwaltungssysteme, darunter DOORS, RequisitePro und CaliberRM nach folgenden fünf Kriterien

verglichen:

1. Sicherstellung der Qualität und Präzision der Anforderungen
2. Verwaltung der Projekterfordernisse und der Prioritäten des Projekts
3. Sicherstellung der Anforderungserfüllung durch die einzelnen Projektphasen
4. Verwaltung von Anforderungsänderungen
5. Gemeinsame Verwaltung von Anforderungen unterschiedlicher Projekte

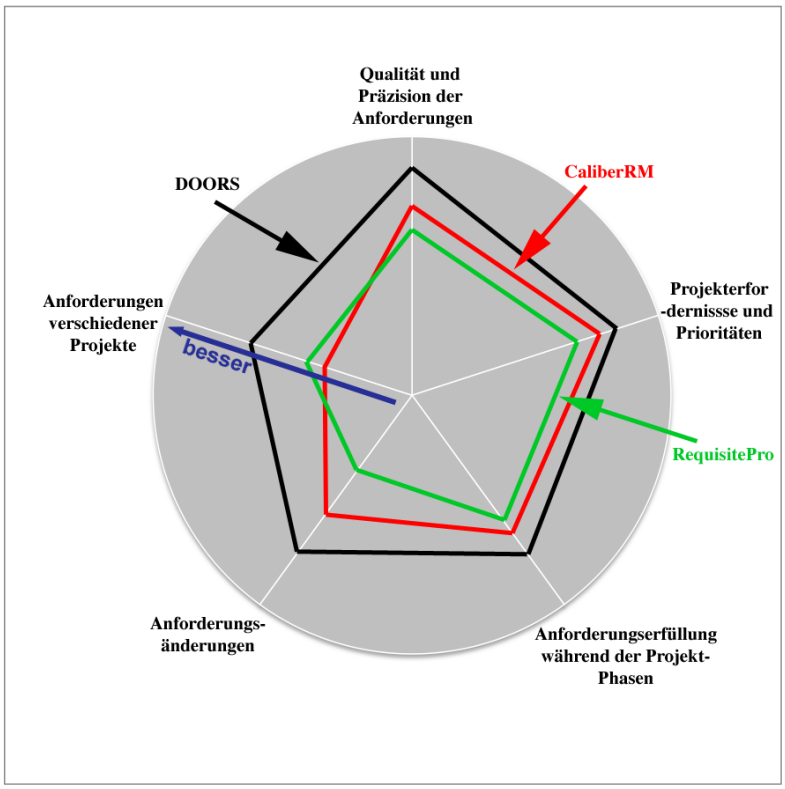


Abbildung 16: Das Ergebnis des Yphise-Berichts [Yph02]

Das Ergebnis des Berichts ist in Abbildung 16 abgebildet.DOORS hat als das beste Anforderungsverwaltungssystem abgeschnitten. Danach folgt CaliberRM knapp vor RequisitePro. DOORS setzt sich in dem Bericht vor allem in den Punkten „Anforderungsänderungen“ und „Anforderungen verschiedener Projekte“ durch. In den anderen Kriterien ist CaliberRM auf ähnlichem Niveau wie DOORS. An dieser Stelle soll erwähnt werden, dass dieser Bericht aus dem Jahr 2002 stammt und sich die Ergebnisse geändert haben könnten. Dennoch hat DOORS 2004 und 2006 erneut als das beste Anforderungsverwaltungssystem in den Vergleichen von Yphise abgeschnitten [Pre06].

2003 hat die META Group in einer weiteren Untersuchung DOORS erneut als den Marktführer der Anforderungsverwaltungssysteme gekürt [HHMS04, S. 19]. Dabei seien vor allem folgende Kriterien entscheidend gewesen:

1. Integration der Testwerkzeuge von HP Software (ehemals Mercury)
2. Integration mit Microsoft Project
3. Unterstützung des gesamten Softwarelebenszyklus

DOORS hat somit in Untersuchungen, bei denen allgemeine Kriterien geprüft wurden, als das beste Anforderungsverwaltungssystem abgeschlossen. Im Folgenden wird gezeigt, welche Kriterien für das Unternehmen entscheidend sind. Hieraus kann letztendlich im Vergleich ein anderes Ergebnis folgen.

5.3 Vergleichskriterien

Zum Vergleich der fünf ausgewählten Anforderungsverwaltungssysteme aus Unterkapitel 5.1 (auf Seite 29f.) wurden einige Mitarbeiter befragt. Grundsätzlich haben die Systeme dieselben Grundfunktionen wie z.B. Anforderungseingabe, Verbindung von Anforderungen, Rückverfolgbarkeit, etc.

Trotzdem gibt es an vielen Stellen Unterschiede zwischen den Systemen. Daher wurden die Mitarbeiter nach ihren fünf wichtigsten Anforderungen an ein neues Anforderungsverwaltungssystem befragt. Im Folgenden werden die Anforderungen aufgezählt und erläutert, wobei sie nach ihrer Wichtigkeit absteigend sortiert sind.

1. Die **Anzeige der Produktanforderungen mehrerer Projekte** wird als die wichtigste Anforderung eingestuft. Bisher ist dies im Unternehmen noch nicht möglich, da die Anforderungen einzelner Projekte in Word-Dateien dokumentiert werden. Somit ist eine Gesamtübersicht über alle Anforderungen nur mit viel Aufwand erstellbar. Oft existieren auch nur Delta-Anforderungen und Verweise zu älteren Lastenheften, auf die Mitarbeiter eventuell keinen Zugriff haben.

2. Die **Möglichkeit, Projekt- und Produktgrenzen zu überschreiten** hat Ähnlichkeiten zur vorherigen Anforderung. Hier erfordert es allerdings zusätzlich, dass Anforderungen verschiedener Projekte oder sogar Produkte nicht nur nebeneinander angezeigt werden können, sondern z.B. Verbindungen zwischen ihnen hergestellt werden können. Dies ist vor allem im Hinblick auf die Rückverfolgbarkeit, die im nächsten Punkt beschrieben wird, wichtig.

3. Die **Rückverfolgbarkeit** ist, wie bereits vorhergehend erwähnt, eine der Grundfunktionen von Anforderungsverwaltungssysteme. Wenn ein Kunde aufgrund eines möglichen Produktfehlers einen Audit durchführt, muss nachvollziehbar sein, ob die angeblich fehlerhaft umgesetzte Kundenanforderung z.B. eine Systemanforderung, eine Designvorschrift, eine Implementierung und einen Testfall impliziert hat. Während der Projektlaufzeit werden darauf hinarbeitend Verbindungen zwischen diesen Elementen gesetzt.

Rückverfolgbarkeit bezeichnet die Möglichkeit, diese Verbindungen jederzeit in irgendeiner Weise, z.B. in einer Rückverfolgbarkeitsmatrix, darstellbar zu machen [RR06, S. 535]. Obwohl dies eine Grundfunktion ist, wurde diese Anforderung als Vergleichskriterium herangezogen. Den Mitarbeitern des Unternehmens ist diese Funktion sehr wichtig.

4. **Benachrichtigungen** sind ein wichtiger Bestandteil der Änderungsverwaltung [VSH01, S. 40ff.], weil Interessenvertreter im Normalfall die Anforderungen an das Produkt während des Projektverlauf ändern [Rup09, S. 345]. Interessierte Personen sollten bei Anforderungsänderungen vom System vollautomatisch z.B. per E-Mail benachrichtigt werden.

5. Die **Versionskontrolle** der Anforderungen ist ein weiterer Bestandteil um nachträglich alle Handlungen während eines Softwareprojekts nachvollziehen zu können. Zusätzlich dazu sollen auch Notizen oder Diskussionen bei Anforderungsänderungen anhängbar sein.

6. Zur Durchführung von Durchsichten, die üblicherweise vor dem Abschluss von Projektphasen stattfinden, muss ein Anforderungsverwaltungssystem eine **Unterstützung der Durchsicht** anbieten.

7. Bei Anforderungsänderungen entsteht meist ein Umsetzungsaufwand, der mit dem bloßen Auge nicht immer erkennbar ist. Mit **Auswirkungsanalysen** kann ein Implementierer eine Aufwandsabschätzung durchführen. Das System sollte den Implementierer bei dieser Aufgabe unterstützen.

8. Neben den üblichen Anforderungen, die in dem System erfasst werden, kann es auch Anforderungen geben die in anderen Dokumenten abgelegt werden. Solche **externen Spezifikationen** sollten mit Hilfe des Anforderungsverwaltungssystems auch rückverfolgbar sein.

9. Produkte erreichen im Laufe eines Projektzeitraums verschiedene Status. Zuständige Personen können mit **digitalen Signaturen** rechtlich bindende Zusagen treffen, ob ein Status erreicht wurde. Das System sollte daher eine Integration digitaler Signaturen unterstützen.

10. Das System sollte eine **automatische Berichterstellung** anbieten. Dabei sollten je nach Projektphase unterschiedliche Ausschnitte aller Anforderungen im Bericht enthalten sein.

11. Das System sollte die Möglichkeit bieten, Nutzergruppen in **Rollenprofile** einzuteilen und diesen verschiedene Rechte zu gewähren.

12. Im Unternehmen findet bisher eine Unterscheidung zwischen Anwendungsfällen und Systemanforderungen nur selten statt. Hierfür soll das System eine Möglichkeit geben **Anforderungstypen unterscheidbar** zu machen.

13. Ein häufiges Problem in der Erhebung ist die Widersprüchlichkeit von Anforderungen [Mey85, S. 7]. Um dies zu verhindern, kann eine **Konsistenzprüfung** durchgeführt werden, die das System unterstützen sollte.

14. Um eine allgegenwärtige Erreichbarkeit des Systems und eine Vereinfachung des Ausrollens des Systems zu erreichen, sollte das System über ein funktionierendes **Web-Interface** verfügen.

5.4 Vergleich

Zur Ermittlung der Erfüllung der vorhergehend genannten Anforderungen wurden verschiedene Quellen ausgewertet: Für die Produkte CaliberRM und Polarion hat das Unternehmen bereits im Vorfeld von den Systemherstellern spezifische Anforderungskataloge ausfüllen lassen. Zusätzlich dazu stellt INCOSE[17] eine umfangreiche Vergleichsmatrix[18] bereit, in der die meisten auf dem Markt verfügbaren Anforderungsverwaltungssysteme einander gegenübergestellt werden. Letztendlich wurden für weitere Details Handbücher und Demos von den Herstellerseiten verwendet.

Im Unterkapitel 5.5 werden die Ergebnisse in einer Tabelle zusammengefasst, um einen Überblick über alle Anforderungen und ihre Erfüllung zu bekommen.

5.4.1 Anzeige der Produktanforderungen mehrerer Projekte

Mit CaliberRM, Polarion und DOORS können alle Anforderungen eines Produkts über mehrere Projekte hinweg betrachtet werden.

Bei Dimensions RM kann diese Funktion nur über einen Hilfsweg erreicht werden. Standardmäßig hat der Nutzer nur die Sicht eines Projekts, da man sich bei der Anmeldung in das System für ein Projekt entscheiden muss. Allerdings können so genannte „Collections“ angelegt werden, in die Anforderungen beliebiger Projekte eingefügt werden können.

Mit RequisitePro ist eine Gesamtsicht über Anforderungen verschiedener Projekte nur mit vielen Umständen möglich, da das System projektgetrieben aufgebaut ist.

5.4.2 Überschreitbarkeit der Projekt- und Produktgrenzen

Bei CaliberRM und Polarion ist es standardmäßig möglich, Anforderungen von verschiedenen Projekten oder Produkten zu verbinden.

Mit RequisitePro ist dies nur über Umwege möglich. Zur Verbindung von Anforderungen verschiedener Projekte kann die Anforderung des anderen Projekts als „externe Spezifikation“ eingebunden werden. Da diese Vorgehensweise allerdings sehr aufwendig ist, wird sie als nicht praktikabel angesehen. Dimensions RM bietet auch schon wie in Unterkapitel 5.4.1 die Möglichkeit, über den Umweg der „Collections“ mit Anforderungen verschiedener Projekte anzulegen und damit die Projektgrenzen zu durchbrechen.

[17] International Council on System Engineerung.

[18] Die Vergleichsmatrix kann hier heruntergeladen werden: `http://www.incose.org/ProductsPubs/products/rmsurvey.aspx`.

Für DOORS konnte diese Anforderung nicht geprüft werden, da die Quellen dazu keine Aussage lieferten und ein Produkttest aufgrund des hohen zeitlichen Aufwands einer Installation nicht möglich war.

5.4.3 Rückverfolgbarkeit

Eine Rückverfolgbarkeit ist grundsätzlich bei allen Systemen möglich, da dies, wie bereits am Anfang von Kapitel 5.3 (auf Seite 32f.) erwähnt, eine der Grundfunktionen eines Anforderungsverwaltungssystems ist.

DOORS ragt hier leicht aus der Masse heraus, da es in diesem Punkt eine besonders gute Handhabung bietet. Per „Drag&Drop“ können Verbindungsrichtungen eingestellt werden.

5.4.4 Benachrichtigung

Alle Systeme bis auf DOORS bieten eine Benachrichtigung bei Änderungen der Anforderungen an. In CaliberRM können die Regeln für Benachrichtigungen in einer benutzerfreundlichen Oberfläche angepasst werden. Bei Polarion müssen die Regeln anhand von XML-Files in das System übertragen werden.

Sowohl das Benutzerhandbuch als auch das Administratorenhandbuch von DOORS äußern sich nicht zu einer solchen Funktion. Lediglich im Bezug auf eingereichte Anforderungsentwürfe von Benutzern, gibt es eine Benachrichtigungsfunktion. Bei Annahme oder Ablehnung des Entwurfs kann eine E-Mail an den Ersteller verschickt werden. Somit existiert zumindest teilweise eine Benachrichtigungsfunktion und sie könnte mit einer Softwareanpassung wahrscheinlich auch für weitere Ereignisse genutzt werden.

5.4.5 Versionskontrolle

Alle Systeme bieten standardmäßig eine Versionskontrolle der Anforderungen inklusive einer Gegenüberstellung zweier Versionen an.

Die Funktion, Kommentaren anzuhängen, erfüllt nur RequisitePro nicht. Polarion bietet hier aufgrund der Nutzung von Web 2.0 Funktionen sogar die Möglichkeit, Diskussionen im Form eines Forums zu eröffnen.

Dimensions RM bietet darüber hinaus die Möglichkeit an, verschiedene Versionen einer Anforderung mit anderen Objekten zu verbinden. Dies ist gerade im Hinblick auf die Situation wichtig, wenn ein Projekt eines älteren Produkts später für die Integration in ein anderes, neues Produkt verbessert wird. Dann sollen die Verbindungen der alten Anforderungen des Projekts zum alten Produkt bestehen bleiben.

5.4.6 Unterstützung der Durchsicht

Alle Systeme haben die Funktion, Berichte zu erstellen, womit der Prozess zur Durchsicht unterstützt werden kann. Berichte beinhalten eine Auswahl von Anforderungen und weitere Darstellungen wie z.B. eine Rückverfolgungsmatrix.

Polarion, Dimensions RM und DOORS bieten zusätzlich an, den eigenen Ablauf im System abzubilden und somit den Prozess zur Durchsicht noch besser zu unterstützen. Bei Polarion

wird der Ablaufplan in XML erstellt. In Dimensions RM können Skripte erstellt werden, die eine Folge von Dokumenten, Rückverfolgbarkeitsmatrizen und anderen Berichtstypen generieren. Diese Folge kann so definiert werden, dass sie den Ablaufplan abbildet. DOORS unterstützt den Prozess zur Durchsicht durch einen speziellen „Change Proposal Workflow“.

5.4.7 Auswirkungsanalyse

Alle Systeme können die Auswirkungsanalyse nur eingeschränkt unterstützen. Der Aufwand durch Änderung einer Anforderung kann nicht automatisch geschätzt werden.

Es kann lediglich automatisch angezeigt werden, welche Verbindungen zwischen den Anforderungen bei einer Änderung verdächtig[19] sind. Ein Verdacht kann bedeuten, dass bei Änderung einer Anforderung die damit verbundenen Anforderungen auch geändert werden müssen.

Bei Polarion muss diese Funktion durch eine manuelle Anpassung zunächst eingestellt werden. Bei Erstellung von Anforderungen und Verbindungen muss hierbei angegeben werden, ob bei der Änderung einer bestimmten Anforderung diese Verbindung als verdächtig gekennzeichnet werden soll.

5.4.8 Externe Spezifikationen

Nur CaliberRM unterstützt eine Einbindung und Rückverfolgbarkeit von externen Spezifikationen. Neben der standardmäßigen Integrationsmöglichkeit von der im Unternehmen benutzten Entwicklungsumgebung ClearCase stellt Borland eine API[20] zur Verfügung, mit der jede Schnittstelle zu den im Unternehmen benutzten Systemen erstellt werden kann.

Polarion bietet die Integration von ClearCase nur gegen Gebühr als Zusatzsoftware an.

DOORS bietet zwar das Einbinden externer Spezifikationen an, ignoriert diese Verbindungen allerdings bei vielen ihrer Funktionen.

Bei Dimensions RM kann die Einbindung externer Spezifikationen über das Importieren der externen Datei erfolgen. Dieser Umweg wird allerdings wegen des hohen Aufwands nicht als praktikabel angesehen. Andere Möglichkeiten zur Verbindung mit externen Spezifikationen waren im Benutzerhandbuch nicht zu finden.

RequisitePro unterstützt das Einbinden externer Spezifikationen nicht bzw. auch nur über den Umweg, die externen Spezifikationen zu importieren. Auch hier wird dieser Umweg aufgrund des hohen Aufwands als nicht praktikabel angesehen.

5.4.9 Digitale Signaturen

Nur DOORS besitzt die Möglichkeit, digitale Signaturen in das System zu integrieren und somit bestimmte Ausschnitte der Anforderungen zu bestimmtem Projektphasen rechtlich bindend zu akzeptieren. Alle anderen Systeme geben in den Benutzerhandbüchern keinen Hinweis darauf, eine ähnliche Funktionalität aufzuweisen.

[19] So genannte „Suspect links“.
[20] Eine API ist ein Application Programming Interface, zu deutsch Programmierschnittstelle.

5.4.10 Automatische Berichterstellung

Bis auf Polarion bieten alle Systeme einfache Möglichkeiten an, Berichte automatisch und mit wenig Aufwand zu erstellen. Für Polarion müssen die Vorlagen in XML-Dateien erstellt werden. Für Unternehmen bedeutet dies, sofern sie kein Personal mit XML-Kenntnissen haben, dass Zusatzkosten für einen Berater entstehen, der diese Anpassung durchführt.

CaliberRM erzeugt über die „Document Factory“ ein Word-Dokument, das je nach Einstellung eine bestimmte Auswahl der Anforderungen enthält. Bei RequisitePro und DOORS können maßgeschneiderte Anfragen je nach Projektphase abgespeichert werden und so immer wieder mit wenig Aufwand erzeugt werden. Dimensions RM verfügt über einen „Document View“, der nach Filterung als Bericht verwendet werden kann.

5.4.11 Rollenprofile

Klare, anpassbare Rollenprofile können in allen Systemen vollständig eingestellt werden. Hierbei sind bis auf kleine Unterschiede in der Handhabung keine Differenzen erkennbar.

5.4.12 Unterscheidung zwischen Anforderungstypen

Eine klare Unterscheidung zwischen den Anforderungstypen bei ihrer Erfassung um den Benutzer vor einer falschen Anforderungsaufnahme zu beschützen, bietet kein System zur vollständigen Zufriedenheit an. Grundsätzlich unterstützen alle Systeme selbst definierbare Anforderungstypen, sodass zur Unterscheidung zwischen Anwendungsfällen und Systemanforderungen beispielsweise verschiedene Felder auszufüllen sind.

5.4.13 Konsistenzprüfung

Eine Konsistenzprüfung ist mit keinem der getesteten Systeme möglich. Hierunter versteht man beispielsweise das Erkennen eines Widerspruchs zwischen zwei Anforderungen.

Um eine solche Konsistenzprüfung durchzuführen, müssten Anforderungen mit semantischen Informationen angereichert werden, die maschinenlesbar sind. Dies ist momentan ein Forschungsgebiet und daher noch nicht in der Praxis einsetzbar. Für interessierte Leser sei hierbei angemerkt, dass im Rahmen des RECAA[21]-Projekts des Instituts für Programmstrukturen und Datenorganisation am KIT[22] ein solcher Ansatz verfolgt wird.

5.4.14 Web-Interface

Über eine Web-Interface verfügen alle Systeme. Im Fall von Polarion ist dies sogar der einzig mögliche Weg, um auf das System zuzugreifen. Das Web-Interface von Dimensions RM hat als einziges System keinen administrativen Zugang.

[21] Requirements Engineering Complete Automation Approach
[22] Das KIT ist das Karlsruher Institut für Technologie, ehemals Universität Karlsruhe (TH).

5.5 Zusammenfassung

Die Ergebnisse des gesamten Vergleichs können in Tabelle 4 betrachtet werden. Die verwendeten Symbole haben folgende Bedeutung:

1. Volle Unterstützung einer Anforderung: • •
2. Teilweise Unterstützung einer Anforderung oder über einen Umweg: ○ •
3. Keine oder mangelhafte Unterstützung einer Anforderung: ○ ○

Insgesamt hat für die Anforderungen des Unternehmens CaliberRM von Borland am besten abgeschnitten. Schwächen hat das System allerdings trotzdem in der Unterstützung der Durchsicht und bei der Integration der digitalen Signaturen. In beiden Fällen ist der Marktführer DOORS von IBM Rational besser aufgestellt als CaliberRM.

Das andere Anforderungsverwaltungssystem von IBM Rational, RequisitePro, hat am schlechtesten abgeschnitten. Vor einigen Jahren wurde es bereits im Unternehmen getestet und in einem Pilotprojekt eingesetzt. Allerdings war das Pilotprojekt für die Neueinführung eines Anfoderungsverwaltungssystems zu komplex. Darüber hinaus waren die Mitarbeiter mit dem System nicht zufrieden, sodass das System nicht global im Unternehmen ausgerollt wurde.

