Certified Professional for Requirements Engineering

Handbuch

Requirements Management

Practitioner | Specialist

Stan Bühne

Andrea Herrmann

2.1.0 | 5. April 2024

IREB

Nutzungsbedingungen

1. Jede Einzelperson und alle Seminaranbieter dürfen dieses Handbuch als Grundlage für Seminare verwenden, sofern die Inhaber der Urheberrechte als Quelle und Besitzer des Urheberrechts anerkannt und benannt werden. Des Weiteren darf dieses Handbuch zu Werbungszwecken nur mit Einwilligung des IREB e.V. verwendet werden.

2. Jede Einzelperson oder Gruppe von Einzelpersonen darf dieses Handbuch als Grundlage für Artikel, Bücher oder andere abgeleitete Veröffentlichungen verwenden, sofern die Autoren und IREB e.V. als Quelle und Besitzer des Urheberrechts genannt werden.

Gendergerechte Formulierungen

Wir haben in diesem Dokument bewusst auf geschlechtsspezifische Formulierungen verzichtet.

Selbstverständlich unterstützen wir als IREB geschlechtssensible Formulierungen. Wir sehen aber auch die Notwendigkeit, komplexe Sachverhalte so zu formulieren, dass sie leicht verstanden werden können.

Texte, die eigentlich eine männliche und weibliche Form fordern, wären weniger gut lesbar und damit schwieriger zu verstehen. Ziel dieses Dokuments ist es jedoch, Inhalte präzise und klar darzustellen und zu vermitteln. Da wir dem Leser helfen wollen, den Fokus auf den Inhalt zu richten, verwenden wir in diesem Dokument bewusst nur die männliche Form von Personen.

Dies soll nicht als Ausdruck von mangelndem Respekt verstanden werden.

Danksagung

Wir danken dem IREB e.V. für die Gelegenheit, dieses Buch zu schreiben sowie für die durchgängige organisatorische Unterstützung. Die IREB-Arbeitsgruppe „Requirements Management" hat maßgeblich zur Erstellung der Inhalte beigetragen, da der von der Arbeitsgruppe erstellte Lehrplan das Grundgerüst dieses Buchs bildet und seine Mitglieder uns mit Kommentaren, Anmerkungen und Zusatzmaterial zur Seite standen. Aus diesem Grunde bedanken wir uns herzlich bei der gesamten Arbeitsgruppe (in alphabetischer Reihenfolge): Frank Engel, Sven Eselgrimm, Günter Halmans, Frank Houdek, Patrick Mäder, Alexander Rachmann, Thomas Schölzl, Amin Soesanto, Frank Stöckel und Malik Tayeh. Ein besonderer Dank geht an Prof. Dr. Martin Glinz und Dr. Thorsten Weyer, die diesem Werk durch ihre gründliche Begutachtung und ihre konstruktiven Kommentare seinen letzten Schliff gegeben haben.

Dieses Handbuch wurde erstellt von (in alphabetischer Reihenfolge):

Stan Bühne, Dr. Andrea Herrmann.

Inhaltsverzeichnis

Das IREB CPRE Modul Requirements Management

Wer als Requirements Engineer, Business Analyst, Berater, Demand Manager oder Projektleiter in System- und Softwareentwicklungsprojekten arbeitet, weiß, dass es für die erfolgreiche Umsetzung eines Projektes noch lange *nicht* damit getan ist, seine Stakeholder zu kennen und deren abgestimmte Anforderungen anfänglich zu dokumentieren!

Nein, selbst falls alle Anforderungen zu Projektbeginn gut strukturiert, abgestimmt und abgenommen waren, werden sie sich bis zum Projektende bzw. „Go Live" ändern – und zwar immer zu den ungünstigsten Zeitpunkten. Und auch während des Betriebs ändern sich die Anforderungen und sollten zu Dokumentationszwecken bis zur Außerbetriebnahme des Systems aktuell gehalten werden.

Dies liegt aber weder am Requirements Engineer bzw. an schlecht ausgewählten Methoden noch an den involvierten Stakeholdern – sondern meist einfach an der Natur der Dinge und sich im Laufe der Zeit verändernder Randbedingungen.

Um in solchen Momenten ein unkontrolliertes „Fire-Fighting" zu vermeiden und damit Sie als Requirements Manager jederzeit über den Stand der Anforderungen oder über die Auswirkung etwaiger Änderungswünsche auskunftsfähig sind, ist das **Requirements Management (RM)** unabdingbar, je komplexer das Projekt, umso mehr.

Zum RM gehören auf der einen Seite der Medaille die bewusste Verwaltung von Anforderungen im klassischen Sinne (z.B. mittels Attributierung, Sichtenbildung, Verfolgbarkeit, etc.) sowie das Änderungsmanagement von Anforderungen. Auf der anderen Seite gehört aber auch die vorherige Planung und Überwachung der definierten **Requirements Engineering (RE)** Prozesse – im Sinne von: „Wie ermittele, dokumentiere und überprüfe ich meine Anforderungen, um kontinuierlich über den Status berichten zu können und um auf Änderungen geplant reagieren zu können?" – zum RM.

In diesem Handbuch für das IREB CPRE Modul Requirements Management – Practitioner – und – Specialist – möchten wir das Requirements Management aus diesem Grund von beiden Seiten betrachten. Dazu stellen wir die wesentlichen Konzepte des Requirements Management vor, beschreiben aber immer auch den notwendigen planerischen Aspekt, um eine bewusste Verwaltung von Anforderungen zu ermöglichen.

Zur bewussten Verwaltung von Anforderungen (RM) muss der Requirements Manager bereits zu Beginn des RE-Prozesses folgendes planen und festlegen:

- Welche Anforderungsarten zu berücksichtigen sind, in welcher Darstellungsform und in welcher Detaillierung diese Anforderungen spezifiziert werden müssen
- Welche Fragen er auf Basis seiner Anforderungen beantworten muss und welche Sichten für die unterschiedlichen Stakeholder notwendig sind
- Durch welche Kriterien Anforderungen bewertet werden, um eine Priorisierung unterstützen zu können
- Wie Anforderungen und Anforderungs-Dokumente zu versionieren sind
- Wie und zu welchem Zeitpunkt mit Änderungen umgegangen werden soll
- Zwischen welchen Anforderungen und weiteren Entwicklungs-Artefakten Verfolgbarkeit geschaffen werden muss

- Ob und wie Anforderungsvarianten innerhalb der Anforderungs-Spezifikation zu dokumentieren sind
- Welche Berichte über den Anforderungsstand gefordert sind, welche Informationen diese zu enthalten haben und über welche Quellen (z.B. aus der Dokumentation der Attribute) diese Informationen ermittelt werden können
- Wie der exakte RE-Prozess (bzw. Ablauf von Aktivitäten) für das Projekt auszusehen hat, und wie der Prozess überwacht und evtl. verbessert werden kann

Das Ergebnis der obigen Überlegungen wird in einem **Requirements Management Plan{** XE "Requirements-Management-Plan" **} (RMP)** dokumentiert. Mit dessen Hilfe können sowohl der RE-Prozess als Ganzes als auch das Berichtswesen, Priorisierungen und Änderungen (als Teil des Requirements Management) geplant und strukturiert ablaufen.

Das Requirements Management wird durch den Requirements Engineer**{** XE "Requirements Engineer" **}** geplant und durchgeführt oder durch eine separate Rolle eines **Requirements Managers{ XE "Requirements Manager" }**. Der Requirements Manager plant, verwaltet und überwacht im Rahmen des RM den RE-Prozess mit seinen Artefakten, berichtet z.B. an den Auftraggeber bzw. Projektmanager. Der Requirements Manager koordiniert auch Änderungen.

Der RMP sorgt dafür, dass der RE-Prozess aktiv überwacht werden kann und spätere Entscheidungen bewusst und nachvollziehbar getroffen werden können. Das heißt nicht, dass das RE kein iterativ-inkrementeller und kreativer Prozess ist – aber der RE-Prozess sollte kreativ und bewusst geplant sein und nicht chaotisch ablaufen!

Dieses Handbuch unterstützt den Requirements Manager bei der Erstellung des RMPs, indem es die dafür nötigen Konzepte und Begriffe erklärt und passende Methoden vorstellt.

Zusätzlich zeigt dieses Buch, wie RE und RM in agilen Projekten umgesetzt werden kann, denn auch in agilen Projekten werden Anforderungen dokumentiert (z.B. User Stories), und auch agile Projekte müssen mit Änderungen, Priorisierungen etc. umgehen können.

In der Praxis ist ein gezieltes RM in komplexen Projekten ohne den Einsatz von Werkzeugen nur schwer vorstellbar. Daher beschreiben wir im letzten Kapitel Möglichkeiten der Unterstützung durch – sowie Grenzen von – RM-Werkzeugen.

Mit diesen Themen vermittelt das Handbuch zum Modul Requirements Manangement – Practitioner – und – Spezialist – des IREB Certified Professional for Requirements Engineering (CPRE) das Rüstzeug, um Anforderungen bewusst und strukturiert zu verwalten.

Viel Spaß beim Lesen!

Weitere Informationen zum IREB Certified Professional for Requirements Engineering Modul Requirements Management – Practitioner – und – Specialist – finden Sie unter: http://www.ireb.org

Vorwort

Das vorliegende *Handbuch Requirements Management nach IREB Standard* ergänzt den Lehrplan des International Requirements Engineering Boards für das Modul Requirements

Management Practitioner und Specialist Version 2.0.00 vom 01.07.2022 und orientiert sich am IREB-Glossar [Glin2014].

Das Handbuch richtet sich sowohl an Trainingsanbieter, die Seminare zum Requirements Management Practitioner und/oder Specialist nach dem IREB-Standard anbieten wollen, als auch an Trainingsteilnehmer und interessierte Praktiker, die einen detaillierten Einblick in den Lehrstoff dieses Moduls und in das Requirements Management nach IREB Standard erhalten möchten.

Dieses Handbuch ist **kein** Ersatz für Schulungen oder Lehrbücher zu diesem Thema. Es stellt vielmehr ein Bindeglied zwischen dem komprimiert gehaltenen Lehrplan (der lediglich die Lernziele des Moduls nennt und erläutert) und der Vielzahl an Literatur dar, die zum Thema Requirements Management in den letzten Jahrzehnten entstanden ist.

Die in diesem Handbuch beschriebenen Inhalte, zusammen mit den Hinweisen auf vertiefende Literatur, sollen Trainingsanbieter dabei unterstützen, die Trainingsteilnehmer fokussiert auf die Zertifizierungsprüfung vorzubereiten. Trainingsteilnehmern und interessierten Praktikern bietet dieses Handbuch die Möglichkeit, die Kenntnisse im Bereich des Requirements Management zu vertiefen und die detaillierten Inhalte auf Basis der Literaturempfehlungen aufzuarbeiten. Darüber hinaus soll dieses Handbuch auch als Referenz dienen, um z.B. nach erfolgreicher Zertifizierung das Wissen über die verschiedenen Themenbereiche des Requirements Managements auffrischen zu können.

Neben den zum Lehrplan vertiefenden und prüfungsrelevanten Inhalten, bietet das Handbuch in jedem Kapitel erläuternde Beispiele anhand eines durchgängigen Fallbeispiels. Das Fallbeispiel ist mit dem links stehenden Randsymbol gekennzeichnet. Diese Inhalte sind nicht direkt prüfungsrelevant, aber zum besseren Verständnis des Inhaltes durchaus zu empfehlen.

Darüber hinaus bieten wir dem interessierten Leser über die Prüfung hinausgehende Informationen, welche nicht prüfungsrelevant sind. Sofern diese Inhalte in den Schreib- und Lesefluss passen, wurden diese in das jeweilige Kapitel integriert und mit einer roten Seitenmarkierung (siehe rechts) als nicht prüfungsrelevant gekennzeichnet.

Die ergänzenden Inhalte in Anhang A bis C sind ebenfalls nicht prüfungsrelevant.

Verbesserungs- und Korrekturvorschläge sind jederzeit herzlich willkommen!

E-Mail-Kontakt: info@ireb.org

Wir wünschen Ihnen viel Spaß beim Studium dieses Handbuchs und viel Erfolg bei der Zertifizierungsprüfung zum IREB Certified Professional for Requirements Engineering Modul Requirements Management – Practitioner – oder – Specialist.

Stan Bühne

Andrea Herrmann

Herbst 2015

Versionshistorie

Version	Datum	Kommentar	Autor
1.1.0	1. September 2019	Beseitigung von Inkonsistenzen gegenüber dem Lehrplan. Fehlerbehebungen im Zusammenhang mit der Übersetzung ins Englische	Andrea Herrmann, Stefan Sturm
2.0.0	1. Juli 2022	Umstellung auf Practitioner und Specialist	
2.1.0	5. April 2024	Neues Corporate Design umgesetzt	

1 Was ist Requirements Management?

Dieses Kapitel definiert zunächst, warum, für wen und wozu Requirements Management wichtig ist, welche Aufgaben dazu gehören und wer diese durchführt.

1.1 Definition des Requirements Managements

Wie für so viele Begriffe gibt es auch für den Begriff „Requirements Management" unterschiedliche Definitionen. Auch das Verhältnis des Requirements Management (RM) zum Requirements Engineering (RE) ist uneinheitlich definiert. Mal gilt RM als Teil des RE (wie im CPRE-Foundation Level [IREB2015]), mal umgekehrt RE als Teil des RM (z.B. in [Schi2001]). Das CMMI [SEI2011] dagegen sieht beide als gleichwertig.

Definition 1.1:

Wir definieren *Requirements Management* als Teil des RE.

Requirements Engineering{XE "Requirements Engineering"} ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen mit den folgenden Zielen:

1. Die relevanten Anforderungen zu kennen, Konsens unter den Stakeholdern über die Anforderungen herzustellen, die Anforderungen konform zu vorgegebenen Standards zu dokumentieren und die Anforderungen systematisch zu managen.
2. Die Wünsche und Bedürfnisse der Stakeholder zu verstehen und zu dokumentieren.
3. Die Anforderungen zu spezifizieren und zu managen, um das Risiko zu minimieren, dass das System nicht den Wünschen und Bedürfnissen der Stakeholder entspricht [Glin2014] und [PoRu2015].

Requirements Management{XE "Requirements Management"}: Der Prozess des Verwaltens existierender Anforderungen und anforderungsbezogener Artefakte. Dies schließt insbesondere die Dokumentation, das Ändern und die Verfolgbarkeit von Anforderungen mit ein [Glin2014]. Auch das Verwalten des RE-Prozesses gehört dazu, d.h. das Planen, Steuern und Kontrollieren des RE-Prozesses. Im Folgenden nennen wir einige zusätzliche Definitionen, um die in der Fachliteratur vorhandene Vielfalt darzustellen. IREB hat sich hier auf eine klare Definition festgelegt, damit zertifizierte Requirements Manager und Requirements Engineers mit demselben Begriff immer dasselbe meinen. Eindeutige Begriffe vereinfachen nicht nur grundsätzlich die Zusammenarbeit im Berufsleben, sondern wenn, wie in der Einführung dieses Handbuchs empfohlen, zusätzlich zur Rolle des Requirements Engineer noch eine Rolle des Requirements Managers eingeführt wird, folgt aus der Definition des Verhältnisses zwischen den beiden Disziplinen RE und RM auch die Aufgabenteilung zwischen diesen beiden Rollen.

Auch Ebert [Eber2012] definiert: „Requirements Management (RM) ist ein Teil des Requirements Engineering, der sich mit Pflege, Verwaltung und Weiterentwicklung von

Anforderungen im Lebenszyklus befasst." Dieser Lebenszyklus ist im Zusammenhang mit dem RE sehr wichtig. Es genügt demnach nicht, Anforderungen nur einmalig zu erheben. Anforderungen müssen über den gesamten Lebenszyklus der Software / des Systems hinweg verwaltet werden. Hierbei ist zu berücksichtigen, dass nicht nur die Software / das System selbst, sondern auch jede einzelne Anforderung einen eigenen Lebenszyklus durchläuft.

Laut Pohl [Pohl2010] kann das RM in drei wesentliche Teilaktivitäten zerlegt werden:

1. Beobachtung des Systemkontextes, um Kontext-Änderungen aufzudecken
2. Verwalten und Ausführen der RE-Aktivitäten (d.h. RM als Prozessmanagement)
3. Verwaltung der Anforderungen bzw. zugehörigen Artefakte während des Entwicklungsprozesses

Laut Standard ISO/IEC/IEEE 29148:2011 ISO29148 ist RM definiert als „activities that ensure requirements are identified, documented, maintained, communicated and traced throughout the life cycle of a system, product, or service". Beim RM geht es nicht nur um die Verwaltung der Anforderungen, sondern aller damit zusammenhängender Informationen: „Maintain throughout the system life cycle the set of system requirements together with the associated rationale, decisions and assumptions." ([ISO15288], 6.4.2.3 b).

Um die Definitionen und Empfehlungen dieses Handbuchs zu illustrieren, verwenden wir das Beispiel eines Online-Banking-Systems. Wir nehmen an, das System existiere bereits. Eine vollständige und qualitätsgesicherte Anforderungs-Spezifikation nach den Regeln der Kunst war die Grundlage seiner Entwicklung. Jedoch ist es damit nicht genug. Die Anforderungen an das System ändern sich durch Gesetzesänderungen (wie z.B. die SEPA-Umstellung), durch das ständige Bemühen, das Online-Banking bei gleichbleibender Benutzerfreundlichkeit noch sicherer zu machen oder bei gleichbleibender Sicherheit noch benutzerfreundlicher, durch technische Neuerungen, durch Ideen des Produktmanagements für neue Funktionalitäten oder durch Änderungen von Geschäftsprozessen innerhalb der Bank, die sich auch auf das Online-Banking auswirken. Nun gilt es, trotz dieser vielen Änderungsideen aus allen Richtungen den Überblick zu behalten, vorab abzuschätzen, welche Kosten und andere Folgen eine Idee bei ihrer Umsetzung mit sich bringen würde, und wohl begründet die Änderungen umzusetzen, zurückzustellen oder abzulehnen. Und dies unter Einbeziehung aller Stakeholder wie der IT-Abteilung, dem Produktmanagement, dem Vorstand, dem Datenschutzbeauftragten und dem Kundenbeirat. Doch auch unser Requirements Engineer ist einer der Stakeholder.

Unser Requirements Engineer heißt Peter Reber. Er ist 35 Jahre alt und arbeitet bereits seit zehn Jahren bei unserer Beispiel-Bank. Er kennt alle Stakeholder sehr gut und kontaktiert sie gerne auch ohne besonderen Anlass, um zu erfahren, was es Neues gibt. Das Online-Banking wurde erfolgreich eingeführt, bevor Peter seine Tätigkeit bei der Bank aufnahm.

Seitdem er von seinem Vorgänger eingearbeitet wurde, verwaltet Peter alleinverantwortlich die Anforderungen des Online-Bankings gewissenhaft nach den Regeln des RE.

Seine ruhigen Zeiten gehen jedoch zu Ende, als das gesamte Online-Banking auf ein neues Corporate Design mit neuen Farben, Schriftarten, Fachbegriffen und Logos umgestellt wird. Diese Gelegenheit möchte man nutzen, um neue Funktionalität einzuführen, die Sicherheit und die Benutzerfreundlichkeit zu erhöhen. Eine Gruppe von Experten soll innerhalb kurzer Zeit Änderungsanforderungen erheben oder definieren, also RE betreiben: mehrere externe Business Analysten analysieren die Geschäftsprozesse, ein Team von IT-Sicherheits-Experten betreibt Risiko-Analysen, der Usability-Experte entwirft alternative Oberflächen-Designs und verbessert die Barrierefreiheit, ein Moderator trifft sich zu einem Ideenworkshop mit dem Kundenbeirat. Diese alle betreiben RE. Peter Reber soll deren Arbeit organisieren und koordinieren. Das heißt, während die anderen RE machen, kümmert er sich um das RM.

1.2 Aufgaben im Requirements Management

Wir definieren hier die Aufgaben, die zum RM gehören. Somit stellt diese Definition auch eine Rollenbeschreibung dar: Der Requirements Manager verantwortet das RM und führt die zum RM gehörigen Aufgaben selbst durch oder überwacht deren Durchführung.

Drei grundsätzliche Rahmenbedingungen [RuSo2009] erhöhen die Komplexität der RM-Aufgaben:

- Anforderungen müssen von mehreren Personen genutzt werden
- Anforderungen sollen wiederverwendet werden
- Anforderungen ändern sich

Das RM ist verantwortlich dafür, die Regeln und Techniken bereitzustellen, die benötigt werden, damit Anforderungen und andere Informationen so abgelegt werden können, dass jeder Beteiligte das findet, was er braucht. Dies muss vorab geplant werden. Darum erstellt der Requirements Manager vor dem Beginn des RE-Prozesses den **Requirements-Management-Plan (RMP)**.

Definition 1-2:

Der *Requirements Management Plan*{ XE "Requirements-Management-Plan" } umfasst:

- Die Anforderungs-Landschaft, d.h. die zu verwaltenden Anforderungs-Artefakt-Typen und deren Detaillierungsgrad (siehe Kapitel 2)
- Attribute und Sichten auf die Anforderungen (siehe Kapitel 3)
- Priorisierungskriterien und -techniken (Kapitel 4)
- Versionsverwaltung von Anforderungen sowie den Änderungsprozess (Kapitel 5)
- Verwaltung der Verfolgbarkeit von Anforderungen (Kapitel 6)
- Variantenmanagement (Kapitel 7)
- Berichtswesen (Kapitel 8)
- RE-Prozess und Aktivitäten für dessen Verbesserung (Kapitel 9 und 10)
- Zu verwendende Werkzeuge (Kapitel 11)

Dieses Handbuch behandelt in den folgenden Kapiteln die Inhalte des RMP und schlägt für die jeweils durchzuführenden Aktivitäten unterschiedliche Techniken vor. Es wird jeweils am Ende jedes Kapitels zusammengefasst, welche Inhalte der RMP enthalten soll.

Während des RE-Prozesses führt das RM diese Aufgaben entsprechend des Planes aus: Sichten und Berichte werden erstellt und aktualisiert, Anforderungen für Releases ausgewählt, Änderungen systematisch verwaltet und priorisiert, Produktlinien definiert und verwaltet, Werkzeuge eingeführt und der RE-Prozess überwacht und verbessert.

Das Requirements Management wird durch den Requirements Engineer geplant und durchgeführt oder durch eine separate Rolle eines Requirements Managers. Der Requirements Manager verhält sich dann zum Requirements Engineer wie der Qualitätsmanager zum Tester. Eine solche Aufgabenteilung ist gerade dann sinnvoll, wenn in einem komplexen, zeitkritischen Projekt im RE und RM so viel Arbeit anfällt, dass mehrere Personen Anforderungen erheben, abstimmen und verwalten müssen. Dann braucht es eine Rolle, welche den Prozess gestaltet und überwacht, Informationen zusammenführt und auswertet.

Peter Rebers erste Aufgabe ist also die Erstellung des RMPs. Er könnte auch organisieren, dass dieser erstellt wird. In unserem Fall erstellt Peter den RMP aufgrund seiner großen Erfahrung selbst. Er kann sich dafür Hilfe von den vielen vorhandenen RE-Experten einholen und sollte diesen Plan ohnehin mit ihnen abstimmen. Denn vielleicht haben diese spezielle Anforderungen an das Requirements Manangement. In diesem Handbuch begleiten wir Peter und sein Team Schritt für Schritt bei ihren Aufgaben.

1.3 Ziele und Nutzen des Requirements Managements

Ziel des Requirements Management ist es, Anforderungen und andere Artefakte (z.B. Interview-Protokolle und Lastenheft) so zu verwalten, dass die Anforderungen mit vertretbarem Aufwand systematisch durchsucht, gruppiert, bewertet, geändert und verfolgt werden können. Dabei versucht das RM, gleichzeitig die Bedürfnisse vieler verschiedener Beteiligter zu erfüllen. Die Bedürfnisse sind dabei im Wesentlichen vom konkreten Projektkontext abhängig. So unterscheiden sich diese beispielsweise in Projekten zur kundenspezifischen Software- bzw. Produktentwicklung von internen Projekten durch die IT-Abteilung.

Das RM liefert unter anderem Antworten auf die nachfolgenden Fragestellungen, basierend auf den in Klammern genannten Techniken:

- Welche Anforderungsarten werden unterschieden? (Anforderungs-Landschaft)
- Auf welchen Detaillierungsebenen werden Anforderungen dokumentiert? (Anforderungs-Landschaft)
- Welche Anforderungen sind bereits abgenommen? (Attributierung)
- Welche Anforderungen stammen aus welcher Quelle? (Attributierung)
- Welche Anforderungen sind dringend und wichtig und somit Kandidaten für das nächste Release? (Bewertung und Priorisierung)
- Welche Anforderung erzeugt zu hohe Kosten bei zu geringem Nutzen? (Bewertung und Priorisierung)
- Welche Anforderungen gehören zu einer bestimmten Basislinie der Software? (Versionsverwaltung)
- Welche Version der Anforderung wurde im System umgesetzt? (Versionierung)
- Wer hat die Anforderung zuletzt verändert und warum? (Versionierung)
- Welche technische Komponente gehört zu welcher Anforderung? (Verfolgbarkeit)
- Welche Testfälle gehören zu welcher Anforderung? (Verfolgbarkeit)
- Welche Anforderung ist Teil des ausgelieferten Systems / Produkts? (Verfolgbarkeit)
- Worin unterscheiden sich die beiden Varianten des Produkts? (Variantenmanagement)
- Welcher Anteil der Anforderungen wurde bereits umgesetzt und getestet? (Berichtswesen)
- Wie lange braucht ein Change Request im Durchschnitt bis zu seiner Umsetzung? (Berichtswesen)

- Hat sich der RE-Prozess durch eine bestimmte Maßnahme verbessert? (Berichtswesen)

Grundsätzlich lohnt sich RM nicht nur bei größeren Projekten, sondern auch bei kleinen. Wobei bei überschaubaren Projekten das Kernteam das RM sogar oft im Kopf erledigt und noch genau weiß, wann welche Anforderung warum geändert wurde.