Tabelle 4: Zusammenfassung der Ergebnisse des Vergleichs

Anforderung	**CaliberRM**	**Dimensions RM**	**DOORS**	**Polarion**	**RequisitePro**
1. Anzeige der Produktanforderungen mehrerer Projekte	● ●	○ ●	● ●	● ●	○ ○
2. Überschreitbarkeit der Projekt- und Produktgrenzen	● ●	○ ●	k.A.[1)]	● ●	○ ○
3. Rückverfolgbarkeit	● ●	● ●	● ●	● ●	● ●
4. Benachrichtigung	● ●	● ●	○ ●	● ●	● ●
5. Versionskontrolle	● ●	● ●	● ●	● ●	○ ●
6. Unterstützung der Durchsicht	○ ●	● ●	● ●	● ●	○ ●
7. Auswirkungsanalyse (eingeschränkte Version)	● ●	● ●	● ●	○ ●	● ●
8. Externe Spezifikationen	● ●	○ ○	○ ●	○ ●	○ ○
9. Digitale Signierung	○ ○	○ ○	● ●	○ ○	○ ○
10. Automatische Berichterstattung	● ●	● ●	● ●	○ ●	● ●
11. Rollenprofile	● ●	● ●	● ●	● ●	● ●
12. Unterscheidung zwischen Anforderungstypen	○ ●	○ ●	○ ●	○ ●	○ ●
13. Konsistenzprüfung	○ ○	○ ○	○ ○	○ ○	○ ○
14. Web-Interface	● ●	○ ●	● ●	● ●	● ●
Gesamtwertung	**1.**	4.	2.	3.	5.

[1)] Unterstützung der Anforderung konnte nicht ermittelt werden

● ● Volle Unterstützung einer Anforderung

● ○ Teilweise Unterstützung einer Anforderung oder über einen Umweg

○ ○ Keine oder mangelhafte Unterstützung einer Anforderung

6 Optimierung der Anforderungsprozesse

In diesem Kapitel wird durch eine Anpassung der bestehenden Anforderungsprozesse des Unternehmens ein Entwurf zur Optimierung dieser Prozesse erstellt. Zunächst werden dafür in Unterkapitel 6.1 aus den in Unterkapitel 3.3 vorgestellten Anforderungsprozessen diejenigen identifiziert, die für das Unternehmen sinnvoll einsetzbar wären. Danach wird in Unterkapitel 6.2 jeder identifizierte Prozess mit Teilen der bestehenden Anforderungsprozesse des Unternehmens verglichen, um Potentiale aufzudecken. An dieser Stelle wird auch erwähnt, welche Änderungen der Anforderungsprozesse vorgeschlagen werden. Abschließend werden in Unterkapitel 6.3 alle Optimierungen in einem Entwurf zusammengefasst.

6.1 Identifikation nützlicher Prozesse

Die Auswahl der weiterhin betrachteten Prozesse und die Begründung dafür wird im Folgenden diskutiert. Unter den recherchierten Prozessen gibt es zum einen Prozesse, die die Anforderungserhebung im Gesamten beschreiben und zum anderen Prozesse, die nur Teile der Anforderungserhebung beschreiben. Bei einer differenzierten Betrachtung beider Typen wird im Folgenden für erstere der Begriff **Gesamtprozesse** und für zweitere der Begriff **Teilprozesse** verwendet. Da eine Optimierung anhand eines Gesamtprozesses aufwendiger ist als die Optimierung anhand eines Teilprozesses, wird im Folgenden zwischen solchen Prozessen unterschieden.

6.1.1 Betrachtete Gesamtprozesse

HOOD Requirements Development Process Dieser Prozess basiert auf einem globalen Ansatz und betrachtet daher die Anforderungserhebung im Gesamten, das heißt beginnend bei der Anforderungsaufnahme bis hin zur Implementierung und den anschließenden Tests. Im Unternehmen ist in den letzten Jahren eine Tendenz zum teilweisen Einsatz iterativer Softwareentwicklung zu erkennen. Dies ist laut den „Life Cycle“ Dokumenten nicht vorgesehen, sodass davon auszugehen ist, dass die Mitarbeiter eine Anpassung hin zu iterativen Prozesselementen erwünschen. In den durchgeführten Interviews im Rahmen der Ist-Analyse wurde diese Annahme bestätigt.

Der HOOD Requirements Development Process beinhaltet einen Aktionswirbel zur Anforderungsdefinition auf jeder Informationsebene. In diesem Aktionswirbel werden iterativ mehrere Aktivitäten durchgeführt. Da eine iterative Vorgehensweise von den Mitarbeitern erwünscht ist, ist von einer hohen Akzeptanz bei Umsetzung dieses Prozesses auszugehen.

Trotz dieser Iterationen schreibt Hoods Prozess auch klar vor, wann die Anforderungen auf einer Informationsebene fertig gestellt werden. Dies ist ein weiterer Wunsch, den viele Mitarbeiter in den Interviews geäußert hatten. Nach der Erstellung der Anforderungen ist eine Änderung dieser nur auf Änderungsantrag und mit Durchführung einer Auswirkungsanalyse möglich.

Da dieser Prozess in die Unternehmenskultur passt und zusätzlich eine Verbesserung des Prozesses darstellen kann, wird er zur Optimierung in Unterkapitel 6.2 näher betrachtet.

Iterative Anforderungserhebung Die iterative Anforderungserhebung eignet sich besonders für Produkte, deren Anforderungen sich während der Entwicklung verändern können (vgl. Seite 16). Die Historie der Produktentwicklungen beim Unternehmen zeigt, dass sich Anforderungen immer während des Projektzeitaums entwickelt haben. Dies liegt unter anderem daran, dass es keine Auftragsprojekte gibt, in denen ein Kunde klar definierte Anforderungen verlangt. Oft ändern oder konkretisieren sich im Laufe der Zeit die Wünsche der Marketingabteilung. Daneben spricht die vorher erwähnte Eignung iterativer Prozesse für eine nähere Betrachtung bei der Optimierung in Unterkapitel 6.2.

Volere Requirements Process Der Volere Requirements Process wird trotz des geringeren Umfangs im Vergleich zum HOOD Requirements Development Process (vgl. Seite 13f.) in die Gruppe der Gesamtprozesse eingeordnet. Dies wird dadurch begründet, dass der Prozess alle Schritte detailliert beschreibt, die dafür benötigt werden, um eine Anforderungsspezifikation zu erstellen.

In den Mitarbeiterinterviews wurde festgestellt, dass Anforderungen oft deswegen nicht im Detail korrekt erstellt werden, weil die Wahrscheinlichkeit, dass sie sich später wieder ändern, sehr hoch ist. Mit der Implementierung eines Änderungsverwaltungsprozesses, der bisher im Unternehmen nicht stattfindet, könnte diese Wahrscheinlichkeit gesenkt werden. Gleichzeitig müsste sicher gestellt werden, dass die Anforderungen korrekt erhoben wurden. Mit dem Volere Requirements Process könnten die Prozesse des Unternehmen so verbessert werden, dass die SRS (die Anforderungsspezifikation) qualitativ besser wird. Aus diesen Gründen wird der Volere Requirements Process in Unterkapitel 6.2 näher betrachtet.

6.1.2 Betrachtete Teilprozesse

Change-Control Process Der Change-Control Process ist ein generischer Änderungsverwaltungsprozess, der in dieser Form im Unternehmen umgesetzt werden könnte. Dieser Prozess ist in der Gruppe der Teilprozesse aufgeführt, da er nur eine Teildisziplin der Anforderungserhebung ist.

Im Unternehmen ist die Änderungsverwaltung bisher nur bedingt integriert (vgl. Kapitel 4). Daher ist eine nähere Betrachtung dieses Prozesses zur Optimierung in Unterkapitel 6.2 zwingend notwendig.

Requirements Engineering Process nach Kotonya / Sommerville Kotonya und Sommerville beschreiben in ihrem Prozess zwar einen gesamten Prozess zur Anforderungerhebung. Allerdings sind hiervon nur Teilprozesse für eine nähere Betrachtung interessant, da der Gesamtprozess als einfache Ableitung vom Wasserfallmodell zu allgemein definiert ist. Auf der anderen Seite sind spezielle Teilprozesse sehr detailliert beschrieben und teilweise im Unternehmen einsetzbar.

Zwei dieser Prozesse sind die Änderungsverwaltung und der Prototypenbau[23]. Die Änderungsverwaltung ist, wie vorhergehend schon erwähnt, im Unternehmen bisher nur bedingt

[23]Prototypenbau ist die Vorgehensweise der Erstellung eines Prototyps um Anforderungen vom Nutzer besser aufnehmen zu können.

integriert. Beim Prototypenbau ist die Situation sehr ähnlich. Bisher wurde ein erstellter Prototyp meist auch zur Entwicklung weiter verwendet, was beim Prototypenbau eigentlich nicht vorgesehen ist[24] [Wie03, S. 236f.].

Im Folgenden werden daher Teile dieses Prozesses für eine Optimierung in Unterkapitel 6.2 näher betrachtet.

6.1.3 Nicht betrachtete Prozesse

Architecture Tradeoff Analysis Method (ATAM) Die Autoren des ATAM wurden kontaktiert, da die Verbreitung dieser Methode nicht bekannt ist. Das Software Engineering Institute der Carnegie Mellon University gab diesbezüglich an, dass eine weite Verbreitung noch nicht stattgefunden hat, obwohl es namhafte Erstanwender wie z.B. Bosch oder Boeing gibt.

ATAM prüft, wie in Unterkapitel 3.3.5 auf Seite 16 beschrieben, ob die Architektur den erhobenen Anforderungen entspricht. Diese Prüfung ist nicht direkter Bestandteil der Anforderungserhebung bzw. der Anforderungsprozesse und fällt damit nicht in den Umfang dieser Arbeit.

Aus den beiden vorhergehend genannten Gründen wird ATAM nicht weiter behandelt.

Rational Unified Process Wie in Unterkapitel 3.3.2 (auf Seite 10f.) angesprochen, macht die Implementierung des Rational Unified Process nur dann Sinn, wenn auch Software der Produktgruppe IBM Rational eingesetzt wird [HHMS04, S. 23]. Da sich das Unternehmen, wie in Kapitel 5.5 (auf Seite 38) empfohlen, für CaliberRM der Firma Borland entschieden hat, wird daher der Rational Unified Process nicht weiter betrachtet.

Stringente Anforderungserhebung Die stringente Anforderungserhebung eignet sich vor allem für Projekte, deren Produkte für einen längerfristigen Einsatz vorgesehen sind oder wenn Anforderungen bereits bekannt sind oder als stabil gelten (vgl. Seite 15f.).

Dies ist im Unternehmen nur bedingt der Fall, da die Produkte des Unternehmens zwar oft für einen längerfristigen Einsatz vorgesehen sind, allerdings regelmäßig neue Produktversionen erstellt werden. Die Anforderungen dieser Produkte gelten zusätzlich als volatil. Projekte, in denen anfangs klar definierte Anforderungen existieren und sich im Projektverlauf nicht oder kaum ändern, gibt es laut Aussage des Anforderungsanalysten im Unternehmen nicht. Aus diesen Gründen, fällt die stringente Anforderungserhebung nicht in eine nähere Betrachtung.

V-Modell Die Anforderungserhebung, die im V-Modell integriert ist, ist zu schwach ausgeprägt, um eine Verbesserung für den Prozess des Unternehmens darzustellen (vgl. Seite 9f.). Des weiteren existieren auch keine Teilprozesse, die für eine weitere Betrachtung in Frage kommen könnten.

[24] Eine Ausnahme bildet hier das Vorgehensmodell des evolutionären Prototypenbaus, das allerdings im Unternehmen nicht durchgeführt wurde.

6.2 Vergleich des aktuellen Prozesses mit empfohlenen Prozessen

6.2.1 Gesamtprozesse

HOOD Requirements Development Process Der aktuelle Prozess im Unternehmen hat viele Ähnlichkeiten mit dem HOOD Requirements Development Process. Das zu Grunde liegende Informationsmodell (siehe Abbildung 4 auf Seite 11) liegt auch implizit in den aktuellen Anforderungsprozessen des Unternehmens vor (vergleiche mit Appendix A).

Der Übergang von den Kundenanforderungen zu den Systemanforderungen entspricht dem Proposal-to-Investigation Checkpoint. Zu diesem Zeitpunkt hat der Kunde, repräsentiert durch den Produktmanager, seinen noch sehr abstrakten Auftrag in Koordination mit dem Auftragnehmer, in diesem Fall die F&E-Abteilung, bestimmt.

Der Übergang von den Systemanforderungen zu den Systemdesignanforderungen entspricht dem Definition Checkpoint. An diesem Punkt ist eine SRS, also ein Lastenheft, erstellt, das die konkreten Systemanforderungen beinhaltet. Diese wurden größtenteils vom Anforderungsanalysten erstellt, der den Anweisungen des Produktmanager folgt aber gleichzeitig in Kontakt mit der F&E-Abteilung ist.

Der Übergang von den Systemdesignanforderungen zu den Subsystemanforderungen entspricht dem Investigation-to-Definition Checkpoint. Eine ERS, entsprechend dem Pflichtenheft, wird an diesem Kontrollpunkt erstellt. Dies ist eine Verpflichtung der F&E-Abteilung gegenüber dem Produktmanager, welche Produktbestandteile implementiert werden. Kurz vor diesem Kontrollpunkt werden noch einige Aktivitäten durchgeführt, die die Bezeichnung „Non-RM-Tool Activity“ tragen. Dabei handelt es sich größtenteils um Projektplanungsaktivitäten, die für die Anforderungserhebung irrelevant sind und daher keine Auswirkung auf den Vergleich mit dem Informationsmodell haben.

Der Übergang von den Subsystemanforderungen zu den Designanforderungen und weiter zur Implementierung verschwimmt in den Prozessen des Unternehmens, da sie hierfür nicht verschiedene Kontrollpunkte besitzen. An dieser Stelle muss differenziert werden, was erzwungene Kontrollpunkte sind und was tatsächlich von den Mitarbeitern ausgeführt wird. Der Lifecycle gibt strikt vor, welche Kontrollpunkte existieren. Diese Kontrollpunkte werden auch genau so eingehalten, da die Qualitätsicherung an jedem Kontrollpunkt Dokumente formell überprüft. Da zur Erreichung einer Implementierung die zuletzt erwähnten Übergänge (inklusive der beinhalteten Aktivitäten) des Informationsmodells notwendig sind, werden sie trotz des Nichtvorhandenseins von Kontrollpunkten im Lifecyle von der F&E-Abteilung implizit durchgeführt.

Die Aktivität „Create internal design“ zielt darauf ab, die Subsystemanforderungen zu definieren. Diese werden dann wiederum von den Aktivitäten „Define implementation milestones“ und „Create detailled design for current milestone“ entsprechend den Designanforderungen des Informationsmodells weiter verfeinert. Letztendlich erfolgt der Übergang zur Implementierung, wenn die Aktivität „Implement / fix units for the current milestone“ startet.

Die letzten drei Ebenen des Informationsmodells sind im Unternehmen nicht so ausgeprägt wie die ersten drei Ebenen. Dies kann aber auch darauf zurückgeführt werden, dass in den letzten drei Ebenen die meiste Arbeit nur in der F&E-Abteilung intern stattfindet und kaum externe Kommunikation notwendig ist. Eine genaue Regulierung anhand von Kontrollpunk-

ten würde damit eventuell die Flexibilität der Abteilung einschränken. Ob dies tatsächlich der Grund ist, warum das Unternehmen keine weiteren Kontrollpunkte eingeführt hat und die Kontrolle dem Projektmanager überlässt, wird in dieser Arbeit nicht untersucht. Dies sind Aktivitäten, die die Anforderungserhebung nicht betreffen und daher in Appendix A auch dementsprechend markiert sind. Es ist ersichtlich, dass die Basis der aktuellen Prozesse des Unternehmens und die Basis des HOOD Requirements Development nahezu identisch sind. Damit ist sichergestellt, dass keine grundlegenden Änderungen an den Prozessen des Unternehmens zur Anpassung an den HOOD Requirements Development Process notwendig sind. Gleichzeitig ist durch die Übereinstimmung der Kontrollpunkte mit den Ebenen des Informationsmodells ein einfacher Vergleich der verschiedenen Aktivitäten möglich.

Vor dem Proposal-to-Investigation Checkpoint werden die Kundenanforderungen definiert (s.o.). Diese sollten auf einer abstrakteren Ebene formuliert sein als die Systemanforderungen. Im Unternehmen werden auf dieser Stufe Anforderungen erstellt, die später mit den Systemanforderungen verschmelzen und sich nur dadurch unterscheiden, dass es die ersten (unvollständigen) Pflichtanforderungen[25] sind. Da die Kundenanforderungen in der Kundensprache formuliert werden sollten [HW05, S. 117], bieten sich hier eher Anwendungsfälle oder Szenarien zur Darstellung der Kundenanforderungen an. In dieser Darstellungart beschreibt der Kunde in Geschichten, welche Arbeitsschritte das Produkt können soll. Ausgehend davon kann der Anforderungsanalyst danach die Systemanforderungen definieren.

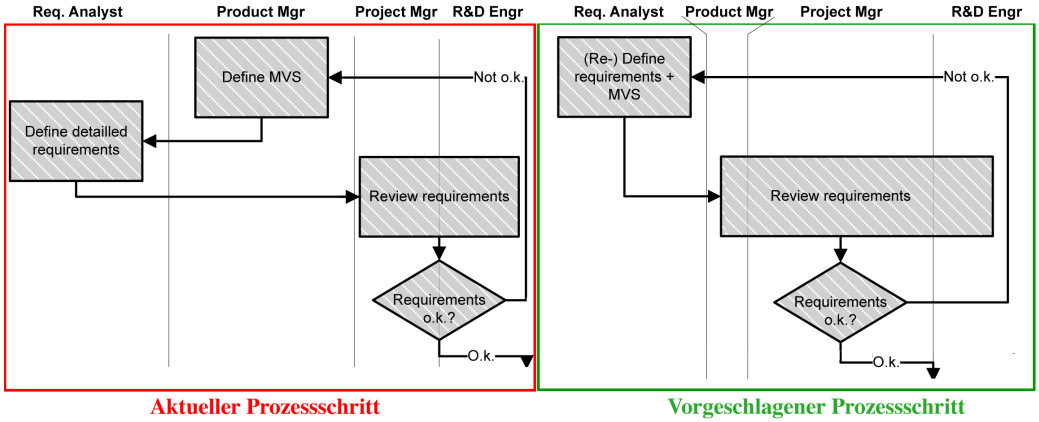


Abbildung 17: Der Anforderungsanalyst erstellt zukünftig die gesamten Systemanforderungen

Da die Kundenanforderungen mit Erstellung der Anwendungsfälle und Szenarien abgeschlossen sind, sollte daraufhin der Anforderungsanalyst die Systemanforderungen definieren. Bisher erstellt der Produktmanager (Kunde) zuerst die Pflichtanforderungen. Da der Anforderungsanalyst allerdings die Systemanforderungen definiert und er im Normalfall zur Erstellung von Anforderungen ausgebildet ist (z.B. Beachtung der Qualitätskriterien von Anforderungen [HW05, S. 94], soll er die gesamten Anforderungen erstellen (siehe Abbildung 17[26]). Der

[25] Im Appendix sind Pflichtanforderungen als „MVS“ dargestellt.
[26] Die Spaltenbreiten der Diagrammausschnitte wurden wegen Platzgründen verändert.

Produktmanager definiert danach mit einem Attribut, welche Anforderungen Pflichtanforderungen sind. Somit ist er nicht an der verbalen Gestaltung der Anforderungen beteiligt. Durch diese Änderung soll eine höhere Qualität der Systemanforderungen erreicht werden, sodass weniger nachträgliche Änderungen notwendig sind, wenn beispielsweise die F&E-Abteilung die Systemanforderungen nicht versteht.

Der Aktionswirbel zur Definition der Anforderungen (siehe Abbildung 6 auf Seite 12) wird meistens im Unternehmen in leichter Abwandlung durchgeführt.

Die Aktivitäten „Ermitteln“ und „Spezifizieren“ werden im Unternehmen durch die Aktivität des „Erstellens“ durchgeführt. Die Aktivität „Ermitteln“ wird nicht explizit erwähnt, da zur Ermittlung die Anforderungen der abstrakteren Ebene betrachtet werden müssen. Hood beschreibt die Aktivität „Ermitteln“ so, dass mit der „Sammlung von vagen Anforderungen“ begonnen werden soll [HW05, S. 75]. Dies ist nur bei der erstmaligen Ermittlung von Kundenanforderungen erforderlich, die der Produktmanager beim Erstellen des Produktentwurfs durchführt. Bei Ableitung der Anforderung auf eine speziellere Ebene, liegen diese vagen Anforderungen auf der abstrakteren Ebene bereits vor.

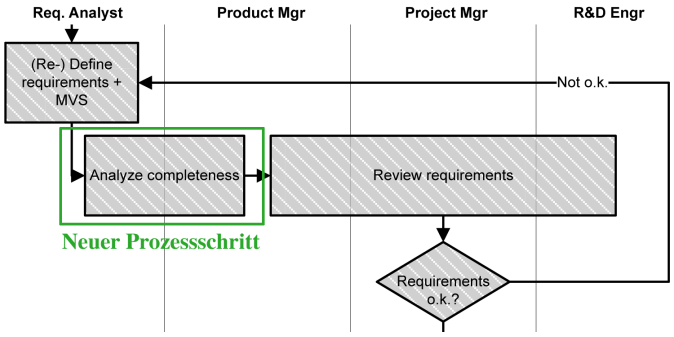


Abbildung 18: Vor jeder Durchsicht wird eine Analyse hinsichtlich der Vollständigkeit der Anforderungen durchgeführt

Die Aktivität „Analysieren“ wird im Unternehmen laut einzelner Aussagen immer als Vorbereitung auf die Durchsicht durchgeführt, die explizit stattfindet. Wie weit diese Analyse allerdings geht, ist nicht geklärt. Laut Hood soll mit dieser Aktivität die „Vollständigkeit der Anforderungen“ überprüft werden [HW05, S. 75]. Die hohe Anzahl an durchgeführten Durchsichten auf einer Informationsebene im Unternehmen könnte einen Hinweis darauf geben, dass die Analyse vor den Durchsichten nicht ausführlich genug durchgeführt wird. Teilweise finden Analysen laut Aussage mancher Mitarbeiter sogar erst während der Durchsichten statt. Um dieser offensichtlichen Ineffizienz entgegenzutreten, soll eine explizite Aktivität „Analyze completeness“ vor jeder Durchsicht durchgeführt werden (siehe Abbildung 18). Diese Aktivität wird an insgesamt drei Stellen in den Anforderungsprozessen eingefügt; jeweils vor den Durchsichten der Kundenanforderungen, der Systemanforderungen und der Systemdesignanforderungen.

Insgesamt werden alle Aktivitäten so oft wiederholt, bis in der Durchsicht die Anforderungen akzeptiert und weitergegeben werden. Damit ist die Iterativität des Aktionswirbels als weiteres wichtiges Element des HOOD Requirements Development Processes gegeben.