RM ist umso wichtiger und schwieriger [RuSo2009]...

- ... je größer die Zahl der Anforderungen ist
- ... je länger die geschätzte Lebensdauer des Produktes ist
- ... je mehr Änderungen zu erwarten sind
- ... je größer die Anzahl der Beteiligten am RE-Prozess ist
- ... je schlechter die Stakeholder zu erreichen oder einzubeziehen sind
- ... je höher die Qualitätsansprüche an das System sind
- ... je mehr Wiederverwendung betrieben werden soll
- ... je komplexer der Entwicklungsprozess ist
- ... je inhomogener die Stakeholder-Meinungen sind
- ... je mehr Releases entwickelt werden
- ... je wichtiger die Nutzung von Normen für das Projekt ist

Ein gutes Requirements Management ...[RuSo2009]

- ... erhöht die Qualität von Anforderungen, Produkten und Prozessen
- ... reduziert Projektkosten und Projektlaufzeit
- ... vereinfacht die Überwachung komplexer Projekte während aller Phasen
- ... verbessert die Kommunikation innerhalb der und zwischen den Teams
- ... erhöht die Kundenzufriedenheit
- ... reduziert das Projektrisiko

RM ist eine komplexe Aufgabe: Jeder Stakeholder sollte jederzeit auf die aktuellen Informationen zugreifen können und auch über die Änderungen informiert werden, die ihn betreffen, jedoch ohne mit unnötiger Information belastet zu werden. Dies soll auch dann gelten, wenn die Stakeholder weltweit verteilt arbeiten und wenn Ansprechpartner wechseln. Gleichzeitig soll aber auch der notwendige Datenschutz beachtet werden und jeder nur auf diejenigen Daten zugreifen, die er tatsächlich für seine Arbeit benötigt. Die durch das RM gesammelten Daten führen zu einer gewissen Komplexität, die nicht nur durch die pure Menge der Anforderungen und zugehöriger Informationen entsteht, sondern auch durch deren gegenseitige Abhängigkeit und die zeitliche Dimension von Versionen und Anforderungs-Basislinien (Baselines).

Das RM vereinfacht das RE:

- Strukturieren der Anforderungen und des Anforderungs-Dokuments (z.B. durch Attributierung, Sortieren und Filtern),
- Vereinheitlichen der Terminologie (z.B. mit einem Glossar),
- Definition klarer Prozesse und Arbeitsschritte, die durchzuführen sind (z.B. im Änderungsprozess).

Für Peter Reber stellt sich also die Frage nach dem Nutzen des RM nicht wirklich. Es ist zu erwarten, dass diese Vielzahl an Stakeholdern und Requirements Engineers eine große, vielleicht sogar zu große Anzahl an Änderungsanforderungen finden werden. Diese müssen miteinander abgestimmt und die relevantesten für das erste Release ausgewählt werden. Natürlich steht dieses Projekt wie die meisten unter Zeitdruck und besitzt nur ein vorab festgelegtes Budget.

1.4 Der Requirements-Management-Plan

Im Projektmanagement kennt man seit Jahr und Tag die Notwendigkeit eines Projektmanagement-Plans [PMI2013]. Dieser Plan beschreibt, wie ein konkretes Projekt durchgeführt werden soll. Neben dem Projektplan enthält er die Planung, wie beispielsweise Risikomanagement betrieben werden soll, welche Kommunikations- und Besprechungskultur gelebt werden soll, wer welche Verantwortung übernimmt, etc. Dieser Plan ermöglicht es dem Projektmanager, alle Projektmitglieder auf denselben Kenntnisstand zu bringen, bezüglich der Frage, wie innerhalb des Projektes gearbeitet werden soll. Darüber hinaus bietet der Plan die Möglichkeit zur Kontrolle des Prozesses.

Analog zum Projektmanagementplan sehen wir den Requirements-Management-Plan (RMP). Der RMP beschreibt u.a. die Planung wie der RE-Prozess aufgesetzt werden soll, wer welche Aufgaben verantwortet, welche Anforderungen wie dokumentiert werden sollen, wie diese Anforderungen verwaltet werden, ob und welche Werkzeuge zum Einsatz kommen. Der RMP beschreibt kurz gefasst kurz alle Aspekte, die im RE und RM für eine neue Entwicklung bzw. für die kontinuierliche Weiterentwicklung eines bspw. Produktes zu berücksichtigen sind. Der RMP beschreibt damit den Rahmen für den gesamten RE-Prozess.

In den folgenden Kapiteln beschreiben wir die wesentlichen Inhalte der RM und geben Ihnen in jedem dieser Kapitel einen Hinweis, welche der beschriebenen Aspekte in einen RMP aufgenommen werden sollten. Im Anhang A finden Sie darüber hinaus eine mögliche Vorlage für einen RMP zur eigenen Anwendung.

Hinweis: In der Praxis ist der RMP oft kein eigenständiges Dokument, sondern Bestandteil des Projektplans, des Konfigurationsmanagement-Plans oder anderer Vorgabedokumente zum Entwicklungsprozess. Die Struktur des RMP im Anhang soll im Wesentlichen dazu dienen, Ihnen für das Thema RMP einen Rahmen zu geben.

1.5 Relevante Normen

Damit die Entwicklung von Software und technischen Systemen nachvollziehbar und wiederholbar in hoher Qualität erfolgt, wurden diverse Normen und Standards entwickelt. Diese beschreiben die durchzuführenden Aktivitäten, zu erstellenden Artefakte und anzuwendenden Techniken. Das RM ist von diesen Standards durchgängig als wichtiger Beitrag zur Sicherung der Ergebnisqualität anerkannt. Darum machen diese Standards auch

Aussagen über die Durchführung des RM. Allerdings sind die Aussagen der verschiedenen Standards nicht unbedingt miteinander konsistent und miteinander verträglich. Beispielsweise verwenden sie unterschiedliche Begriffe für ein entsprechendes Artefakt und schlagen jeweils unterschiedliche Kapitel dafür vor.

Einige wichtige Normen, die den gesamten Software- oder Systementwicklungsprozess umfassen, und dabei auch Aussagen über RE und RM machen, sind die folgenden:

- Das Prozess-Reifegradmodell CMMI (Version 1.3) [SEI2010] betrachtet unter anderem die Prozesse „Anforderungsentwicklung (Requirements Development) " und „Anforderungsmanagement (Requirements Management)", wobei die zugeordneten Ziele sich teilweise deutlich von den Festlegungen des IREB unterscheiden.
- Die ISO 9000 / ISO 9001 [ISO9000] ist eine Norm für Qualitätsmanagement in Unternehmen. Die ISO 9001:2008 („Quality Management Systems – Requirements") legt Mindestanforderungen an ein Qualitätsmanagementsystem fest und beschreibt beispielsweise Anforderungen an die Produktrealisierung, Messung und Verbesserung und adressiert somit Themen wie Identifizierbarkeit oder Verfolgbarkeit von Anforderungen (vgl. Clause 7.5.3 „*Identification and Traceability*").
- In der ISO/IEC 12207:2008 [ISO12207] bzw. 15288:2008 [ISO15288] („Software life cycle processes" bzw. „Systems and software engineering – Systems life cycle processes") werden sämtliche Prozesse zur Entwicklung von Systemen und Software definiert. Die Aufgaben „*System requirements analysis*" und „*Software requirements analysis*" des ISO/IEC 12207 und „*Stakeholder Requirements Definition Process*" sowie „*Requirements Analysis Process*" des ISO/IEC 15288 umfassen die Tätigkeiten des RE und RM.
- Die IEC 61508 [DIN61508] („Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme") beschäftigt sich mit der Definition von Anforderungen an die funktionale Sicherheit von Systemen und deren Umsetzung, einschließlich quantitativer Sicherheitsschätzungen. Insbesondere dem Thema Verfolgbarkeit wird hier eine besondere Bedeutung beigemessen. Die Norm definiert Sicherheitsintegritätsstufen (SIL) 1 bis 4, die das Gefährdungsrisiko beschreiben.
- Je höher die Stufe, umso höher das Gefährdungspotenzial, und damit steigen auch die Anforderungen an die Verlässlichkeit des Systems.
- SOX (Sarbanes-Oxley Act) [USCo2002] ist ein US-Bundesgesetz als Reaktion auf Bilanzskandale, welches die Verlässlichkeit der Berichterstattung von Unternehmen, die am öffentlichen Kapitalmarkt in den USA notiert sind, verbessern soll. Der Kern des Sarbanes-Oxley Act ist zu wissen, wer, zu welchem Zeitpunkt, welche Änderung durchführen ließ und betrifft somit auch Kernaufgaben des RM.

RE und RM ist zudem beispielsweise in den folgenden Standards zu finden:

- VDI-Richtlinie 2519 Blatt 1 – Vorgehensweise bei der Erstellung von Lasten-/Pflichtenheften [VDI2001] ist der deutsche Standard zur Beschreibung von Lasten- und Pflichtenheften.

- Der IEEE 830-1998 („Recommended Practice for Software Requirements Specifications") [IEEE830] definiert Begriffe des RE und RM, insbesondere Qualitätseigenschaften von Anforderungen und die Kapitel für eine Spezifikation ("software requirements specification"). Viele dieser Definitionen sind auch in den CPRE Foundation Level eingegangen [IREB2015].
- ISO/IEC/IEEE 29148:2011 („Systems and software engineering – Life cycle processes – Requirements engineering") ISO29148 definiert Qualitätseigenschaften und Attribute von Anforderungen und empfiehlt den iterativen Umgang mit Anforderungen während des gesamten Lebenszyklus.
- IEEE Standard 1233 „Guide for Developing of System Requirements Specifications" [IEEE1233] beschreibt die Entwicklung von Anforderungen, Spezifikationen und deren Verwaltung in der gesamten Produktentwicklung. Er beschreibt die Erhebung und Definition von Anforderungen, das Änderungsmanagement und die Organisation von Anforderungen im Projekt.
- ISO / IEC 14102:1995 (*„Evaluation and Selection of CASE Tools"*) ISO14102 beschreibt Anforderungen an CASE Tools, d.h. Werkzeuge zur computerunterstützten Softwareentwicklung, kann aber auch speziell für die Auswahl von RE- und RM-Werkzeugen verwendet werden.
- ISO/IEC 25010:2011 („Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models") [ISO25010] beschreibt zwei Qualitätsmodelle für nichtfunktionale Anforderungen – eines für „Quality in use" und eines für die Produktqualität. Mit Hilfe dieser Qualitätsmodelle können nichtfunktionale Anforderungen an Software und Computersysteme einheitlich erhoben und spezifiziert werden.
- Die ISO 29110 („Lifecycle process standard for Very Small and Medium Entities (VSME)") [ISO29110] beschreibt einen System-Lebenszyklus einschließlich RE für kleine und mittlere Einheiten, also Projekte von ca. unter 25 Personen.
- Die europäische Norm ISO 9241 [ISO9241], die auch als DIN-Norm anerkannt ist, beschreibt Richtlinien der Mensch-Computer-Interaktion, und zwar sowohl eine Liste von Qualitätsanforderungen, die eine benutzerfreundliche Software einhalten muss, als auch den Entwicklungs- und Prüfprozess einer solchen Software.

Hinzu kommen noch branchenspezifische Normen und Standards wie DO 178 B/ED-12B und DIN EN 14160 für die Luftfahrt, IEEE/EIA Std. 12207:1998 für das Militär, FDA-535, FDA-938 und EN62304 für die Medizintechnik, EN 50128 für die Bahntechnik oder ITU X.290-X.296 (ISO/IEC 9646-x) oder ETSI ES 201 873-x (TTCN-3) für die Telekommunikation.

Diese Normen gelten nicht automatisch und vor allem nicht gleichzeitig, da sie nicht vollständig miteinander verträglich sind. Jede Firma und jedes Projekt wählt die passenden Standards aus, die sie in originaler oder in angepasster Form anwenden werden. Manchmal verlangt der Kunde die Einhaltung eines bestimmten Standards.

Zusätzlich zu Normen, Standards und Richtlinien sind firmenspezifische Normen des Software-Herstellers oder des Kunden zu beachten. Diese wiederum können aufbauend auf öffentlichen Standards entwickelt sein können und Aspekte des RE und RM beinhalten.

Peter Reber verwendet wegen seiner Verständlichkeit und Praxisnähe für seine Arbeit zunächst das (hier vorliegende) IREB-Handbuch zum Requirements Management, das aus Standards heraus entwickelt wurde. Außerdem gelten die Firmen-Richtlinien über die Durchführung von IT-Projekten und die neue Corporate Identity.

1.6 Vertiefende Literatur

Zur Vertiefung empfehlen wir die im vorigen Abschnitt genannten Standards.

Weitere Definitionen finden Sie im IREB-Glossar [Glin2014].

2 Requirements Information Model

In diesem Kapitel betrachten wir, wie Sie die verschiedenen Aspekte Ihrer projektspezifischen Anforderungs-Landschaft festlegen und durch ein Requirements-Information-Model beschreiben können.

Im Rahmen der Dokumentation von Anforderungen stoßen wir immer wieder auf eine Vielzahl von Grundsatzfragen, die noch nichts mit den konkreten Inhalten der Anforderungen zu tun haben, aber frühzeitig festgelegt werden müssen, z.B.:

- Welche Anforderungsarten existieren und müssen betrachtet werden?
- Wie können Anforderungen nach ihrer Lösungsabhängigkeit klassifiziert werden?
- Wie sollen diese Anforderungen dokumentiert und dargestellt werden?
- Wie detailliert muss die Anforderung beschrieben werden?

Diese Fragen sollten grundsätzlich bereits zu Beginn des RE geklärt werden. Aber nicht immer laufen Projekte so ideal, wie man es sich wünscht. Manchmal werden Projekte von Dritten übernommen, die keine Festlegungen dieser Art getroffen haben. Manchmal erlauben es die zum Projektstart gegebenen Randbedingungen nicht, einen RMP zu erstellen bzw. sich über die Struktur seiner Anforderungen Gedanken zu machen und man „sammelt" in einem ersten Schritt nur, ohne die Anforderungen zu klassifizieren. Wichtig ist allerdings, dass Sie sich zu gegebener Zeit – und zwar möglichst frühzeitig – mit der Planung Ihrer Anforderungs-Landschaft beschäftigen. Denn als Requirements Manager sind Sie neben der Verwaltung der Anforderungs-Artefakte dafür zuständig, die Aktivitäten im RE-Prozess zu managen – das heißt, diese zu planen, zu überwachen und entsprechend zu steuern (Definition 1-1).

Definition 2-1:

Anforderungs-Artefakt{ XE "Anforderungs-Artefakt" } nach [Glin2014][Pohl2010]:

„Ein Anforderungs-Artefakt ist eine dokumentierte Anforderung".

Immer dann, wenn wir in den folgenden Kapiteln von Anforderungs-Artefakten sprechen, sprechen wir von einer dokumentierten Anforderung. Wenn wir den allgemeineren Begriff „Artefakt" verwenden, sprechen wir hingegen von dokumentierten Artefakten auf unterschiedlichen Entwicklungsstufen, z.B. Testfälle, Architekturbeschreibungen etc. (vgl. IREB-Glossar [Glin2014]).

Definition 2-2:

Anforderungs-Landschaft{ XE "Anforderungs-Landschaft" }: Eine Anforderungs-Landschaft ist eine Festlegung der:

1. zu verwendenden Klassifizierung in Bezug auf die Anforderungsarten
2. zu verwendenden Klassifizierung in Bezug auf die Lösungsunabhängigkeit der Anforderung
3. notwendigen Abstraktionsebenen (Detaillierungsebenen) je Anforderungs-Artefakt-Typ
4. zu verwendenden Darstellungsformen je Anforderungs-Artefakt-Typ

In den folgenden Abschnitten gehen wir auf die unterschiedlichen Dimensionen der Anforderungs-Landschaft ein und geben Hinweise, an welcher Stelle sich ein Requirements Manager Gedanken machen muss, um ein „**Requirements-Information-Model{** XE "Requirements Information Model" **}**" (RIM) zur Beschreibung der Anforderungslandschaft zu erstellen.

2.1 Grundlagen (Klassifizierung von Anforderungen)

Wenn wir von Anforderungen sprechen, sprechen wir oftmals von den unterschiedlichsten Ausprägungen einer Anforderung. Eine Anforderung kann sich beispielsweise durch ihren Detaillierungsgrad unterscheiden. So können Anforderungen sehr detailliert und konkret eine Funktion beschreiben oder aber auch eine sehr abstrakte Forderung an das Gesamtsystem sein. Neben ihrem Detaillierungsgrad können sich Anforderungen in ihrer Art des Inhalts unterscheiden. So kann eine Anforderung die Qualität eines Systems (z.B. Korrektheit) beschreiben oder aber funktionale Eigenschaften fordern. Darüber hinaus können sich Anforderungen in ihrer Lösungsunabhängigkeit unterscheiden, z.B. generischen Produktzielen stehen beispielsweise Anforderungen an die Datenstruktur gegenüber. Dies ist grundsätzlich nicht neu und bereits im Certified Professional for Requirements Engineering – Foundation Level angerissen worden [IREB2015].

Zur besseren Orientierung und zum gezielten Aufbau Ihrer Anforderungslandschaft wollen wir Anforderungen aus diesem Grund nach den folgenden Dimensionen klassifizieren:

- nach ihrer Anforderungsart
- nach ihrer Lösungsunabhängigkeit
- nach ihrer Detaillierungsebene (bzw. Abstraktionsebene)

Die genannten Dimensionen sind orthogonal zueinander, d.h. auch auf einer Ebene mit einem hohen Detaillierungsgrad gibt es beispielsweise Ziele – also lösungsunabhängige Anforderungen.

2.1.1 Klassifizierung nach Anforderungsarten

Eine Frage, die im Rahmen des Anforderungsmanagements sehr häufig gestellt wird ist: „Welche Arten von Anforderungen sind für die Erhebung und Dokumentation zu berücksichtigen?“. Diese Frage lässt sich mit einem kurzen Rückblick auf den Certified Professional for Requirements Engineering – Foundation Level beantworten [IREB2015].

- **Funktionale Anforderungen{** XE "Funktionale Anforderung" **}**: Funktionale Anforderungen beschreiben die Funktionalität, die das geplante System bereitstellen soll. Diese Anforderungen beschreiben, was das geplante System leisten soll, z.B. mit welchen Daten sich ein Kunde am Geldautomaten autorisieren muss, um eine Geldabhebung zu autorisieren.

Definition 2-3:

Funktionale Anforderung nach [PoRu2015]:

„Eine funktionale Anforderung ist eine Anforderung bezüglich des Ergebnisses eines Verhaltens, das von einer Funktion des Systems bereitgestellt werden soll.“

- **Qualitätsanforderungen{** XE "Qualitätsanforderung" **}**: Qualitätsanforderungen beschreiben gewünschte Qualitäten des geplanten Systems und beeinflussen dadurch die Systemarchitektur. Diese Klasse beschreibt bspw. Anforderungen an die Zuverlässigkeit, Sicherheit, Skalierbarkeit oder Performanz des geplanten Systems bzw. einzelner Funktionen.

Definition 2-4:

Qualitätsanforderung nach [PoRu2015]:

„Eine Qualitätsanforderung ist eine Anforderung, die sich auf ein Qualitätsmerkmal bezieht, das nicht durch funktionale Anforderungen abgedeckt wird.“

- **Randbedingungen{** XE "Randbedingung" **}**: Randbedingungen sind organisatorische, gesetzliche oder technische Vorgaben (i.d.R. Einschränkungen) zur Realisierung des geplanten Systems. Dies können unterschiedlichste Bedingungen sein – angefangen von zeitlichen Vorgaben zur Umsetzung bis hin zu konkreten Technologievorgaben für die Umsetzung.

Definition 2-5:

Randbedingung nach [PoRu2015]:

„Eine Randbedingung ist eine Anforderung, die den Lösungsraum jenseits dessen einschränkt, was notwendig ist, um die funktionalen Anforderungen und die Qualitätsanforderungen zu erfüllen."

Die Spezifika der verschiedenen Anforderungsarten sowie deren weitergehende Kategorisierung werden in der Literatur ausführlich behandelt (vgl., [Pohl2010], [PoRu2015]und [Eber2012]).

Neben der oben eingeführten Klassifikation von Anforderungen findet man in der Requirements-Engineering-Literatur noch einige weitere Klassifikationen, für die häufig ein Bezug zu den drei oben genannten Anforderungsarten hergestellt werden kann, z.B.:

- [RuSo2009]: funktionale Anforderungen, technologische Anforderungen, Qualitätsanforderungen, Anforderungen an die Benutzeroberfläche, Anforderungen an sonstige Lieferbestandteile, Anforderungen an durchzuführende Tätigkeiten, rechtlich-vertragliche Anforderungen.
- [WiBe2013]: Business-Anforderungen, Businessregeln, Randbedingungen, Schnittstellenanforderungen, Features, funktionale Anforderungen, nicht-funktionale Anforderungen, Qualitätsanforderungen, System-Anforderungen, Benutzer-Anforderungen.
- [RoRo2014]: funktionale Anforderungen, nicht-funktionale Anforderungen{ XE "Nicht-funktionale Anforderung" }, Randbedingungen.
- [Youn2014]: Business-Anforderungen, Benutzer-Anforderungen, Produkt-Anforderungen, Umgebungs-Anforderungen, System-Anforderungen, funktionale Anforderungen, Performance-Anforderungen, Interface-Anforderungen usw.

Entscheidend für ein gutes Anforderungsmanagement ist allerdings nicht, nach welcher Klassifizierung Sie Anforderungen unterscheiden, sondern vielmehr das Bewusstsein um die Existenz verschiedener Anforderungsarten, die zur vollständigen Beschreibung der gewünschten Änderung oder des geplanten Systems berücksichtigt werden müssen.

Keine der in diesem Kapitel vorgestellten Anforderungsarten stellt einen allgemeingültigen Standard dar. Die Aufstellung soll Ihnen lediglich die Bandbreite an Anforderungsarten aufzeigen, die sich über die letzten Jahre etabliert haben.

2.1.2 Klassifizierung nach Lösungsabhängigkeit von Anforderungen

Unabhängig von der Anforderungsart weisen Anforderungen häufig eine sehr unterschiedliche Lösungsabhängigkeit auf. So finden wir in Anforderungsbeschreibungen häufig eine Vermischung von:

- Zielen, die durch ein System erreicht werden sollen (i.d.R. nahezu lösungsunabhängige Beschreibung des zu erreichenden Ziels, z.B. einfacher und sicherer Zugang zu Bargeld für alle Bankkunden).
- Systemabläufe bzw. Prozesse, die durch ein System unterstützt werden sollen (i.d.R. nur indirekter Lösungsbezug durch fachliche Vorgaben zum gewünschten Systemverhalten oder Prozessablauf, z.B. Beschreibung eines Ablaufs zur Authentifizierung am Bankautomaten).
- Konkrete Eigenschaften und Merkmale, die ein System erfüllen soll (i.d.R. direkte Lösungsabhängigkeit durch fachliche und operative Vorgaben zum gewünschten System, z.B. eindeutige Festlegung der relevanten Daten zur Authentifizierung eines Benutzers am Bankautomaten).

Für eine bewusste Unterscheidung der Anforderungen mit unterschiedlicher Lösungsabhängigkeit empfehlen wir die explizite Klassifizierung von Anforderungs-Artefakten in Abhängigkeit zum Lösungsbezug bzw. der Lösungsabhängigkeit, z.B. in Ziele, Szenarien und lösungsabhängige Anforderungen (vgl. [Pohl2010]).

Zielorientierte Beschreibungen (Ziele):

- Zielorientierte Beschreibungen dokumentieren (durch Ziele) die Intention des Systems, ohne dabei auf die Umsetzung (Lösung) einzugehen. Sie sind damit die abstrakteste Form einer dokumentierten Anforderung und fordern bspw., dass ein Kunde in jeder deutschen Stadt Geld von seinem Konto abheben können muss – unabhängig davon wie das umgesetzt werden kann.
- Ziele lassen sich beispielsweise rein textuell in natürlicher Sprache, durch Und-Oder-Bäume oder eigenständige Notationen wie z.B. i* (I-Star) beschreiben. Unabhängig von der Darstellungsform geht es bei Zielen aber vor allem darum, ein Systemverständnis zu erlangen, um so den benötigten Mehrwert des geplanten Systems zu erkennen.