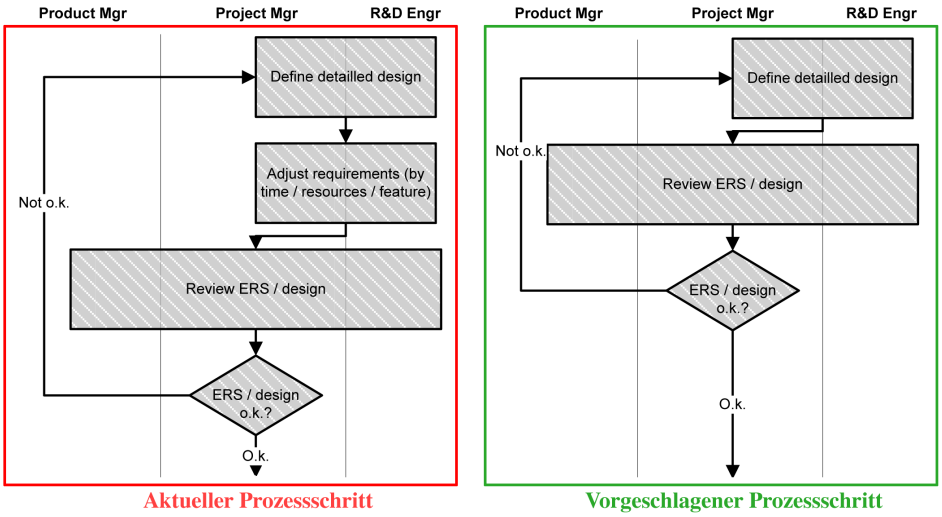


Abbildung 19: Das regelmäßige Ändern der Systemanforderungen wird nicht empfohlen. Dies soll zukünftig über einen Änderungsantrag geschehen.

Nach dem Definition Checkpoint werden die Systemdesignanforderungen erstellt. Hierbei werden üblicherweise bei der Definition der Systemdesignanforderungen nachträgliche Änderungen an den Systemanforderungen durchgeführt, die eine Abstraktionsebene höher liegen. Alle Änderungen an vorher akzeptierten Anforderungen sollen zukünfig über einen Anforderungsänderungsantrag erfolgen, der in Unterkapitel 6.2.2 eingeführt wird. Die Aktivität „Adjust requirements“ wird daher eliminiert (siehe Abbildung 19).

In der Testphase gibt es momentan einen „System test“ und einen „Release candidate test“. Die Testfälle des letzteren Tests werden willkürlich, basierend auf subjektiven Erfahrungswerten, aus den Testfällen des ersten Tests ausgewählt. Da dieses Vorgehen von der Erfahrung und dem Wissen des Testmanagers abhängt, kann es in manchen Projekten zu mangelhaften Tests führen. Fehler könnten dadurch erst beim tatsächlichen Produktabnehmer auftreten. Dazu kommt, dass im Unternehmen ein Testen nach den Arbeitsabläufen des potentiellen Abnehmers selten stattfindet.

Da vorhergehend bereits die Erstellung von Anwendungsfällen und Szenarien als neue Aktivität eingeführt wurde, können auf dieser Basis Testfälle einer neuen Testart erstellt werden. Nachdem die Anwendungsfälle und Szenarien akzeptiert wurden, sollen parallel zur Definition der Systemanforderungen Testfälle auf Basis dieser Kundenanforderungen definiert werden (siehe Abbildung 20). Sie sollen später in der Testphase anstelle des bisher willkürlich erstellten „Release candidate test“ durchgeführt werden. Vorteil dieser Art von Tests ist, dass hierbei aus der Kundensicht die Software getestet wird. Andere Tests orientieren sich ausschließlich an der Implementierung.

Iterative Anforderungserhebung Die Literatur zur iterativen Anforderungserhebung (z.B. [Ebe08]) bietet nur wenige Informationen und erklärt lediglich das dahinter liegende Konzept.

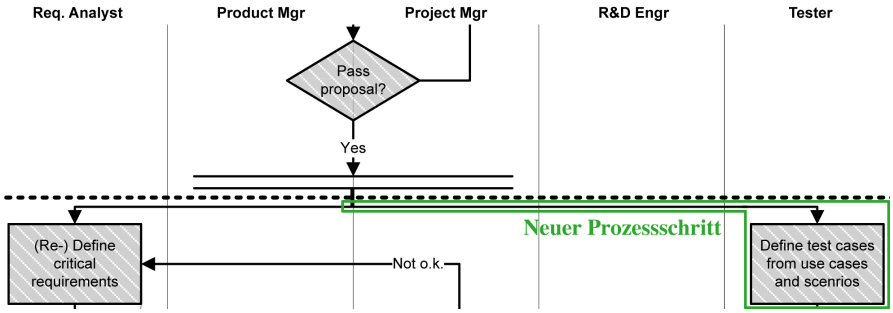


Abbildung 20: Der neue Prozessschritt ermöglicht ein weiteres kundenorientiertes Testverfahren

Eine genaue Beschreibung der durchzuführenden Aktivitäten existiert nicht. Trotzdem lässt dich dieses Konzept gleichzeitig mit dem HOOD Requirements Development Process auf die Anforderungsprozesse des Unternehmens übertragen.

Grob lässt sich die iterative Anforderungserhebung so beschreiben, dass zunächst im Projekt alle kritischen Anforderungen erhoben werden. Dies sind Anforderungen, die einen Einfluss auf die Architektur, Schnittstellen oder die Projektdefinition haben. Danach wird geplant, welche und wieviele Inkremente[27] bis zur Vollendung des Projekts optimal sind. Danach werden die Inkremente kontinuierlich und überlappend entwickelt. Jedes Inkrement baut dabei auf die vorherigen Inkremente auf und erweitert diese [Ebe08, S. 73].

Dieses Vorgehen findet im Unternehmen auf Implementierungsebene bereits teilweise bei der Erhebung der Systemdesignanforderungen statt (vgl. Seite 25). Eine korrekte Durchführung der iterativen Anforderungserhebung müsste bei der Erhebung der Systemanforderungen ansetzen.

Mitarbeitern würden nach Rücksprache ein solches optionales Vorgehen begrüßen. Ein weiteres Argument für eine Anpassung an eine iterative Anforderungserhebung liegt in der Nutzung eines Anforderungsverwaltungssystems. Mit Hilfe eines solchen Systems ist die Handhabung von Inkrementen vereinfacht, da alle Anforderungen an einer Stelle gespeichert sind. Die Problematik, die bei Anforderungen von Inkrementen entsteht, wenn sie in verschiedenen Dokumenten niedergeschrieben werden, ist somit umgangen. Eine Ansicht der einzelnen Anforderungen eines Inkrements wäre mit wenigen Schritten möglich. Ein zusätzliches Attribut würde die Zugehörigkeit einer Anforderung zu einem Inkrement bestimmen. Die Filterung eines bestimmten Inkrements würde dann die Anforderungen eines Inkrements ausgeben.

Im angepassten Prozess des Unternehmens wird die Bildung der Inkremente mit der Erhebung der Systemanforderungen, also direkt nach dem Proposal-to-Investigation Checkpoint, stattfinden. Zunächst werden die kritischen Anforderungen mit dem Aktionswirbel des HOOD Requirements Definition Process erhoben. Dies beinhaltet die Aktivitäten „Definition“ (1), „Analyse auf Vollständigkeit“ (2) und „Durchsicht“ (3) (siehe Abbildung 21).

[27] Inkremente sind nicht zu verwechseln mit Iterationen, was man fälschlicherweise durch die Namensgebung dieser Art der Anforderungserhebung denken könnte. Eine Bildung von Iterationen sind trennbare Teilprojekte. Inkremente hingegen basieren aufeinander und können ohne das vorherige Inkrement nicht weiterentwickelt werden [Ebe08, S. 73].

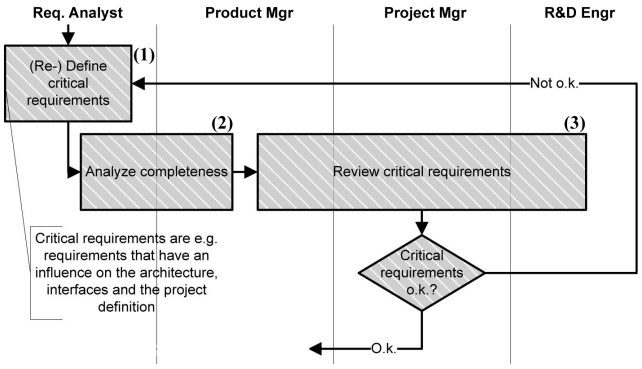


Abbildung 21: Die neuen Prozessschritte zur Erhebung der kritischen Anforderungen

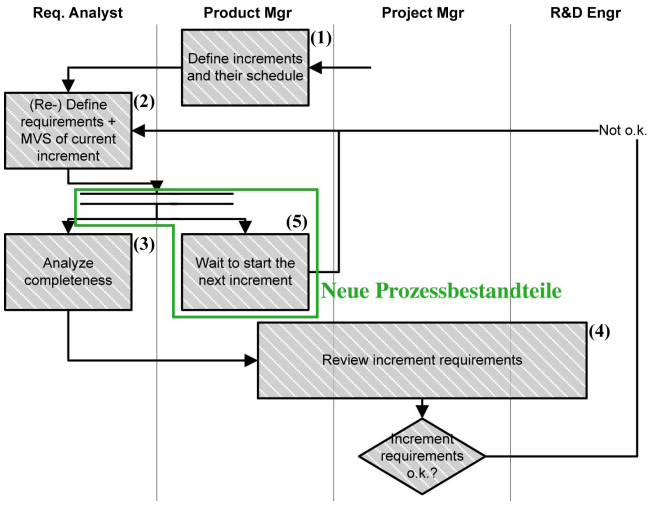


Abbildung 22: Die neuen Prozessbestandteile zur Erstellung von Inkrementen und deren Anforderungserhebung

Wenn nach mehreren Iterationen die kritischen Anforderungen erhoben wurden, werden die Inkremente vom Produktmanager definiert und zeitlich geplant (1). Danach erfolgt die übliche Erhebung der Systemanforderungen (2) bis (4). In diesem Fall werden allerdings nur die Anforderungen des aktuellen Inkrements erhoben. Zur Erhebung dieser Anforderungen wird wieder der Aktionswirbel des HOOD Requirements Definition Process angewendet. Parallel kann nach der zuvor bestimmten Wartezeit das nächste Inkrement gestartet werden (5) (siehe Abbildung 22). Die Arbeitsabläufe der einzelnen Inkremente bauen aufeinander auf und fließen später nach der Implementierung und vor der Übergabe an die Testabteilung wieder zusammen.

Volere Requirements Process Der Volere Requirements Process kann mit der iterativen Anforderungserhebung und dem HOOD Requirements Development Process kombiniert

werden. Volere gibt nicht vor, wie agil die Anforderungserhebung durchgeführt werden soll. Grundsätzlich gibt es drei Arten der Agilität:

1. Bei so genannten „Hasenprojekten“ werden zuerst Anforderungen eines einzelnen Anwendungsfalls erhoben und implementiert. Danach wird Feedback vom Kunden eingeholt, um darauf mit den Anforderungen folgender Anwendungsfälle eingehen zu können [RR06, S. 19].

2. Bei „Pferdeprojekten“ handelt es sich um einen Mittelweg der Agilität. Hierbei gibt es ein ähnliches Vorgehen wie bei Hasenprojekten, allerdings wird dabei kein Feedback des Kunden für die weiteren Anforderungen eingeholt [RR06, S. 20].

3. „Elefantenprojekte“ repräsentieren die am wenigsten agilen Projekte. Hierbei wird zunächst eine endgültige Anforderungsspezifikation erhoben, bevor mit der Implementierung begonnen wird. Die Anforderungsspezifikation kann trotzdem iterativ oder inkrementell erstellt werden. Dieses Vorgehen wird oft für pharmazeutische oder Luftfahrtprojekte eingesetzt [RR06, S. 20].

Durch die Offenheit gegenüber der Ausprägung der Agilität der Anforderungsprozesse ist die Kombinationsfähigkeit mit der iterativen Anforderungserhebung begründet. Die Begründung für eine Kombination mit dem HOOD Requirements Development Process liegt darin, dass Volere nur einen Teil von Hoods vorgeschlagenen Prozess spezifiziert. Hood betrachtet den Prozess gesamtheitlich und bezieht nicht nur die Prozesse bis zum ersten Design mit ein. Aus diesen Gründen wird im Folgenden auf den Optimierungsvorschlägen der vorigen beiden Paragraphen aufgebaut.

Insgesamt gibt es auch beim Volere Requirements Process viele Ähnlichkeiten zu den aktuellen bzw. angepassten Anforderungsprozessen des Unternehmens.

Zunächst soll in einem „Project Blastoff“ die Machbarkeit des Projekts geprüft werden [RR06, S. 22f.]. Dies ist auch Ziel der Aktivitäten vor dem Proposal-to-Investigation Checkpoint. Hierzu soll zunächst der Umfang des Projekts bestimmt werden, das auch eine Eingangsgröße für die Anforderungsdefinition nach Hood ist. Daneben sollen die Projektziele bestimmt werden und darauf basierend eine Start- oder Abbruchentscheidung getroffen werden, die wiederum der Aktivität entspricht, in der über die Annahme des Entwurfs entschieden wird.

Im darauf folgenden Prozessschritt unterscheidet Volere zwischen dem Erheben[28] und dem Niederschreiben der Anforderungen. Diese Unterscheidung findet in den Aktivitäten des Unternehmens bisher kaum statt, sondern wird implizit in der Aktivität zur Definition der Anforderungen angenommen. Zur Erhebung soll der Anforderungsanalyst aus den vorliegenden Materialien Anwendungsfälle ableiten und daraus wiederum mit verschiedenen Techniken wie Szenarien, Workshops etc. die Anforderungen ableiten. Einige Techniken davon eignen sich nur, wenn auch der letztendliche Benutzer des Systems verfügbar ist. Da das Unternehmen Produkte für den anonymen Markt erstellt, ist ein Kunde in den meisten Fällen nicht verfügbar. Eine bisher noch nicht betrachtete Technik, die den Kunden als Ressource nicht unbedingt benötigt, ist der Prototypenbau.

[28] Erheben ist eine sinngemäße Überstzung aus dem Original auf englisch: „Trawling for Requirements“ [RR06, S. 24].

Prototypenbau eignet sich vor allem dann, wenn der Anforderungsanalyst die Ableitung der Anforderungen nicht versteht, wenn Anwendungsfälle komplex sind oder wenn ein Produkt solch eine Neuheit ist, dass man sich die Anforderungen gedanklich nicht vorstellen kann [RR06, S. 25]. Es existieren verschiedene Arten von Prototypen in der Softwareentwicklung. Ein Prototyp, bei dem die Benutzeroberfläche auf Papier aufgezeichnet wird, verbessert die Vorstellung, wie die Software benutzt wird. Ein elektronischer Prototyp mit einer funktionsgetreuen Logik ermöglicht die experimentelle Nutzung der Software.

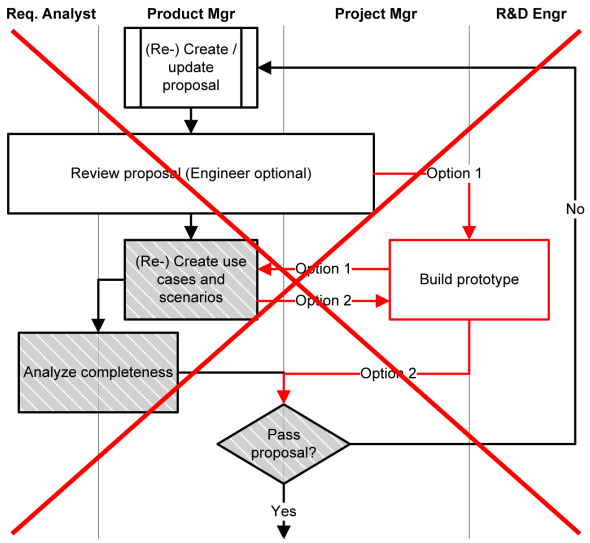


Abbildung 23: Der Prototypenbau ist zu diesem frühen Zeitpunkt der Anforderungsprozesse aufgrund fehlender Informationen nicht möglich

Vor der Definition der Anwendungsfälle (Option 1) in der Phase des Entwurfs kann noch kein Prototypenbau durchgeführt werden, da die verfügbaren Informationen zu gering sind. Danach (Option 2) fehlen noch die kritischen Rahmenbedingungen um einen Prototyp zur Analyse der Vollständigkeit zu erstellen (siehe Abbildung 23). Somit wäre der erste Einsatzpunkt für die Erstellung eines Prototypen nach der Definition der kritischen Anforderungen. Direkt nach der Aktivität der Definition der Inkremente (1) könnte optional, je nach Komplexität, für je ein Inkrement ein einfacher Prototyp von der F&E-Abteilung erstellt werden (2). Der Produktmanager würde daraufhin diesen Prototyp untersuchen (3), um danach dem Anforderungsanalysten Informationen bereitzustellen, der darauf basierend die Anforderungen niederschreibt (4) (siehe Abbildung 24).

Es ist einer der meisten Trugschlüsse, dass man alle Anforderungen sammeln müsse, bevor mit der Implementierung begonnen werden könne [RR06, S. 30]. Wenn zunächst nur die wichtigsten Bestandteile einer Software implementiert werden, können diejenigen, die die Software beauftragt haben, sehen, ob sie mit ihren Wünschen richtig liegen. Eine Planänderung zu diesem Zeitpunkt ist billiger, da der Aufwand für die Erstellung einer gesamten Anforderungsspezifikation und teilweisen Implementierung nicht entstanden ist. Daher schlägt Volere vor, für jeden Anwendungsfall ein Inkrement zu starten und damit auch je nach Bedarf

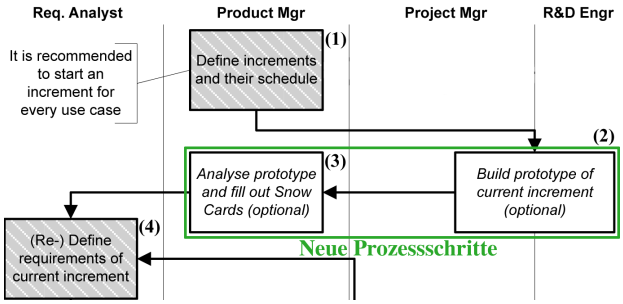


Abbildung 24: Der neue optionale Prozessschritt zur Erstellung eines Prototyps, mit dessen Hilfe die Anforderungen aufgenommen werden können

einen Prototypen zu erstellen. Dies könnte man als eine Möglichkeit für die Definition der Inkremente vorschlagen, wofür der Produktmanager zuständig wäre.

Zur Unterstützung des Niederschreibens der Anforderungen, das der Aktivität der Definition der Anforderungen im Unternehmen gleichkommt, bietet Volere als Werkzeug die Snow Card an. Die Snow Card (siehe Abbildung 25) kann während der Untersuchung des Prototyps verwendet werden, damit nicht nur der Anforderungsanalyst, sondern auch andere, nicht zur Anforderungsspezifikation ausgebildete, Projektinteressierte Anforderungen festhalten können. Daraufhin kann der Anforderungsanalyst die Informationen der Snow Card genau definieren und in das Anforderungsverwaltungssystem eingeben.

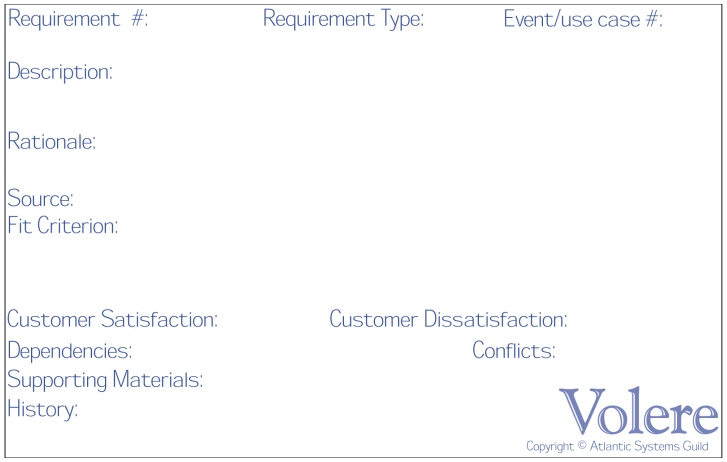


Abbildung 25: Die Snow Card als Werkzeug zur Anforderungsaufnahme

Als weiteres Werkzeug zum Niederschreiben der Anforderungen bietet Volere das Requirements Specification Template an. Dabei handelt es sich um eine Vorlage, die den Anforderungsanalysten dabei unterstützt, eine lückenlose Anforderungsspezifikation zu erstellen [RR06, S. 451ff.]. Die Vorlage besteht aus allen vorkommenden Anforderungstypen inklusive ihrer Inhalte und Beschreibungen, die sich in folgende Gruppen gliedern:

1. Projekttreiber

2. Rahmenbedingungen des Projekts
3. Funktionale Anforderungen
4. Nicht-funktionale Anforderungen
5. Projektprobleme

Da im Unternehmen die Anforderungen zukünftig im Anforderungsverwaltungssystem CaliberRM erfasst werden sollen, scheidet eine manuelle Erstellung einer Anforderungsspezifikation aus. Anforderungen werden dann ausschließlich im System erfasst, das automatisch eine bestimmte Auswahl von Anforderungen als Dokument ausgeben kann.

Die Vorlage kann aber auch auf andere Weise verwendet werden. Da die Anforderungstypen und ihre Attribute im Zuge der Neueinführung des Systems neu bestimmt werden sollen, bietet es sich dabei an, auf die lückenlose Aufstellung aller Anforderungstypen zurückzugreifen. Die Auflistung der neu definierten Anforderungstypen und ihrer Attribute mit Erklärungen sind im Appendix C angehängt. Die Anforderungstypen wurden nach Einreichung eines Vorschlags gemeinsam mit Mitarbeitern des Unternehmens angepasst. Folgende Anforderungstypen wurden für das Unternehmen identifiziert:

1. Rahmenbedingungen des Produkts
2. Anwendungsfälle und Szenarien
3. Funktionale Anforderungen
4. Nicht-funktionale Anforderungen
5. Designanforderungen
6. Testfälle
7. Ideen und Verbesserungen

Die jeweiligen Attribute wurden aus den bestehenden Anforderungstypen des Unternehmens, den Standardeinstellungen in CaliberRM, der Vorlage von Volere und der Snow Card abgeleitet. Dabei wurde auch festgehalten, welche Attribute von CaliberRM automatisch ausgefüllt werden und welche Attribute alle Anforderungstypen gemein haben.

Als nächsten Prozessschritt sieht der Volere Requirements Process eine Qualitätsprüfung vor, in der die Anforderungen auf verschiedene Qualitätskriterien hin überprüft werden. Dazu zählen zum Beispiel die Vollständigkeit, die Relevanz, die Testbarkeit, der Zusammenhang, die Rückverfolgbarkeit und andere Kriterien (vgl. Unterkapitel 2.2). Für die Durchführung dieser Qualitätsprüfung soll die Testabteilung eingesetzt werden [RR06, S. 28]. Die Übernahme dieses Prozessschritts in die angepassten Anforderungsprozesse des Unternehmens ist sinnvoll, da laut Aussagen der Mitarbeiter die Anforderungsspezifikationen des Unternehmens teilweise qualitativ nicht hochwertig genug seien. Da der Anforderungsanalyst bereits die Vollständigkeit der Anforderungen eines Inkrements überprüft (1), soll ein Tester parallel dazu die weiteren Qualitätskriterien überprüfen (2). Die Ergebnisse dieser Qualitätsprüfung fließen dann in die Durchsicht (3) ein, in dem darüber entschieden wird, ob an bestimmten Stellen eine Nachbesserung stattfinden soll (siehe Abbildung 26).

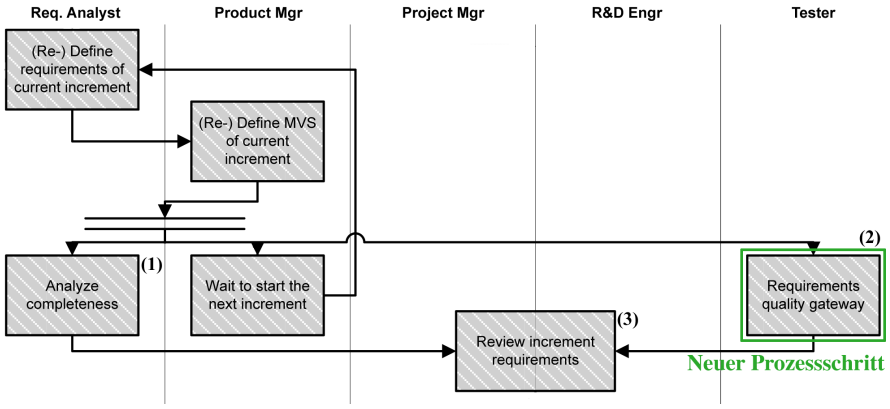


Abbildung 26: Ein neuer Prozessschritt zur Erhöhung der Anforderungsqualität

6.2.2 Teilprozesse

Change-Control Process Das Änderungsverwaltungssystem CaliberRM bietet in der Standardausführung keine Möglichkeit, Änderungsanträge zu bearbeiten. Im Unternehmen existiert darüber hinaus ein anderes System, mit dem Änderungen gelegentlich verwaltet werden. Eine Einarbeitung in dieses System würde allerdings den Aufwand für eine beiläufige Behandlung des Themas Änderungsprozess übersteigen. Daher wird im Folgenden der Change-Control Process beschrieben und Ansätze aufgezeigt, wie dieser Prozess im Unternehmen umgesetzt werden könnte. Eine genaue Anpassung an die Bedürfnisse des Unternehmens und damit auch eine Einarbeitung dieses Teilprozesses in die Anforderungsprozesse kann aus vorhergehend genanntem Grund nicht durchgeführt werden.

Wiegers beschreibt den Change-Control Process sehr detailliert und geht dabei auf die Punkte „Änderungsstatus“, „Rollen“ und „Richtlinien“ ein [Wie03, S. 331ff.]. Der Prozess kann vom generischen Zustandsübergangsdiagramm (siehe Abbildung 27) abgeleitet werden.

Zunächst erstellt ein Antragsteller einen Änderungsantrag, der dadurch den Status „übermittelt“ (1) hat. Danach muss ein Begutachter den Aufwand zur Umsetzung der Änderung mit einer Auswirkungsanalyse abschätzen (2). Basierend darauf entscheidet ein Komitee darüber, ob der Änderungsantrag abgelehnt (3) oder angenommen (4) wird. Sofern er angenommen wird, muss das Komitee der durchzuführenden Änderungen eine Priorität oder eine Frist zuordnen. Außerdem bestimmt das Komitee je eine Person zur Durchführung der Änderung und zur Verifizierung der Ergebnisse.

Danach wird die Änderung durchgeführt. Nach der Durchführung (5) wird die Änderung verifiziert und entweder zur erneuten Überarbeitung zurückgegeben (4) oder als verifiziert (6) gekennzeichnet. Sofern sie verifiziert wurde, kann die Änderung in das Gesamtprodukt eingesetzt werden. Damit wird der Änderungsantrag geschlossen (7). Vor jedem dieser Schritte kann der Änderungsantrag storniert (8) werden.

Die Tätigkeitsbereiche mancher Rollen wurden bereits teilweise angesprochen. In Tabelle 5 sind die Rollenbezeichnungen, ihre Beschreibung und die dazu passenden Rollen des

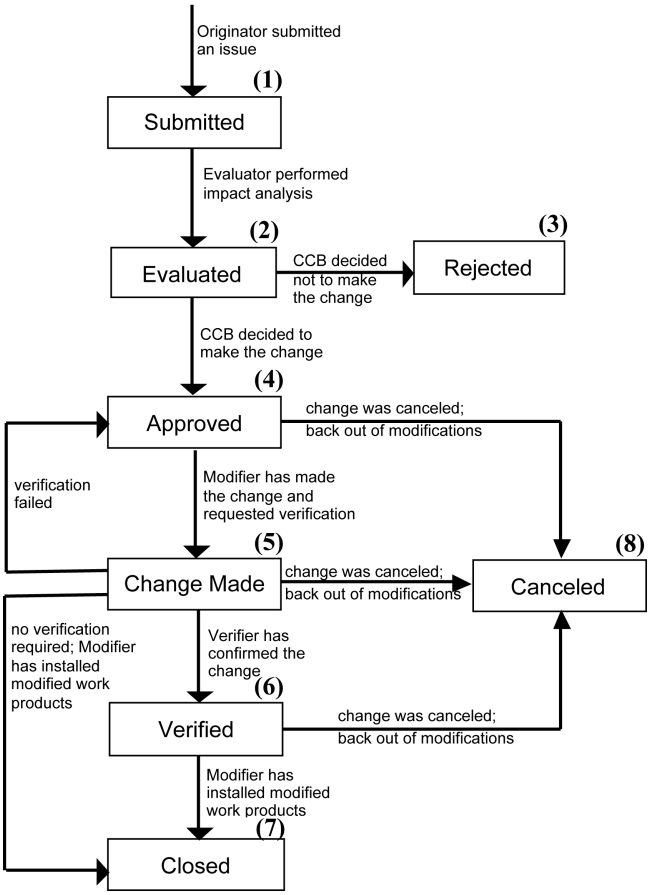


Abbildung 27: Das Zustandsübergangsdiagramm des Change-Control Process nach Wiegers [Wie03, S. 335]

Unternehmen aufgeführt.

Hierbei sei angemerkt, dass die Rolle des Antragempfängers je nach Werkzeugunterstützung wegfallen könnte. Im optimalen Fall sollte ein System den Änderungsprozess abbilden können und automatisch den Änderungsantrag je nach Status an die richtigen Rollen weiterleiten. Außerdem sollte die doppelte Besetzung des F&E-Ingenieurs bei Einführung des Änderungsprozesses kritisch beobachtet werden. Hier könnten unter Umständen Interessenskonflikte entstehen, da die Person eine realistische Auswirkungsanalyse durchführen soll, die die Entscheidung über den Änderungsantrag beeinflusst, gleichzeitig aber auch für die Umsetzung dieser Änderung verantwortlich wäre. Sollte dieser Interessenskonflikt eine Auswirkung auf die Ergebnisqualität der Auswirkungsanalyse haben, könnte der Anforderungsanalyst diese Aufgabe übernehmen.

Zur erfolgreichen Umsetzung dieses Änderungsprozesses ist laut Wiegers eine klare Kommunikation von Richtlinien unerlässlich [Wie03, S. 332]. Für das Unternehmen relevante Richtlinien sind die folgenden:

Tabelle 5: Die Rollen im Change-Control Process mit Empfehlungen für das Unternehmen [Wie03, S. 335]

Rolle	**Beschreibung**	**Rolle im Unternehmen**
Antragsteller	Person, die den Änderungsantrag erstellt.	Jeder Involvierte
Antragempfänger	Person, die die neuen Änderungsanträge entgegennimmt und weitergibt.	Anforderungsanalyst oder Produktmanager
Gutachter	Person, die die Auswirkung einer Änderung untersucht.	F&E-Ingenieur
Komitee	Spezifische Gruppe für ein Projekt, die entscheidet ob ein Änderungsantrag angenommen oder abgelehnt wird.	Produktmanager und F&E-Projektmanager
Komiteevorsitz	Person, die die finale Entscheidungsbefugnis hat, falls das Komitee sich nicht einigt. Er wählt den Gutachter und den Modifizierer aus.	Produktmanager
Modifizierer	Person, die die Änderung eines angenommenen Antrags durchführt.	F&E-Ingenieur
Verifizierer	Person, die die Änderung verifiziert.	Tester

1. Alle Änderungsanträge, die mehr als eine Person betreffen, müssen dem vorhergehend beschriebenen Prozess folgen.
2. Bevor ein Änderungsantrag nicht angenommen wurde, dürfen bis auf die Auswirkungsanalyse keine Entwurfs- oder Implementierungsaufgaben angefangen werden.
3. Alle Projektinvolvierten müssen die Änderungen einsehen können.
4. Bei jedem Änderungsantrag muss eine Auswirkungsanalyse durchgeführt werden.
5. Zwischen einem Änderungsantrag und der geänderten Anforderung muss eine Verbindung zur Rückverfolgbarkeit erstellt werden.

Trotz dieser strikten Richtlinien soll auch eine Ausnahme möglich sein: Zur Beschleunigung von risiko- und kostenarmen Änderungen sollen diese nur einen komprimierten Entscheidungszyklus durchlaufen.

Der Change-Control Process könnte in der vorhergehend beschriebenen Form im Unternehmen mit leichter Anpassung und Werkzeugunterstützung umgesetzt werden. Jede Änderung

an Anforderungen nach den jeweiligen Kontrollpunkten müsste dann diesen Prozess durchlaufen. Bei dem bereits erwähnten Pilotprojekt zur Einführung von CaliberRM wird beobachtet werden, ob bereits zu diesem Zeitpunkt ein Änderungsprozess neben den sich ändernden Anforderungsprozessen einführbar ist oder ob dies erst in einem folgenden Projekt geschehen kann.

Requirements Engineering Process nach Kotonya / Sommerville In diesem Abschnitt wird der Änderungsprozess des vorherigen Paragraphen gegebenenfalls um die Ausführungen von Kotonya / Sommerville erweitert und der Prototypenbau als Teilprozess der Anforderungsprozesse detailliert behandelt.

Abbildung 28: Die drei Stufen des Änderungsverwaltungsprozesses [KS00, S. 124]

In einem groben Überblick zeigen Kotonya und Sommerville den Zusammenhang zwischen drei Stufen, die in einem Änderungsprozess durchlaufen werden müssen (siehe Abbildung 28). Zunächst muss ein identifiziertes Problem analysiert und die dazugehörige Änderung spezifiziert werden. Danach wird die Änderung analysiert und die Kosten zur Durchführung abgeschätzt. In der dritten Stufe wird die Änderung implementiert. Nur die zweite Stufe wird detailliert behandelt. Die anderen beiden Stufen, die von Wiegers Change-Control Process zumindest teilweise angeschnitten wurden, werden nicht näher spezifiziert, da ihre Ausgestaltung laut der Autoren vom eingereichten Änderungsproblem abhängt.

Die Detaillierung der zweiten Stufe ist in Abbildung 29 zu sehen.

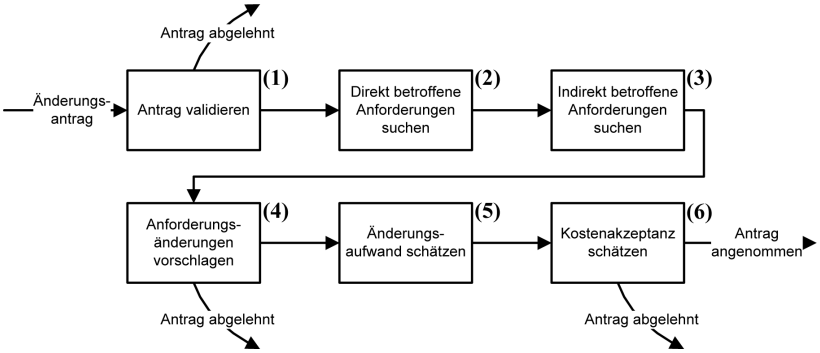


Abbildung 29: Die Stufe zur Änderungsanalyse und Kostenschätzung im Detail [KS00, S. 124]

Die Aktivität der Validierung des Änderungsantrags (1) wurde von Wiegers nicht explizit aufgeführt. Dort wurde lediglich angenommen, dass ein ankommender Änderungsantrag valide ist. Diese Aktivität sollte somit vorgeschaltet sein, um einen unnötigen Aufwand durch eine Auswirkungsanalyse nicht valider Änderungsanträge zu vermeiden.

Die folgenden vier Aktivitäten sind eine Verfeinerung des Übergangs vom Zustand „eingereichter Antrag“ zum Zustand „abgeschätzer Änderungsaufwand“ (siehe Abbildung 27). Zunächst muss ein Anforderungsanalyst oder ein F&E-Ingenieur die direkt betroffenen Anforderungen identifizieren (2). Danach muss er die davon abhängigen Anforderungen ausfindig machen (3) und abschließend alle Anforderungsänderungen vorschlagen (4). Die abhängigen Anforderungen können dabei bei korrektem Einsatz des Anforderungsverwaltungssystems automatisiert ausfindig gemacht werden, indem eine Abhängigkeitsmatrix ausgehend von den sich ändernden Anforderungen erstellt wird. Darauf basierend muss eine Aufwandsabschätzung durchgeführt werden (5). Dies muss manuell geschehen. Der F&E-Ingenieur eignet sich dafür am besten, da meistens eine Abschätzung getroffen werden muss, die sich auf den Implementierungsaufwand bezieht. Die Aktivitäten (2) bis (5) sollten als Verfeinerung zum Änderungsprozess hinzugefügt werden, um eine klare Anleitung zu haben, wie ein Änderungsantrag abgeschätzt wird.

Als letzter Schritt ist vorgesehen, Verhandlungen mit dem Kunden durchzuführen, um herauszufinden ob die Kosten für ihn akzeptabel sind (6). Da im Unternehmen keine Auftragsprojekte durchgeführt werden, ist dieser Schritt nicht notwendig. Nach einer Aufwandsabschätzung wird die zuständige Person des Unternehmens, im Normalfall der Produktmanager mit eventueller Hilfe des Projektmanagers darüber entscheiden, ob die Änderung durchgeführt wird.

Neben dem Prozess zählen Kotonya und Sommerville fünf Attribute auf, die in einem Änderungsantrag auf jeden Fall vorkommen sollten [KS00, S. 126]. Diese Vorschläge sind aufgrund ihrer Wichtigkeit in üblichen Änderungsverwaltungssystemen bereits umgesetzt. Zur Vollständigkeit werden sie im Folgenden aufgelistet:

1. Textfeld zur Dokumentation der Ergebnisse auf jeder Stufe des Änderungsprozesses
2. Datumfeld, in dem jeder Zustandsübergang festgehalten wird
3. Verantwortlichkeitsfeld, in dem festgehalten wird, wer für welche Aktivität verantwirtlich ist
4. Statusfeld, in dem festgehalten wird, in welchem Zustand sich der Änderungsantrag befindet
5. Kommentarfeld, in dem sonstige relevante Informationen erfasst werden können

Insgesamt wurde nun der Änderungsprozess von Wiegers um ein paar Prozessschritte und die eben erwähnten Attribute verfeinert. An der gesamten Struktur des Änderungsprozesses hat sich nichts geändert.

Darüber hinaus vertiefen Kotonya und Sommerville das Thema Prototypenbau, indem sie die verschiedenen Arten des Prototypenbaus vorstellen. Dies wurde im Rahmen des Volere Requirements Process bereits ansatzweise behandelt und als neuer Prozessschritt eingeführt (siehe Abbildung 24 auf Seite 51).

Grundlegend gibt es laut Kotonya und Sommerville zwei verschiedene Gruppen von Prototypen: Die Wegwerf-Prototypen und die evolutionären Prototypen [KS00, S. 73]. Wegwerf-

Protoypen sind zur Unterstützung der Aufnahme und Entwicklung von Anforderungen gedacht, wohingegen die evolutionären Prototypen dafür verwendet werden, um dem Kunden möglichst schnell ein funktionierendes Teilsystem zu liefern [KS00, S. 74]. Da in dieser Arbeit der Fokus auf der Thematik der Anforderungserhebung liegt, soll im Folgenden nur der Wegwerf-Prototyp näher betrachtet werden, der sich aufgrund der schnelleren Umsetzung besser eignet. Die weitere Umsetzung eines Wegwerf-Prototypen sollte unbedingt unterbunden werden, da die langfristigen Probleme wie z.B. die aufwendige Wartbarkeit aufgrund fehlender Strukturierung gegenüber den Vorteilen einer schnelleren Implementierung überwiegen [KS00, S. 74].

Neben der Möglichkeit des Experimentierens mit einer endproduktähnlichen Software gibt es weitere Gründe, die für das Erstellen eines Prototypen sprechen [KS00, S. 74]:

- Validierung des Nutzens und der Machbarkeit vor Entstehung hoher Entwicklungskosten
- Einzig effektiver Weg zur Entwicklung von Benutzerschnittstellen
- Entwicklung von Systemtests ausgehend vom Prototypen
- Aufdeckung von unvollständigen oder inkonsistenten Anforderungen

Zur Umsetzung eines Prototypen gibt es drei verschiedene Ansätze [KS00, S. 76]:

1. Beim **Papierprototypenbau** wird ein Modell der Software gezeichnet, das dem Endprodukt vom Aussehen ähnlich sein soll. Dieser Ansatz ist kostengünstig, da keine ausführbare Software entwickelt werden muss. Trotzdem ist es ein effektiver Ansatz, da Analysten und Endnutzer sehen können, wie das Produkt benutzt werden könnte.
2. Beim **'Zauberer von Oz'-Prototypenbau** wird eine Benutzeroberfläche in elektronischer Form bereitgestellt hinter der allerdings keine Logik steckt. Ein Nutzer kann trotzdem mit diesem Prototypen interagieren, da im Hintergrund eine andere Person, der so genannte 'Zauberer von Oz' die Reaktionen des Systems simuliert. Auch dieser Ansatz ist relativ günstig, da kaum Entwicklungsaufwand notwendig ist. Bei häufigen Experimenten mit dem Prototypen kann der Aufwand allerdings steigen, da immer eine Person benötigt wird, die die Reaktionen des Systems simuliert.
3. Der aufwendigste Ansatz ist die Entwicklung eines **ausführbaren Prototypen**. Zur Entwicklung muss allerdings eine Programmiersprache benutzt werden, mit der die tatsächliche Implementierung der Software nicht stattfinden wird. Andernfalls könnte sonst die Gefahr bestehen, dass der Prototyp weiterentwickelt wird. Für diesen Ansatz kommen Programmiersprachen wie z.B. 4GL[29], Visual Basic oder Java in Frage.

Der Prozessschritt des Prototypenbaus soll dadurch erweitert werden, dass im Unternehmen einer der ersten beiden Ansätze je nach Komplexität des behandelten Anwendungsfalls

[29] 4GL heißt „Fourth Generation Languages“ und steht für Sprachen, die im Gegensatz zu anderen Sprachen wie BASIC oder FORTRAN einfacher zu erlernen und zu benutzen sind [OM08].

angewendet wird. Der dritte Ansatz sollte vom Unternehmen zunächst nicht behandelt werden, da er aufwendiger ist und innerhalb eines Projekts im Normalfall mehrere Prototypen erstellt werden sollen. Des Weiteren wäre hierfür zunächst dafür zu sorgen, dass die Mitarbeiter der F&E-Abteilung eine der geforderten Sprachen beherrschen.

6.3 Zusammenfassung der Prozessoptimierung

In diesem Kapitel wurden alle Optimierungsvorschläge einzeln aufgezeigt und in die Anforderungsprozesse eingearbeitet. Alle Ergebnisse aus Unterkapitel 6.2 sind in Appendix B zusammengefasst. Neben diesen Änderungen wurden weitere kleinere Änderungen durchgeführt, die nicht im vorherigen Unterkapitel erwähnt wurden. Dies sind triviale Optimierungen, die damit zusammenhängen, dass manche bisher manuellen Vorgänge nun vom Anforderungsverwaltungssystem automatisch durchgeführt werden. Dies betrifft vor allem alle Aktivitäten, in denen eine Rückverfolgbarkeitsmatrix erstellt werden muss. Ein formelles Lastenheft (SRS) oder Pflichtenheft (ERS) muss nun ebenfalls nicht mehr erstellt werden, da dies ein einfacher Datenbankauszug aus dem Anforderungsverwaltungssystem ist.

Insgesamt sind nun mehr Aktivitäten als zuvor in der Darstellung zu sehen. Dies bedeutet nicht zwangsweise, dass es sich daher um mehr Aufwand handelt. Beispielsweise ist auf Seite 1 zu sehen, dass sich zwei Anforderungsaufnahmezyklen wiederholen, die zuvor in einem Zyklus zusammengefasst waren. Daneben sind an manchen Stellen neue, teilweise optionale, Aktivitäten eingeführt worden.

Das Ziel der Prozessoptimierung war nicht eine Verschlankung der Anforderungsprozesse sondern eine Verbesserung der Anforderungsqualität. Die hauptsächlichen Optimierungen werden im Folgenden abschließend wiederholt:

- Systemanforderungen, die im Lastenheft (SRS) abgelegt wurden, schreibt zukünftig nur noch der Anforderungsanalyst. Er hat als einziger die Kompetenz um Anforderungen zu formulieren. Der Produktmanager wird nun Anwendungsfälle verfassen, die der Anforderungsanalyst zur Definition der Systemanforderungen nutzen kann.

- Nach der Definition von Anforderungen jeglicher Art wird vor einer Durchsicht zunächst die Vollständigkeit analysiert. Dies wird bewirken, dass Durchsichten zielführender durchgeführt werden können. Außerdem erhöht dies genauso wie im vorherigen Absatz die Qualität der Anforderungen.