Szenario-orientierte Beschreibungen (Szenarien):

- Szenario-orientierte Beschreibungen dokumentieren (durch Szenarien) auf eine beispielhafte Art und Weise den gewünschten zu unterstützenden Ablauf aus Benutzersicht (teilweise auch aus Systemsicht). Szenarien beschreiben somit mögliche Interaktionsfolgen zur Erfüllung eines oder mehrerer Ziele. Szenarien geben hierbei oftmals den für die Anforderung notwendigen Kontext mit, indem sie bspw. den Ablauf einer Geldabhebung an einem Geldautomaten beschreiben. Szenarien umfassen aus diesem Grund in der Regel mehrere atomare Anforderungen.

- Szenarien lassen sich beispielsweise rein textuell als eine Art Geschichte durch strukturierte Vorlagen (z.B. Use Case Templates), aber auch modellbasiert durch Aktivitätsdiagramme, Geschäftsprozessmodelle (z.B. BPMN), Sequenzdiagramme etc. beschreiben, siehe IREB CPRE Modul "Requirements Modeling" CHQW2022.

Lösungsorientierte Beschreibungen (lösungsorientierte Anforderungen):

- Lösungsorientierte Beschreibungen dokumentieren (durch lösungsorientierte Anforderungen) konkrete Anforderungen z.B. an die Funktionalität oder Performanz eines Systems oder einzelner Komponenten. Sie beschreiben die zur Erfüllung der Ziele und zur Umsetzung der Szenarien notwendigen Daten, Funktionen, Systemverhalten, Zustände sowie die erforderliche Qualität. Sie sind damit als die „klassischen" Anforderungen zu verstehen, die in eine Lösung münden müssen, z.B. „Das System muss dem Kunden nach einer erfolgreichen Autorisierung die Möglichkeit geben, Bargeldbeträge zwischen 50 € und 500 € abzuheben. Der ausgewählte Betrag muss glatt durch 10 € dividierbar sein, um eine Auszahlung zu ermöglichen".
- Lösungsorientierte Anforderungen lassen sich entweder natürlich-sprachlich als klassische textuelle Anforderung oder über modellbasierte Notationen (z.B. UML) beschreiben. Lösungsorientierte Anforderungen umfassen in der Regel alle Anforderungen an die klassischen System-Sichten: Daten, Funktionen und Verhalten des geplanten Systems. Sie sind damit die konkretesten Beschreibungen, siehe IREB CPRE Modul "Requirements Modeling" CHQW2022.

2.1.3 Detaillierungsebenen für Anforderungen – Twin-Peaks-Model

Die Detaillierung von Anforderungen ist in der Praxis nur sehr selten ein strikt sequenzieller (Wasserfall-)Prozess, der mit grobgranularen Anforderungs-Artefakten startet und diese dann schrittweise zu feingranularen Anforderungs-Artefakten macht, auf Basis derer dann im nächsten Schritt eine Systemarchitektur erstellt wird. In der Praxis gibt es in der Regel bereits zum Start auf der einen Seite sehr unterschiedliche detaillierte Anforderungen und auf der anderen Seite ein frühzeitiges Zusammenspiel von Anforderungen und Systemarchitektur, sodass es eine gegenseitige Beeinflussung von Systemarchitektur- bzw. Lösungsentscheidungen und Anforderungen gibt.

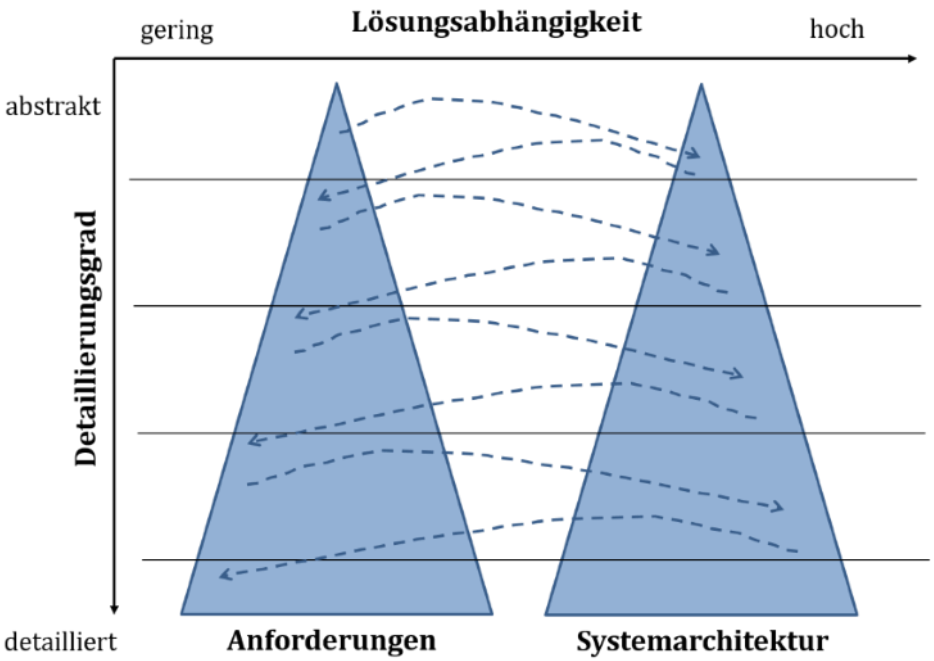


Abbildung 1: Twin-Peaks-Model

Abbildung 1 verdeutlicht diesen Zusammenhang im sogenannten Twin-Peaks-Model **{** XE "Twin-Peaks-Modell" **}**[Nuse2001]. Die vertikale Achse stellt den Detaillierungsgrad der Anforderungen bzw. der Systemarchitektur dar, während die horizontale Achse die Lösungsabhängigkeit darstellt, d.h. die zunehmende Ausrichtung von der Problembeschreibung zur Implementierung. Die Abbildung zeigt, dass iterativ eine detaillierter werdende Anforderungsbeschreibung (links im Bild) und parallel dazu eine immer detaillierter werdende Systemarchitektur (rechts im Bild) entstehen und sich beide gegenseitig ergänzen. Auch wenn die Abbildung vereinfacht die gleichen Ebenen für die Detaillierung von Anforderungen und für die Systemarchitektur zeigt, sind hier durchaus unterschiedliche Detaillierungsebenen möglich. Die Abbildung soll im Wesentlichen die Notwendigkeit von unterschiedlichen Detaillierungsebenen für die Dokumentation von Anforderungen aufzeigen (vgl. auch [BBHK2014]).

Eine allgemeine Einigkeit, wie viel Detaillierungsebenen auf der Anforderungsseite notwendig und sinnvoll sind, existiert leider nicht. Die notwendige Detaillierung**{** XE "Abstraktions-Ebene" **}** für Anforderungen hängt von vielen Faktoren ab, wie die folgenden Beispiele verdeutlichen sollen:

- **Systemkontext und Domäne**: Soll innerhalb einer wohlbekannten Systemumgebung nur eine kleine Änderung durchgeführt werden, kann ein geringerer Detailgrad notwendig sein als bei einer vollständigen Neuentwicklung.

- **Expertise und Nähe der Beteiligten**: In einem Projektumfeld mit erfahrenen und eingespielten Requirements Engineers, Architekten, Entwicklern und Testern ist oftmals eine geringere Detailstufe notwendig als bei einem verteilten Projektteam und Entwicklungsteam mit Lieferantenbeziehungen.
- **Akzeptierte Freiheitsgrade in der Umsetzung**: In einem Projektumfeld, in dem Auftraggeber und Stakeholder rein ergebnisorientiert denken und die Art und Weise wie das Ergebnis erreicht wird unerheblich ist (z.B. Darstellung der Kontobewegungen), muss der Lösungsraum durch Anforderungsdetaillierungen weniger eingeschränkt werden als in einem sicherheitskritischen Umfeld (z.B. Autorisierung beim Login).

Die Anzahl der Detaillierungsebenen bzw. der Detailierungsgrad der Anforderungen ist aus diesem Grunde individuell festzulegen (z.B. projektspezifisch). Dieser kann sich sogar in Abhängigkeit des betrachteten Systemgegenstands innerhalb eines Projektes unterscheiden. Grundsätzlich gilt, dass eine Anforderung so weit detailliert wird, bis:

- Bei allen Stakeholdern ein **gemeinsames Verständnis** bezüglich der Anforderungen erreicht wurde und jedem bewusst ist, was exakt gefordert wird. Dies gilt vor allem für den Stakeholder-Kreis, der die Anforderung umsetzen soll.
- Die verbleibenden Freiheitsgrade für die Konstruktion der Lösung so gering sind, dass eine weitere Präzisierung mehr Kosten als Nutzen verursachen würde. Also Detaillierung der Anforderungen bis ein **akzeptiertes Restrisiko** besteht, dass sich aufgrund noch verbleibender Freiheitsgrade eine nicht gewünschte Lösung ergibt.
- Die Anforderungen soweit spezifiziert sind, dass die spätere Lösung mittels der vorliegenden Anforderungen **eindeutig überprüfbar** (testbar) ist, sprich, dass eine Abnahme auf Basis der Spezifikation erfolgen kann.

Hinweis: in vielen Projekten haben sich drei Detaillierungsebenen bewährt – auch wenn diese in der Regel immer anders benannt sind, scheint dies ein praktikables Level zu sein (z.B. Ebene der Produkt-Anforderungen, Ebene der Benutzer-Anforderungen, Ebene der System-Anforderungen).

Die Literatur bietet darüber hinaus eine Reihe von Vorschlägen zur Strukturierung der Anforderungen auf unterschiedlichen Detaillierungsebenen.

- [WiBe2013] schlagen die Unterteilung in „Business-Requirements", „User Requirements" und „System Requirements" vor. [Eber2012] verfolgt die Detaillierung von Anforderungen über die Aufteilung nach Markt-, Produkt- und Komponentenanforderungen.
- [RuSo2009] beschreiben fünf Detaillierungsebenen vom groben Gesamtvorhaben mit seinen Zielen bis hin zur technischen Spezifikation und der Trennung in Hardware, Software und sonstigen Komponenten.
- [PHAB2012] definieren drei Detaillierungsebenen für die Domäne von eingebetteten softwareintensiven Systemen (Embedded Systems) „Functional Layer", „Logical Layer" und „Technical Layer".

- [BBHK2014] beschreiben ebenfalls drei Detaillierungsebenen „System Layer", „Function Group Layer", „Hardware/Software Layer" für die Detaillierung von Anforderungen bei software-intensiven eingebetteten Systemen.

2.2 Darstellungsformen zur Dokumentation von Anforderungen

Für die Dokumentation von Anforderungen existiert eine Vielzahl unterschiedlicher Darstellungsformen (bzw. Beschreibungsformen). Welche Darstellungsform für die Dokumentation von Anforderungen verwendet wird, hängt von unterschiedlichen Faktoren ab, z.B.:

- Zweck der Dokumentation (z.B. formale Prüfung, Review, Diskussion),
- Adressat der Information (z.B. Produktmanager, Architekt, Tester, Entwickler),
- Klassifizierung der Anforderung (z.B. Anwendungsfall, Performanz-Anforderung).

Darüber hinaus ist die gewählte Darstellungsform ebenfalls von der Erfahrung und den persönlichen „Vorlieben" desjenigen, der die Anforderung dokumentiert, abhängig.

Nachfolgend wollen wir folgende Darstellungsformen zur Dokumentation unterscheiden:

- **Textuelle Darstellung von Anforderungen durch natürliche Sprache.**
  Natürliche Sprachen (z.B. Deutsch, Englisch, Spanisch) sind Sprachen, die im täglichen Gebrauch verwendet werden, um Informationen zu dokumentieren und auszutauschen. Bei textuellen Beschreibungen finden wir beispielsweise folgende Darstellungsformen für Anforderungen:
  - Prosa in Reinform
  - Satzschablonen (z.B. „DAS SYSTEM muss/sollte/wird PROZESSWORT")
  - Vorlagen zur Strukturierung (z.B. zur Beschreibung von Anwendungsfällen)
- **Modellbasierte Darstellung von Anforderungen durch Modellierungssprachen:**
  Modellierungssprachen gehören im Vergleich zu natürlichen Sprachen zu den künstlich erschaffenen Sprachen. Modellierungssprachen zur Dokumentation von Anforderungen sind beispielsweise:
  - Unified Modeling Language (UML)
  - Business Process Model and Notation (BPMN)
  - Ereignisgesteuerte Prozesskette (EPK)
  - System-Modellierungssprache (SysML)
  - Entity-Relationship Model (ERM)
  - Petri-Netze
- **Formalisierte Darstellung von Anforderungen durch formale Sprachen**:
  Formale Sprachen gehören ebenfalls zu den künstlich erschaffenen Sprachen. Bei formalen Sprachen steht die Kommunikation in der Regel nicht im Vordergrund, sondern vielmehr die widerspruchsfreie Beschreibung. Zu formalen Sprachen gehören bspw.:
  - Mathematisch-algebraische Beschreibungen
  - Beschreibungsformen der Mengenlehre
  - logische Beschreibungen und Operatoren

Um die Dokumentation im Rahmen des RE-Prozesses – aber auch um die Verwaltung und Pflege im Rahmen des RM kontrollierbar zu gestalten – sollten Sie als Requirements Manager so früh wie möglich festlegen, welche Anforderungsart mit welcher Lösungsabhängigkeit auf welcher Detaillierungsebene durch welche Darstellungsformen persistiert werden.

Legen Sie ebenfalls frühzeitig fest, in welcher Landessprache textuelle und modellbasierte Anforderungen zu dokumentieren sind, um unnötigen doppelten Aufwand für eine spätere Übersetzung zu vermeiden.

Hinweis: In der Regel wird die Sprache der Anforderungen durch die Projektsprache oder die Landessprache der beteiligten Stakeholder und Lieferanten mitbestimmt. Eine Festlegung kann aber auch lauten, dass Anforderungen der Fachabteilungen in Deutsch verfasst werden, um eine bessere Beteiligung und Akzeptanz zu erlangen, und System-Anforderungen, die durch einen Lieferanten umgesetzt werden müssen, in Englisch, d.h. die Dokumentationssprache kann abhängig von der Detaillierungsebene unterschiedlich sein.

2.3 Beschreibung einer Anforderungs-Landschaft durch ein Requirements-Information-Model

In diesem Kapitel beschreiben wir, wie eine Anforderungs-Landschaft beschrieben und durch ein **Requirements-Information-Model{** XE "Requirements Information Model" **}** (RIM) dokumentiert werden kann. Wie in Definition 2-2 erläutert, legt eine Anforderungs-Landschaft die folgenden Dimensionen fest:

- *die zu verwendende Klassifizierung in Bezug auf die Anforderungsarten*
- *die zu verwendende Klassifizierung in Bezug auf ihre Lösungsabhängigkeit*
- *die notwendigen Detaillierungsebenen je Anforderungs-Artefakt-Typ*
- *die zu verwendenden Darstellungsformen je Anforderungs-Artefakt-Typ*

Zur Beschreibung der Anforderungs-Landschaft kann eine tabellarische Auflistung dienen. Hier dokumentieren Sie die unterschiedlichen Dimensionen der Anforderungs-Landschaft (Anforderungsart, Lösungsabhängigkeit, Detaillierungsebenen, Darstellungsform). Tabelle 1 zeigt ein Beispiel für eine Anforderungs-Landschaft.

		Lösungsabhängigkeit		
Detaillierungsebene	**Anforderungsart**	**Gering (Ziel)**	**Mittel (Szenario)**	**Hoch (Lösungsorientierte Anf.)**
Ebene 1: Geschäfts-Ebene	Randbedingung	Geschäftsziel (textuell)	N.R.	N.R.
	Qualitätsanforderung	Service-Qualität (textuell)	N.R.	N.R.
	Funktionale Anforderung	N.R.	Geschäftsprozess (BPMN)	Geschäftsregel (textuell)
Ebene 2: Benutzer-Ebene	Randbedingung	Usability / Benutzbarkeits-Ziel	N.R.	N.R.
	Qualitätsanforderung	N.R.	Benutzer-Oberfläche (Mock-Up)	N.R.
	Funktionale Anforderung	N.R.	Nutzer-Anwendungsfall (Use Case Diagramme,Vorlagen)	Nutzer-Anforderung (textuell, ER-Modelle)
Ebene 3: System-Ebene	Randbedingung	N.R.	N.R.	Schnittstellen Richtlinien (textuell)
	Qualitätsanforderung	System-Qualitätsziel (textuell)	N.R.	Systemqualität (textuell)
	Funktionale Anforderung	N.R.	Systemanwendungsfall (MSC, AD)	Schnittstellenanforde-rungen (textuell, MSC)

Tabelle 1: Beispiel für die Festlegung einer Anforderungs-Landschaft

In Spalte 1 sind die Detaillierungsebenen beschrieben (hier Detaillierung 1: Geschäfts-Ebene, Detaillierung 2: Benutzer-Ebene und Detaillierung 3: System-Ebene). In Spalte 2 ist die Klassifizierung nach Anforderungsarten (hier Randbedingung, Qualitätsanforderung und funktionale Anforderung) beschrieben. In den Spalten 3-5 die Lösungsabhängigkeit der

Anforderung durch die Klassifizierung in (Ziele, Szenarien und lösungsorientierte Anforderungen) beschrieben. Diese Tabelle beschreibt somit alle theoretisch möglichen Kombinationen für Anforderungs-Artefakt-Typen.

Über die Zellen können Sie nun auswählen, welche Anforderungs-Artefakt-Typen für Ihre Spezifikation relevant sind bzw. nicht relevant sind (N.R.). Für die relevanten Anforderungskandidaten (**Anforderungsklasse**) können Sie nun die gewünschten Darstellungsformen je Artefakt-Typ für Ihre Anforderungs-Landschaft festlegen (z.B. Benutzer-Ebene, lösungsorientierte Beschreibung für funktionale Anforderungen erfolgt durch Entity-Relationship Modelle oder in textueller Form). Zusätzlich haben Sie hier noch die Möglichkeit, den ausgewählten Anforderungs-Kandidaten einen dedizierten unternehmensspezifischen Bezeichner zu geben (z.B. Benutzer-Ebene, lösungsorientierte Beschreibung für funktionale Anforderungen = Benutzeranforderungen).

Bei der Festlegung der Anforderungs-Landschaft ist immer zwischen dem Nutzen, den eine umfangreichere Anforderungsdokumentation bringt, und den Kosten, die dadurch entstehen, abzuwägen (vgl. [Glin2014], [Davi2005]). So kann es durchaus sein, dass Sie Anforderungen bspw. nur auf zwei Detaillierungsebenen auf Basis von Szenarien und lösungsorientierten Anforderungen beschreiben.

Die Festlegung zur Anforderungs-Landschaft sollte allerdings explizit getroffen werden – und nicht irgendwie passieren – und z.B. im RMP dokumentiert werden, sodass für alle Beteiligten klar ersichtlich ist, welche Anforderungs-Artefakt-Typen auf welcher Detaillierungen-Ebene zu dokumentieren sind. Hierzu bietet sich eine tabellarische Beschreibung der Anforderungs-Landschaft an (vgl. Tabelle 1). Neben der tabellarischen Beschreibung ist die Erstellung eines Informationsmodells in Form eines Entity-Relationship Diagramms oder eines Klassendiagramms sinnvoll, um Beziehungen zwischen den Artefakt-Typen auf einer bzw. auf den unterschiedlichen Detaillierungsebenen zu beschreiben. Dieses beschreibende Informationsmodell nennen wir im Folgenden **Requirements-Information-Model (RIM)**.

Das Requirements-Information-Model kann neben den oben getroffenen Festlegungen, welche Anforderungsart mit welcher Lösungsabhängigkeit auf welcher Detaillierungsebene durch welche Form dokumentiert wird, um weitere Aspekte erweitert werden:

- Welche Attribute werden für welche Artefakt-Typen verwendet? (siehe Kapitel 3)
- Welche Sichtenbildung wird unterstützt? (siehe Kapitel 3)
- Welche Bewertungskriterien sind für Anforderungen vorgesehen? (siehe Kapitel 4)
- Welchen Rollen obliegt die Pflege und Änderung? (siehe Kapitel 5)
- Welche Verfolgbarkeits-Beziehungen werden zwischen Anforderungs-Artefakten sowie vor- und nachgelagerten Artefakten dokumentiert? (siehe Kapitel 6)
- Wie werden Varianten von Anforderungen dokumentiert? (siehe Kapitel 7)

Mit diesen Informationen stellt das Requirements-Information-Model einen wesentlichen Teil des Requirements-Management-Plan (RMP) dar. Aus diesem Grund muss das RIM für alle Stakeholder im Zugriff sein und jederzeit eingesehen werden können.

In Abbildung 2 zeigen wir das RIM von Peter Reber. Peter Reber hat sein RIM, in Anlehnung an Tabelle 1, in drei Detaillierungsebenen unterteilt (Geschäfts-Ebene, Benutzer-Ebene, System-Ebene). Aus den theoretisch 27 möglichen Anforderungs-Artefakt-Typen wurden durch Peter Reber 13 ausgewählt, um die Anforderungen für das neue Banksystem zu spezifizieren.

Ziele auf Geschäfts-Ebene (Detaillierungsebene 1) werden in Geschäftsziele, Geschäftsregeln, Service-Qualität und Geschäftsprozesse unterschieden.

Geschäftsziele beschreiben hier i.d.R. Randbedingungen, die in der Umsetzung zu berücksichtigen sind, z.B. geplanter Starttermin, Unternehmensrichtlinien etc.

Geschäftsprozesse können der Kategorie der Szenarien zugeordnet werden und beschreiben überwiegend funktionale Anforderungen. Die Geschäftsprozesse werden bei Peter durch BPMN beschrieben.

Geschäftsregeln beschreiben funktionale Anforderungen auf hoher Ebene, die im Folgenden berücksichtigt werden müssen und den Lösungsraum einschränken. Unter Geschäftsregeln versteht Peter beispielsweise Begrenzungen der Höhe für Online-Überweisungen pro Tag.

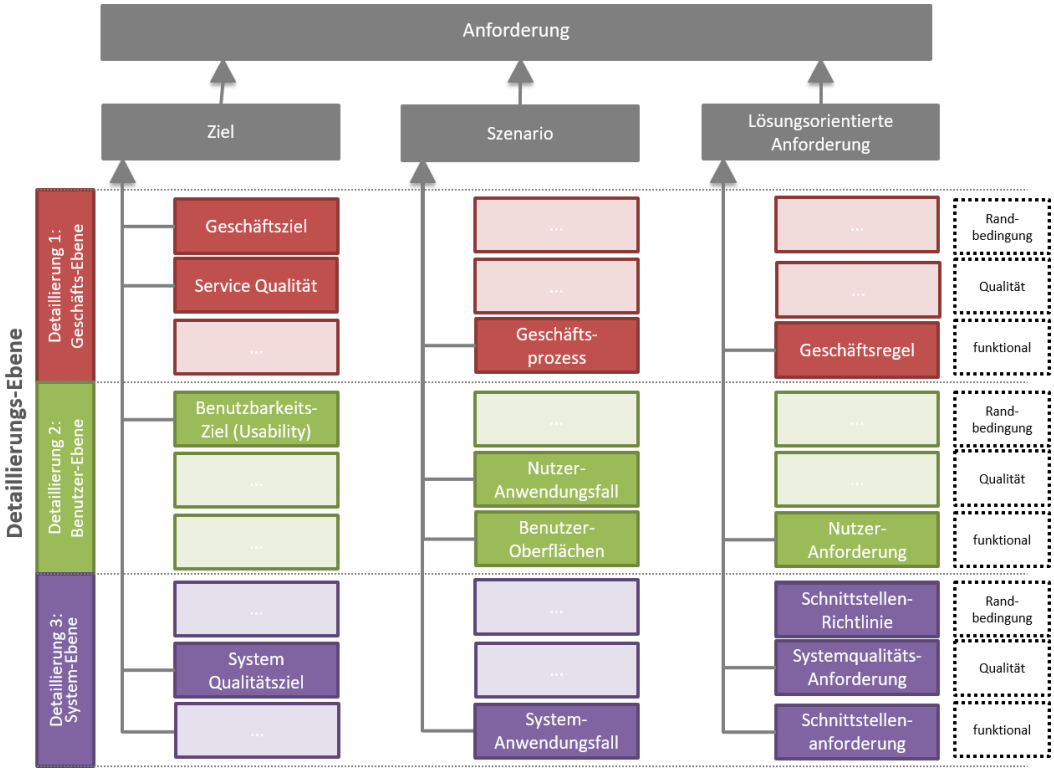


Abbildung 2: beispielhaftes Requirements-Information-Model

Auf der Benutzer-Ebene möchte Peter Benutzbarkeits-Ziele (Usability), Anforderungen an die Benutzer Oberfläche (GUI) sowie Nutzer-Anwendungsfälle und Nutzer-Anforderungen dokumentieren. Nutzer-

Anwendungsfälle sollen beispielsweise durch Use Case Diagramme und Templates beschrieben werden.

Aus lösungsorientierter Sicht werden Benutzer-Anforderungen und Benutzeroberflächen-Anforderungen beschrieben. Hierzu sollen Mock-Ups, textuelle Beschreibungen oder ER-Modelle genutzt werden.