- Die grundlegendste Anpassung der Anforderungsprozesse erfolgte bei der Vorgehensweise der Anforderungserhebung. Es gibt nun ein inkrementelles Vorgehen, dass sich bis hin zur abschließenden Implementierung durchzieht. Nach Erhebung der für die grundlegende Architektur kritischen Anforderungen werden Inkremente definiert, die aufeinander aufbauen. Für jedes Inkrement werden zunächst Anforderungen erhoben, die Architektur entworfen und implementiert. Dies wird so oft wiederholt, bis das letzte Inkrement vollendet wurde.

- Als zwei weitere Hilfsmittel zur Erhöhung der Anforderungsqualität wurden explizit der Prototypenbau und die Qualitätsschranke eingeführt. Der Prototypenbau soll bei komplexen Inkrementen eingesetzt werden um den Anforderungsanalysten bereits zu diesem

frühen Zeitpunkt zu unterstützen und nicht erst bei der Implementierung fehlende oder falsche Anforderungen zu korrigieren. Die Qualitätsschranke soll darüber hinaus die Anforderungen jedes Inkrements nach Erhebung kritisch betrachten und nach definierten Qualitätskriterien hin untersuchen.

- Der nächste Schritt, den das Unternehmen gehen muss, um die Vorschläge umzusetzen, ist die Einführung der Anforderungsprozesse in einem Pilotprojekt und je nach Erfolg im gesamten Unternehmen. Der Erfolg des Optimierungsvorschlags kann mit Hilfe von Kennzahlen gemessen werden, die in Kapitel 7 eingeführt werden. Wie die Einführung der angepassten Anforderungsprozesse werkzeug- und methodentechnisch unterstützt werden kann, wird in Kapitel 2 erklärt.

7 Evaluierung

Dieses Kapitels zeigt Möglichkeiten auf, mit denen der Erfolg der Prozessoptimierung gemessen werden kann. Dabei werden zum Einen in Unterkapitel 7.1 Vergleichskennzahlen erläutert, mit denen Projekte nach den alten Anforderungsprozessen mit Projekten nach den optimierten Anforderungsprozessen verglichen werden können. Dieser Vergleich kann im Rahmen dieser Arbeit nicht durchgeführt werden, da das Pilotprojekt, in dem die optimierten Anforderungsprozesse eingeführt werden, erst nach Abschluss der Arbeit beginnen wird. Um die Messung dieser Kennzahlen zu unterstützen, ist in Appendix B eine Berechnungshilfe angehängt.

Zum Anderen werden in Unterkapitel 7.2 Projektverlaufskennzahlen erläutert, die während eines Projekts Indikatoren für eine erfolgreiche Arbeit sind. Ein Erfassen dieser Kennzahlen ermöglicht dem Projekt- oder Produktmanager ein frühzeitiges Eingreifen.

Die meisten Kennzahlen stammen aus der Fachliteratur. Andere Kennzahlen wurden selbstständig erarbeitet und sind speziell auf das Unternehmen zugeschnitten. Das Unternehmen sollte dieses Kapitel als Hilfestellung verstehen und die vorgeschlagenen Kennzahlen mit Einführung der neuen Anforderungsprozesse messen.

7.1 Vergleichskennzahlen

Änderungsquote Die Häufigkeit der Anforderungsänderungen wurde von vielen Mitarbeitern in den Befragungen und auch während inoffizieller Gespräche bemängelt. Grund für häufige Änderungen können nicht sauber abgeschlossene Analysen oder Anforderungserhebungen sein [Ebe08, S. 276]. Genau an diesem Punkt wurde mit der Anpassung der Anforderungsprozesse angesetzt, sodass eine höhere Anforderungsqualität und dadurch weniger nachträgliche Änderungen die Folge sein sollten.

Um dies zu messen, wird die folgende Kennzahl eingeführt:

$$\ddot{A}nderungsquote = \#\ddot{A}nderungen/\#Anforderungen$$

Da es sich hierbei um eine Quote gemessen an der Gesamtzahl der Anforderungen handelt, lässt sich diese Kennzahl gut über mehrere Projekte hinweg vergleichen. Trotzdem sollten zwei Projekte ausgewählt werden, die von der Größe her miteinander vergleichbar sind, da die Projektgröße einen Einfluss auf die Änderungsquote haben kann. Des Weiteren sollten die beiden zu vergleichenden Projekte einen ähnlichen Anteil neuer Funktionalitäten aufweisen [Ebe08, S. 276], da manche Anforderungen neuer Technologien erst nach Implementierung auffallen.

Anforderungsfehlerdichte / -quote Anforderungsfehler können fehlende Anforderungen, Inkonsistenzen, falsche Inhalte, ungenaue Formulierungen etc. sein [Ebe08, S. 276]. Fehler können eine Ursache für Anforderungsänderungen sein. Sie sind ein großes Hindernis für die spätere Implementierung.

Es gibt für den dokumentenorientierten und den informationsorientierten Ansatz[30] jeweils

[30] Die Begriffe „dokumentenorientierte“ und „informationsorientierte“ Anforderungserhebung wurden in Kapitel 4.2.1 eingeführt.

zwei leicht verschiedene Kennzahlen:

$$Fehlerdichte = \#Fehler/\#Spezifikationsseiten$$

$$Fehlerquote = \#Fehler/\#Anforderungen$$

Die Fehlerdichte soll im dokumentenorientierten Ansatz verwendet werden, da hier zur Berechnung einer Fehlerquote die Anzahl der Anforderungen in einer Spezifikation nicht leicht ermittelbar wäre. Die Fehlerquote hingegen soll im informationsbasierten Ansatz verwendet werden. Eine Ähnlichkeit der zu vergleichenden Projekte ist nicht zwingend notwendig.

Aufwand für Rückverfolgbarkeitsmatrizen Da im Unternehmen ein sehr hoher Aufwand betrieben wird, um die Rückverfolgbarkeitsmatrizen zu erstellen und dies zukünftig automatisiert mit Hilfe des neu eingeführten Anforderungsverwaltungssystems ablaufen wird, sollte die erhoffte Zeitersparnis gemessen werden. Dafür muss der bisherige Zeitaufwand zur Erstellung aller Rückverfolgbarkeitsmatrizen eines gesamten Projekts den zukünftigen Zeitaufwänden zur Verbindung der Anforderungen im System gegenübergestellt werden.

Hierbei sollte nicht vergessen werden, dass die Verbindung der Anforderungen auch anderen Zwecken, wie z.B. das Durchführen einer Auswirkungsanalyse, dienen kann. Wenn möglich, sollten auch diese Zeitersparnisse gemessen werden.

Rückverfolgbarkeit der Anforderungen Die Rückverfolgbarkeit der Anforderungen ist ein wichtiges Thema für das Unternehmen, da es in Kundenaudits nachweisen können muss, dass bestimmte Testfälle zu bestimmten Systemanforderungen durchgeführt wurden etc. Gemessen wird dabei die Quote der Anforderungen, die ausreichend verbunden sind [Ebe08, S. 277].

Da eine umfassende Untersuchung jeder Anforderung zu aufwendig wäre, sollte eine Stichprobe der Anforderungen gezogen werden, die dann im Detail untersucht wird.

Eine beispielhafte Berechnung eines Auszugs der Kennzahlen ist in Tabelle 6 abgebildet. Eine Gesamtübersicht aller Kennzahlen mit Berechnungshilfen ist in Appendix B angehängt.

Tabelle 6: Beispielhafte Berechnung von zwei Kennzahlen.

Kennzahl	**Berechnung**
Änderungsquote	***26,35%***
# Anzahl Änderungen	2.685
# Anzahl Anforderungen	10.190
Fehlerdichte	***2,36***
#Fehler	231
#Spezifikationsseiten	98
...	...

7.2 Projektverlaufskennzahlen

Ändernde Anforderungen pro Entwicklungsphase Diese Kennzahl ähnelt der Vergleichskennzahl „Änderungsquote“. In diesem Fall werden die Änderungen pro Entwicklungsphase gemessen [Ebe08, S. 276]. Dabei gibt es je nach Neuigkeitsgrad des Produkts typische Änderungsraten. Die Änderungsrate sollte im Projektlebenslauf abnehmen.

Wird diese typische Änderungsrate über- oder unterschritten, sollte ein Produkt- oder Projektmanager eingreifen. Eine zu hohe Änderungsrate könnte den Aufwand für das Produkt in die Höhe treiben. Eine zu niedrige Änderungsrate könnte ein Hinweis darauf sein, dass der Kunde nicht ausreichend involviert ist und das Produkt zum Schluss nicht akzeptiert.

Status der Anforderungen Der gesamte Status der Anforderungen gibt einen Hinweis darauf, in welcher Phase sich das Projekt befindet und ist damit das „primäre Maß, um Projekte zu verfolgen“ [Ebe08, S. 274].

Es gibt drei grundlegend verschiedene Status für Anforderungen: „Vereinbart“, „Entwickelt“ und „Abgeschlossen“. Wöchentlich sollten die Gesamtzahl aller Anforderungen sowie die vereinbarten, entwickelten und abgeschlossenen Anforderungen gezählt werden. Der Produktmanager kann diese Zahlen mit seinem Projektplan vergleichen und erkennen, ob sich die Erfüllung der Anforderungen so entwickelt, wie geplant. Ist dies nicht der Fall, kann er darauf reagieren, indem er beispielsweise das Entwicklerteam erweitert.

Um einen transparenten Umgang mit dem Projektstatus zu sichern, sollte dieser auf einer elektronischen Wandtafel im Unternehmen sichtbar gemacht werden. Damit ist der Stand der Dinge für jeden schnell ersichtlich und motiviert das Erreichen von vereinbarten Zielen.

8 Einführung der Anforderungsprozesse

In diesem Kapitel werden die Möglichkeiten aufgezeigt, mit denen die neuen, angepassten Anforderungsprozesse im Unternehmen eingeführt werden können. Hierzu soll zunächst in Unterkapitel 2.1 der Umgang mit Veränderungsprozesse aus einer psychologischen Sichtweise erklärt werden. Danach folgt in Unterkapitel 2.2 eine detaillierte Beschreibung des HOOD Capability Model (kurz: HCM) für die Anforderungserhebung. Es ist ein Reifegradmodell zur Messung der Anforderungsprozesse und kann den Veränderungsprozess im Unternehmen unterstützen.

8.1 Psychologie von Veränderungsprozessen

Notwendige Veränderungen in Unternehmen werden meist durch Druck von Außen ausgelöst. Deshalb akzeptieren Mitarbeiter Veränderungen wenig. Laut einer Studie von McKinsey aus dem Jahr 2008 wird nur jede dritte Veränderung in einem Unternehmen erfolgreich durchgeführt [AK09, S. 101].

Ein Veränderungsprozess besteht aus drei Stufen, an denen eine Veränderung jeweils scheitern kann. Zunächst müssen Barrieren in der Denkweise überwunden werden, indem das Unternehmen seine Mitarbeiter **auftaut** und sie positiv gegenüber Veränderungen motiviert. Nach dieser Stufe sind die Mitarbeiter bereit zu **lernen**. Auch auf dieser Stufe gilt es, die Mitarbeiter beim Lernen der Veränderung zu unterstützen. Auf der letzten Stufe muss man die erlernten Veränderungen **einfrieren**, damit die Mitarbeiter nicht wieder ihre alten Prozesse ausüben [HW05, S. 163f.].

Werden nicht alle drei Stufen für einen Veränderungsprozess durchschritten, so kann dies zu einem Fehlschlagen führen. Eine hohe organisatorische Unterstützung ist erforderlich, sodass die Veränderungen nicht zum Erliegen kommen [HW05, S. 170]

8.1.1 Auftauen

[HW05] unterscheidet zwischen dem Überwinden der Barrieren und dem Auftauen der Mitarbeiter. Aufgrund der herrschenden Meinung der Literatur (z.B. [Sch97]) und wegen der ähnlichen Vorgehensweise in beiden Stufen, wird hier zwischen beiden Stufen nicht unterschieden.

Barrieren sind demotivierende Kräfte, die aufgehoben werden müssen, damit Menschen die Realität erkennen können [HW05, S.164]. Menschen ignorieren oder blenden oft Signale aus, die Anzeichen für einen drohenden Misserfolg sind.

Schein hat dabei folgende Maßnahmen identifiziert um Barrieren zu überwinden [Sch97, S. 320]:

1. Unbequemlichkeit und Unausgeglichenheit in der aktuellen Situation schaffen
2. Hinweisen auf nötige Änderungen mit wichtigen Unternehmenszielen verknüpfen
3. Relative Sicherheit schaffen, um offen für neue Arbeitsweisen zu sein

Der Mangel an Unbequemlichkeit oder Angst in der aktuellen Situation kann durch eine künstliche Unbequemlichkeit oder Angst überbrückt werden. Das Unternehmen könnte fordern, dass die Spezifikationen zur formellen Prüfung in einer neuen Form vorgelegt werden müssen, die manuell schwierig, mit dem einzuführenden Anforderungsverwaltungssystem hingegen leicht eingehalten werden kann.

Das Ignorieren von Hinweisen auf die Notwendigkeit von Änderungen kann gelöst werden, indem die Hinweise mehrfach und auf verschiedenen Wegen ins Gedächtnis der Mitarbeiter gerufen werden. Hierfür könnte das Unternehmen die in Kapitel 7 eingeführten Kennzahlen verwenden und besonders auffällige Hinweise auf Verbesserungspotenziale intern publik machen.

Zur Beseitigung des Mangels an Sicherheit und Angst vor Identitäts- und Integritätsverlust bei erfolgter Änderung rät Schein, eine *relative Sicherheit*[31] aufzubauen. Dafür muss entweder die Situation, in der sich der Mitarbeiter anpasst, für ihn verbessert werden oder es muss die Situation verschlechtert werden, in der sich der Mitarbeiter nicht anpasst. Das Unternehmen könnte zum einen Fehler beim Ausführen der neuen Prozesse tolerieren und zum anderen klar darstellen, dass Marktanteile und damit Arbeitsplätze in Gefahr sind, wenn die Qualität der Anforderungen nicht verbessert wird, was mit den neuen Prozessen mittelfristig geändert werden soll.

8.1.2 Lernen

Nachdem Barrieren eliminiert wurden und eine positive Motivation geschaffen wurde, ist der Mensch bereit umzudenken und Neues zu lernen. Der Lernprozess muss dabei gelenkt werden.

Verschiedene Lernmodelle können angewandt werden. Der Demingkreis (siehe Abbildung 2) ist ein vierphasiger Problemlösungsprozess, der typischerweise zur Verbesserung von Geschäftsprozessen eingesetzt wird.

In einem Kreislauf sollen vier Aktivitäten wiederholt werden: Zunächst soll geplant werden, was gemacht wird. Dies ist im Rahmen der neuen Prozesse dieser Bachelorarbeit bereits geschehen. Danach soll das Neue ausprobiert werden. Die Grundlage für das Ausprobieren bildet dieses Kapitel. Danach soll das Geschaffene beobachtet werden um aus dieser Erfahrung die Handlung entsprechend zu verbessern.

Hierbei kann das Unternehmen seine Mitarbeiter unterstützen, indem es bei der Beobachtung und Anpassung aktive Hilfestellung gibt. Zum einen können die Mitarbeiter anhand der einzuführenden Kennzahlen aus Kapitel 7 beobachten, wie sich z.B. die Qualität der Anforderungen verbessert. Zum anderen sollte das Unternehmen zur kontinuierlichen Anpassung der Prozesse an die Bedürfnisse der Mitarbeiter Personal einstellen oder einkaufen (z.B. Change Coaches), das sich vornehmlich darum kümmert.

8.1.3 Einfrieren

Damit das neu Erlernte langfristig beibehalten wird und die Mitarbeiter sich nicht wieder die alten Arbeitsabläufe angewöhnen, müssen die neuen Prozesse „eingefroren“ werden. Dabei

[31] Relative Sicherheit bedeutet, dass sich der Mitarbeiter bei Anpassung an die Änderung in einer *besseren* Situation befinden muss, als wenn er sich nicht anpassen würde.

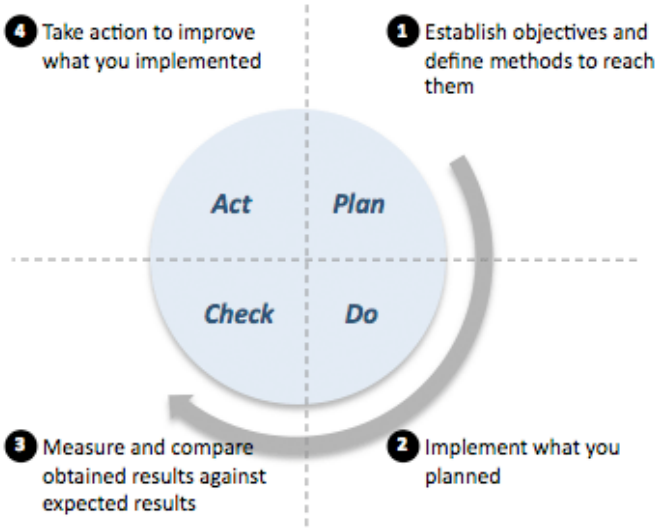


Abbildung 30: Der Demingkreis [Dem00]

müssen bestätigende Zahlen gefunden werden [HW05, S. 168], die von den Mitarbeitern anerkannt werden. Hierfür können die Kennzahlen aus Kapitel 7 eingesetzt werden, die bereits beim Auftauen dafür verwendet wurden, die bisherigen Prozesse in Frage zu stellen. Werden keine bestätigende Zahlen gefunden, muss der Veränderungsprozess fortgeführt werden, bis eine Lösung gefunden wird, die bestätigende Zahlen liefert.

8.2 Einführung des HOOD Capability Model (HCM)

Das HOOD Capability Model (kurz „HCM“) ist eine Methode zum „Messen und Kategorisieren von Prozessen“ [HW05, S. 170]. Es kann die Fähigkeiten einer Organisation ähnlich wie z.B. CMMI [32] [CKS07] oder SPICE[33] [Hör06] bewerten. Gleichzeitig dient es der Unterstützung des Veränderungsprozesses für die Anforderungserhebung, da es das „große Ziel“ in mehrere Schritte unterteilt, die auf dem Weg dahin durchgeführt werden müssen.

HCM setzt direkt an der ersten Stufe des Veränderungsprozesses an. Es nimmt die Ängste der Mitarbeiter vor Fehlern durch klare und einfache, kleine Schritte. Jeder Mitarbeiter kann die Vorteile und Risiken einer Veränderung erkennen und erhält somit einer höhere relative Sicherheit. Dazu kommt, dass sich Menschen von schrittweise messbaren Erfolgen einfacher motivieren lassen. Der Fortschritt der Veränderung lässt sich in diskreten Zeitabständen bestimmen.

Das HCM unterteilt die Fähigkeiten zur Anforderungserhebung grob in drei Stufen (siehe Tabelle 1). Die einzelnen Stufen können selbst wiederum aus mehreren Unterstufen bestehen. Im Folgenden wird erklärt, was getan werden muss, um die jeweiligen Stufen zu erreichen. Ein Unternehmen, das nicht auf Stufe 1, demnach also auf Stufe 0 ist, besitzt laut HCM

[32] CMMI steht für **C**apability **M**aturity **M**odel **I**ntegration.
[33] SPICE steht für **S**oftware **P**rocess **I**mprovement and **C**apability D**e**termination

„keine“ Anforderungserhebung.

Tabelle 7: Bedingungen für die drei Stufen des HOOD Capability Model [HFPW08, S. 227]

	Stufe 1	**Stufe 2**	**Stufe 3**
Umfang	• Identifizieren von Schnittstellen • Projektinteressierte & Rollen	• Definieren von Schnittstellen	
Modellieren	• Umfang	• Sequenz, Zustände, Daten, Algorithmen • Für den Leserkreis angepasst	
Erheben	• Anforderungen aus existierenden Dokumenten	• Priorisierung	
Spezifizieren	• Atomar • Identifizierbar (als Anforderung) • Identifizierbar gegenüber anderen Anforderungen • Struktur	• Verständlich • Nachweisbar • Realisierbar • Eindeutig • Widerspruchsfrei • Redundanzfrei • Abstraktionsebene (Ableitung)	• Vollständig • Rückverfolgbar • Abstraktionsebene (Lösungsvorgabe)
Analyse	• Atomar • Identifizierbar (als Anforderung) • Identifizierbar gegenüber anderen Anforderungen • Struktur	• Verständlich • Nachweisbar • Realisierbar • Eindeutig • Widerspruchsfrei • Redundanzfrei • Abstraktionsebene (Ableitung)	• Vollständig • Rückverfolgbar • Abstraktionsebene (Lösungsvorgabe)
Durchsicht	• Durchführen • Dokumentieren	• Alle Rollen • Jede Iteration • Eingangskriterien	

Zu beachten ist, dass es sich bei den drei Stufen nicht um eine vorgeschriebene Struktur hin zum perfekten Prozess handelt. Beispielsweise kann es sein, dass manche Unternehmen, die sich noch auf Stufe 1 befinden, bereits rückverfolgbare Anforderungen aus Stufe 3 besitzen. Dem Unternehmen bleibt die Freiheit zu entscheiden, welche Arbeitsabläufe zu welchem Zeitpunkt eingeführt werden. Den Stufenaufbau und die Abfolge der zu erreichenden Ziele hat die HOOD Group aus ihrer Lehrerfahrung in Organisationen identifiziert [HW05, S. 184].

8.2.1 HCM Stufe 1

Ein Unternehmen auf Stufe 1 wird dadurch charakterisiert, dass die erhobenen Anforderungen in der Regel atomar und identifizierbar sind und aus ehemaligen Lastenheften gewonnen werden. Auf dieser Stufe geht es zunächst darum, Anforderungen als solche z.B. in einem Text zu identifizieren. Laut Hood ist Erfolg die Erfüllung von Anforderungen [HW05, S. 175]. Diese Fähigkeit bildet somit die Grundlage für die Anforderungserhebung. Eine Kurzübersicht über die zu erfüllenden Bedingungen für Stufe 1 ist in Tabelle 2 dargestellt. Die Unterstufen kombinieren Elemente der jeweiligen Kategorien (Umfang, Modellieren, etc.).

Tabelle 8: Bedingungen für die Unterstufen von HCM Stufe 1 [HW05, S. 177ff.]