Auf der IT-Ebene finden sich Schnittstellen-Richtlinien, System-Qualitätsziele, System-Anwendungsfälle, Schnittstellen-Anforderungen und Systemqualitäts-Anforderungen. Die Darstellungsformen auf der System-Ebene sind stärker an die IT-Entwicklung angelehnt, sodass hier neben textuellen Anforderungen auch modellbasierte Notationen wie Aktivitätsdiagramme und Message-Sequence-Charts (Sequenzdiagramme) zum Einsatz kommen können.

Im vorliegenden Modell wurde nur die Dimension der Lösungsabhängigkeit auf unterschiedlichen Detaillierungsebenen beschrieben. Die Dimension der Anforderungsart (Randbedingung, Qualitätsanforderung, funktionale Anforderung), ist in diesem Modell nur strukturell am rechten Rand dargestellt. Die entsprechende Darstellungsform wurde im RIM vollständig außen vorgelassen, da die Darstellung von mehr als zwei Dimensionen schnell unübersichtlich wird. Bei der Verwendung eines Modellierungs-Werkzeugs können Sie hier eventuell verschiedene Sichten auf das Informationsmodell anbieten, um eine Modell-Sicht nicht zu sehr zu überfrachten.

Eine weitere Möglichkeit, mehr Details in ein RIM zu bringen, ist die Verwendung von Annotationen an einer Klasse, um die unterschiedlichen Dimensionen an der Klasse zu beschreiben (siehe Abbildung 3).

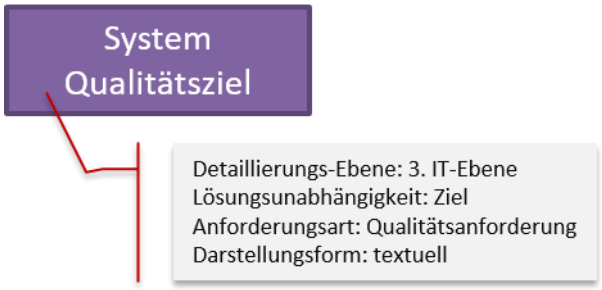


Abbildung 3: Beispiel RIM mit Annotation der Dimensionen-Details

Alternativ gibt es neben dem RIM die Möglichkeit, dass die Darstellungsformen oder die Zuordnung zu Detaillierungsebenen über eine zusätzliche tabellarische Beschreibung erfolgt (vgl. Tabelle 1).

Zur Prüfung des RIM können folgende Kontrollfragen angewendet werden:

- **Prüfung auf formale Vollständigkeit**:
  Ist für jede Anforderungsklasse ersichtlich, welche Anforderungsart dahintersteht, welche Lösungsabhängigkeit diese Anforderungsklasse hat, auf welcher Detaillierungsebene diese Anforderungsklasse existiert und in welcher Darstellungsform sie dokumentiert wird?
- **Prüfung auf inhaltliche Zusammenhänge**:
  Ist im RIM klar ersichtlich, welche Detaillierungsebenen existieren und wie diese zusammenhängen? Ist klar ersichtlich wie die Anforderungen auf den verschiedenen Detaillierungsebenen voneinander abhängen?
- **Prüfung auf Angemessenheit**:
  Ist die Gesamtheit der gewählten Anforderungsklassen geeignet, um hinreichend detaillierte, vollständige und konsistente Anforderungen zu dokumentieren, sodass die nachfolgenden Aktivitäten (z.B. Entwicklung und Test) ihrer Aufgabe vollständig nachkommen können?

2.4 Inhalte für den RMP

Mit der Anforderungs-Landschaft erstellen Sie für Ihren RMP einen wesentlichen ersten Bestandteil. Mit der Anforderungs-Landschaft legen Sie fest, welche Anforderungs-Artefakt-Typen Sie berücksichtigen wollen, auf wie vielen Detaillierungsebenen Sie Anforderungen definieren wollen und in welchen Darstellungsformen welche Anforderungs-Artefakt-Typen spezifiziert werden sollen (vgl. Tabelle 1 und Abbildung 2). Durch die Beschreibung der Anforderungs-Landschaft (z.B. durch ein RIM) können Sie über den RMP dafür sorgen, dass alle am Projekt beteiligten Stakeholder ein einheitliches Verständnis bekommen, durch welche Anforderungs-Artefakt-Typen Anforderungen dokumentiert werden, auf welchen Detaillierungsebenen und mittels welcher Darstellungs-Formen Anforderungen dokumentiert werden.

2.5 Vertiefende Literatur

[BBHK2014] Braun, P.; Broy, M.; Houdek, F.; Kirchmayr, M.; Müller, M.; Penzenstadler, B.; Pohl, K.; Weyer, T.: Guiding Requirements Engineering for software-intensive embedded systems in the automotive industry. Computer Science – R&D 29(1): 21-43 (2014).

[Eber2012] C. Ebert: Systematisches Requirements Engineering. Dpunkt, 4. Auflage, 2012

[CHQW2022] Thorsten Cziharz, Peter Hruschka, Stefan Queins, Thorsten Weyer: Handbuch Requirements Modeling, Aus- und Weiterbildung zum IREB Certified Professional for Requirements Engineering, Advanced Level Requirements. IREB, Version 2.0.0. Juli 2022.

[PHAB2012] Pohl, K., Hönninger, H., Achatz, R., Broy, M. (Eds.): Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology, Springer 2012.

[Pohl2010] K. Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. Springer, 2010.

[WiBe2013] K. Wiegers and J. Beatty: Software Requirements. 3rd Edition. Microsoft Press, 2013.

3 Attributierung und Sichten bei Anforderungen

In diesem Kapitel geht es darum, wie das RM Anforderungs-Attribute{ XE "Anforderungs-Attribut" } festlegt und welche in Projekten benötigt werden. Außerdem geht es um das Erstellen, Nutzen und Ändern von Attributierungsschemata und Sichten.

Definition 3-1:

Ein *Attribut* ist eine charakteristische Eigenschaft einer Einheit.

(aus dem IREB-Glossar [Glin2014])

Der Standard ISO/IEC/IEEE 29148:2011 ISO29148] fügt der Definition eines Attributs noch den Aspekt hinzu, wie Attribute ausgewertet werden: „eine inhärente Eigenschaft oder Charakteristikum einer Einheit, die quantitativ oder qualitativ durch einen Menschen oder automatisiert unterschieden werden kann".

Attribute sind im Zusammenhang mit dem RM also Eigenschaften der Anforderungen, z.B. der Bearbeitungsstatus (Attribut „Status"). Attribute werden als Metainformationen üblicherweise nicht mit der Anforderungsbeschreibung vermischt, sondern getrennt dokumentiert und verwaltet, z.B. in einer tabellarischen Liste von Anforderungen als eigene Spalte und in einer Anforderungsdatenbank als eigenes Feld. Nicht nur textuelle Anforderungen können Attribute haben, sondern auch Elemente eines UML-Modells [CHQW2022]. Trotz Namensgleichheit sind Anforderungs-Attribute nicht synonym zu den Attributen einer Klasse im Klassendiagramm. Diese sind Inhalt der Anforderung, aber keine Metainformation, also keine Anforderungs-Attribute im Sinne dieses Kapitels. Anforderungen aller Arten, Detaillierungsebenen und Typen können Attribute besitzen, manchmal haben sie jedoch nicht dieselben. Auch Änderungsanträge / Change Requests haben Attribute (siehe Kapitel 5). Ganze Dokumente können durch Attribute charakterisiert werden, z.B. einen Status oder eine Versionsnummer.

3.1 Ziele der Attributierung und Beispiele ihrer Verwendung in Managementtätigkeiten

Wie die obigen Definitionen andeuten, dienen Attribute im RM dazu, Anforderungen zu kategorisieren und zwar im Hinblick auf Metainformationen, die z.B. für die Release-Planung oder Verwaltung nötig sind. Dadurch können Sie sich einen Überblick über die Anforderungen verschaffen. Gerade bei umfangreichen Projekten überblickt niemand mehr alle Anforderungen. Hier helfen Attribute bei jeder durchzuführenden anforderungsbezogenen Aktivität im Software Engineering, sich auf die wesentlichen Informationen zu konzentrieren, d.h. auf die betroffenen Anforderungen und deren relevante Eigenschaften. Je nach Aktivität interessiert natürlich ein anderer Auszug aus den vorhandenen Informationen.

Attribute einer Anforderung beantworten typischerweise eine Menge wichtiger Fragen, wie z.B.: „Wer hat eine Anforderung wann das letzte Mal geändert?" oder „Welche Anforderungen sind für Release 1 geplant?" oder „Wie viel Aufwand verursacht Release 1 voraussichtlich insgesamt?". Praktischerweise schreibt man nicht alle eine Anforderung betreffenden Meta-Informationen in ein einziges Freitext-Feld. Jedes Attribut wird in einem eigenen Feld verwaltet.

Oft werden hier Wertelisten vorgeben, die für alle Anforderungen einheitlich sind. Beispielsweise das Attribut „Priorität" lässt sich leichter auswerten, wenn es nur die Werte „gering", „normal" und „hoch" zulässt oder auch eine andere Abstufung bzw. Werteliste. Wäre es ein Freitextfeld, stünden hier Kommentare wie „ziemlich wichtig" oder „Herr Meier sagt, die Anforderung sei wichtig". Solche Inhalte unterstützen die Unterscheidung zwischen den wichtigsten und weniger wichtigen Anforderungen nicht gerade! Dies fällt viel leichter, wenn man sich durch einfaches Filtern eine Liste aller als „hoch" eingestuften Anforderungen anzeigen lassen kann. Welches Format und welche Attributwerte beispielsweise für die Priorität am meisten Sinn machen, hängt u.a. davon ab, welche Sichten und Entscheidungen durch die Prioritäten unterstützt werden sollen (vgl. Kapitel 4).

Die Attributierung von Anforderungen hat also zum Ziel, dass Teammitglieder und andere Stakeholder im Rahmen eines Requirements-Engineering-Prozesses strukturiert Informationen zu Anforderungen dokumentieren und auswerten können [Pohl2010].

Welche Attribute benötigt werden und welche Werte dort erlaubt werden, sollte man sich anfangs gut überlegen, da es ist nicht einfach, ein Attributierungsschema nachträglich zu ändern (siehe Kapitel 3.5). Leider gibt es kein Attributierungsschema, das immer und überall ideal passt. Entscheidend ist stets, welche Attribute man später auf welche Weise auswerten möchte.

Verschiedene Autoren schlagen unterschiedliche Zusammenstellungen an Attributen vor, die sich nach ihrer Erfahrung praktisch bewährt haben. Zu den wichtigen Anforderungs-Attributen gehören laut CPRE Foundation Level die Attribute in Tabelle 2 und Tabelle 3 [PoRu2015].

Attributtyp	Bedeutung
Identifikator	Kurze, eindeutige Identifikation eines Anforderungs-Artefakts in der Menge der betrachteten Anforderungen
Name	Eindeutiger, charakteristischer Name
Beschreibung	Beschreibt in komprimierter Form den Inhalt der Anforderungen
Version	Aktueller Versionsstand der Anforderung
Autor	Benennt den/ die Autor/in der Anforderung
Quelle	Benennt die Quelle bzw. Quellen der Anforderung

Attributtyp	Bedeutung
Begründung	Beschreibt, weshalb diese Anforderung für das geplante System von Bedeutung ist
Stabilität	Benennt die voraussichtliche Stabilität der Anforderung. Stabilität ist dabei der Umfang, in dem künftig noch Veränderungen bzgl. dieser Anforderung erwartet werden. Mögliche Unterscheidung: "stabil", "volatil"
Kritikalität	Im Sinne einer Abschätzung der Schadenshöhe und Eintrittswahrscheinlichkeit
Priorität	Benennt die Priorität der Anforderung hinsichtlich der gewählten Merkmale zur Priorisierung, z.B. „Bedeutung für die Akzeptanz am Markt", „Reihenfolge der Umsetzung", „Schaden bzw. Opportunitätskosten durch Nichtrealisierung"

Tabelle 2: Häufig verwendete Attributtypen [PoRu2015].

Ein weiterer Vorschlag für eine Zusammenstellung von Attributen:

Attributtyp	Bedeutung
Verantwortliche(r)	Benennt die Person, Stakeholdergruppe bzw. Organisation(seinheit), die für diese Anforderung inhaltlich verantwortlich ist
Anforderungsart	Benennt abhängig vom eingesetzten Differenzierungsschema den Typ der Anforderung (z.B. „funktionale Anforderung", „Qualitätsanforderung" oder „Randbedingung")
Status bzgl. des Inhalts	Benennt den aktuellen Status des Inhalts der Anforderung, z.B. „Idee", „Konzept", „detaillierter Inhalt"
Status bzgl. der Überprüfung	Benennt den aktuellen Status der Validierung, z.B. „ungeprüft", „in Prüfung", „überprüft", „fehlerhaft", „in Korrektur"
Status bzgl. der Einigung	Benennt den aktuellen Status der Abstimmung, z.B. „nicht abgestimmt", „abgestimmt", „konfliktär"
Aufwand	Prognostizierter / tatsächlicher Umsetzungsaufwand dieser Anforderung
Release	Name und / oder Nummer des Releases, in dem die Anforderung umgesetzt werden soll

Attributtyp	Bedeutung
Juristische Verbindlichkeit	Gibt den Grad der juristischen Verbindlichkeit der Anforderung an, z.B. „muss", „soll" und „kann"
Querbezüge	Benennt die Beziehungen zu anderen Anforderungen: zum Beispiel wenn bekannt ist, dass die Realisierung dieser Anforderung die vorherige Realisierung einer anderen Anforderung voraussetzt (vgl. Kapitel 6)
Allgemeine Informationen	In diesem Attribut können beliebige, für relevant erachtete Informationen zu dieser Anforderung dokumentiert werden: zum Beispiel wenn die Abstimmung dieser Anforderung auf dem nächsten Treffen mit dem Auftraggeber vorgesehen ist

Tabelle 3: Häufig verwendete Attributtypen [PoRu2015].

Die obige Liste enthält alle der im Standard ISO/IEC/IEEE 29148:2011 ISO29148 genannten Attribute und enthält darüber hinaus noch weitere Attribute.

Für die Dokumentation der Verfolgbarkeit wäre es zusätzlich zur Dokumentation der Quelle noch nützlich, in einem weiteren Attribut zu dokumentieren, in welcher technischen Komponente eine Anforderung umgesetzt wurde (Attribut „technische Komponente") und durch welche Testfälle die Anforderung getestet wird (Attribut „Testfälle") (vgl. Kapitel 1).

Ein weiteres Attributierungsschema mit 7 Attribut-Kategorien empfiehlt Pohl [Pohl2010]. Dieses unterscheidet die Kategorien „Identifizierbarkeit", „Kontextbeziehungen", „Dokumentationsaspekte", „Inhaltsaspekte", „Übereinstimmungsaspekte", „Validierungsaspekte" und „Managementaspekte". Jede der genannten Kategorien enthält eine Reihe von möglichen Attributen.

- **Identifizierbarkeit:** Dies sind diejenigen Attribute, welche die Identifizierbarkeit eines Attributs ermöglichen. Dazu gehören die ID und der Name, der möglichst sprechend den Inhalt der Anforderung beschreibt.
- **Kontextbeziehungen:** Diese Attribute dokumentieren die Beziehungen der Anforderungen zum Kontext, z.B. die Quelle, die Begründung, außerdem die verantwortliche Person und evtl. betroffene Stakeholder, mit denen man eine Anforderungsänderung abstimmen müsste.
- **Dokumentationsaspekte:** Hier wird abgelegt, in welcher Darstellungsform eine Anforderung spezifiziert wird (Freitext, UML-Modell, Textschablone, etc.), ein Link zu einem Dokument, in dem die Spezifikationsregeln abgelegt sind, und der Validierungsstatus der Anforderung(sdokumentation) (z.B. ungeprüft / in Prüfung / partiell geprüft / überprüft / in Korrektur / freigegeben).
- **Inhaltsaspekte:** Diese Attribute dokumentieren und klassifizieren die Inhalte der Anforderung. Dazu gehören insbesondere die Beschreibung, aber auch die Anforderungsart, Anmerkungen des Erstellers, der Status des Inhalts (Idee / grober

Inhalt / detaillierter Inhalt) und Querbezüge zu anderen Entwicklungs-Artefakten (Verfolgbarkeits-Beziehungen).

- **Übereinstimmungsaspekte:** Diese Attribute dokumentieren die Übereinstimmung der Stakeholder, z.B. einen Übereinstimmungsstatus (nicht bekannt / konfliktär / in Übereinstimmung / übereingestimmt), einen Validierungsstatus der Übereinstimmung (ungeprüft / in Prüfung / partiell geprüft / überprüft / in Korrektur / freigegeben), ein Freitextfeld jeweils für erkannte Konflikte und Entscheidungen.
- **Validierungsaspekte:** Die Validierung überprüft die Qualität einer Anforderung im Hinblick auf die drei Dimensionen Inhalt, Dokumentation und Übereinstimmung. Hier kann dokumentiert werden: Konformität zu Eingangskriterien der Validierung (d.h.: Kann mit der Validierung begonnen werden?), Techniken zur Validierung, aktueller Validierungsschritt, Status der Validierung insgesamt (ungeprüft / in Prüfung / partiell geprüft / überprüft / in Korrektur / freigegeben).
- **Managementaspekte:** Diese Attribute dokumentieren den Status einer Anforderung und andere Managementinformationen. Dazu gehören Stabilität, Kritikalität und Priorität, juristische Verbindlichkeit und weitere Status-Informationen. Auch der Autor der Anforderung, Version, Änderungshistorie, System-Release, Soll- und Ist-Aufwand.

Tipp aus der Praxis: Überlegen Sie sich frühzeitig ganz genau, welche Attribute Ihr Attributierungsschema enthalten soll, um spätere Änderungen möglichst zu vermeiden. Fügen Sie nur solche Attribute dem Attributierungsschema hinzu, die zwei Kriterien erfüllen: (1) Sie sind sicher, dass die zuständige Person während ihres Arbeitsablaufs dieses Attribut pflegen wird; (2) Es ist klar, wer wann welchen Nutzen von diesem Attribut hat, indem er es auswertet. Beide Kriterien müssen erfüllt sein, damit sich der Aufwand für die Dokumentation dieser Meta-Information lohnt.

3.2 Was ist ein Attributierungsschema?

Definition 3-2:

„Die Menge aller definierten Attribute für eine Klasse von Anforderungen (z.B. funktionale Anforderungen, Qualitätsanforderungen) wird als das *Attributierungsschema* bezeichnet.“ ([PoRu2015], Kap. 8.1.2)

Ein Attributierungsschema beschreibt die für ein Projekt und/oder ein Unternehmen relevanten Attribute für Anforderungen [RuSo2009]. Zum Attributierungsschema gehören außer den Namen und der Definition der Attribute auch deren Format (Text oder Zahl? Wie viele Zeichen darf das Attribut haben?) und die Festlegung der erlaubten Werte bzw. Wertebereiche.

Ein Attributierungsschema für Anforderungen bereitzustellen (schablonen- bzw. template-basiert), führt zu folgenden Vorteilen im RM [Pohl2010]:

- **Genaue und einheitliche Festlegung der benötigten Informationen**: Durch ein vorgegebenes Schema ist definiert, welche Informationen bzw. Attribute zu Anforderungen erfasst werden müssen und welche Werte für diese Informationen erlaubt sind.
- **Erkennung von Lücken**: Es können Lücken in der Bearbeitung von Anforderungen entdeckt werden, wenn bestimmte Attribute leer sind.
- **Unterstützung der Einarbeitung von Mitarbeitern**: Mitarbeiter, die z.B. in einem vorherigen Projekt bereits mit dem gleichen oder ähnlichen Attributierungsschema gearbeitet haben, finden schnell die notwendigen Informationen bzw. die Stellen, an denen bestimmte Informationen zur Anforderung dokumentiert werden sollen.
- **Vorfinden von gleichen Informationen an jeweils gleicher Stelle**: Da alle Anforderungen im Rahmen eines Projektes auf der Basis desselben Attributierungsschema dokumentiert werden, ist festgelegt, wo welche Informationen – wie z.B. der Autor – zu einer Anforderung zu finden sind.

Das Attributierungsschema ist Teil des RMP, aber nicht der Anforderungs-Landschaft. Manchmal gibt es Unternehmensstandards, oder die Verwendung eines offiziellen Standards oder Branchenstandards ist vorgeschrieben. Diese enthalten meist Vorgaben über die zu verwendenden Attribute und deren Werte. Dieses standardisierte Attributierungsschema erlaubt dann projektübergreifende und vergleichbare Auswertungen über das RM (vgl. Kapitel 8).

Außer dem Requirements Manager nutzen auch noch andere Rollen im Entwicklungsprozess und im Unternehmen die in den Anforderungsattributen festgehaltenen Meta-Informationen zu Anforderungen. Beispielsweise interessiert sich der Projektmanager regelmäßig für den Bearbeitungsstand der Anforderungen. Bei der Definition eines Attributierungsschemas ist also auch der Informationsbedarf anderer Stakeholder des RE-Prozesses zu berücksichtigen. Mit den verschiedenen Rollen und deren Informationsbedarf beschäftigen wir uns im Zusammenhang mit Sichten weiter unten erneut (siehe Kapitel 3.6). Letztlich sind die benötigten Sichten die Grundlage für die Definition des Attributierungsschemas.

3.3 Nutzen eines Attributierungsschemas

Eine Vielzahl von RM-Aufgaben und andere Managementaufgaben werden durch Attribute unterstützt:

- Sichten: Attribute sind die Grundlage für die Definition und Umsetzung von Sichten in einem Werkzeug (siehe Kapitel 3.6).
- **Priorisierung:** Die jeweilige Priorität einer Anforderung wird in einem oder mehreren entsprechenden Attributen dokumentiert. Dabei können mehrere Attribute für unterschiedliche Priorisierungskriterien definiert werden. Diese Prioritäten unterstützen wiederum Entscheidungen, die auf den Prioritäten beruhen, z.B. die Release-Planung. Üblicherweise möchte man die wichtigsten Anforderungen zuerst umsetzen. Diese Wichtigkeit kann wiederum von mehreren Attributen abhängen, z.B.

wenn Nutzen und Kosten gegeneinander abgewogen werden. Mehr über Priorisierung finden Sie in Kapitel 4.

- **Projektmanagement:** Den Projektmanager interessiert der prognostizierte (oder auch der tatsächliche) Realisierungsaufwand je Anforderung. Dieser wird in einem entsprechenden Attribut erfasst. Dieses Attribut erlaubt nicht nur durch Summenbildung den Gesamtaufwand für eine Gruppe von Anforderungen (beispielsweise aller wichtigen Anforderungen oder der Anforderungen eines Releases) zu ermitteln. Es unterstützt auch während des Projektes das Projekt-Monitoring.
  Projektmanagementberichte, in denen nur über die reine Anzahl von Anforderungen in einem bestimmten Status berichtet wird, verfälschen die Sicht auf den Projektstand, weil nicht jede Anforderung gleich umfangreich ist. Gewichtet man in diesen Berichten die Anforderungen aber nach ihrem Realisierungsaufwand, so erhält man ein sehr gutes Bild vom Status des Gesamtprojektes. Das gilt insbesondere, wenn das Status-Attribut der Anforderungen den gesamten Lebenszyklus abdeckt, also nicht nur, ob eine Anforderung abgenommen und an die Entwicklung übergeben wurde, sondern auch, ob sie realisiert wurde, ob deren Umsetzung bereits getestet ist, ob sie von der Qualitätssicherung als fehlerfrei freigegeben wurde, ob der Kunde die zugehörige Funktionalität abgenommen hat und ob sie ausgeliefert ist.
- **Release-Management:** Die Prioritäten unterstützen die Release-Definition und das Release-Management, d.h. die Verwaltung der verschiedenen an Kunden auszuliefernden Software-Stände. Das Release-Management wird durch ein entsprechendes Attribut „Release" unterstützt. Dieses Attribut dokumentiert, welche Anforderungen in welchem Release umgesetzt sind. Vielfach wird hierbei auch zwischen dem gewünschten und dem geplanten Release unterschieden, um den oftmals auftretenden Unterschied zwischen diesen beiden Wirklichkeiten darstellen zu können (vgl. Kapitel 5).
- **Risikomanagement:** Die Attribute „Kritikalität", aber auch „Stabilität" und „juristische Verbindlichkeit" unterstützen das Identifizieren und Abschätzen von Risiken, die mit einer Anforderung verbunden sind. Diese Risikobetrachtung ist wiederum auch für den Projektleiter und die Release-Planung relevant. Entscheidend für die Definition von Attributen zur Unterstützung des Risikomanagements sind die Kriterien, welche im Risikomanagement verfolgt werden, und welche Auswertungen notwendig sind.
- **Verfolgbarkeit:** Die Verfolgbarkeit von Anforderungen ist wichtig, wenn man während des Änderungsmanagements die Auswirkungen von Anforderungsänderungen vorhersehen oder Konflikte zwischen einander widersprechenden Anforderungen lösen möchte. Die Verfolgbarkeit sollte in beide Richtungen gegeben sein: Zur Quelle einer Anforderung und zu späteren Artefakten wie technischen Komponenten und Testfällen. Aber auch Abhängigkeiten zwischen Anforderungen desselben Typs und Verfeinerungs-Beziehungen zwischen Anforderungen auf verschiedenen Detaillierungsebenen sollten dokumentiert sein – wenn ihr Nutzen den Aufwand rechtfertigt. Mehr über Verfolgbarkeit von Anforderungen erfahren Sie in Kapitel 6.

- **Variantenmanagement:** Im Rahmen des Variantenmanagements (siehe Kapitel 7) können mit Hilfe von Attributen Anforderungen bestimmten Varianten und Produkt-Konfigurationen zugeordnet werden.
- **Berichtswesen:** Attribute bilden eine Grundlage für Berichte, wie zum Beispiel eine Auswertung über die jeweilige Anzahl von Anforderungen in einem bestimmten Status (wie z.B. „freigegeben" oder „getestet"). Mehr über RM-Berichte finden Sie in Kapitel 8.

3.4 Entwurf eines Attributierungsschemas

Das Attributierungsschema{ XE "Attributierungsschema" } ist Teil des RMPs. Die Definition eines Attributierungsschemas sollte vor dem Beginn der Anforderungs-Dokumentation erfolgen und mit allen Stakeholdern des RE-Prozesses abgestimmt sein. Nachträgliche Ergänzungen und Änderungen lassen sich meist nur unter sehr hohem Aufwand durchführen.

Wir empfehlen für den Entwurf eines Attributierungsschemas in einem konkreten Projekt folgende Schritte, die in den folgenden Abschnitten jeweils genauer betrachtet werden:

1. Identifizieren von Attribut-Quellen
2. Auswahl der Attribute
3. Definition von zulässigen Attributwerten und Eigenschaften der Attribute
4. Definition von Abhängigkeiten zwischen Attributen und ihren Werten
5. Bereitstellung einer Erfassungsunterstützung
6. Dokumentation des Attributierungsschemas

3.4.1 Identifizieren von Attribut-Quellen

Zur Auswahl von Attributen ist es zunächst notwendig, die relevanten Quellen für Attribute zu identifizieren. Quellen, die zur Auswahl von Attributen dienen können, sind z.B.:

- Ein Attributierungsschema aus einem ähnlichen Projekt (z.B. ähnlich hinsichtlich des Umfangs, der Anzahl von beteiligten Mitarbeitern etc.)
- Ein Referenzschema des Unternehmens oder ein anderer Standard wie in Kapitel 3.1
- Regeln im Unternehmen, die z.B. festlegen, welche Attribute in allen Attributierungsschemata in allen Projekten eingesetzt werden müssen
- Stakeholder des RE-Prozesses

Schemata aus Standards können Sie übernehmen oder wie benötigt anpassen.

Werden in einem (in der Regel größeren) Unternehmen in vielen unterschiedlichen Projekten Attributierungsschemata definiert (z.B. durch Wiederverwendung und anschließende Anpassung), ist es zweckmäßig, allgemeine Regeln für die Bildung eines Attributierungsschemas festzulegen. Hierbei kann z.B. definiert werden, dass die Auswahl eines Attributs „Stabilität" für ein Attributierungsschema auch die Auswahl des Attributs „Risiko" nach sich zieht, und somit beide Attribute im Attributierungsschema für das jeweilige Projekt vorhanden sein müssen. Dies ist etwa dann sinnvoll, wenn das Unternehmen

bei der Beurteilung der Stabilität einer Anforderung parallel immer auch abschätzen möchte, wie hoch das Risiko der Anforderung im Hinblick auf z.B. die Zeitplanung zu bewerten ist.

Des Weiteren kann ein Unternehmen für die Auswahl von Attributen für ein Attributierungsschema zuvor allgemein festlegen, welche Attribute immer und in jedem Projekt berücksichtigt werden müssen, um z.B. projektübergreifende Auswertungen zu ermöglichen. Diese Attribute sind konsequenterweise in jedem Projekt vorhanden und sollten als Pflichtfelder deklariert sein.

Eine weitere Quelle von Attributen für ein Attributierungsschema bilden die Stakeholder des RE-Prozesses (siehe auch Kapitel 3.6). Es werden zunächst die Stakeholder identifiziert und dann deren Bedürfnisse.

Peter Reber beschließt, das Attributierungsschema des IREB als ersten Entwurf zu verwenden (siehe Tabelle 2 und Tabelle 3).

Die Stakeholder des RE-Prozesses sind der Projektleiter, die Business Analysten, die IT-Sicherheits-Experten, die Usability-Expertin und der Kundenbeirat, aber auch gesetzliche Risikomanagementanforderungen wie BASEL II.

Bei der Übernahme eines Referenzsystems werden leicht auch unnötige Attribute übernommen. Deshalb geht Peter Reber so vor, dass er zunächst jeden Stakeholder einzeln nach seinen Bedürfnissen befragt. Wird dabei ein Attribut des Referenzschemas nicht genannt, so wird er prüfen, ob es wirklich niemand braucht. Vielleicht haben die Stakeholder nur vergessen, es zu nennen.

Im ersten Schritt nennen die Stakeholder folgende Attribute:

- Projektleiter: Status und Aufwand der Anforderungen, Release, für das eine Anforderung geplant ist, und Status der Artefakte, vor allem aber auch ein über den Aufwand gewichteter Überblick, welcher Anteil der Anforderungen (eines Release) sich in welchem Status befindet
- Business Analyst: ein Prioritätswert, der den Nutzen einer Anforderung für die Bank misst
- IT-Sicherheits-Experten: Kritikalität
- Usability-Expertin und Kundenbeirat: Priorität im Sinne von „Nutzen für die Benutzer / Kunden der Bank"
- BASEL II: „Autor" und „Version" erlauben die Nachvollziehbarkeit, wer wann was geändert hat, „Begründung" dokumentiert auch den Grund. Die Attribute „Stabilität" und „juristische Verbindlichkeit" dienen dem Risikomanagement.

3.4.2 Auswahl der Attribute

Die Attribute müssen zielgerichtet und für das Projekt passend ausgewählt werden, damit sie wirklich nützen. Dies gilt sowohl dann, wenn Sie ein Referenzschema verwenden, als auch wenn Sie ein Attributierungsschema neu definieren.

Wenn Sie ein Referenzschema als Grundlage verwenden, prüfen Sie für jedes einzelne Attribut, ob Sie es brauchen und wofür. Es wird dann übernommen, angepasst oder (projektspezifisch) entfernt. Außerdem können Sie neue Attribute ergänzen. Wenn Sie kein Referenzschema verwenden, müssen Sie ein Attributierungsschema neu erstellen. Dazu gilt es, entsprechende Attribute zu identifizieren, z.B. durch die Befragung der relevanten Stakeholder (siehe Schritt 1).

Die ID (Identifikation) einer Anforderung spielt eine besondere Bedeutung bei der Attributierung von Anforderungen. Sie dient dazu, jede Anforderung eindeutig identifizieren zu können, und ist ein Pflichtattribut in jedem Attributierungsschema. Im Unternehmen muss festgelegt werden, in welchem Kontext die Eindeutigkeit gegeben sein soll. Dabei kann der Kontext sich an der Organisationsstruktur orientieren und z.B. festlegen, dass die Eindeutigkeit nur innerhalb von Abteilungen gelten muss. Eine andere Möglichkeit der Abgrenzung kann in technischen Randbedingungen liegen, z.B. dass die Eindeutigkeit nur in der jeweils genutzten Datenbank gegeben sein muss.

Hierbei ist festzulegen, wie mit IDs von Anforderungen verfahren wird, die zwischen diesen Datenbanken ausgetauscht werden, da ein solcher Austausch gegebenenfalls zu mehrdeutigen (weil mehrfach vorkommenden) IDs führen kann.

Um Attribute für ein Attributierungsschema auszuwählen sowie zu beurteilen, ob ein Schema vollständig ist, können die in [Pohl2010] vorgestellten sieben Kategorien als Checkliste eingesetzt werden: „Identifizierbarkeit", „Kontextbeziehungen", „Dokumentationsaspekte", „Inhaltsaspekte", „Übereinstimmungsaspekte", „Validierungsaspekte" und „Managementaspekte" (vgl. Kapitel 3.1). Alle sieben Kategorien sollen in einem Attributierungsschema abgedeckt sein.

Um ein Attributierungsschema mit Hilfe der genannten Kategorien zu definieren, führen Sie am besten folgende Aktivitäten durch:

- Für jede Kategorie prüfen, welche der vorgeschlagenen Attribute bereits für das Projekt ausgewählt wurden (z.B. durch die Wahl eines Referenzschemas). Dabei ist zu beachten, dass manchmal in verschiedenen Schemata dasselbe Attribut verschiedene Namen trägt oder unter derselben Bezeichnung ein anderer Inhalt verstanden wird. Zum Beispiel kann das in der Kategorie „Kontextbeziehungen" aufgeführte Attribut „Quelle" bereits durch ein Attribut „Basis" in einem Referenzschema abgebildet sein. Um diese Übereinstimmung feststellen zu können, ist es notwendig, dass die Semantik des jeweiligen Attributs dokumentiert und nachvollziehbar ist.
- Systematische Betrachtung der einzelnen Kategorien und der dort vorgeschlagenen Attribute. Bei dieser Aktivität muss für jedes vorgeschlagene Attribut der Nutzen des Attributes für das aktuelle Projekt bewertet und geprüft werden. Nur solche Attribute

sollen verwendet werden, für die bereits klar ist, welcher Stakeholder es für welchen Zweck benutzen wird.

- Erweiterung der Kategorien oder des Referenzschemas. Wenn bei der Analyse zur Auswahl von Attributen ein neues Attribut identifiziert wird, das weder im Referenzschema noch in einer der Kategorien vorhanden ist, so sollte analysiert werden, ob eine Erweiterung einer Kategorie oder des Referenzschemas sinnvoll ist. In dieser Aktivität ist abzuwägen, ob ein neues Attribut im Referenzschema oder in einer Kategorie eingefügt werden soll, um als Vorschlag für die Definition eines Attributierungsschemas in folgenden Projekten zu dienen. Hier gilt es abzuschätzen, wie groß die Wahrscheinlichkeit ist, dass ein neues Attribut in den meisten der folgenden Projekte genutzt werden wird. In diesem Fall ist die Erweiterung des Referenzschemas sinnvoll. Wird das neue Attribut wahrscheinlich eher selten angewendet, so sollte es in einer der Kategorien dokumentiert werden.

In Prozessen zur Definition eines Attributierungsschemas werden vielfach zunächst sehr viele Attribute festgelegt, da z.B. unterschiedliche Stakeholder mit unterschiedlichen Perspektiven Informationen zu einer Anforderung erfassen und auswerten wollen. In der Praxis zeigt sich dann oft, dass Attribute aus gutem Grund nicht gefüllt und nicht genutzt werden. Dann muss das Attributierungsschema angepasst werden, was nicht immer ganz unproblematisch ist (vgl. Kapitel 3.5).

Um derartige spätere Anpassungen weitestgehend zu vermeiden, sollten die Attribute auf eine pragmatisch einsetzbare Menge beschränkt werden. Es sollen daher nur Attribute verwendet werden, die ein klares Ziel oder eine bestimmte Aufgabe unterstützen. Ein solches Ziel kann sein, dass der Projektleiter eine Earned-Value-Analyse durchführen möchte.

Dazu muss er u.a. den Fertigstellungsgrad des Projektes kennen, d.h. eine nach Aufwand gewichtete Berechnung, welcher Anteil der Anforderungen bereits fertiggestellt ist. Hierzu braucht jede Anforderung zwei Attribute: Ein Attribut „Aufwand" und ein Status-Attribut, das den Wert „abgeschlossen" einnimmt, sobald die Arbeiten an der Anforderung abgeschlossen sind. Für eine noch differenziertere Berechnung des Fertigstellungsgrades können auch für frühere Status-Werte bereits Teilerfüllungsgrade berücksichtigt werden (z.B. einen Erfüllungsgrad von 80 %, wenn die Anforderung umgesetzt, aber noch nicht getestet ist). Dabei kann die Formel für den Fertigstellungsgrad natürlich nur Status-Werte verwenden, die das Attribut auch tatsächlich einnehmen wird. Dies klingt zunächst selbstverständlich, doch bei Änderungen des Attributierungsschemas kann die Konsistenz zwischen Attributierungsschema und Sichten/Berichten allzu leicht verloren gehen, und das auch bereits während der Definition des Schemas.

Bei der Diskussion mit den Stakeholdern stellt sich heraus, dass sie es für unnötig kompliziert halten, drei verschiedene Status-Attribute zu verwenden. Es wird darum beschlossen, dass es nur ein einziges Attribut „Status" geben soll. Es kann die folgenden Werte annehmen, um den Lebenszyklus einer Anforderung oder Änderung abzubilden: in Bearbeitung, in Prüfung, freigegeben, geändert, zurückgewiesen, gelöscht, umgesetzt, getestet, abgeschlossen.

Nicht genannt wurden von den Stakeholdern (vgl. Tabelle 2 und Tabelle 3) folgende Attribute: Identifikator, Name, Beschreibung, Quelle, Verantwortlicher, Anforderungsart, Querbezüge, allgemeine Informationen. Diese Attribute sind für den Requirements Manager selbst wichtig. Hinzu kommen noch die unterschiedlichen Dimensionen der Anforderungs-Landschaft aus Kapitel 2, von denen nur die Anforderungsart bereits im Schema enthalten ist.

So entspricht im ersten Entwurf das Attributierungsschema dem vom IREB vorgeschlagenen, mit den folgenden drei Unterschieden: Es gibt nur ein einziges Status-Attribut statt drei, und es gibt zwei verschiedene Prioritätswerte, die jeweils den Nutzen für verschiedene Stakeholder messen: aus Sicht der Bank und aus Sicht der Bank-Kunden. Zum Anforderungstyp (="Anforderungsart") kommen noch die Anforderungs-Attribute Lösungsunabhängigkeit, Darstellungsform und Detaillierungsebene hinzu.

Tipp aus der Praxis: Weniger ist mehr. Im Zweifel lassen Sie ein Attribut erstmal weg. Es ist besser, Sie fügen später ein neues Attribut hinzu und pflegen die Inhalte nach, als dass ein Attribut gar nicht, unsinnig oder widerwillig befüllt und hinterher womöglich nie gebraucht wird. Das frustriert die Mitarbeiter und stellt auch den Sinn anderer Attribute in Frage.

3.4.3 Definition von zulässigen Attributwerten und Eigenschaften der Attribute

Das Attributierungsschema definiert für die Attribute auch die gültigen Werte. Beispielsweise kann für das Attribut „Risiko" bzw. „Kritikalität" festgelegt werden, dass nur die Werte „hoch", „mittel", „niedrig" oder „kein" erlaubt, um das Risiko zu quantifizieren. Dies ist sinnvoll, weil man erstens Risiken sowieso nicht viel genauer vorherbestimmen kann und zweitens dieser Wert nur dazu dient, um die besonders kritischen von den weniger kritischen Risiken zu unterscheiden. In einer Risikomanagement-Sicht werden die Risiken dann gerne nach Kategorien dargestellt und für jede Kategorie andere Maßnahmen definiert von „Risiko tragen" über „Maßnahmen ergreifen" bis hin zu „Projekt abbrechen". Es ist notwendig, dass Werte klar definiert sind. Beispielsweise: Wann gilt ein Risiko einer Anforderung als „hoch"? Woran wird dies gemessen? An der Wahrscheinlichkeit eines Problems, dem möglichen Schaden oder dem Produkt aus beiden? Wie hoch muss das Risiko sein, damit es als „hoch" gilt?

Beispielsweise könnte man den Wert „hoch" nur vergeben, wenn eine Betriebsunterbrechung zu befürchten ist. Diese Definitionen müssen an all diejenigen kommuniziert werden, die diesen Status pflegen oder auswerten, u.a. auch im RM-Werkzeug im Hilfetext hinterlegt werden.

Des Weiteren muss für die ausgewählten Attribute in dem Attributierungsschema festgelegt werden, ob es sich jeweils um ein Pflichtattribut oder ein optionales Attribut handelt. Beides

hat seine Vor- und Nachteile. Werden wichtige Attribute nicht befüllt, dann werden darauf basierende Berichte nicht vollständig sein. Wird beispielsweise das Risiko für eine kritische Anforderung nicht eingetragen, z.B. weil noch nicht klar ist, ob das Risiko „hoch" oder gar „katastrophal" ist, so wird wegen des leer gelassenen Feldes das Risikomanagement genau diese Anforderung übersehen. Schlimmstenfalls wird so ein Risiko übersehen, welches das ganze Projekt in Frage stellt.

Beispielsweise beim Online-Banking kann es taktisch günstiger sein, riskante Funktionen lieber nicht einzuführen, solange man die dadurch verursachten Risiken nicht sicher unter Kontrolle hat. Ist das Attribut „Risiko" ein Pflichtattribut, hätte der Ersteller zumindest „hoch" eingetragen und in einem Bemerkungsfeld angemerkt, dass diese Anforderung nochmal geprüft und evtl. höhergestuft werden muss. Diese wäre aber schon mal beim Risikomanagement in der Kategorie der kritischen Anforderungen aufgetaucht. Andererseits können Pflichtfelder die Ersteller von Anforderungen dazu zwingen, verfrühte Aussagen zu machen, die eventuell später nicht mehr geprüft und korrigiert werden. Manche Einschätzungen sind in dem Moment, wo eine Anforderung neu angelegt wird, oft noch nicht möglich. Solange ein Attribut leer bleibt, kann eine entsprechende Sicht aufzeigen, dass diese Anforderung noch zu bearbeiten ist, und das Expertenteam für Sicherheit wird sich um die noch nicht klassifizierte Anforderung kümmern.

Eine weitere festzulegende Eigenschaft ist, ob in dem Feld pro Anforderung mehrere Werte oder nur ein Wert ausgewählt werden kann.

Statt die Auswahl eines einzelnen Wertes zu erzwingen, kann man auch als Attributwert zur Auswahl anbieten: „alle möglichen Werte sind gültig" oder „kein Wert ist gültig". Im Fall „kein Wert ist gültig" ist darauf zu achten, dass dieser Wert semantisch von einem Nichtausfüllen des Attributs unterschieden wird (also nicht gleichgesetzt wird), denn die Auswahl des Wertes „kein Wert ist gültig" bedeutet eine bewusste Aussage, während das Nichtausfüllen zwei Interpretationen zuließe: Erstens die vorgegebenen Auswahlmöglichkeiten treffen nicht zu oder zweitens: Das Attribut wurde nicht bearbeitet.

Hier finden Sie das Attributierungsschema unserer Beispiel-Bank. Pflichtfelder sind durch ein Sternchen markiert:

Attribut	Bedeutung	Werte
Identifikator*	Kurze, eindeutige Identifikation	Das bankinterne Schema ist: Projektkürzel + fortlaufende Nummer, hier also OBA0001, OBA0002 usw.
Name*	Eindeutiger, charakteristischer Name	Freitext
Beschreibung*	Beschreibt in komprimierter Form den Inhalt der Anforderungen	Freitext
Version*	Aktueller Versionsstand der Anforderung	Das Format der Versionierung wird noch definiert.
Autor*	Autor der Anforderung	Schreiben dürfen nur: Peter Reber, Martin Geldmann (Business Analyst), Anja Streng (IT-Sicherheits-Expertin), Kirsten Uba (Usability-Expertin)
Quelle*	Benennt die Quelle bzw. Quellen der Anforderung	Liste aller Stakeholder
Begründung	Beschreibt, weshalb diese Anforderung für das geplante System von Bedeutung ist	Freitext
Stabilität	Benennt die voraussichtliche Stabilität der Anforderung.	"stabil", "volatil"
Kritikalität	Im Sinne einer Abschätzung der Schadenshöhe multipliziert mit der Eintrittswahrscheinlichkeit (Risiko)	„gering", „mittel", „hoch"
Priorität für Bank	Misst den Nutzen der Anforderungen für die Bank	„gering", „mittel", „hoch"

Priorität für Kunden	Misst den Nutzen der Anforderungen für die Bank-Kunden	„gering", „mittel", „hoch"
Verantwortliche(r)	Benennt die Person, Stakeholdergruppe bzw. Organisation(seinheit), die für diese Anforderung inhaltlich verantwortlich ist	Liste aller Stakeholder
Anforderungsart	Typ der Anforderung	„funktionale Anforderung", „Qualitätsanforderung" oder „Randbedingung"
Lösungsabhängigkeit	Lösungsabhängigkeit	„Ziel", „Szenario", „Lösungsorientierte Anforderung"
Detaillierungsebene	Detaillierungsebene	Level 1: Geschäfts-Scope Level 2: Benutzer-Scope, Level 3: IT-Scope
Status*	Zustand im Lebenszyklus der Anforderung	in Bearbeitung, in Prüfung, freigegeben, geändert, zurückgewiesen, gelöscht, umgesetzt, getestet, abgeschlossen
Aufwand	Prognostizierter Umsetzungsaufwand dieser Anforderung in Tagen; bei Anforderungen im Status „abgeschlossen" steht hier der tatsächliche Aufwand	Nur ganz- und halbzahlige Werte
Release	Nummer des Releases, in dem die Anforderung umgesetzt werden soll	Hat die Form Jahr/Quartal, also z.B. 2014/3
Juristische Verbindlichkeit	Gibt den Grad der juristischen Verbindlichkeit der Anforderung an	„kann", „soll", „muss"

3.4.4 Definition von Abhängigkeiten zwischen Attributen und ihren Werten

Attribute können in Bezug auf ihre Werte untereinander abhängen. Bei der Definition eines Attributierungsschemas kann z.B. festgelegt werden, dass bestimmte Kombinationen zweier

Attribute mit vordefinierten Attributwerten nicht erlaubt sind. Beispielsweise kann unterbunden werden, dass eine Anforderung mit dem Eintrag „volatil" im Attribut „Stabilität" gleichzeitig im Attribut „Status" den Eintrag „freigegeben" erhält. Damit kann sichergestellt werden, dass z.B. nur als stabil eingeschätzte Anforderungen zur Entwicklung freigegeben werden.

Es kann aber auch sinnvoll sein, diese beiden Attribute in einem Attribut zusammenzufassen und dort nur noch die erlaubten Kombinationen anzubieten. Dies ist vor allem dann eine Lösung, wenn das eingesetzte Werkzeug die Berücksichtigung von Attributwert-Abhängigkeiten nicht unterstützt.

Im Rahmen des Variantenmanagements kann die Zuordnung von Anforderungen zu bestimmten Varianten untersagt sein. Darüber hinaus ist es denkbar, dass bestimmte Variantenkombinationen nicht erlaubt sind.

Es kann auch festgelegt werden, dass beliebige Übergänge von einem Attributwert zu einem anderen nicht erlaubt sind. Insbesondere die Übergänge von einem Status zum anderen sollten sich an den definierten Lebenszyklus der Anforderung halten (vgl. Kapitel 5).

Abhängigkeiten zwischen Attributen und ihren Werten können auch durch eine Hierarchisierung von Anforderungen entstehen. Wird eine Anforderung A zum Beispiel durch die beiden Anforderungen A.1 und A.2 detailliert, so ist für die Attribute des Attributierungsschemas festzulegen, ob A in einem Attribut hinsichtlich der Auswahl eines Wertes von den Werten aus A.1 und A.2 abhängt oder nicht.

Für ein Attribut „Status" ist z.B. die Festlegung sinnvoll, dass die Anforderung A den Wert „freigegeben" nur erhalten kann, wenn die Anforderungen A.1 und A.2 beide ebenfalls den Wert „freigegeben" besitzen (und nicht etwa A.2 sich noch im Status „in Bearbeitung" befindet). Und es wäre auch nicht erklärbar, wenn die Anforderung A eine geringere Priorität hat als A.1 oder A.2 alleine.

Peter Reber entscheidet zusammen mit den Stakeholdern, dass sie keine Einschränkungen machen wollen, außer die in der Anforderungs-Landschaft (vgl. Kapitel 2) vorgegeben, und dass zwischen den Anforderungsstatus nicht jeder Übergang erlaubt ist. Im RIM wurde u.a. festgelegt, dass Geschäftsziele und Geschäftsprozesse zu Detaillierungsebene Level 1 gehören. Geschäftsziele werden textuell beschrieben und Geschäftsprozesse in Use Case Templates. Dies gibt Einschränkungen für die erlaubten Attributkombinationen vor.

Damit die Statusübergänge dem vordefinierten Lebenszyklus der Anforderungen folgen und das versehentliche Überspringen von Schritten (z.B. Genehmigungen) nicht möglich ist, werden nur die in Abbildung 11 dargestellten Übergänge erlaubt.

3.4.5 Bereitstellung einer Erfassungsunterstützung

Für jedes Attribut ist immer eine Person verantwortlich, konkret für dessen Erfassung und regelmäßige Vollständigkeits- und Plausibilitätsprüfung. Per Default ist das der Requirements Manager. Dieser kann die Verantwortung für einzelne Attribute an andere Projektbeteiligte delegieren.

Zusätzlich zur Anforderung noch die Attribute zu erfassen, bedeutet für den Attributverantwortlichen einen zusätzlichen Aufwand. Das Anklicken eines Wertes in einer Dropdown-Liste ist zwar schnell gemacht, aufwendig und zeitraubend ist jedoch, dass er die dafür nötigen Informationen bei verschiedenen Stakeholdern erheben und widersprüchliche Aussagen klären muss. Darum ist in vielen Fällen eine technische Unterstützung durch ein Werkzeug bei der Erfassung und Verwaltung der Attribute sehr wichtig.