Stufe 1.1	**Stufe 1.2**	**Stufe 1.3**	**Stufe 1.4**
Umfang: • Identifizieren von Schnittstellen • Projektinteressierte & Rollen *Modellieren*: • Umfang *Durchsicht*: • Durchführen • Dokumentieren	*Erheben:* • Anforderungen aus existierenden Dokumenten	*Spezifizieren:* • Atomar • Identifizierbar (als Anforderung) • Identifizierbar gegenüber anderen Anforderungen *Analyse:* • Atomar • Identifizierbar (als Anforderung) • Identifizierbar gegenüber anderen Anforderungen	*Spezifizieren:* • Struktur *Analyse:* • Struktur

Stufe 1.1 Auf der ersten Unterstufe soll ein Unternehmen die Projektbeteiligten und damit einhergehend die Schnittstellen identifizieren und die Methodik der Durchsicht einführen [HW05, S. 176f.].

Durch die Festlegung der Schnittstellen wird genau abgesteckt, wo die Grenzen des Systems liegen. Damit wird der Umfang des Projekts definiert. Die Schnittstellen und Grenzen sollten in Form eines Blockdiagramms modelliert werden. Damit dieser Umfang nicht zu klein gewählt wird, müssen alle Projektinteressierten und ihre Rollen aufgelistet werden. Diese Liste ist wichtig, da klar festgelegt wird, wer für das Erheben und das Prüfen der Anforderungen berücksichtigt wird.

Die Arbeitsprodukte müssen danach in einer Durchsicht von einem Entscheidungsgremium bezüglich ihres erreichten Reifegrads beurteilt werden. Bei zu geringen Reifegrad, also schlechter Qualität, ist Nacharbeit notwendig, bis das Entscheidungsgremium zufrieden gestellt ist. Durchsichten müssen dokumentiert werden.

Stufe 1.2 Auf der zweiten Unterstufe kommt die Erkenntnis hinzu, dass die meisten Projekte Weiterentwicklungen bereits bestehender Systeme sind [HW05, S. 178]. Um den Aufwand für die Anforderungserhebung zu minimieren, können daher die Spezifikationen der Vorgängersysteme verwendet werden. Sie können übernommen und weiterentwickelt werden.

Stufe 1.3 Die Anforderungseigenschaften „Identifizierbarkeit“ und „Atomizität“ sind auf der dritten Unterstufe gefordert [HW05, S. 178f.]. Atomizität bedeutet, dass Anforderungen aus einem Text extrahiert und isoliert von anderen Informationen einzeln dargestellt werden. Daneben müssen sie, um identifizierbar zu sein, klar als Anforderungen in der Spezifikation gekennzeichnet sein und eine eindeutige Kennzeichnung besitzen. Diese Eigenschaften müssen beim Erheben und Niederschreiben der Anforderungen beachtet werden und danach in einer Analyse von einem definierten Personenkreis überprüft werden.

Stufe 1.4 Auf der vierten Unterstufe wird die Struktur der Spezifikation behandelt [HW05, S. 179]. Dabei soll die Spezifikation eine vorgegeben Struktur einhalten, sodass Anforderungen einem Thema oder Kapitel zugeordnet werden können.

Empfehlungen für das Unternehmen Im aktuellen Zustand würde das Unternehmen Stufe 1.1 nicht erreichen. Dafür müsste es je Projekt die Projektinteressierten und ihre Rollen auflisten und den Umfang des Projekt in einem Blockdiagramm darstellen.

Die Bedingung für Stufe 1.2 wird bereits teilweise erfüllt. Bei Produktweiterentwicklungen werden allerdings nur die Delta-Anforderungen (vgl. Seite 5) niedergeschrieben und die alten Anforderungen als bekannt angenommen. Um neuen Mitarbeitern diese Hürde zu nehmen, sollten auch die alten Anforderungen wieder in die neue Spezifikation integriert werden. Vor allem ist das Unternehmen dazu angeregt, nicht mehr zu beachtende Anforderungen auszusortieren. Diese Altlasten machen bereits heute einen beträchtlichen Aufwand aus, da sie z.B. bei der Auswahl der Testfälle immer überlesen werden müssen.

Zum Erreichen der Stufe 1.3 müsste neben der Sicherstellung eindeutiger Identifikationsnummern für Anforderungen Analysen für die Spezifikationsvorschriften eingeführt werden. Momentan sind eindeutige und unveränderliche Identifikationsnummern aufgrund des verwendeten Werkzeugs (Word) nicht möglich. Durch den Einsatz eines Anforderungsverwaltungssystems würde diese Bedingung erfüllt.

Für die Erreichung der Stufe 1.4 und damit dem Erreichen der Stufe 1 müssten zusätzlich zum Vorhergehenden keine weiteren Verbesserungen durchgeführt werden, da aufgrund der „Life Cycle“-Dokumente bereits eine Struktur zur Spezifikation besteht.

8.2.2 HCM Stufe 2

Ein Unternehmen auf Stufe 2 wird dadurch charakterisiert, dass die erhobenen Anforderungen in der Regel eindeutig und prüfbar sind und sich alle relevanten Rollen an der Anforderungserhebung beteiligen. Diese Stufe ist sehr aufwendig einzuführen. Testfälle können aber zielgerichtet entlang der einzelnen Anforderungen entwickelt werden [HW05, S. 180]. Ein Unternehmen auf dieser Stufe ist „fähig“, eine erfolgreiche Anforderungserhebung durchzuführen. Eine Kurzübersicht über die zu erfüllenden Bedingungen für Stufe 2 ist in Tabelle 3

dargestellt. Die Unterstufen kombinieren Elemente der jeweiligen Kategorien (Umfang, Modellieren, etc.).

Tabelle 9: Bedingungen für die Unterstufen von HCM Stufe 2 [HW05, S. 181ff.]

Stufe 2.1	**Stufe 2.2**	**Stufe 2.3**
Umfang: • Definieren von Schnittstellen *Modellieren*: • Sequenz, Zustände, Daten, Algorithmen • Für den Leserkreis angepasst *Erheben:* • Priorisierung *Spezifizieren:* • Verständlich • Nachweisbar *Analyse:* • Verständlich • Nachweisbar	*Spezifizieren:* • Realisierbar • Eindeutig *Analyse:* • Realisierbar • Eindeutig	*Spezifizieren:* • Widerspruchsfrei • Redundanzfrei • Abstraktionsebene (Ableitung) *Analyse:* • Widerspruchsfrei • Redundanzfrei • Abstraktionsebene (Ableitung) *Durchsicht:* • Alle Rollen • Jede Iteration • Eingangskriterien

Stufe 2.1 Auf der ersten Unterstufe geht es darum, die einzelnen Anforderungen verständlicher und detailreicher zu machen, damit sie prüfbar werden [HW05, S. 181]. Zunächst müssen die Schnittstellen, die bereits auf Stufe 1.1 identifiziert wurden, im Detail definiert und dokumentiert werden.

Zur besseren Verständlichkeit müssen verschiedene Modellierungstechniken ausgewählt und verwendet werden. Die Modelle sowohl vom Auftragnehmer, als auch vom Auftraggeber verstanden werden. Darüber hinaus müssen Anforderungen in Spezifikationen verständlich beschrieben werden. Eine Analyse der Verständlichkeit von Anforderungen muss ebenso erfolgen.

Zur Detaillierung werden Anforderungen priorisiert. Des Weiteren muss beim Spezifizieren der Anforderungen darauf geachtet werden, dass diese nachweisbar sind. Diese Eigenschaft muss analysiert werden.

Stufe 2.2 Auf der zweiten Unterstufe sollen weitere Eigenschaften von Anforderungen umgesetzt werden [HW05, S. 182]. Anforderungen müssen so spezifiziert werden, dass sie realisierbar und eindeutig sind. Eindeutigkeit bedeutet in diesem Zusammenhang, dass verschie-

dene Projektbeteiligte dasselbe von einer Anforderungsbeschreibung verstehen sollen. Die Realisierbarkeit und die Eindeutigkeit müssen auf ihre Erfüllung hin analysiert werden.

Stufe 2.3 Auf der dritten Unterstufe soll die Konsistenz der Anforderungen sichergestellt werden [HW05, S.182f.]. Hierfür werden drei weitere Eigenschaften von Anforderungen eingeführt, auf die geachtet werden muss: Anforderungen sollen sich nicht gegenseitig widersprechen, Anforderungen oder Teile davon (wobei diese auf Grund der Atomizität nicht vorliegen dürften) dürfen sich nicht wiederholen und die Anforderungen müssen immer korrekt von einer übergeordneten Abstraktionsebene abgeleitet werden. Alle drei Eigenschaften müssen in der Analyse überprüft werden.

Darüber hinaus wird in dieser Unterstufe die Struktur der Durchsichten festgelegt. Die Ergebnisse von Analysen müssen allen relevanten Projektinteressierten und nicht wie auf Stufe 1.1 einem frei ausgewählten Entscheidungsgremium zur Durchsicht vorgelegt werden. Die Durchsichten müssen iterativ durchgeführt werden. Die Kriterien der Durchsichten müssen dokumentiert werden.

Empfehlungen für das Unternehmen Nachdem die Bedingungen zum Erreichen der Stufe 1 umgesetzt wurden, können die folgenden Schritte angegangen werden. Zur Erreichung der Stufe 2.1 muss eine neue Technik im Unternehmen eingeführt werden: Das Modellieren. Daneben müssen die bereits identifizierten Schnittstellen detailliert definiert werden und die Verständlichkeit und Nachweisbarkeit der Anforderungen explizit im Zuge des Quality Gateways (siehe Seite 53) geprüft werden.

Die folgenden Schritte gelten nicht als Empfehlung für das erste Pilotprojekt zur Einführung der neuen Prozesse, da in der kurzen Zeit nicht zuviel Neues angegangen werden sollte. Sie werden zur Vollständigkeit hier aufgeführt. Zur Erreichung von Stufe 2.2 und damit einhergehend auch 2.3 müssten folgende Anforderungskriterien im Quality Gateway geprüft werden: Realisierbarkeit, Eindeutigkeit, Widerspruchsfreiheit, Redundanzfreiheit und eine korrekte Ableitung der nächsthöheren Abstraktionsebene. Letzteres Kriterium wird durch diese neue Struktur zur Anforderungserhebung verbessert umgesetzt. Zusätzlich müssten zukünftig alle Rollen für Durchsichten herangezogen werden.

8.2.3 HCM Stufe 3

Ein Unternehmen auf Stufe 3 wird dadurch charakterisiert, dass die erhobenen Anforderungen in der Regel vollständig und zur Quelle rückverfolgbar sind. Unternehmen auf dieser Stufe können sich als Experten in der Anforderungserhebung bezeichnen [HW05, S. 183]. Eine Kurzübersicht über die zu erfüllenden Bedingungen für Stufe 3 ist in Tabelle 4 dargestellt. [HW05, S. 184f.]:

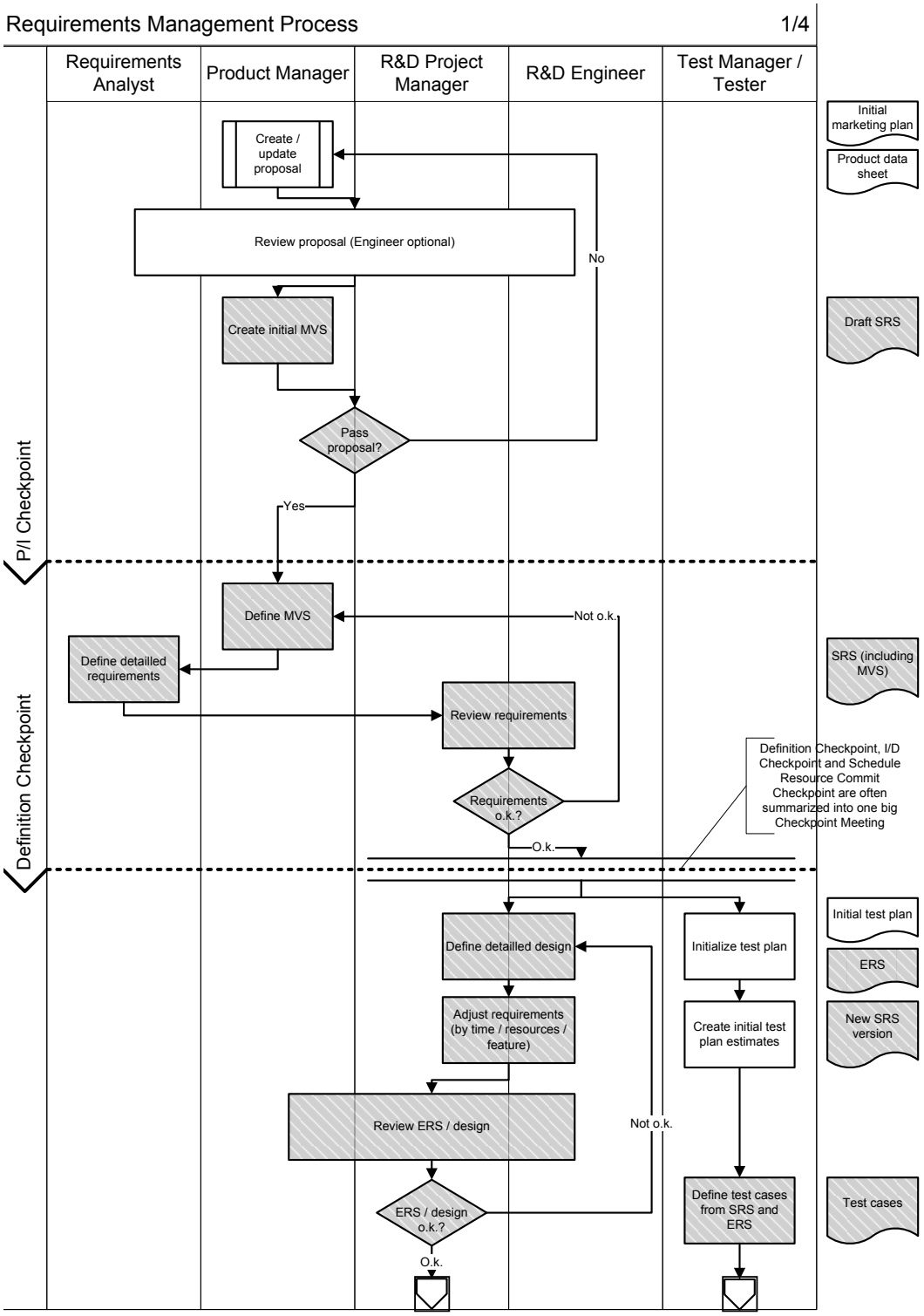
Bei der Spezifikation muss darauf geachtet werden, dass jede Anforderung, die bei der Modellierung erarbeitet wurde, spezifiziert wird. Daneben muss bei der Eintragung einer Anforderung jedes Mal der Ursprung dieser Anforderung dokumentiert werden. Es muss darauf geachtet werden, dass die Spezifikation so abstrakt wie möglich gehalten wird, um eine Lösung des Problems und damit die Freiheiten eines Programmierers nicht einzuschränken. Die drei genannten Eigenschaften müssen in der Analyse überprüft werden.

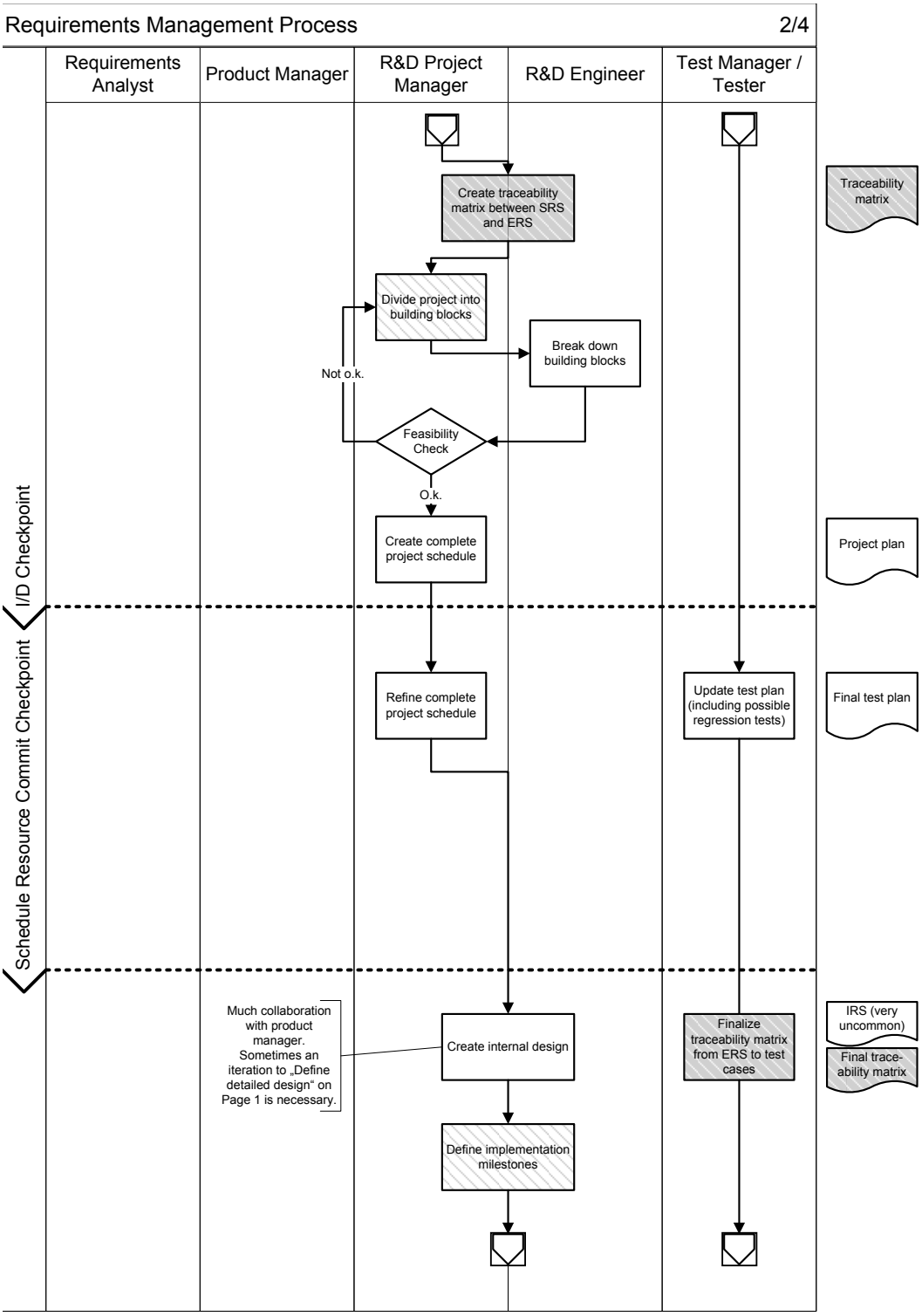
Tabelle 10: Bedingungen für HCM 3 [HW05, S. 181ff.]

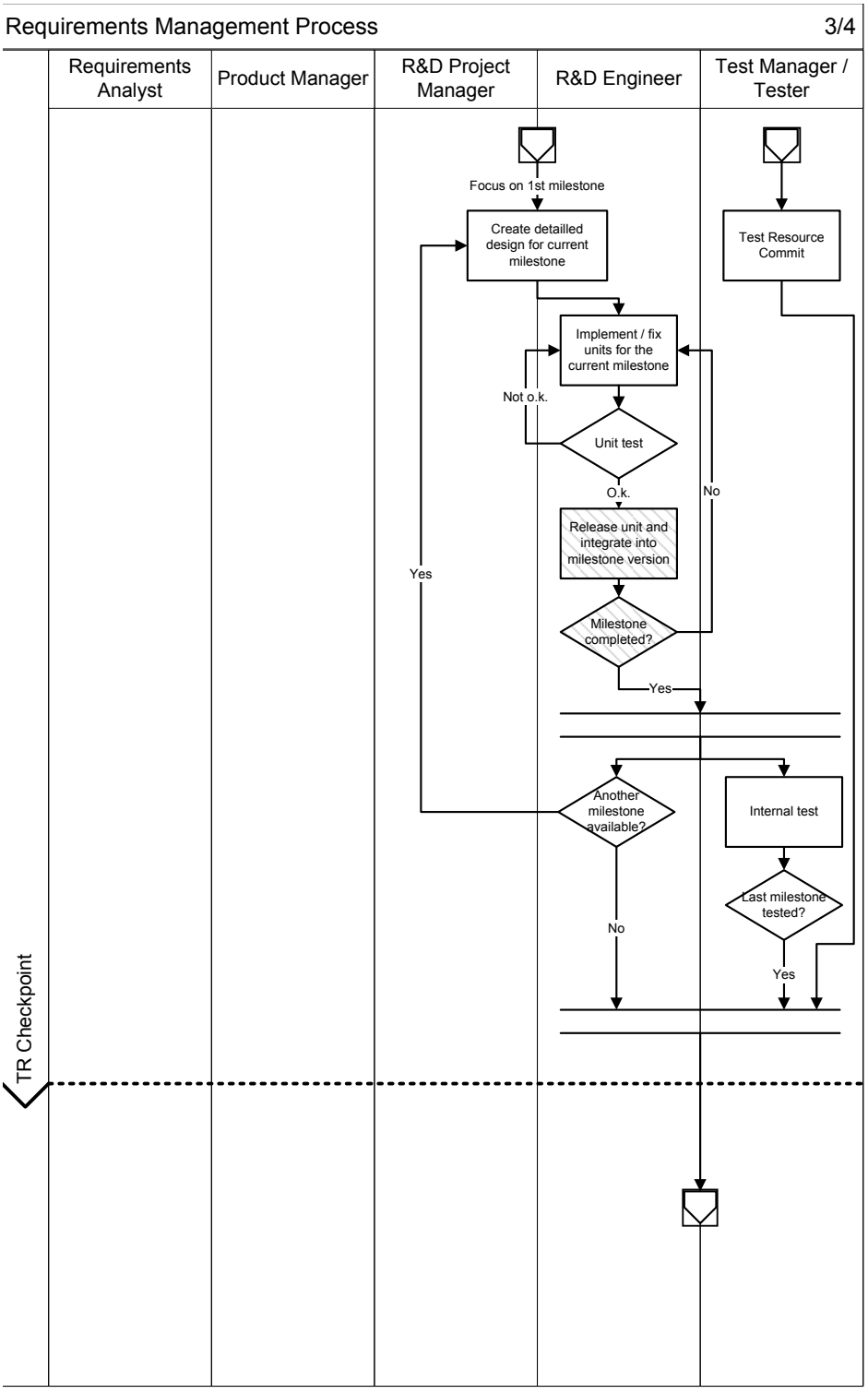
Stufe 3
Spezifizieren: • Vollständig • Rückverfolgbar • Abstraktionsebene (Lösungsvorgabe) *Analyse:* • Vollständig • Rückverfolgbar • Abstraktionsebene (Lösungsvorgabe)