Hilfreich ist z.B. die Festlegung von Standardwerten („Default") für Attribute, die automatisch bei neu angelegten Anforderungen gesetzt werden. Der Default-Wert kann der häufigste Wert sein oder der am wenigsten bedeutungsvolle oder derjenige, den alle neu angelegten Anforderungen haben, z.B. Stabilität „volatil". Allerdings ist zu beachten, dass es eventuell nicht erwünscht ist, dass sich als „mittel" klassifizierte Anforderungen nicht von denen unterscheiden, die neu angelegt wurden und noch gar nicht klassifiziert wurden. Nützlich sind Default-Werte besonders bei Pflichtfeldern, die beim Anlegen unbedingt befüllt werden müssen.

Man kann auch sinnvoll zwei Attribute zu einem „kombinierten" Attribut zusammenfassen, wenn nur einige wenige Kombinationsmöglichkeiten der jeweiligen Attributwerte erlaubt sind. Hilfreich sind darüber hinaus gängige Eingabefunktionen wie das Markieren oder Demarkieren einer gesamten Werteliste mit einer Funktion, um sie z.B. für eine Anforderung auszuwählen (sofern in dem Attribut mehrere Werte erlaubt sind) oder umgekehrt das Auswählen einer Liste von Anforderungen und gemeinsame Setzen eines Attributs.

Eine weitere Erfassungshilfe kann ein Werkzeug dadurch bereitstellen, dass abhängige Attributwerte von unterschiedlichen Attributen automatisch eingefügt werden (gegebenenfalls mit Rückfrage). Z.B. kann ein Werkzeug einer Anforderung, die bereits den Status „freigegeben" hatte und anschließend geändert wird, automatisch (evtl. mit einem Hinweis) wieder den Status „in Bearbeitung" zuweisen. Darüber hinaus lassen sich Erfassungshilfen realisieren, indem beispielsweise bei hierarchisch abhängigen Anforderungen Werte, die im Elternknoten gesetzt werden, automatisch auf die „Kindknoten" übertragen werden.

Manche Werte sollten im Sinne einer unanfechtbaren Nachweisbarkeit ohnehin automatisch gesetzt werden, z.B. Autor (=Benutzername des Autors) und Datum einer Änderung.

Unser Attributierungsschema wird nun noch um Default-Werte für die Attribute ergänzt wo sinnvoll:

Attribut	Werte	Default-Wert
Identifikator	Das bankinterne Schema ist: Projektkürzel + fortlaufende Nummer, hier also OBA0001, OBA0002 usw.	Automatisch hochgezählt
Name	Freitext	
Beschreibung	Freitext	
Version	Das Format der Versionierung wird noch definiert.	1
Autor	Schreiben dürfen nur: Peter Reber, Martin Geldmann (Business Analyst), Anja Streng (IT-Sicherheits-Expertin), Kirsten Uba (Usability-Expertin)	Benutzername des Autors
Quelle	Liste aller Stakeholder	
Begründung	Freitext	
Stabilität	"stabil", "volatil"	„volatil"
Kritikalität	„gering", „mittel", „hoch"	
Priorität für Bank	„gering", „mittel", „hoch"	
Priorität für Kunden	„gering", „mittel", „hoch"	
Verantwortliche(r)	Liste aller Stakeholder	Peter Reber
Anforderungsart	„funktionale Anforderung", „Qualitätsanforderung" oder „Randbedingung"	„funktionale Anforderung"
Lösungsabhängigkeit	„Ziel", „Szenario", „Lösungsorientierte Anforderung"	
Detaillierungsebene	Level 1: Geschäfts-Scope Level 2: Benutzer-Scope, Level 3: IT-Scope	

Status	in Bearbeitung, in Prüfung, freigegeben, geändert, zurückgewiesen, gelöscht, umgesetzt, getestet, abgeschlossen	„in Bearbeitung"
Aufwand	Nur ganz- und halbzahlige Werte (in Personentagen)	
Release	Hat die Form Jahr/Quartal, also z.B. 2014/3	
Juristische Verbindlichkeit	„kann", „soll", „muss"	
Querbezüge	„wird getestet durch", „verfeinert", „steht in Konflikt zu", „ersetzt"	
Allgemeine Informationen	Freitext	

3.4.6 Dokumentation des Attributierungsschemas

Attributierungsschemata werden abhängig vom Komplexitätsgrad (z.B. bezüglich der Anzahl der Attribute, Abhängigkeiten zwischen Attributen oder deren Ausprägungen) in Tabellenform oder in einem Informationsmodell (ähnlich wie in Kapitel 2.3) dargestellt. Ein RM-Werkzeug bildet dann das entsprechende Attributierungsschema ab, indem es die entsprechenden Felder zur Verfügung stellt.

3.5 Änderungsmanagement von Attributierungsschemata

Nachträgliche Änderungen an einem Attributierungsschema während des Projektverlaufs sollten möglichst vermieden werden [RuSo2009]. Was bei einer nachträglichen Änderung an einem Attributierungsschema zu beachten ist, hängt von der Art der Änderung ab.

3.5.1 Hinzufügen, Ändern oder Löschen eines Attributs

Wird ein neues Attribut hinzugefügt, sollten die bisher dokumentierten Anforderungen hinsichtlich des neuen Attributs nachgepflegt werden. Dies kann aufwendig werden. Beim Ändern des Namens sind evtl. die Bezeichnungen in verschiedenen Dokumenten, Sichten und Berichten nicht mehr miteinander konsistent. Soll ein Attribut gelöscht werden, verursacht dies oft Schwierigkeiten, wenn eine Sicht, ein Bericht oder eine Schnittstelle zu einem anderen System dieses Attribut abfragt. Statt es zu löschen, kann man seinen Namen ergänzen durch „(nicht mehr verwendet)".

3.5.2 Hinzufügen, Ändern oder Löschen möglicher Attributwerte (Wertebereich)

Das Hinzufügen eines Attributwertes für ein existierendes Attribut ist in der Regel für das zugrundeliegende Werkzeug kein Problem. Fachlich ist zu untersuchen, ob die Anforderungen, für die dieses Attribut bereits gesetzt wurde, noch einmal analysiert werden müssen und ob gegebenenfalls der neue Attributwert besser passt.

Würde beispielsweise bei der Kritikalität ein neuer Attributwert „sehr hoch" ergänzt, wären alle Anforderungen mit der bisherigen Kritikalität „hoch" nochmal zu prüfen, ob sie nicht stattdessen „sehr hoch" kritisch sind.

Beim Löschen von Attributwerten aus einem Wertebereich ist darauf zu achten, dass Anforderungen durch den nun leeren Eintrag im Attribut nicht inkonsistent werden. Probleme bereiten dann vor allem Pflichtfelder, da die Anforderung in dem betrachteten Attribut einen gültigen Wert besitzen muss. In diesem Fall kann die Lösung darin liegen, den Standardwert (Default-Wert) in das Feld einzutragen. Bei Attributen mit abhängigen Attributwerten ist sicherzustellen, dass durch die Entfernung eines Attributwertes keine unerlaubten Attributkombinationen entstehen.

Bei Änderungen von Attributwerten ist zu beachten, dass die Änderungen in allen Anforderungen, die den ursprünglichen Wert beinhalten, durchgeführt werden. Speziell wenn Anforderungen zwischen unterschiedlichen Systemen ausgetauscht werden, kann es zu Inkonsistenzen kommen, wenn z.B. in einer Datenbank die Änderung eines Anforderungswertes nicht durchgeführt wurde (weil nicht verlangt). Generell muss beim Hinzufügen, Ändern sowie Löschen von Attributwerten entschieden werden, ob diese Änderung auf bereits erfasste Anforderungen Auswirkungen hat oder nur für zukünftige Anforderungen gelten soll.

3.5.3 Hinzufügen oder Löschen von Beziehungen zwischen Attributen

Fügt man eine Beziehung bzw. Abhängigkeit zwischen Attributen hinzu, kann das dazu führen, dass einige Attributkombinationen der schon erfassten Anforderungen plötzlich nicht mehr erlaubt sind. Wird zum Beispiel die Einschränkung hinzugefügt, dass die Auswahl eines Wertes für das Attribut „Stabilität" nun immer auch zum Ausfüllen des Attributs „Risiko" führen muss (und damit immer gefüllt ist, wenn auch „Stabilität" gefüllt ist), so muss im Nachhinein untersucht werden, welche Anforderungen zwar einen Wert für „Stabilität", aber keinen für „Risiko" besitzen und entsprechend nachgepflegt werden müssen.

Wird eine Beziehung zwischen Attributen gelöscht, ist das meist unkritisch. Ab sofort ist mehr erlaubt als zuvor. Eventuell kann man prüfen, ob in einigen Fällen die bisher verbotene Attributkombination Sinn machen würde.

3.5.4 Ändern von Vorbelegungen für Attributtyp

Das Ändern von Vorbelegungen sollte sich zunächst nur auf die Eingabe von neuen Anforderungen auswirken. In diesem Zusammenhang sollte aber analysiert werden, ob Anforderungen, welche die bisherige Vorbelegung erhalten haben, noch mit dem richtigen Wert belegt sind oder gegebenenfalls angepasst werden müssen.

3.5.5 Ändern der Verbindlichkeit von Attributen („Pflichtfelder“ und „optionale Felder“)

Wird ein Pflichtfeld in ein optionales Feld geändert, dann führt dies in der Regel zu keinem Folgeaufwand. Ist umgekehrt eine Änderung von einem optionalen Attribut zu einem Pflichtattribut geplant, so ist sicherzustellen, dass für alle bereits dokumentierten Anforderungen das Attribut mit einem passenden Wert gefüllt wird. Dabei kann es sein, dass hierbei nicht ein Standardwert herangezogen werden kann, sondern individuelle Werte vergeben werden müssen.

Generell ist bei Änderungen an Attributierungsschemata zu analysieren, inwieweit Sichten, Berichte und Schnittstellen zu anderen Werkzeugen davon betroffen sind. Wurden beispielsweise in dem Werkzeug Skripte erstellt, die ein bestimmtes Attribut abprüfen oder verarbeiten, kann eine Änderung an diesem Attribut dazu führen, dass die entsprechenden Skripte nicht mehr lauffähig sind.

3.6 Ziele und Arten von Sichten

Projekte umfassen oft hunderte oder gar tausende Anforderungen. Diese Fülle von Anforderungen kann von den Projektbeteiligten in der Regel nicht mehr überblickt werden. Daher benötigt man Konzepte, um diese Komplexität zu verringern. Eine ganz wichtige Bedeutung kommt hier dem Sichtenkonzept zu. Eine Sicht ist eine reduzierte Darstellung der Anforderungen, z.B. reduziert auf nur einen Teil der Anforderungen oder auf einen Teil der Informationen (u.a. der Attribute). Grundlage hierfür ist das Filtern und Sortieren von Anforderungen.

Definition 3-4:

Eine *Sicht* ist eine zweckgerichtete Abstraktion auf die Anforderungen, die nur noch solche Anforderungen und damit zusammenhängende Informationen umfasst, die für den jeweiligen Verwendungszweck (z.B. Stakeholder, Entscheidungsbedarf) relevant sind. Eine Sicht ist technisch gesehen aber auch eine vordefinierte, wiederverwendbare Kombination von Filter- und Sortiereinstellungen sowie Abstraktionen und Aggregationen.

Das IREB Foundation Level unterscheidet zwei grundsätzliche Arten von Sichten [IREB2015]:

- **Selektive Sichten{** XE "Selektive Sicht" **}:** Darstellung einer Teilmenge der Anforderungen, die einen bestimmten Attributwert haben, z.B. alle Anforderungen mit Status „in Bearbeitung".
- **Verdichtende Sichten{** XE "Verdichtende Sicht" **}:** Darstellung verdichteter Informationen zu den ausgewählten Anforderungen, d.h. von Informationen, die so nicht in der Original-Anforderungsliste stehen, sondern erst berechnet werden, z.B. die Anzahl aller Anforderungen mit Kritikalität „hoch" und Status „in Bearbeitung".

Hinzu kommen noch *projektive Sichten{*XE "projektive Sicht" **}**: In projektiven Sichten werden diejenigen Attribute, welche für die betrachtete Sicht nicht relevant sind, ausgeblendet, um nur die für eine Sicht relevanten Informationen zu einer Anforderung anzuzeigen.

In einem RE-Werkzeug kann eine Sicht einmalig definiert und dann abgespeichert werden. Dank der regelmäßigen Verwendung genau derselben Sicht entstehen im Zeitverlauf des Projektes Berichte, die miteinander vergleichbar sind und den Projektverlauf dokumentieren.

Das Attributierungsschema ist die Grundlage für die Definition einer Sicht, denn jede Sicht basiert auf den Informationen, die im Attributierungsschema festgelegt sind. Bei der Erstellung eines Attributierungsschemas sind aus diesem Grunde die später zu erstellenden Sichten unbedingt mit zu berücksichtigen.

Stakeholder eines Projektes haben oftmals einen spezifischen Informationsbedarf für die Durchführung ihrer Aufgaben. Verschiedene Rollen benötigen andere Informationen und damit Sichten auf die Anforderungen:

- Die Stakeholder, insbesondere diejenigen, die Anforderungen definieren, möchten wissen, woher eine Anforderung stammt (die Quelle, z.B. Stakeholder oder Dokument) und welches Ziel sie unterstützt., Diese Informationen werden benötigt um sicher zu stellen, dass alle Anforderungen tatsächlich nötig sind (=Rückwärts-Verfolgbarkeit oder Pre-Requirements-Specification Traceability). Außerdem interessiert sie auch die Vorwärts-Verfolgbarkeit (=Post-Requirements-Specification Traceability), d.h. die Verfolgbarkeit von Anforderungen zu nachfolgenden Entwicklungs-Artefakten, z.B. Systemarchitektur, Implementierung und Test [IREB2015]. Mit welchem Test kann

er beispielsweise bei der Abnahme prüfen, ob eine bestimmte Anforderung korrekt umgesetzt wurde?

- **Requirements Engineer und Requirements Manager** möchten die Qualität der Anforderungen sicherstellen, insbesondere auch deren Konsistenz zwischen Dokumenten und Detaillierungsebenen. Auch hierfür ist die Verfolgbarkeit wichtig, aber auch ein Überblick über deren Status.
- **Projektleiter:** Ihn interessiert der Status des Projektes (oder eines Release) mit dem Ziel, eine Prognose über den Restaufwand und die Restdauer des RE oder des Projektes abgeben zu können. Er könnte sogar anforderungsbasiertes Projektmanagement betreiben, z.B. anhand der Status der Anforderungen den Fertigstellungsgrad des Projektes beurteilen. Dies ist in der agilen Entwicklung üblich, aber grundsätzlich immer dann möglich, wenn die Qualität der Attribut-Inhalte stimmt, d.h. der voraussichtliche Umsetzungsaufwand jeder Anforderung hier abgelegt ist, der Bearbeitungsstand aus dem Status hervorgeht und entsprechende Auswertungen im Werkzeug vorhanden sind. Bei Anträgen auf Anforderungsänderungen möchte der Projektleiter deren voraussichtliche Auswirkungen vorhersehen können, z.B. Aufwand, Nebenwirkungen und neu geschaffene Risiken. Für solche Beurteilungen ist die Verfolgbarkeit sehr wichtig. Außerdem möchte er wissen, wer der Ansprechpartner für eine bestimmte Anforderung ist. Das spricht für ein Attribut „Quelle", „Verantwortlicher" und / oder „nächster Bearbeiter".
- Der **Architekt** benötigt vor allem einen strukturierten Blick auf die Anforderungen, z.B. deren Gruppierung nach technischen Kriterien für die Zuordnung zu Komponenten. Hierzu sind Attributtypen wie "Schnittstellen-Anforderung" nützlich.
- **Entwickler** möchten Zugriff auf die Originalanforderungen, die zu der Komponente gehören, die er als nächstes umsetzt. Dies wird durch eine Verfolgbarkeit zwischen Anforderungen und Komponenten unterstützt.
- **Tester und Testmanagement** möchten wissen, welche und wie viele Anforderungen noch zu testen sind und wie viele sie bereits als fehlerfrei bestätigt haben. So lässt sich der Testfortschritt messen und der Restaufwand des Testens einschätzen. Es ist auch wichtig zu wissen, welche Testfälle bei einer Änderung erneut durchgeführt werden müssen.

Die oben genannten Sichten lassen sich relativ leicht durch Filtern und Sortieren der Anforderungen nach ihren Attributwerten und durch Auswertungen wie Summenbildung über Attribute erzeugen. Die unbedingt notwendige Voraussetzung hierfür ist jedoch, dass die entsprechenden Attributwerte vorhanden und gut gepflegt sind, und außerdem das RE-Werkzeug die nötigen Auswertungen erlaubt.

Zusätzlich zur übersichtlichen Fokussierung und Darstellung der vorhandenen Informationen für die verschiedenen Rollen, können Sichten auch den Zugriff auf Anforderungen rollenspezifisch regeln. Häufig ist es gar nicht erwünscht, dass jeder Projektbeteiligte auf alle Informationen zugreifen kann und eine rollenbasierte Sichten Generierung erforderlich ist (vgl. [Pohl2010]). Ein solches Berechtigungskonzept kann in einem Werkzeug konkret so realisiert werden, dass bestimmte Rollen nur bestimmte Sichten zu sehen bekommen.

3.7 Definition von Sichten und Risiken von Sichten

Der Prozess zur Definition von Sichten umfasst u.a. folgende Schritte:

- **Stakeholder-Identifikation:** Feststellen der Stakeholder, welche eine oder mehrere Sichten benötigen.
- **Wiederverwendung:** Es können Sichten aus anderen Projekten oder aus einem Referenzprojekt als Vorlage für zu definierende Sichten herangezogen werden.
- **Festlegung der Ziele:** Für jeden Stakeholder muss erarbeitet werden, welches Ziel er mit seinen Sichten verfolgt. Daraus folgt, welche Informationen herausgefiltert oder verdichtet werden sollen oder welche Sortierung initial eingestellt werden soll. In diesem Zusammenhang müssen auch Rechte von Rollen und Sichten festgelegt werden, d.h. welche Rolle soll welche Sicht aktivieren können. Dabei ist es effizient, wenn eine Sicht von mehreren Rollen genutzt werden kann.
- **Festlegung notwendiger Attribute und Abgleich mit dem Attributierungsschema:** Um die Ziele einer Sicht erfüllen zu können, muss sichergestellt werden, dass die notwendigen Informationen auch erhoben werden können und somit die entsprechenden Attribute auch zur Verfügung stehen. Eine Auswertung über die Anzahl von Anforderungen, die sich noch im volatilen Zustand befinden, lässt sich nur generieren, wenn dieser Zustand auch in einem entsprechenden Attribut dokumentiert ist. Der Abgleich mit dem Attributierungsschema führt oft auch zur Entdeckung neuer Sichten, weil die Stakeholdern beim Betrachten des Attributierungsschemas erkennen, welche Auswertungen noch möglich wären. Die Definition von Sichten und Attributierungsschema beeinflussen einander gegenseitig.
- **Implementierung der Sicht:** Schließlich müssen die vordefinierten Sichten in dem zugrundeliegenden Werkzeug umgesetzt und getestet werden.

Manchmal ist Anwendern eines RM-Systems nicht bewusst, dass sich die komplexen Informationen zu den Anforderungen durch Sichten bedarfsgerecht einschränken lassen. Sie arbeiten dann oft mit einer globalen und allumfassenden Sicht und empfinden ungerechterweise das Werkzeug als zu umfangreich sowie gegebenenfalls in ihrer Arbeit behindernd. Gleichzeitig bergen Sichten auch Gefahren.

Beispielsweise könnte in einer Sicht zu viel Kontext verloren gehen: Erstellt man z.B. eine Sicht, in der atomare Anforderungen ohne jeglichen Kontext (z.B. Use Cases) in einer Liste aufgeführt sind, wird diese Übersicht nur eingeschränkt aussagekräftig sein. Um solche wenig effektiven Sichten möglichst zu vermeiden, müssen beim Definieren einer Sicht immer die dahinterliegenden Ziele der Stakeholder berücksichtigt werden.

3.8 Umsetzung einer Sicht

Die meisten Sichten setzt man in einem RM-Werkzeug um, indem man nach Attributwerten selektiert / filtert und sortiert (vgl. auch [Pohl2010]).

Beim Selektieren blendet man entweder durch einen vordefinierten Filter Anforderungen aus, die gerade nicht interessieren, oder Attribute der Anforderungen oder beides kombiniert. Das heißt, die Sicht präsentiert nur einen Auszug aus den vorhandenen Daten.

Beispielsweise kann ein Tester durch den Filter auf ein bestimmtes Release die Anforderungen selektieren, die für das aktuell zu testende Release relevant sind. Diese erkennt er daran, dass sie im Attribut „Release" den gesuchten Wert haben. Gleichzeitig ist die Information über den prognostizierten Aufwand zur Realisierung einer Anforderung wahrscheinlich für einen Tester nicht relevant, sodass das entsprechende Attribut für die spezifische Sicht des Testers ausgeblendet werden kann. Zusätzlich kann noch nach Kritikalität sortiert werden, damit der Tester bei den kritischsten Anforderungen mit dem Testen beginnen kann. Durch Sortieren kann ein Stakeholder den Fokus auf Anforderungen ändern (z.B. alle Anforderungen mit hoher Kritikalität am Anfang einer Liste), ohne Anforderungen auszublenden.

Verdichtende Sichten zeigen auch Informationen, die nicht in dieser Form in den Anforderungen enthalten sind. Solche entstehen z.B. durch das Bilden von Summen und Teilsummen oder durch das Berechnen von prozentualen Anteilen. So kann z.B. ermittelt werden, wie hoch der Anteil von Anforderungen mit dem Status „getestet" ist, ggf. auch gewichtet nach dem Aufwand. Damit lässt sich über einen bestimmten Zeitraum feststellen, wie sich der Fertigstellungsgrad der Anforderungen weiterentwickelt.

Verdichtende Sichten lassen sich mit selektiven Sichten kombinieren, um z.B. den Fertigstellungsgrad der Anforderungen in Bezug auf ein bestimmtes Release zu bestimmen. Dabei wird die Verdichtung auf eine gefilterte Menge von Anforderungen berechnet.

Durch Filtern erstellt man beispielsweise diese Sichten:

- alle freigegeben Anforderungen
- alle zu einem bestimmten Release gehörenden Anforderungen
- alle bereits getesteten Anforderungen
- alle Anforderungen, für die eine bestimmte Person verantwortlich ist
- alle Anforderungen, die ein Entwickler beim Umsetzen einer bestimmten Komponente zu berücksichtigen hat

Durch Sortierung erhält man beispielsweise solche Sichten:

- eine Darstellung der Anforderungen in der Reihenfolge ihrer Kritikalität
- eine Sortierung der Anforderungen nach der verantwortlichen Person zeigt die Verteilung der Arbeit auf die Teammitglieder auf

3.9 Optimierung von Attributierung und Sichtenbildung

Die Praxis zeigt, dass in manchen Projekten die Attribute nicht im erwarteten Maße mit Werten belegt werden – manchmal sogar mit gutem Grund. Daher ist es zweckmäßig, regelmäßig und zu definierten Zeitpunkten im Projekt zu überprüfen, ob und wie ein Attribut noch weiter verwendet werden soll [RuSo2009].

Grundsätzlich sollten nur diejenigen Attribute beibehalten werden, die in einer Sicht oder einem Bericht für mindestens einen Stakeholder einen Nutzen bringen. Diese nötigen Attribute sollen aber so gut wie möglich gepflegt werden.

Ein leicht zu prüfender Mangel der Attribute ist eine fehlende Befüllung. Um unbefüllte Attribute zu finden, kann man entweder eine Sicht dafür definieren oder nach dem entsprechenden Attribut sortieren. Die Anforderungen mit den leeren Feldern stehen in der Liste dann entweder ganz oben oder ganz unten.

Will man, dass ein Attribut immer befüllt wird, kann man dies am einfachsten sicherzustellen, indem man das Attribut als Pflichtfeld definiert. Dies erzwingt eine Eingabe beim Anlegen einer Anforderung. Allerdings ist dies selten die optimale Lösung. Dabei ist zu beachten, dass die Definition von zu vielen Pflichtfeldern die Bearbeitung sehr behindern kann und dass zu dem Zeitpunkt, wenn eine Anforderung das erste Mal erfasst wird, manche Informationen noch nicht vorliegen, wie beispielsweise die Kostenschätzung. Insofern sollten Pflichtfelder nur sparsam und mit Augenmaß deklariert werden.

Bei optionalen Attributen, deren Erfassung nicht zwingend eingefordert wird, können Auswertungen über die bisherige Verwendung folgende Schlussfolgerungen erlauben:

- **Das Attribut wurde weder in einer Sicht noch in einem Bericht herangezogen:**
  Dies deutet darauf hin, dass das betrachtete Attribut kein spezifisches Ziel unterstützt und vermutlich auch keinen der Stakeholder interessiert. Damit steht sein Sinn überhaupt in Frage, denn jedes vorhandene Attribut verursacht Pflegeaufwand.
- **Das Attribut ist immer nur mit demselben Wert gefüllt, z.B. dem Default-Wert:**
  In diesem Fall scheinen sich die Anforderungen in Bezug auf dieses Attribut nicht zu unterscheiden, d.h. die vorgeschlagene Auswahlliste von Werten passt nicht. Hier kann man entweder das Attribut wegfallen lassen (weil es niemandem nutzt) oder die Auswahlliste anpassen. Im letzteren Fall sollten die Hinweise aus Kapitel 3.5 über das Ändern von Attributierungsschemata berücksichtigt werden.
- **Das Attribut ist nie gefüllt:**
  Wenn das Attribut bewusst nicht gefüllt wurde, ist diese Information möglicherweise nicht wichtig. Bestätigt sich diese Annahme, sollte das Attribut entfernt werden. Oftmals liegt der Grund für das Nichtausfüllen von Attributen aber auch darin, dass den Anwendern die Definition oder der Nutzen des Attributs nicht bekannt ist oder sie für sich keinen Nutzen sehen, weil die Information von einem anderen Stakeholder genutzt wird. Dann sollten die Anwender (nach)geschult werden, um ihnen die Wertschöpfung des jeweiligen Attributs zu verdeutlichen. In diesem Fall kann das Attribut weiter eingesetzt werden.
- **Das Attribut ist nur bei wenigen Anforderungen gefüllt:**
  Hier ist zu hinterfragen, ob das mit dem Attribut verbundene Ziel erreicht werden kann oder überhaupt noch relevant ist. Ist dies nicht der Fall, kann das Attribut entfernt werden. Stellt sich aber heraus, dass das Attribut wichtig ist, kann es gegebenenfalls als Pflichtfeld deklariert werden, was einen Eintrag in der Zukunft erzwingt.
  In dem Fall muss eine Nacherfassung bei den bereits dokumentierten Anforderungen erfolgen, die keinen Wert in dem betrachteten Attribut besitzen (z.B. automatisierte Befüllung mit einem Standardwert).

- **Das Attribut ist in Einzelfällen nicht gefüllt:**
  Auch für dieses Attribut muss zunächst ermittelt werden, ob es für das Projekt weiterhin relevant ist. Wenn ja, sollte der Requirements Engineer die entsprechenden Anforderungen vervollständigen. Wird das Attribut nicht mehr als wesentlich eingeschätzt, kann es entweder entfernt oder es können alternativ die Lücken toleriert werden.
- **Das Attribut ist immer gefüllt:**
  In diesem Fall ist keine weitere Aktivität notwendig.

Außer die Befüllung der Attribute zu prüfen, sollte man nicht vergessen, auch die Stakeholder über ihre Zufriedenheit zu befragen. Vielleicht vermissen sie in ihren Sichten Informationen, die ebenfalls im Attributierungsschema fehlen. Dann sollte das fehlende Attribut oder der fehlende Attributwert neu eingeführt und wenn nötig für die bereits erfassten Anforderungen nachbefüllt werden. Vielleicht wird aber auch ein Attribut oder Wert obsolet. Wenn es von einem Stakeholder in seiner Sicht nicht mehr gebraucht wird, stellt sich die Frage, ob noch andere Sichten es verwenden oder es eventuell gelöscht werden kann. Änderungen am Attributierungsschema sind mit Bedacht durchzuführen (siehe Kapitel 3.5).

3.10 Inhalte für den RMP

Der RMP dokumentiert das Attributierungsschema. Dieses beschreibt, welche Anforderungs-Attribute verwendet werden sollen. Jedem Attribut werden der Name, eine Beschreibung, die verantwortliche Person, die zulässigen Werte und Abhängigkeiten zu anderen Attributen dokumentiert. Das Attributierungsschema stellt man beispielsweise in tabellarischer Form dar (vgl. Tabelle 2) oder in einem Informationsmodell.

Im RMP werden auch die Sichten festgelegt, die unterstützt werden sollen. Für jede Sicht sind das Ziel und der sie verwendende Stakeholder dokumentiert, sowie natürlich die anzuzeigenden Attribute, die anzuwendenden Filter und voreingestellten Sortierungen. Es ist sichergestellt, dass das Attributierungsschema alle dafür nötigen Attribute enthält.

3.11 Vertiefende Literatur

[Pohl2010] K. Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. Springer, 2010.

[RuSo2009] C. Rupp & die SOPHISTen: Requirements-Engineering und -Management. Hanser, 5. aktualisierte und erweiterte Auflage, 2009.

4 Bewertung und Priorisierung von Anforderungen

4.1 Motivation und Schwierigkeiten der Priorisierung von Anforderungen

Nicht alle Anforderungen sind gleich wichtig. Das stellt sich spätestens dann heraus, wenn innerhalb eines fest vorgegebenen Kostenrahmens oder Zeitraums nicht alle Anforderungen umgesetzt werden können. Dann steht man vor der Entscheidung, entweder das Budget zu erhöhen, später zu liefern oder am Lieferumfang Abstriche zu machen. Und plötzlich sind dann manche Anforderungen – Funktionen oder Qualitätsanforderungen – doch nicht mehr unverzichtbar. Manchmal begegnet einem auch der Fall, dass zwei Anforderungen aus technischen Gründen nicht beide oder nicht beide gleich gut umgesetzt werden können (=Anforderungskonflikt). Um diesen Konflikt zu lösen, muss entschieden werden, welche der beiden Anforderungen wichtiger ist. In vielen Fällen gilt: „Schedule should drive requirements" [Davi2005]. Das heißt, wenn die Zeit knapp wird, müssen Anforderungen oftmals weichen.

Definition 4-1:

Die *Priorität{ XE "Priorität" }* (oder Wichtigkeit) einer Anforderung dokumentiert die Bedeutung einer Anforderung im Vergleich zu anderen Anforderungen in Bezug auf ein definiertes Kriterium (IREB-Glossar [Glin2014]).

„Im Vergleich" bedeutet, dass diese Priorität nicht unbedingt Absolutwerte verlangt wie z.B. Bedeutung gemessen in Euro oder Implementierungsaufwand in Personentagen. Da sie dem Vergleich der Anforderungen dient, kann es sich auch um einen relativen Wert handeln, z.B. auf einer Punkteskala von 1 bis 10. Sie wollen ja nur wissen, welche Anforderung wichtiger ist als eine andere.

Die Priorität einer Anforderung bildet die Grundlage einiger Entscheidungen im Software- oder Systementwicklungsprozess. Solche Entscheidungen sind außer der Konfliktlösung zwischen Anforderung und der Release-Planung auch technische Entscheidungen und die Priorisierung beim Testen. Je wichtiger oder kritischer die Anforderung, die durch einen Testfall geprüft wird, umso wichtiger auch der Testfall und die damit gefundenen Fehler, und umso gründlicher sollte man diese Anforderung testen.

Definition 4-2:

Priorisierung{ XE "Priorisierung" } bezeichnet die Tätigkeit zur Ermittlung der Prioritäten von Anforderungen.

Die Priorisierung bereitet die Anforderungsverhandlung (requirements negotiation), Anforderungsauswahl (requirements selection) oder Release-Planung (release planning) vor.

Die Priorisierung wird dadurch erschwert, dass die Bedeutung einer Anforderung letztlich von vielen Faktoren abhängt, insbesondere:

- den Kriterien, die man anlegt,
- der Perspektive, die man einnimmt (d.h. für verschiedene Stakeholder ist eine Anforderung verschieden wichtig),
- der Entscheidung, die damit unterstützt werden soll (d.h. bedeutet eine Niedrigpriorisierung, dass diese Anforderung nie geliefert wird oder kommt sie nur einfach vier Wochen später als die anderen?)
- und dem Zeitpunkt, zu dem sie geschätzt wird.

Zählen beispielsweise nur finanzielle Kriterien wie Kosten und Nutzen, oder auch die Kundenzufriedenheit, oder zielt man auf Risikominimierung? In einem sicherheitskritischen System wird z.B. Risikoverringerung wichtiger sein als Benutzerfreundlichkeit, in anderen Systemen ist es umgekehrt.

4.2 Grundlagen der Bewertung

Die Bewertung der Anforderungen ist die Grundlage für deren Priorisierung. Die Priorität – also Wichtigkeit – einer Anforderung wird oft aus mehreren Bewertungskriterien ermittelt, z.B. durch den Abgleich von Kosten und Nutzen dieser Anforderung.

Als Bewertungskriterien werden beispielsweise verwendet:

- Implementierungsaufwand oder andere Kosten
- Bedeutung oder Nutzen für den Benutzer oder andere Stakeholder,
- Voraussichtliche Nutzungshäufigkeit einer Funktion
- Rechtliche Verbindlichkeit einer Anforderung
- Abhängigkeiten zwischen Anforderungen (grundlegende Anforderungen sind früher zu implementieren als die von ihnen abhängigen)
- Kritikalität (auch Risiko genannt),
- Stabilität oder Innovationsgrad (vgl. Kano-Priorisierung in Kapitel 4.5.8).

Die Prioritäten der Anforderungen korrelieren natürlich mit den Prioritäten der damit verknüpften System-Funktionen, Testfälle und den im Rahmen des Tests aufgedeckten Fehlern. Mit anderen Worten: Ist eine wichtige Anforderung bzw. Funktionalität fehlerhaft, so wiegt dieser Fehler schwerer als ein ähnlicher Fehler in einer unwichtigen Funktionalität, die nur selten benutzt wird.

Die Bewertung von Anforderungen ist die Aufgabe entsprechend qualifizierter Stakeholder. So können beispielsweise die Endnutzer oder das Produktmanagement den Nutzen einer Anforderung am ehesten einschätzen. Für die Kosten ist das technische Personal der beste Ansprechpartner. Aber auch bereits die Frage, welche Bewertungs- und Priorisierungskriterien verwendet werden sollen, muss mit Stakeholdern abgestimmt werden. Dies tun dann eher die Nutzer der Priorisierungsergebnisse, z.B. der Projektleiter, also die Personen, die aufgrund der Prioritäten Entscheidungen zu treffen haben. Hierbei sind die Priorisierungskriterien exakt zu definieren einschließlich der zu verwendenden Skala und dem Schätzverfahren. Sollen beispielsweise die Kosten durch Expertenschätzung oder durch eine Zählmethode wie die Function Point Methode ermittelt werden?

Genügen Function Points als relatives Maß oder müssen die Aufwände in Personentage umgerechnet werden? Und wer darf bzw. soll diese Bewertung durchführen? Wann (frühestens oder spätestens)?

Quellen für Bewertungskriterien sind z.B.:

- Projektmanagement
- Normen und Standards
- das Attributierungsschema der Anforderungen
- Folgedisziplinen der Entwicklung wie z.B. die Qualitätssicherung

Die Aufgabe des RM ist es, dafür zu sorgen, dass die Prioritäten ermittelt und geeignet dokumentiert werden. Für diese Dokumentation eignen sich die Attribute. Dazu müssen die Priorisierungskriterien Teil des Attributierungsschemas sein (vgl. Kapitel 3).

Das Attributierungsschema unseres Fallbeispiels enthält bereits einige Attribute, die zur Priorisierung der Anforderungen geeignet sind:

- **Stabilität**: Volatile Anforderungen sollten noch nicht umgesetzt, sondern im Release-Plan auf später verschoben werden, bis sie sich stabilisiert haben.
- **Kritikalität**: Besonders kritische (d.h. riskante) Anforderungen werden anders behandelt als unkritische. Z.B. sollen die IT-Sicherheitsexperten für die riskanten Anforderungen eine systematische Risiko-Analyse durchführen. Außerdem sollen die kritischen Anforderungen besonders gründlich getestet werden.
- **Priorität für Bank**: Will man möglichst schnell möglichst viel Return on Investment für die Bank erzeugen, dann sollten die hier „hoch" bewerteten Anforderungen zuerst produktiv gehen.
- **Priorität für Kunden**: Will man möglichst schnell möglichst viel Benutzernutzen liefern, dann sollten die hier „hoch" bewerteten Anforderungen zuerst produktiv gehen.
- **Aufwand**: Der Aufwand kann auf zwei verschiedene Arten die Priorisierung beeinflussen. Das fest vorgegebene Budget für ein Release gibt einen Kostendeckel vor. Man kann bei der Release-Planung daher so viele der wichtigsten Anforderungen auswählen,

dass das Budget nicht gesprengt wird. Außerdem kann man auch aus Nutzen und Aufwand einer Anforderung jeweils das Kosten-Nutzen-Verhältnis dieser Anforderung berechnen und dieses als Priorisierungskriterium verwenden. Entscheidend ist dann für die Bewertung der Wichtigkeit nicht der absolute Nutzen, sondern ob sich die eingesetzten Aufwände lohnen: Liegen die Kosten unter dem Nutzen oder darüber?

- **Release**: Die Release-Nummer einer Anforderung ist bereits das Ergebnis einer Priorisierung, vermutlich aufgrund anderer Priorisierungskriterien.
- **Juristische Verbindlichkeit**: Alle "Muss"-Anforderungen müssen in Release 1 bereits enthalten sein.

4.3 Priorisierung von Anforderungen

Wie wir im vorigen Abschnitt gesehen haben, ist bei der Priorisierung von Anforderungen einiges zu beachten. Am besten geht man dabei systematisch vor, und zwar in der folgenden Reihenfolge:

- **Festlegen der Ziele der Priorisierung:** Wer wird wann Prioritäten brauchen, um Entscheidungen zu treffen? Welche Entscheidungen sind zu treffen? Warum und wofür ist diese Entscheidung wichtig? Welche (höheren) Ziele soll sie unterstützen?
- **Festlegen der Priorisierungskriterien:** Welche Kriterien sind bei der Priorisierung zu verwenden, um diese Ziele zu erreichen? Soll beispielsweise das Kosten-Nutzen-Verhältnis des Gesamtprojektes maximiert werden, so ist es sinnvoll, auch für jede Anforderung deren Kosten-Nutzen-Verhältnis zu ermitteln und dieses Kriterium bei der Release-Planung zu verwenden. Festzulegen ist hierbei auch die Skala und der Wertebereich für jedes Kriterium. Sollen absolute oder relative Werte ermittelt werden? Dies hängt davon ab, welche Art der Entscheidung zu treffen ist und wie feingranular die Bewertung hierfür nötig ist. Soll beispielsweise aus einer Liste von Anforderungen das wichtigste Drittel ausgewählt werden, genügt eine Bewertung auf einer Skala mit 1, 2 oder 3 Punkten. Will man jedoch eine nach Wichtigkeit sortierte Liste erstellen, dann ist eine Ordinalskala am besten. Eine Ordinalskala weist den Anforderungen Werte zu von „wichtigste Anforderung" über „zweitwichtigste Anforderung" bis hinunter zur unwichtigsten. Dargestellt wird sie durch eine fortlaufende ganzzahlige Nummer, die bei 1 für die wichtigste Anforderung starten kann, aber auch umgekehrt die 1 der unwichtigsten Anforderung zuordnen.
- **Festlegung der priorisierenden Stakeholder:** Zunächst werden die Anforderungen in Bezug auf die Bewertungskriterien bewertet. Für jedes Bewertungskriterium sind ggf. andere Stakeholder kompetent genug, um eine verlässliche Schätzung abgeben zu können. Aufbauend auf diesen Bewertungen wird dann eine Person oder auch Personengruppe die Anforderungen priorisieren. Alle diese Stakeholder sind zunächst entsprechend ihrer Kompetenz auszuwählen.

- **Festlegung der zu priorisierenden Anforderungs-Artefakte:** Sind die Anforderungen auf verschiedenen Detaillierungsebenen beschrieben, so möchte man vermutlich nur auf einer einzigen dieser Ebenen priorisieren. Dabei wählt man diejenige, auf der auch die Entscheidung zu treffen ist. Sollen beispielsweise die im ersten Release umzusetzenden Geschäftsprozesse ausgewählt werden, priorisiert man nur die Geschäftsprozesse, und nicht die sie verfeinernden Benutzungsszenarien. Auch ergibt es wenig Sinn, Äpfel mit Birnen zu vergleichen, z.B. Features mit Mock-ups. Bei der Auswahl der zu priorisierenden Anforderungen ist zu beachten, dass sich die Anforderungen auf einer ähnlichen Detaillierungsebene befinden sollten, um das Ergebnis der Priorisierung nicht zu verfälschen [WiBe2013]. Gröbere, abstraktere Anforderungen neigen dazu, eher höhere Prioritäten zu haben als detailliertere, weil eine grobe Anforderung mehrere detaillierte umfasst. Insbesondere geht es hier auch darum, die Anzahl der zu priorisierenden Anforderungen zu begrenzen, weil der Priorisierungsaufwand sonst sehr hoch werden kann. (Mehr dazu später.)
- **Auswahl der Priorisierungstechnik:** Hierbei geht es sowohl um Messverfahren und Schätzmethoden für die Ermittlung der Kriterien als auch um ein Sortierverfahren für die Anforderungen. Verschiedene Priorisierungstechniken werden in Kapitel 4.4 und 4.6 beschrieben. Diese sind meist eher Sortier- als Messverfahren. Festzulegen ist auch, wer wann diese Priorisierung durchführt. Vermutlich werden abhängig von der benötigten Kompetenz verschiedene Experten verschiedene Kriterien bewerten.
- **Ggf. Anpassung des Attributierungsschemas:** Zumeist dokumentiert man die Prioritäten der Anforderungen in Attributen. Die Voraussetzung dafür ist natürlich, dass das Attributierungsschema die entsprechenden Attribute mit den richtigen Wertelisten enthält. Falls dies noch nicht der Fall ist, muss das Attributierungsschema angepasst werden. Was bei dieser kritischen Tätigkeit zu beachten ist, insbesondere wenn das Attributierungsschema schon in Benutzung ist, diskutieren wir in Kapitel 3.
- **Priorisierung:** Nun wird die Priorisierung durchgeführt wie geplant. Es werden alle Bewertungen und Prioritäten dokumentiert einschließlich deren Begründungen und der dabei getroffenen Annahmen.
- **Regelmäßige Überprüfung und ggf. Neupriorisierung der Anforderungen**: Die Dinge ändern sich mit der Zeit, manches wird wichtiger, manches unwichtiger. Manchmal verbessert sich auch unser Wissen von den Tatsachen. Darum ändern sich auch Prioritäten mit der Zeit und dies macht eine regelmäßige Überprüfung der Prioritäten nötig. Der beste Zeitpunkt dafür ist stets, kurz bevor aufgrund der Prioritäten eine wichtige Entscheidung zu treffen ist.

Stakeholder der Priorisierung, Ziele und Kriterien:

In der Fallstudie sollen die Änderungen des Online-Banking-Systems in einzelnen Releases ausgeliefert werden. Die Release-Planung führt der Projektleiter durch, unterstützt von unserem Requirements Manager Peter Reber. Für die Release-Planung werden folgende Regeln festgelegt:

- Stabilität: Volatile Anforderungen kommen grundsätzlich nicht ins nächste Release.

- Juristische Verbindlichkeit: Alle "Muss"-Anforderungen müssen in Release 1 bereits enthalten sein.
- Von den restlichen Anforderungen werden abwechselnd die für die Bank wichtigste Anforderung (Attribut „Priorität der Bank") und dann die für die Kunden wichtigste Anforderung (Attribut „Priorität der Kunden") ausgewählt, bis das Release-Budget verbraucht ist.

Priorisiert werden sollen nur Geschäftsprozesse, da jedes Release nur vollständige Geschäftsprozesse beinhalten soll. Mit halb umgesetzten Geschäftsprozessen kann (meist) niemand etwas anfangen.

Für schnelle Entscheidungen in Krisen und Konflikten möchte der Projektleiter jederzeit eine sortierte Liste der Anforderungen sehen können, die alternativ nach ihrer Bedeutung für die Bank oder für die Kunden sortiert ist. Hier hätte er gerne außer den Geschäftsprozessen auch Benutzer- und Systemanwendungsfälle und außerdem alle lösungsorientierten Anforderungen priorisiert gesehen. Allerdings wurde schnell klar, dass diese detaillierte Priorisierung einen enormen Aufwand bereiten würde.

Stunden- oder gar tagelange Priorisierungsworkshops wären nötig. Daher gibt es zunächst keine Priorisierung der lösungsorientierten Anforderungen (und damit auch keinen Workshop).

Die IT-Sicherheitsexperten verwenden das Attribut „Kritikalität", um Anforderungen für das weitere Vorgehen vorzumerken: Als „hoch" bewertete Anforderungen sollen von der Gruppe einer gründlichen Risiko-Analyse unterzogen werden, einschließlich Fehlerbaum-Analyse, die „mittel"-kritischen Anforderungen einer ganz normalen Risiko-Analyse durch eine einzige Person, und die „niedrig" bewerteten Anforderungen werden nicht weiter betrachtet.

Das Ziel dieses Vorgehens ist es, das Augenmerk auf die besonders relevanten Bereiche zu richten.

Die Usability-Expertin wird sich besonders intensiv mit den Anforderungen beschäftigen, die für die Kunden besonders wichtig sind.

Es sind also alle für die Entscheidungsfindung nötigen Attribute bereits im Attributierungsschema enthalten (vgl. Kapitel 3). Kurzzeitig wurde überlegt, noch ein weiteres Attribut einzuführen, das alle Anforderungen heraushebt, die für die Barrierefreiheit besonders wichtig sind. Bei einer ersten Analyse stellte es sich jedoch heraus, dass dies für beinahe alle Anforderungen gelten würde. Das Attribut würde also seinen Zweck als Unterscheidungs- und Priorisierungskriterium gar nicht erfüllen. Darum wurde dieser Plan aufgegeben.

Die Auswahl der passenden Priorisierungstechniken erfolgt später im Kapitel, nachdem Sie die verschiedenen Techniken kennen gelernt haben.

4.4 Zwei Typen von Priorisierungstechniken

Es gibt sehr viele Priorisierungstechniken. Sie sind mehr oder weniger aufwendig, mehr oder weniger subjektiv und fehleranfällig, und für verschiedene Zwecke besser geeignet als für andere. Wir können hier keine vollständige Übersicht über alle existierenden Techniken geben. Wir präsentieren Ihnen aber die wichtigsten, verbreitetsten und für die Praxis am geeignetsten Techniken.

Wir gehen hier von der Situation aus, dass die Ziele und Kriterien der Priorisierung bereits festgelegt wurden, sowie die Personen, welche abschätzen können (bzw. oft eher vorhersagen), welche Anforderung welches Kriterium wie gut erfüllen wird. Es ist auch definiert, welche Artefakte priorisiert werden sollen. Wenn Sie eine komplexe Anforderungs-Landschaft haben und z.B. die Anforderungen auf verschiedenen Detaillierungsebenen spezifizieren, dann sollten Sie nur Anforderungen miteinander vergleichen, die sich auf demselben Abstraktionsniveau befinden. Alles andere wäre ein Vergleich von Äpfeln mit Birnen. Um zu verlässlichen Ergebnissen zu gelangen, sollten Sie je nach zu treffender Entscheidung nur auf einer einzigen Detaillierungsebene priorisieren, nämlich auf der, wo diese Entscheidung getroffen werden soll. Geht es z.B. darum, für das erste auszuliefernde Release die wichtigsten Geschäftsprozesse auszuwählen, dann priorisieren Sie die Geschäftsprozesse. Wollen Sie dagegen die Implementierungs-Reihenfolge der Anwendungsfälle festlegen, so priorisieren Sie die Anwendungsfälle. Ausgangspunkt der Priorisierung ist also eine unsortierte Liste von Anforderungen auf demselben Abstraktionsniveau.

Die Priorisierungstechnik wandelt diese Liste in eine sortierte Liste um, in der jeder Anforderung ein Prioritätswert zugeordnet ist. Auf der Grundlage dieser Liste sind dann wiederum weitere Tätigkeiten möglich, wie die Release-Planung.

Wir unterscheiden zwei Typen von Priorisierungstechniken{ XE "Priorisierungstechnik" }:

- Bei einer Ad-hoc-Priorisierungstechnik{ XE "Ad-hoc-Priorisierungstechnik" } weist (erfahrungsbasiert) ein Experte jeder Anforderung einen Wert zu. Dies geht schnell, ist jedoch etwas fehleranfälliger.
- Die analytische Priorisierungstechnik{ XE "analytische Priorisierungstechnik" } geht systematischer vor, z.B. durch paarweisen Vergleich von Anforderungen untereinander oder dadurch, dass verschiedene Experten verschiedene Kriterien bewerten, die dann zusammen die Priorität ergeben. Hier wird mehr Aufwand nötig, das Ergebnis ist jedoch sorgfältiger erstellt und zuverlässiger.

4.5 Ad-Hoc Priorisierungstechniken

Die folgenden Ad-hoc-Priorisierungstechniken haben sich in der Praxis bewährt:

- Requirements Triage
- Ranking
- Top-Ten Technik
- Ein-Kriterien-Klassifikation
- Planning Poker

- Zwei-Kriterien-Klassifikation
- 100-Dollar-Technik
- Kano-Klassifikation

4.5.1 Requirements Triage

Requirements Triage{ XE "Requirements Triage" } [Davi2003] ist eine an die Medizin angelehnte Ein-Kriterien-Klassifikation. Sie kann auch zur Vorsortierung von Anforderungen und Vereinfachung der Priorisierung eingesetzt werden. Man ordnet jede Anforderung einer von drei Kategorien zu:

- den Muss-Anforderungen, die unbedingt (z.B. im nächsten Release) umgesetzt werden müssen
- denjenigen Anforderungen, die (noch) nicht nötig sind
- den Kann-Anforderungen, bei denen die Priorität noch nicht eindeutig ist; diese Anforderungen müssen noch genauer priorisiert werden oder deren Umsetzung hängt von den verfügbaren Ressourcen ab

Die Kann-Anforderungen kann man dann noch mit einer anderen Priorisierungstechnik bewerten. Für die anderen beiden Anforderungs-Gruppen ist die Entscheidung schon gefallen. Sind es zu viele Muss-Anforderungen oder möchte man sie in eine Reihenfolge bringen, hat aber auch hier eine detailliertere Priorisierung Sinn.