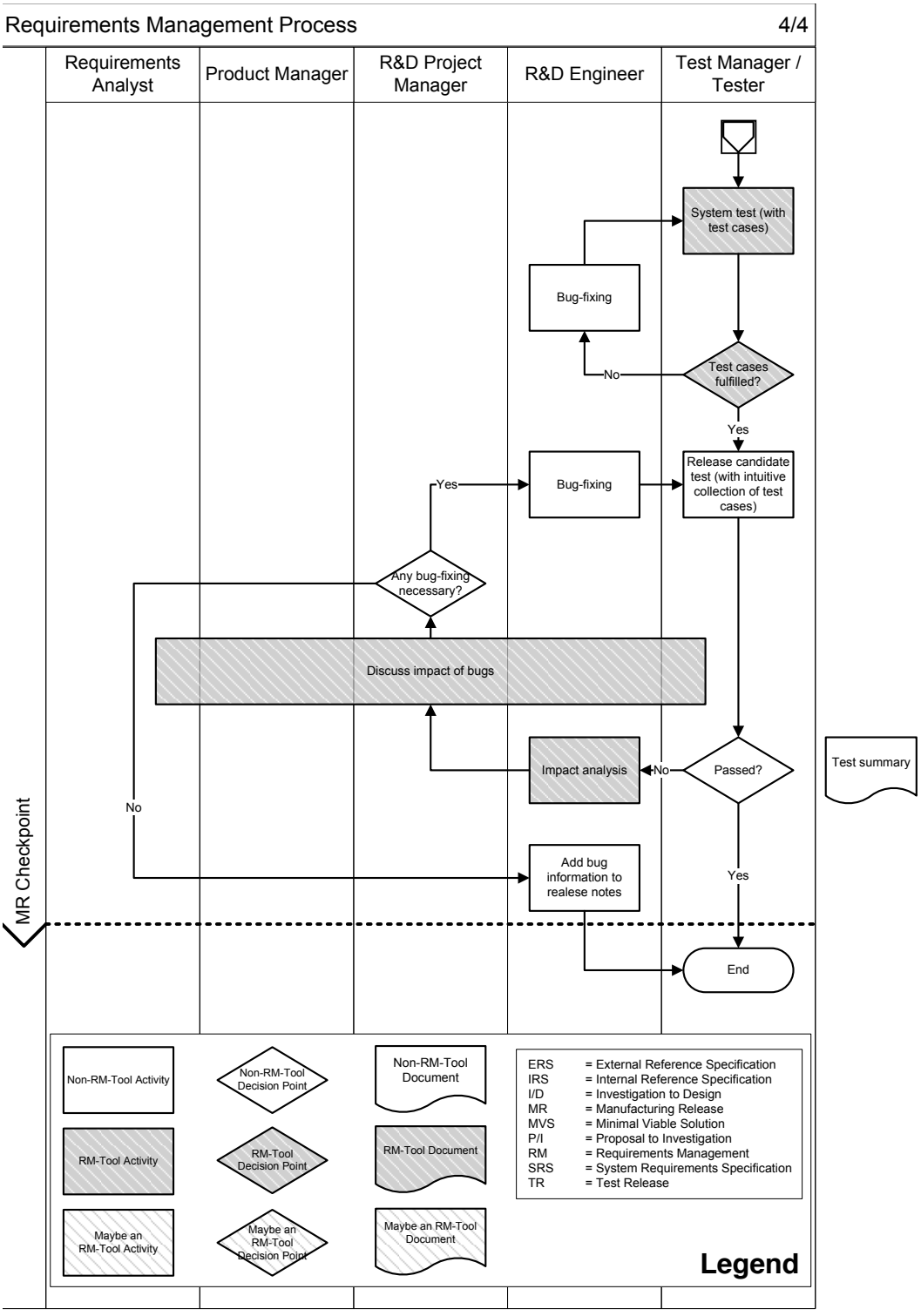
Empfehlungen für das Unternehmen Stufe 3 sollte bis zur langfristigen Etablierung der neuen Prozesse über das Pilotprojekt hinaus nicht näher betrachtet werden. Der Grund hierfür ist, wie auch schon auf Stufe 2 erwähnt, dass zuviel Neues die Ziele der Prozessoptimierung verwischen würde.

A Aktuelle Anforderungsprozesse im Unternehmen

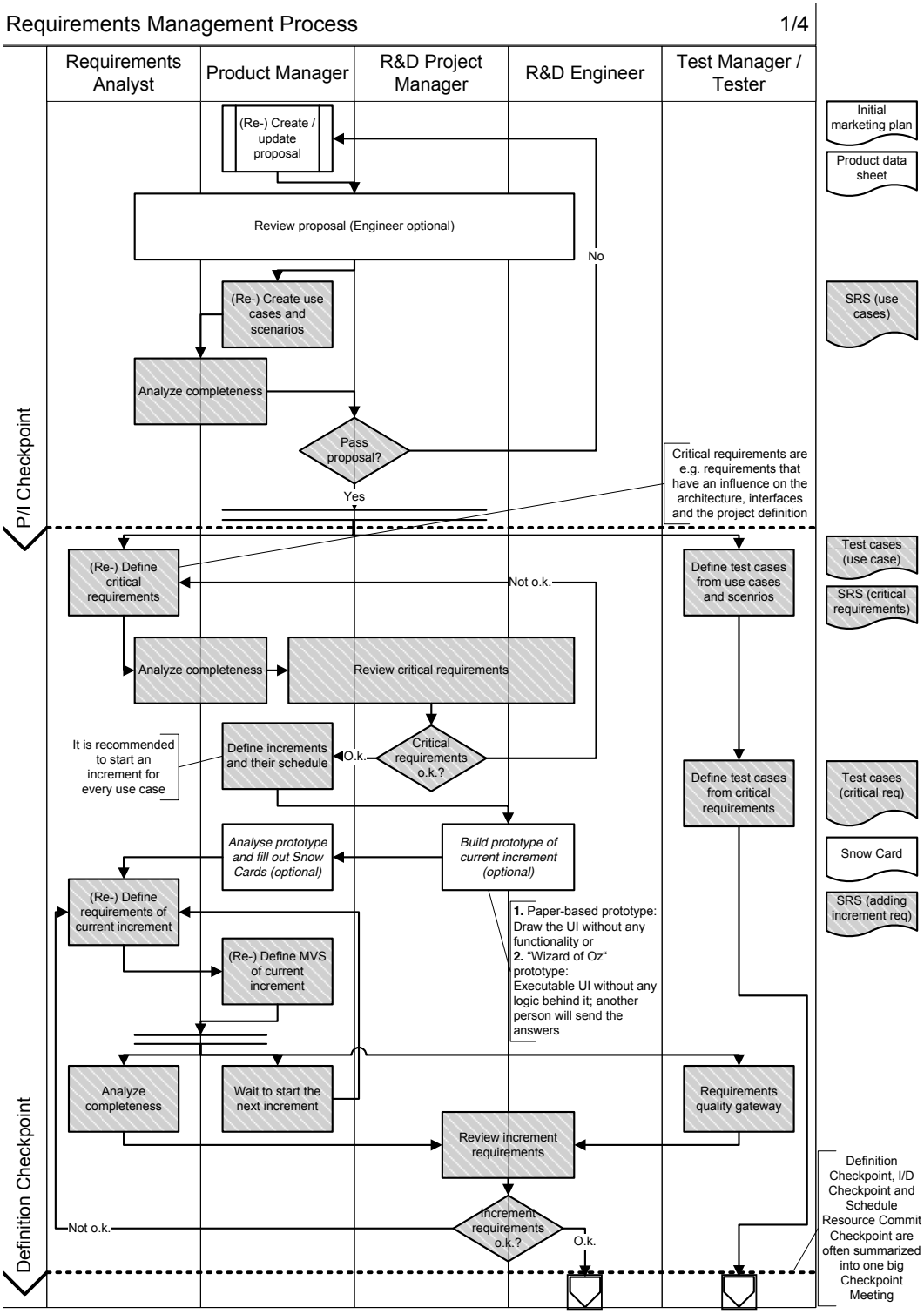


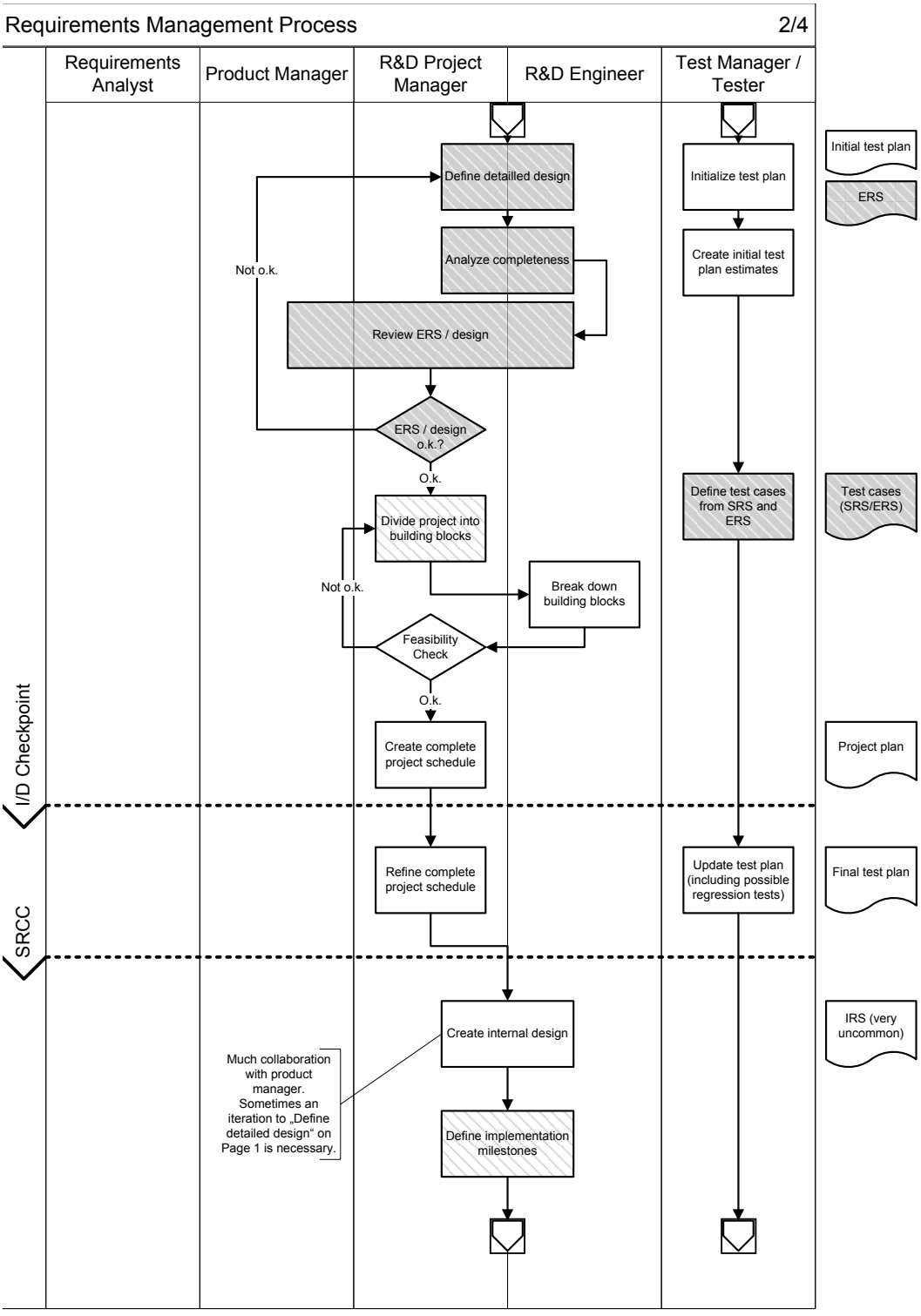

Requirements Management Process
2/4
Requirements Analyst
Product Manager
R&D Project Manager
R&D Engineer
Test Manager / Tester
I/D Checkpoint
Schedule Resource Commit Checkpoint
Create traceability matrix between SRS and ERS
Traceability matrix
Divide project into building blocks
Break down building blocks
Not o.k.
Feasibility Check
O.k.
Create complete project schedule
Project plan
Refine complete project schedule
Update test plan (including possible regression tests)
Final test plan
Much collaboration with product manager. Sometimes an iteration to „Define detailed design“ on Page 1 is necessary.
Create internal design
Finalize traceability matrix from ERS to test cases
IRS (very uncommon)
Final trace-ability matrix
Define implementation milestones

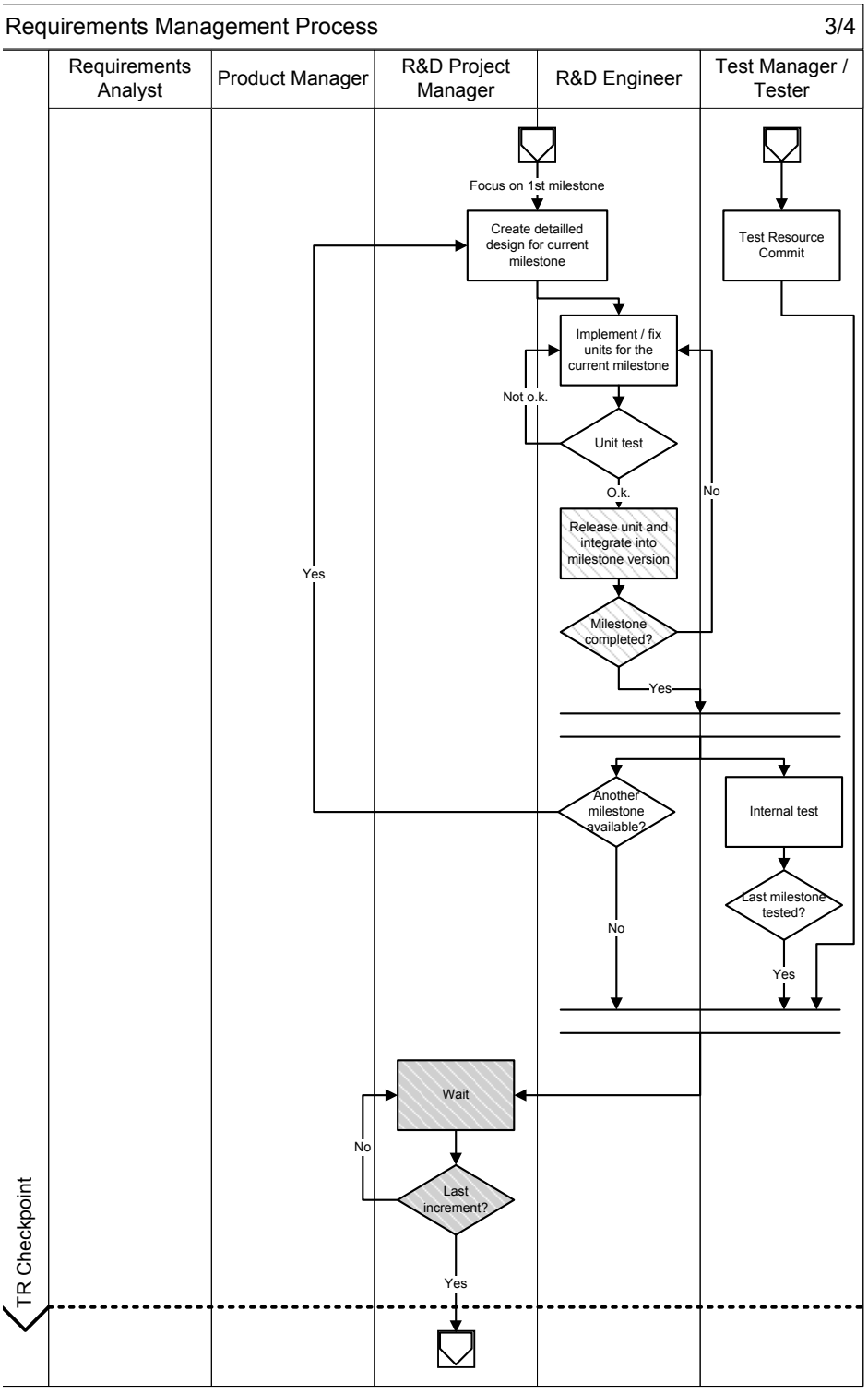

Requirements Management Process
3/4
Requirements Analyst
Product Manager
R&D Project Manager
R&D Engineer
Test Manager / Tester
Focus on 1st milestone
Create detailled design for current milestone
Test Resource Commit
Implement / fix units for the current milestone
Not o.k.
Unit test
O.k.
No
Release unit and integrate into milestone version
Yes
Milestone completed?
Yes
Another milestone available?
Internal test
Last milestone tested?
No
Yes
TR Checkpoint

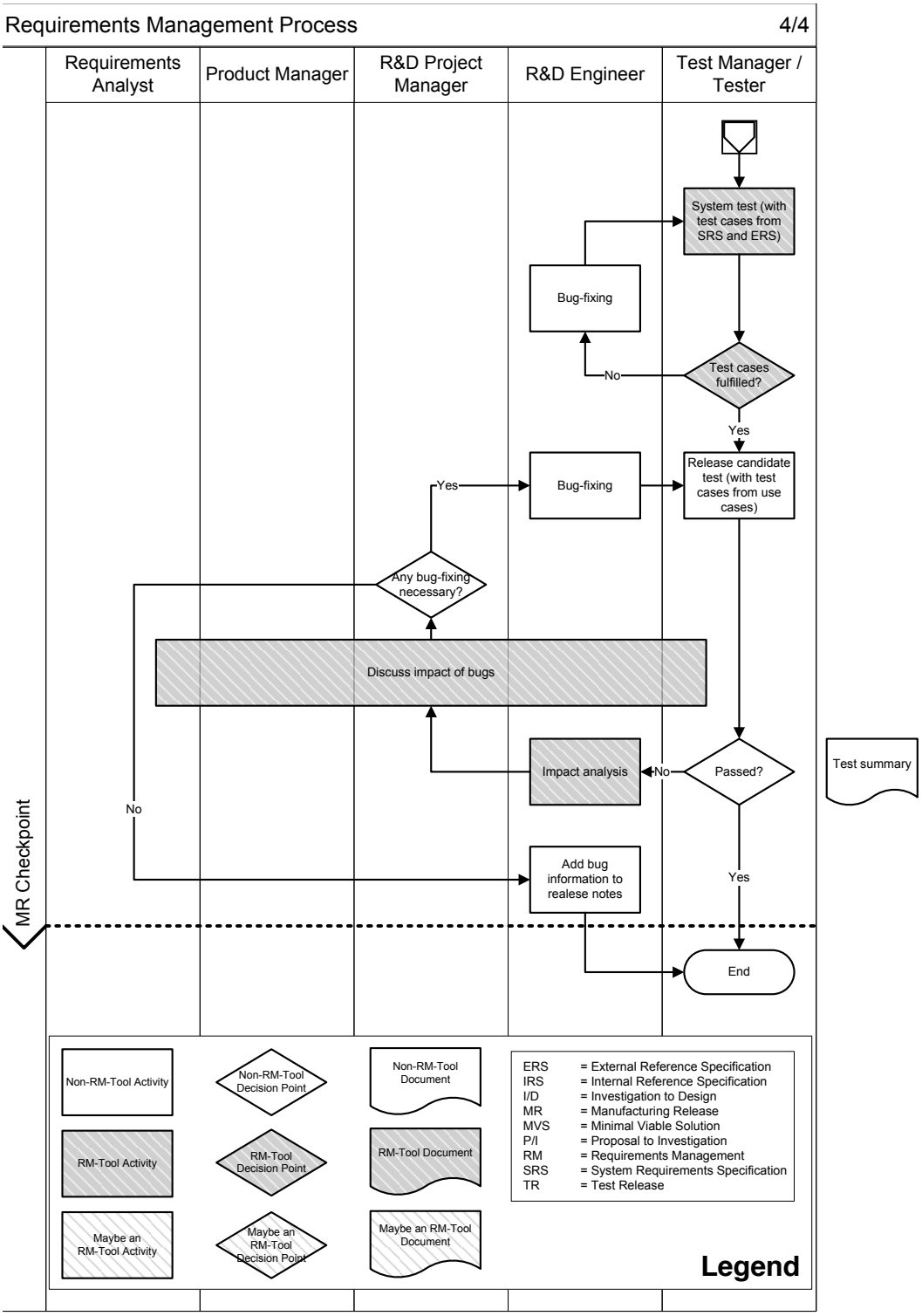

Requirements Management Process
4/4
Requirements Analyst
Product Manager
R&D Project Manager
R&D Engineer
Test Manager / Tester
System test (with test cases)
Bug-fixing
Test cases fulfilled?
No
Yes
Release candidate test (with intuitive collection of test cases)
Bug-fixing
Yes
Any bug-fixing necessary?
Discuss impact of bugs
Impact analysis
No
Passed?
Test summary
No
Add bug information to realese notes
Yes
End
MR Checkpoint
Non-RM-Tool Activity
Non-RM-Tool Decision Point
Non-RM-Tool Document
RM-Tool Activity
RM-Tool Decision Point
RM-Tool Document
Maybe an RM-Tool Activity
Maybe an RM-Tool Decision Point
Maybe an RM-Tool Document
ERS = External Reference Specification
IRS = Internal Reference Specification
I/D = Investigation to Design
MR = Manufacturing Release
MVS = Minimal Viable Solution
P/I = Proposal to Investigation
RM = Requirements Management
SRS = System Requirements Specification
TR = Test Release
Legend

B Angepasste Anforderungsprozesse im Unternehmen




Requirements Management Process
2/4
Requirements Analyst
Product Manager
R&D Project Manager
R&D Engineer
Test Manager / Tester
Define detailled design
Initialize test plan
Initial test plan
ERS
Analyze completeness
Create initial test plan estimates
Not o.k.
Review ERS / design
ERS / design o.k.?
O.k.
Define test cases from SRS and ERS
Test cases (SRS/ERS)
Divide project into building blocks
Not o.k.
Break down building blocks
Feasibility Check
O.k.
Create complete project schedule
Project plan
I/D Checkpoint
Refine complete project schedule
Update test plan (including possible regression tests)
Final test plan
SRCC
Create internal design
IRS (very uncommon)
Much collaboration with product manager. Sometimes an iteration to „Define detailed design“ on Page 1 is necessary.
Define implementation milestones


Requirements Management Process
3/4
Requirements Analyst
Product Manager
R&D Project Manager
R&D Engineer
Test Manager / Tester
Focus on 1st milestone
Create detailled design for current milestone
Test Resource Commit
Implement / fix units for the current milestone
Not o.k.
Unit test
O.k.
No
Release unit and integrate into milestone version
Yes
Milestone completed?
Yes
Another milestone available?
Internal test
Last milestone tested?
No
Yes
Wait
No
Last increment?
Yes
TR Checkpoint


Requirements Management Process
4/4
Requirements Analyst
Product Manager
R&D Project Manager
R&D Engineer
Test Manager / Tester
System test (with test cases from SRS and ERS)
Bug-fixing
Test cases fulfilled?
No
Yes
Release candidate test (with test cases from use cases)
Bug-fixing
Yes
Any bug-fixing necessary?
Discuss impact of bugs
Impact analysis
No
Passed?
Test summary
No
Add bug information to realese notes
Yes
End
MR Checkpoint
Non-RM-Tool Activity
RM-Tool Activity
Maybe an RM-Tool Activity
Non-RM-Tool Decision Point
RM-Tool Decision Point
Maybe an RM-Tool Decision Point
Non-RM-Tool Document
RM-Tool Document
Maybe an RM-Tool Document
ERS = External Reference Specification
IRS = Internal Reference Specification
I/D = Investigation to Design
MR = Manufacturing Release
MVS = Minimal Viable Solution
P/I = Proposal to Investigation
RM = Requirements Management
SRS = System Requirements Specification
TR = Test Release
Legend

C Angepasste Anforderungstypen

Requirements Types

1. Use Case / Scenario

Attribute name	Data type	Description
ID	Integer > 0	Unique identifier
Name	String	Abstract of description for a better overview
Type	Enum {Project Constraint, Use Case / Scenario, Functional Requirement, UI / Usability Requirement, FURPS Requirement, Design Requirement, Test Case Script, Ideas & Enhancements}	Useful in order to categorize by type
Description	String	Natural language statement about what should be fulfilled
External references*	anything	External specifications or other material, that led to this req
Originator	Person	Raiser of the req in first instance
Status	Enum {initial, approved, partly implemented, implemented, tested}	This attribute shows in which status the req is
Rationale*	String	Explains why the req is important
Owner	Persons	Person that is responsible for the req
Conflicts*	String	Description about a conflict between two or more reqs
Customer Satisfaction	Integer >=1 AND <= 5	How happy is the product manager if this req will be implemented
Customer Dissatisfaction	Integer >=1 AND <= 5	How unhappy is the product manager if this req won't be implemented
Priority	Enum {Must, ..., nice to have}	Quantifier, how important it is to implement this req
Validation*	String	Quantified goal the solution has to meet (this will lead to a test case)
Trace to*	Functional Req No. OR UI / Usability Req No. OR FURPS Req No.	Link that shows, how this use case will be implemented
Business Event ID*	Integer > 0	Unique identifier that shows, from which business event it comes from – normally both have the same No.

2. Functional Requirement

Attribute name	Data type	Description
ID	Integer > 0	Unique identifier
Name	String	Abstract of description for a better overview
Type	Enum {Project Constraint, Use Case / Scenario, Functional Requirement, UI / Usability Requirement, FURPS Requirement, Design Requirement, Test Case Script, Idea & Enhancement}	Useful in order to categorize by type
Description	String	Natural language statement about what should be fulfilled
External references*	anything	External specifications or other material, that led to this req
Originator	Person	Raiser of the req in first instance
Increment ID	Integer > 0	Identifier, that indicates to which increment this requirement belongs.
Status	Enum {initial, approved, partly implemented, implemented, tested}	This attribute shows in which status the req is.
Rationale*	String	Explains why the req is important
Owner	Persons	Person that is responsible for the req
Priority	Enum {Must, ..., nice to have}	Quantifier, how important it is to implement this req
Conflicts*	String	Description about a conflict between two or more reqs
Trace from*	Use Case ID	Link that shows, from which use case this req has been created
Trace to*	Test Case Script AND Design Req ID	Link that shows, which test case script tests this req and how this req will be implemented
Validation*	String	Quantified goal the solution has to meet (this will lead to a test case)

3. FURPS Requirement

Attribute name	Data type	Description
ID	Integer > 0	Unique identifier
Name	String	Abstract of description for a better overview
Type	Enum {Project Constraint, Use Case / Scenario, Functional Requirement, UI / Usability Requirement, FURPS Requirement, Design Requirement, Test Case Script, Idea & Enhancement}	Useful in order to categorize by type
Description	String	Natural language statement about what should be fulfilled
External references*	anything	External specifications or other material, that led to this req
Originator	Person	Raiser of the req in first instance
Increment ID	Integer > 0	Identifier, that indicates to which increment this requirement belongs.
Status	Enum {initial, approved, partly implemented, implemented, tested}	This attribute shows in which status the req is.
Rationale*	String	Explains why the req is important
Owner	Persons	Person that is responsible for the req
Priority	Enum {Must, ..., nice to have}	Quantifier, how important it is to implement this req
Conflicts*	String	Description about a conflict between two or more reqs
Trace from*	Use Case ID	Link that shows, from which use case this req has been created
Trace to*	Test Case Script AND Design Req ID	Link that shows, which test case script tests this req and how this req will be implemented
Validation*	String	Quantified goal the solution has to meet (this will lead to a test case)

4. Design Requirement

Attribute name	Data type	Description
ID	Integer > 0	Unique identifier
Name	String	Abstract of description for a better overview
Type	Enum {Project Constraint, Use Case / Scenario, Functional Requirement, UI / Usability Requirement, FURPS Requirement, Design Requirement, Test Case Script, Idea & Enhancement}	Useful in order to categorize by type
Description	String	Natural language statement about what should be fulfilled
External references*	anything	External specifications or other material, that led to this req
Originator	Person	Raiser of the req in first instance
Increment ID	Integer > 0	Identifier, that indicates to which increment this requirement belongs.
Milestone ID	Integer > 0	Identifier, that indicates to which milestone this requirement belongs.
Status	Enum {initial, approved, partly implemented, implemented, tested}	This attribute shows in which status the req is.
Rationale*	String	Explains why the req is important
Owner	Persons	Person that is responsible for the req
Priority	Enum {Must, ..., nice to have}	Quantifier, how important it is to implement this req
Conflicts*	String	Description about a conflict between two or more reqs
Trace from*	Functional Req OR FURPS Req ID	Link that shows, from which req dthis esign req has been created
Trace to*	Test Case Script ID	Link that shows, which test case script tests this req
Validation*	String	Quantified goal the solution has to meet (this will lead to a test case)

5. Test Case Script

Attribute name	Data type	Description
ID	Integer > 0	Unique identifier
Name	String	Abstract of description for a better overview
Type	Enum {Project Constraint, Use Case / Scenario, Functional Requirement, UI / Usability Requirement, FURPS Requirement, Design Requirement, Test Case Script, Idea & Enhancement}	Useful in order to categorize by type
Description*	String	Natural language statement about what should be fulfilled
External references*	anything	External specifications or other material, that led to this req
Originator	Person	Raiser of the req in first instance
Status	Enum {initial, approved, partly implemented, implemented, executed}	This attribute shows in which status the req is.
Owner	Persons	Person that is responsible for the req
Priority	Enum {Must, ..., nice to have}	Quantifier, how important it is to implement this req
Conflicts*	String	Description about a conflict between two or more reqs
Trace from*	Functional Req ID OR FURPS Req ID OR Design Req ID	Link that shows, from which req this test case script has been created

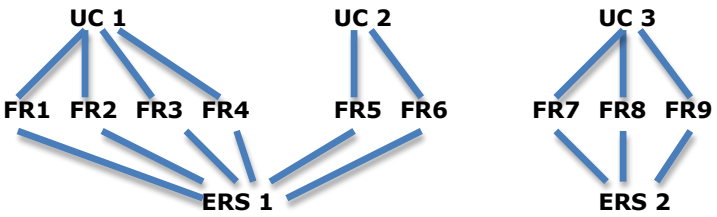
6. Idea & Enhancement

Attribute name	Data type	Description
ID	Integer > 0	Unique identifier
Name	String	Abstract of description for a better overview
Type	Enum {Project Constraint, Use Case / Scenario, Functional Requirement, UI / Usability Requirement, FURPS Requirement, Design Requirement, Test Case Script, Idea & Enhancement}	Useful in order to categorize by type
Description	String	Natural language statement about what should be fulfilled
External references*	anything	External specifications or other material, that led to this req
Originator	Person	Raiser of the req in first instance
Rationale*	String	Explains why the req is important
Customer Satisfaction	Integer >=1 AND <= 5	How happy is the product manager if this req will be implemented
Customer Dissatisfaction	Integer >=1 AND <= 5	How unhappy is the product manager if this req won't be implemented

Legend:

*	Optional attribute
[grey background]	Automatically generated by CaliberRM
[bold font]	**Attribute, that is used by every requirement type**
[normal font]	Attribute, that is not used by every requirement type

Possible structure of requirements:



D Berechnungshilfe für Kennzahlen

Metrics Sheet

	A	B
1	**Benchmark Metrics:**	
2	**rate of changes:**	=B3/B4
3	number of changes:	
4	number of requirements:	
5		
6	**error density:**	=B7/B8
7	*(for paper based requirements)*	
8	number of requirement defects:	
9	number of specification pages:	
10		
11	**rate of errors:**	=B12/B13
12	(for requirements in a database)	
13	number of requirement defects:	
14	number of requirements:	
15		
16	**proportional cost* minimization for traceability:**	=(B17-B18)/B17
17	cost* for creating a traceability matrix manually:	
18	accumulated costs* for setting up links for every requirement:	
19		
20	**cost can be replaced by time*	
21		
22	**rate of traceable requirements:**	=B23/B24
23	number of traceable requirements*:	
24	number of requirements:	
25		
26	**requirements are traceable if no trace exists, that is not displayed in the matrix*	
27		
28	**Success Indicator Metrics:**	
29	**rate of changes of a phase*:**	=B30/B31
30	number of changes in current phase*:	
31	number of current requirements:	
32		
33	**time between two adjacent checkpoints*	
34		
35		
36	**rate of status no. 1 requirements:**	=B37/(B37+B39+B41+B43)
37	number of status no. 1 requirements:	
38	**rate of status no. 2 requirements:**	=B39/(B37+B39+B41+B43)
39	number of status no. 2 requirements:	
40	**rate of status no. 3 requirements:**	=B41/(B37+B39+B41+B43)
41	number of status no. 3 requirements:	
42	**rate of status no. 4 requirements:**	=B43/(B37+B39+B41+B43)
43	number of status no. 4 requirements:	
44		
45	**General notes:**	
46	*For every metric you can take a random sample, if it is too expensive to count everything.*	
47	*Input Data*	*Output Date*

Literatur

[AK09] Aiken, Carolin ; Keller, Scott: The Irrational Side of Change Management. In: *The McKinsey Quarterly* (2009), April, Nr. 2, 101-109. `http://www.ihrim.org/Pubonline/Wire/May09/ChangeMgt.pdf`. – Besucht am 10.03.2010

[AM08] Abts, Dietmaer ; Mülder, Wilhelm: *Grundkurs Wirtschaftsinformatik: Eine kompakte und praxisorientierte Einführung*. 6. Aufl. Vieweg+Teubner Verlag, 2008

[BBB+01] Beck, Kent ; Beedle, Mike ; Bennekum, Arie van ; Cockburn, Alistair ; Cunningham, Ward ; Fowler, Martin ; Grenning, James ; Highsmith, Jim ; Hunt, Andrew ; Jeffries, Ron ; Kern, Jon ; Marick, Brian ; Martin, Robert C. ; Mellor, Steve ; Schwaber, Ken ; Sutherland, Jeff ; Thomas, Dave: *Principles behind the Agile Manifesto*. http://agilemanifesto.org/principles.html, 2001

[BD09] Bortz, Jürgen ; Döring, Nicola: *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler*. 4., überarb. Aufl., Nachdr. 2009. Heidelberg : Springer Medizin Verlag, 2009 (Springer-Lehrbuch). – ISBN 978–3–540–33305–0

[Bec00] Beck, Kent: *Extreme Programming Explained: Embrace Change*. Reading, Mass. [u.a.] : Addison-Wesley, 2000. – ISBN 0–201–61641–6

[Boe81] Boehm, Barry W.: *Software Engineering Economics*. Englewood Cliffs, NJ : Prentice-Hall, 1981 (Prentice-Hall advances in computing science and technology series). – ISBN 0–13–822122–7

[Bra07] Braun, Hans-Peter: *Facility Management: Erfolg in der Immobilienbewirtschaftung*. 5., neu bearb. Aufl. Berlin : Springer, 2007 `http://swbplus.bsz-bw.de/bsz261762540cov.htm`. – ISBN 3–540–34701–1 ; 978–3–540–34701–9

[CKS07] Chrissis, Mary B. ; Konrad, Mike ; Shrum, Sandy: *CMMI: Guidelines for Process Integration and Product Improvement*. 2. ed. Harlow [u.a.] : Addison-Wesley, 2007 (SEI series in software engineering). `http://swbplus.bsz-bw.de/bsz258335394inh.htm`. – ISBN 0–321–27967–0. – CMMI for development, Version 1.2

[DeM79] DeMarco, Tom: *Structured Analysis and System Specification*. Englewood Cliffs, N.J. : Prentice-Hall, 1979 (Prentice-Hall software series). – ISBN 0–13–854380–1

[Dem00] Deming, William E.: *Out of the Crisis*. 1. MIT Press ed. Cambridge, Mass. [u.a.] : MIT Press, 2000. – ISBN 0–262–54115–7

[Drö00] Dröschel, Wolfgang (Hrsg.): *Das V-Modell 97: Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz*. München : Oldenbourg, 2000. – ISBN 3–486–25086–8

[Ebe08] Ebert, Christof: *Systematisches Requirements-Engineering und Management: Anforderungen ermitteln, spezifizieren, analysieren und verwalten*. 2., aktualisierte und erw. Aufl. Heidelberg : dpunkt-Verlag, 2008 http://deposit.d-nb.de/cgi-bin/dokserv?id=3079595&prov=M&dok_var=1&dok_ext=htm. – ISBN 978–3–89864–546–1 ; 3–89864–546–0

[Hau08] Haughey, Duncan: *Why Software Projects Fail and How to Make Them Succeed*. http://www.projectsmart.co.uk/pdf/why-software-projects-fail.pdf. Version: 2008. – Besucht am 10.03.2010

[HFPW08] Hood, Colin (Hrsg.) ; Fichtinger, Stefan (Hrsg.) ; Pautz, Urte (Hrsg.) ; Wiedemann, Simon (Hrsg.): *Requirements Management: The Interface Between Requirements Development and All Other Systems Engineering Processes*. Berlin, Heidelberg : Springer-Verlag Berlin Heidelberg, 2008 http://dx.doi.org/10.1007/978-3-540-68476-3. – ISBN 978–3–540–68476–3. – In: Springer-Online

[HHMS04] Heßeler, Alexander ; Hood, Colin ; Missling, Christian ; Stücka, Renate ; Versteegen, Gerhard (Hrsg.): *Anforderungsmanagement: formale Prozesse, Praxiserfahrungen, Einführungsstrategien und Toolauswahl*. Berlin : Springer, 2004 (Xpert.press). http://swbplus.bsz-bw.de/bsz104174110cov.htm. – ISBN 3–540–00963–9

[HW05] Hood, Colin ; Wiebel, Rupert: *Optimieren von Requirements Management & Engineering: Mit dem HOOD Capability Model*. Berlin : Springer, 2005 (Xpert.press). http://swbplus.bsz-bw.de/bsz116206578cov.htm. – ISBN 3–540–21178–0

[Hör06] Hörmann, Klaus (Hrsg.): *SPICE in der Praxis: Interpretationshilfe für Anwender und Assessoren; basierend auf ISO/IEC 15504 (Stand 2006)*. 1. Aufl. Heidelberg : dpunkt-Verlag, 2006. – ISBN 3–89864–341–7

[IEE90] IEEE Standard Glossary of Software Engineering Terminology. In: *IEEE Std 610.12-1990* (1990), S. 1–84

[JP93] Jarke, M. ; Pohl, K.: Establishing Visions in Context: Towards a Model of Requirements Processes. In: *Proceedings of the 14th International Conference on Information Systems* (1993), S. 23 – 24

[KKC00] Kazman, Rick ; Klein, Mark ; Clements, Paul: *ATAM: Method for Architecture Evaluation*. Carnegie Mellon, Software Engineering Institute, 2000 http://www.sei.cmu.edu/library/abstracts/reports/00tr004.cfm. – Besucht am 10.03.2010

[Kov99] Kovitz, Benjamin L.: *Practical Software Requirements: A Manual of Content and Style*. Greenwich : Manning, 1999. – ISBN 1–884777–59–7

[Kru07] Kruchten, Philippe: *The Rational Unified Process: An Introduction*. 3. ed., 6. pr. Upper Saddle River, NJ [u.a.] : Addison-Wesley, 2007 (Addison-Wesley object technology series). – ISBN 0–321–19770–4

[KS00] Kotonya, Gerald ; Sommerville, Ian: *Requirements Engineering: Processes and Techniques*. Repr. Chichester [u.a.] : Wiley, 2000 (Worldwide series in computer science). – ISBN 0–471–97208–8. – Includes bibliographical references and index

[Law98] Lawrence, B.: Designers Must Do the Modeling. In: *IEEE Software* 15 (1998), March/April, Nr. 2, S. 31–33

[LB08] Lass, Sander ; Brockmann, Carsten: *Geschäftsprozessmanagement: Methoden der Ist-Aufnahme und Projektplanung*. `http://wi.uni-potsdam.de/hp.nsf/0/5A269FE667BAC41EC12574EC004954CE/$File/GPM_Übung_Ist-AufnahmePlanung.pdf`. Version: 2008. – Besucht am 10.03.2010

[LW00] Leffingwell, Dean ; Widrig, Don: *Managing Software Requirements: A Unified Approach*. Reading, Mass. [u.a.] : Addison-Wesley, 2000. – ISBN 0–201–61593–2

[Mey85] Meyer, B.: On Formalism in Specifications. In: *IEEE Software* 2 (1985), January, Nr. 1, S. 6–26. – ISSN 0740–7459

[MP84] McMenamin, Stephen M. ; Palmer, John F.: *Essential Systems Analysis*. Englewood Cliffs, N.J. : Yourdon Pr., 1984 (Yourdon computing series). – ISBN 0–13–287905–0

[OM08] O'Brien, James A. ; Marakas, George M.: *Management Information System: Glossary*. `http://highered.mcgraw-hill.com/sites/0073511544/student_view0/chapter4/glossary.html`. Version: 2008. – Besucht am 10.03.2010

[Pep04] Pepels, Werner: *Marketing: Lehr- und Handbuch*. 4., völlig überarb. und erw. Aufl. München : Oldenbourg, 2004. – ISBN 3–486–27538–0

[Poh07] Pohl, Klaus: *Requirements Engineering: Grundlagen, Prinzipien, Techniken*. 1. Aufl. Heidelberg : dpunkt-Verlag, 2007 `http://www.gbv.de/dms/ilmenau/toc/515698989.PDF`. – ISBN 3–89864–342–5978–3–89864–342–9

[Pre06] Pressebox: *Telelogic DOORS von Yphise zum dritten Mal in Folge als bestes Anforderungsmanagement-Produkt ausgezeichnet*. `http://www.pressebox.de/pressemeldungen/telelogic-deutschland-gmbh/boxid-80711.html`. Version: October 2006. – Besucht am 10.03.2010

[Rei03] Reilly, John: *IBM Completes Acquisition of Rational Software*. `http://www-03.ibm.com/press/us/en/pressrelease/314.wss`. Version: February 2003. – Besucht am 10.03.2010

[RR06] Robertson, Suzanne ; Robertson, James: *Mastering Tte Requirements Process*. 2. ed. Upper Saddle River,NJ : Addison-Wesley, 2006 http://swbplus.bsz-bw.de/bsz253400317inh.pdf. – ISBN 0–321–41949–9

[Rup09] Rupp, Chris (Hrsg.): *Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis*. 5., aktualisierte und erw. Aufl. München : Hanser, 2009. – ISBN 978–3–446–41841–7 ; 3–446–41841–5

[SB02] Schwaber, Ken ; Beedle, Mike: *Agile Software Development with Scrum*. Upper Saddle River, NJ : Prentice Hall, 2002 (Series in agile software development). – ISBN 0–13–067634–9

[Sch97] Schein, Edgar H.: *Organizational Culture and Leadership*. 2. ed., 1. paperback ed. San Francisco : Jossey-Bass, 1997 (The Jossey-Bass business & management series; The Jossey-Bass psychology series). – ISBN 1–55542–487–2

[SG09] Standish-Group: *CHAOS Summary 2009*. http://www.standishgroup.com/newsroom/chaos_2009.php. Version: April 2009. – Besucht am 10.03.2010

[SKT+92] Sheldon, F.T. ; Kavi, K.M. ; Tausworthe, R.C. ; Yu, J.T. ; Brettschneider, R. ; Everett, W.W.: Reliability Measurement: From Theory to Practice. In: *IEEE Software* 9 (1992), July, Nr. 4, 13-20. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=143095&isnumber=3837

[SM95] Stevens, Richard ; Martin, James: What is Requirements Management? In: *Proceedings of the Fifth Annual International Symposium of the NCOSE* 2 (1995), S. 13–18

[SS97] Sommerville, Ian ; Sawyer, Pete: *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, 1997

[Sta09] Stafford, Jan: *Why Software Projects Fail and More Will Fail in 2009*. http://itknowledgeexchange.techtarget.com/software-quality/why-software-projects-fail-and-more-will-fail-in-2009/. Version: January 2009. – Besucht am 10.03.2010

[Sut02] Sutcliffe, Alistair: *User Centred Requirements Engineering*. 1. publ. London : Springer, 2002. – ISBN 1–85233–517–3

[VSH01] Versteegen, Gerhard ; Salomon, Knut ; Heinold, Rainer: *Change-Management bei Softwareprojekten*. Berlin : Springer, 2001 (Xpert.press). http://swbplus.bsz-bw.de/bsz09154226xcov.htm. – ISBN 3–540–67809–3

[Wie03] Wiegers, Karl E.: *Software Requirements: Practical Techniques for Gathering and Managing Requirements throughout the Product Development Cycle*. 2. ed. Redmond, WA : Microsoft Press, 2003 http://www.gbv.de/dms/ilmenau/toc/358899605.PDF. – ISBN 0–7356–1879–8 ; 978–0–7356–1879–4

[Yph02] Yphise: *Requirements Management Tools*. 2002