Analog kann man das Triage natürlich auch einsetzen, wenn es nicht um die Entscheidung geht, welche Anforderungen umgesetzt werden sollen, sondern auch bezüglich anderer Entscheidungen.

4.5.2 Ranking

Beim Ranking{ XE "Ranking" } werden die Anforderungen von den Stakeholdern bezüglich des Priorisierungskriteriums (z.B. Nutzen, Kosten, Dringlichkeit) in eine sortierte Reihenfolge gebracht. Hierbei ist jedes Priorisierungskriterium möglich. Das Ergebnis ist dann die Zuordnung der Anforderungen zu der bereits erwähnten Ordinalskala (vgl. Kapitel 4.3), in der z.B. die wichtigste Anforderung die Nummer 1 erhält, die zweitwichtigste die Nummer 2 und so weiter. Eine übersichtliche Anzahl an Anforderungen kann man tatsächlich ad-hoc auf einem Blick sortieren. Dies kann z.B. erfolgen, indem man die Anforderungen auf Karteikarten schreibt und diese Karten auf einem Tisch sortiert, gerne auch in einer Gruppe.

Handelt es sich um mehr Anforderungen, als man auf einen Blick miteinander vergleichen kann (z.B. mehr als 10), hilft entweder eine Vorsortierung (Triage) oder eine systematische Sortiertechnik. Nach der Vorsortierung durch Requirements Triage wendet man die Priorisierungstechnik nur noch innerhalb einer Kategorie von Anforderungen an, z.B. für die Kann-Anforderungen.

Als systematische Sortiertechnik empfiehlt sich folgendes Verfahren: Man legt eine Anforderung auf den Tisch (oder zeigt sie elektronisch an). Dann nimmt man die nächste und entscheidet, ob sie wichtiger oder unwichtiger ist als die andere und positioniert sie in einer

Liste entsprechend weiter oben oder unten. Es sollten möglichst keine zwei Anforderungen denselben Rang haben, aber eine Gleichbewertung wäre im Ausnahmefall möglich. Auf diese Weise geht man mit jeder weiteren Anforderung vor. Meist ist schnell klar, ob die neue Anforderung relativ wichtig oder unwichtig ist und daher mit den Anforderungen oben oder unten auf der Liste verglichen werden muss. So erhält man zuletzt eine vollständig nach Prioritäten sortierte Liste der Anforderungen.

4.5.3 Top-Ten Technik

Oft ist eine vollständig sortierte bzw. priorisierte Liste der Anforderungen gar nicht nötig. Häufig sucht man einfach nur diejenige Gruppe von Anforderungen, die aktuell am höchsten priorisiert sind und die im nächsten Bearbeitungsschritt verarbeitet werden sollen, z.B. implementiert oder getestet. Die Anzahl der gesuchten Anforderungen ist oft auch schon ungefähr klar. Wenn beispielsweise eine Anforderung im Schnitt vier Tage Implementierungsaufwand verursacht und in der nächsten Iteration Ressourcen für 40 Personentage zur Verfügung stehen, ist das Ziel der Priorisierung, die wichtigsten 10 Anforderungen, die Top Ten{ XE "Top-Ten-Technik" }, zu ermitteln. Die restlichen Anforderungen benötigen keine Priorisierung, außer der Information, dass sie (noch) nicht zu den Top Ten gehören. Die Methode funktioniert natürlich auch für die wichtigsten drei oder 24 Anforderungen.

Zum Vorgehen: Man sammelt in einer ersten Runde alle Kandidaten für die Top Ten. Vermutlich werden dabei nicht exakt zehn Kandidaten herauskommen. Sind es zu viele, wählt man diejenigen aus, die dann doch zunächst mal nicht dazu gehören. Sind es zu wenige, betrachtet man die verworfenen Anforderungen erneut. Das Aussortieren aus der Kandidatenliste ist meist einfacher als die Neubetrachtung aller Anforderungen.

Darum sollte man in der ersten Runde im Zweifel eine Anforderung eher mal den Kandidaten zuordnen, als sie zu verwerfen. Für das Aussortieren von Anforderungen aus der Top Ten Liste kann man eine der anderen Priorisierungstechniken verwenden (z.B. Ranking oder die weiter unten beschriebenen Techniken) und dann die sortierte Liste der Anforderungen nach Platz Nr. 10 abschneiden.

Sind mehrere Stakeholder(gruppen) an der Priorisierung beteiligt, kann man es auch so machen, dass von den fünf Gruppen jede ihre Top Zwei Anforderungen erwählen darf, und diese zusammen ergeben dann die Top Ten Liste.

4.5.4 Ein-Kriterien-Klassifikation

Bei der Ein-Kriterien-Klassifikation{ XE "Ein-Kriterien-Klassifikation" } bewertet man der Reihe nach jede Anforderung nach dem Priorisierungskriterium.

Dabei kann jede sinnvolle Skala verwendet werden, beispielsweise absolute Werte wie Implementierungsaufwand in Personentagen, aber auch relative Werte auf einer selbstdefinierten Punkteskala (z.B. 0 bis 10 Punkte) oder in Kategorien wie niedrig / mittel / hoch oder muss (mandatory) / kann (optional) / nice-to-have. Das geht relativ schnell. Allerdings muss man vorher klar festlegen, welche Punktzahl was bedeutet. Beispielsweise

kann man definieren, dass die Kritikalität 10 nur vergeben werden darf, wenn das Leben von Menschen oder die Existenz der gesamten Firma auf dem Spiel steht, oder dass jede Anforderung, bei deren Verzicht mit Hilfe eines Workaround die Arbeit noch möglich ist, eine Kann-Anforderung ist. Solche Definitionen sind umso wichtiger, je mehr Personen an der Priorisierung beteiligt sind, sei es in der Gruppe oder auch, wenn aus den Einzelschätzungen der Schätzer ein Mittelwert berechnet werden soll. Wird dies nicht wie geschildert durchgeführt, dann wird man bei der Gruppenpriorisierung in vielen Fällen Zeit verschwenden (z.B. bei der Diskussion über die Bedeutung des Punktwertes 10) statt über die Prioritäten selbst zu sprechen.

Vergleicht und aggregiert man die unabhängigen Einschätzungen verschiedener Experten, erhält man im Prinzip zuverlässigere Werte als in der Gruppendiskussion, da Gruppen durch gruppendynamische Effekte nicht immer optimal entscheiden. Man sieht dann aber auch, dass anscheinend verschiedene Schätzer unterschiedlich optimistisch sind. Der eine vergibt z.B. viel öfter den Wert 1 und überhaupt niedrigere Werte, während der andere viel öfter die 10 vergibt und seine Werte insgesamt im Schnitt höher liegen. Oft beruht dieser Unterschied jedoch weniger auf Charakterunterschieden, sondern auf einem unterschiedlichen Verständnis davon, wann die Extremwerte 1 und 10 zu vergeben sind.

4.5.5 Planning Poker

Beim Planungs-Poker{ XE "Planungs-Poker" } werden ebenfalls die Anforderungen im Hinblick auf ein Kriterium bewertet, meist in Bezug auf Kosten oder ihren Nutzen im Hinblick auf die Einplanung der Anforderungen in bestimmte Releases. Diese Technik ist in der agilen Entwicklung verbreitet, funktioniert aber natürlich nicht nur dort.

Diese Technik berücksichtigt die Tücken der Entscheidungsprozesse in Gruppen und führt damit potenziell zu besseren Bewertungen als eine Gruppendiskussion. Gruppendynamische Effekte wurden ja bereits erwähnt. Schätzen die Experten separat, können sie allerdings auch unterschiedliches Verständnis entwickeln oder einen relevanten Faktor übersehen.

Darum verwendet das Planning Poker als Entscheidungsprozess einen pragmatischen Kompromiss zwischen Einzelschätzung und Gruppendiskussion. Außerdem beruht die Bewertung nicht auf einer gleichmäßigen Punkteskala von 1 bis 10, sondern auf den Fibonacci-Zahlen, die sich wohl für diesen Zweck bewährt haben. Jeder der Schätzer, die gemeinsam am Tisch sitzen, erhält einen Satz von Spielkarten mit folgenden Punktwerten: 0, 1, 2, 3, 5, 8, 13, 21, 34. Außerdem gibt es Karten, mit denen der Schätzer Diskussions- oder Pausenbedarf anmelden kann.

Bei der gemeinsamen Priorisierung der Anforderungen in der Gruppe geht man dann so vor:

1. Vorstellung der zu priorisierenden Anforderung (2 Minuten).
2. Jeder schätzt für sich (½ Minute): Jeder Schätzer wählt eine seiner Karten und legt sie verdeckt auf den Tisch, um zu signalisieren, dass er sich entschieden hat. Die anderen Schätzer sehen also nur die Rückseite der Karte.
3. Zeitgleiche Offenlegung der Karten: Sobald alle eine Karte gewählt haben, decken alle gleichzeitig ihre Karten auf.

4. Erklärung der höchsten und niedrigsten Schätzung (1 Minute): Die beiden Personen, die die niedrigste Schätzung und die höchste Schätzung abgegeben haben, erklären jeweils ihren Wert. Meist haben sie andere Annahmen zugrunde gelegt als die anderen, zu Recht oder zu Unrecht. Diese werden nun kurz diskutiert.
5. Jeder schätzt für sich (½ Minute): Die Schätzungen werden, basierend auf dem neu von den Kollegen hinzu gewonnenen Wissen, wiederholt wie zuvor: Jeder wählt eine Karte und legt sie verdeckt vor sich hin.
6. Zeitgleiche Offenlegung der Karten
7. Einigung auf eine Schätzung (1 Minute): In der zweiten Schätzrunde liegen die Werte schon näher beieinander, sind aber nicht notwendigerweise gleich. Man kann sich nun in der Gruppe auf den häufigsten Wert festlegen oder auf den Mittelwert der Schätzungen.

Diese Technik ist für alle denkbaren Priorisierungskriterien durchführbar, ob Kosten, Nutzen oder ein anderes Kriterium.

4.5.6 Zwei-Kriterien-Klassifikation

Manchmal sollen mehrere Priorisierungskriterien zugleich betrachtet werden. Es gibt verschiedene Möglichkeiten, um zwei Kriterien{ XE "Zwei-Kriterien-Klassifikation" } miteinander zu kombinieren:

- **Mit einer Formel:** Interessiert beispielsweise das Kosten-Nutzen-Verhältnis, dann ermittelt man für jede Anforderung in einem Durchgang die Kosten und in einem anderen Durchgang den Nutzen. Vermutlich sind hierbei verschiedene Stakeholder gefragt, z.B. technische Experten für die Kosten und die Benutzer für die Nutzenschätzung. Das Kosten-Nutzen-Verhältnis berechnet sich dann für jede Anforderung aus dem Quotienten zwischen beiden Werten. Die Anforderung mit dem höchsten Nutzen pro investierter Kosteneinheit hat dann die höchste Priorität. Dazu müssen Kosten und Nutzen nicht unbedingt beide in derselben Einheit (z.B. Euro) ermittelt werden. Auch ein Verhältnis in der Einheit „Punkte / Personentag" ist sinnvoll. Zur Dokumentation und Verwendung in Sichten ergibt es Sinn, im RE-Werkzeug nicht nur die Kosten- und Nutzen-Schätzungen in einem eigenen Attribut abzulegen, sondern auch deren Quotient.
- **Mit einer Matrix:** Beispielsweise die Kritikalität einer Anforderung kann man ermitteln als ihr rechnerisches Risiko, das definiert ist als die Wahrscheinlichkeit des Auftretens eines Risikoereignisses multipliziert mit dem beim Eintreten verursachten Schaden. Beim Online-Banking können das recht hohe Risiken sein. Letztlich enthält diese Zahl aber nicht alle relevante Information. Extrem selten auftretende Katastrophen mit beinahe unschätzbar hohem Schaden kommen vielleicht genauso auf 10 € pro Monat wie kleine Versehen, die regelmäßig geschehen und jedes Mal nur 0,01 € Schaden verursachen. Diese Kategorien will man oft getrennt und verschieden behandeln. Statt Wahrscheinlichkeit und Schaden zu multiplizieren, stellt man lieber in einer Risikomatrix die Risiken in einem zweidimensionalen Diagramm dar, um sie zu klassifizieren.

Abbildung 4 zeigt ein Beispiel für eine Matrix, in der Anforderungen nach ihrem Kosten-Nutzen-Verhältnis priorisiert werden. Ganz links oben (Priorität 1) finden sich die „Quick wins", d.h. diejenigen Anforderungen, die bei geringen Kosten einen hohen Nutzen erbringen. Diese erhalten damit Priorität 1, d.h. die höchste Priorität. In den Feldern mit den Prioritäten 2 und 3 ist der Nutzen auch noch höher als die Kosten. Die Anforderungen auf der Diagonale, wo Kosten und Nutzen sich ungefähr die Waage halten, werden in dieselbe Kategorie (Priorität 4) klassifiziert, und so weiter.

Wie Sie die Anforderungen aufgrund der beiden Kriterien in Prioritätskategorien zuordnen, ist Ihre Entscheidung und beeinflusst natürlich das Weitere, z.B. die Release-Planung.

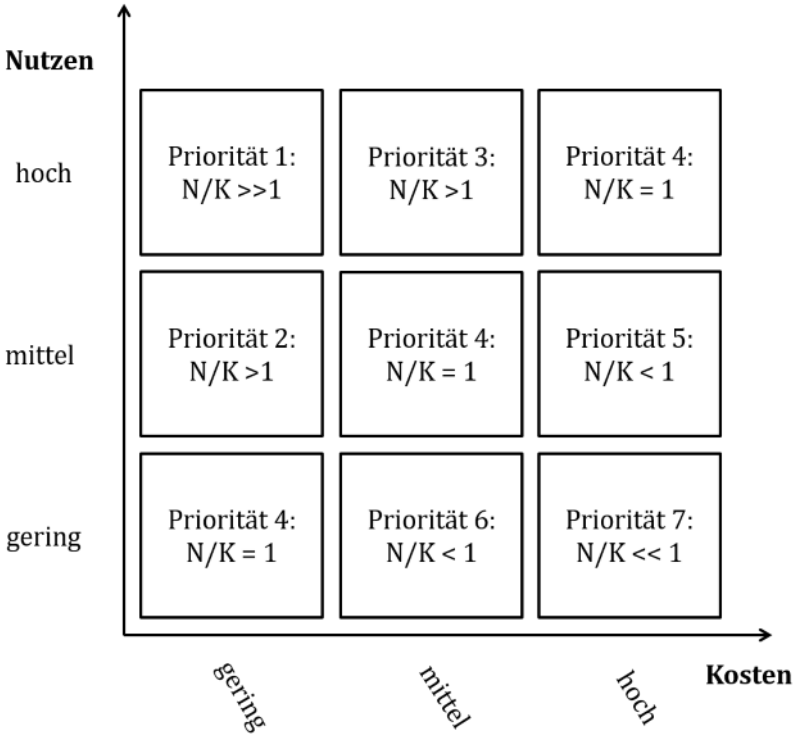


Abbildung 4: Priorisierungsmatrix für Anforderungen nach Kosten und Nutzen. N/K bezeichnet hier das Verhältnis (d.h. den Quotienten) aus Nutzen und Kosten.

Abbildung 5 zeigt ein Beispiel für eine Priorisierungsmatrix nach Risiken (=Risikomatrix). Hier werden die mit einer Anforderung verknüpften Risiken priorisiert und damit die Kritikalität der Anforderung, die dadurch bedroht wird. Beispielsweise bringt die Anforderung (bzw. Funktionalität) der Überweisung im Online-Banking die Gefahr mit sich, dass Hacker die Zugangsdaten des Kontoinhabers ausspionieren und eine unberechtigte Überweisung durchführen. Diese Gefahr ist möglich und schwerwiegend, liegt hier also in einem roten Kästchen. Dies bedeutet, dass unbedingt Gegenmaßnahmen dagegen ergriffen werden müssen.

Bei denjenigen Risiken, die in einem gelben Bereich liegen, wird man abwägen, welche Maßnahmen wirtschaftlich sinnvoll sind. Und die Risiken im grünen Bereich wird man eventuell akzeptieren, außer sie sind mit einfachen Gegenmaßnahmen zu verhindern. Die

drei Bereiche der Matrix bestimmen also darüber, wie mit den jeweiligen Risiken umgegangen werden soll.

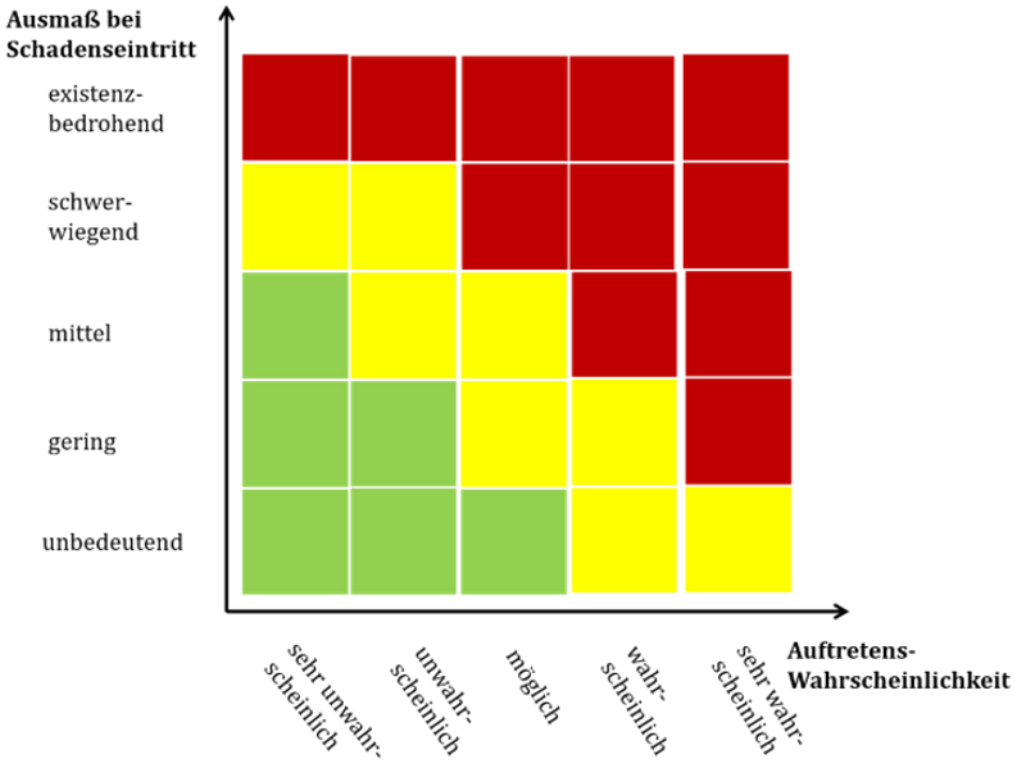


Abbildung 5: Risikomatrix für die Priorisierung von Anforderungen in Bezug auf Risiko bzw. Kritikalität

4.5.7 Die 100-Dollar-Technik

Die 100-Dollar-Technik{ XE "100-Dollar-Technik" } [LeWi2000] eignet sich besonders gut für die Priorisierung mit mehreren Personen, die nicht unbedingt zu einer Gruppendiskussion zusammen kommen müssen.

Bei dieser Technik erhält jeder Stakeholder 100 imaginäre Einheiten (Geld, Zeit, etc.), die er auf die Anforderungen verteilen darf. Dabei soll eine Anforderung, wenn sie ihm z.B. doppelt so viel Geld wert ist wie eine andere, auch doppelt so viele Einheiten / Punkte zugewiesen bekommen. Jeder Stakeholder darf insgesamt nur 100 Einheiten vergeben. Am Ende werden die Punkte, welche die verschiedenen Stakeholder derselben Anforderung zugeteilt haben, aufsummiert. Die Anforderung mit den meisten Punkten hat auch die höchste Priorität.

Diese Technik ist bei größeren Anzahlen an Anforderungen schwierig umzusetzen. Die Stakeholder tun sich dann schwer, alle Anforderungen gegeneinander abzuwägen, und es ist auch schwieriger sicherzustellen, dass nicht mehr als 100 Punkte vergeben werden. Daher empfehlen wir diese Technik vor allem für die Priorisierung grober Anforderungen (z.B. von Features) mit einem hohen Abstraktionsniveau oder für die gegenseitige Priorisierung von ganzen Anforderungsgruppen. Innerhalb der Anforderungsgruppe kann dann wiederum die 100-Dollar-Technik verwendet werden, um die Anforderungen untereinander zu vergleichen.

Bei dieser Technik sind zwei Fallen zu vermeiden: Wenn die Stakeholder es sich einfach machen, vergeben sie an jede Anforderung dieselbe Punktzahl. Beispielsweise von 100 Anforderungen erhält jede genau einen Punkt. Dann werden am Ende alle Anforderungen gleich wichtig sein und die Priorisierung bringt keinen Nutzen. Weisen Sie die Stakeholder darum darauf hin, dass sie ihre Punkte möglichst ungleich verteilen sollen, um zu eindeutigen Aussagen zu kommen. Teilt ein Stakeholder tatsächlich jeder Anforderung gleich viele Punkte zu, dann können Sie diese Bewertungen auch ablehnen und eine Neueinschätzung anfordern.

Außerdem müssen die Stakeholder ihre Bewertungen unabhängig voneinander abgeben, um einander nicht zu beeinflussen. Eine solche unabhängige Stimmabgabe kann per Fragebogen oder Einzelinterview geschehen. In einer Abstimmung in der Gruppe könnte ein Stakeholder die vermeintliche Fehleinschätzung seines Kollegen durch Gegensteuern „reparieren". Aber auch die Kenntnis der Vorlieben der anderen Schätzer (ohne ihre Punktevergabe zu kennen) beeinflusst bereits das Abstimmungsverhalten. Weiß ein Stakeholder, dass die anderen seine Lieblingsanforderung niedrig bewerten werden, während sie anderen ebenfalls wichtigen Anforderungen selbstverständlich viele Punkte geben werden, so wird er seine Punkte auf seine Lieblingsanforderung fokussieren im Vertrauen darauf, dass die anderen Anforderungen ihre Punkte von jemand anderem erhalten. Auf diese Weise sind am Ende auch wieder alle Anforderungen scheinbar gleich wichtig.

Es gibt die 100-Dollar-Technik auch in einer Variante mit 1.000 oder 10.000 Einheiten. Mit mehr Punkten sind natürlich auch differenziertere Bewertungen möglich. Die Priorisierung verursacht dann jedoch auch mehr Aufwand, und die Überprüfung, wie viele Punkte ein Stakeholder insgesamt vergeben hat, ist noch schwieriger. Hierzu benötigt man praktisch dann ein Werkzeug.

4.5.8 Kano-Klassifikation

Im Kano-Modell{ XE "Kano-Modell" } [Kano1984] klassifiziert und priorisiert man Anforderungen in drei Kategorien, wobei im Hinblick auf die Benutzererwartungen klassifiziert wird. Das Kano-Modell ist bereits im Handbuch zum Foundation Level ausführlich beschrieben [PoRu2015] und soll darum hier nur kurz wiederholt werden. Die Anforderungen werden bei dieser Technik einer von drei Kategorien zugeordnet:

- *Basisfaktoren*{XE "Basisfaktor" } sind Anforderungen, deren Erfüllung die Benutzer als selbstverständlich voraussetzen. Darum macht es sie nicht ausdrücklich zufrieden, wenn die Anforderung erfüllt ist, sondern nur sehr unzufrieden, wenn sie nicht erfüllt ist.
- *Leistungsfaktoren*{XE "Leistungsfaktor" } sind Anforderungen, die von den Benutzern ausdrücklich angefordert werden. Deren Erfüllung macht die Benutzer zufrieden, die Nichterfüllung unzufrieden.
- *Begeisterungsfaktoren*{XE "Begeisterungsfaktor" } sind Anforderungen, mit denen der Benutzer gar nicht rechnet. Sind sie nicht erfüllt, fällt ihm das gar nicht auf. Ist sie jedoch erfüllt, dann begeistert ihn diese Innovation.

Um zu ermitteln, zu welcher Kano-Kategorie eine Anforderung gehört, stellt man den Benutzern also zwei Fragen: Wie zufrieden wären Sie, wenn die Anforderung erfüllt ist (Zufriedenheit)? Wie unzufrieden wären Sie, wenn die Anforderung nicht erfüllt ist (Unzufriedenheit)?

Abbildung 6 zeigt eine Matrix für eine Zwei-Kriterien-Priorisierung nach Kano. Sie sehen, dass es außer den drei oben genannten Anforderungs-Kategorien auch noch eine weitere gibt, die allerdings selten auftritt, nämlich die unerhebliche Anforderung.

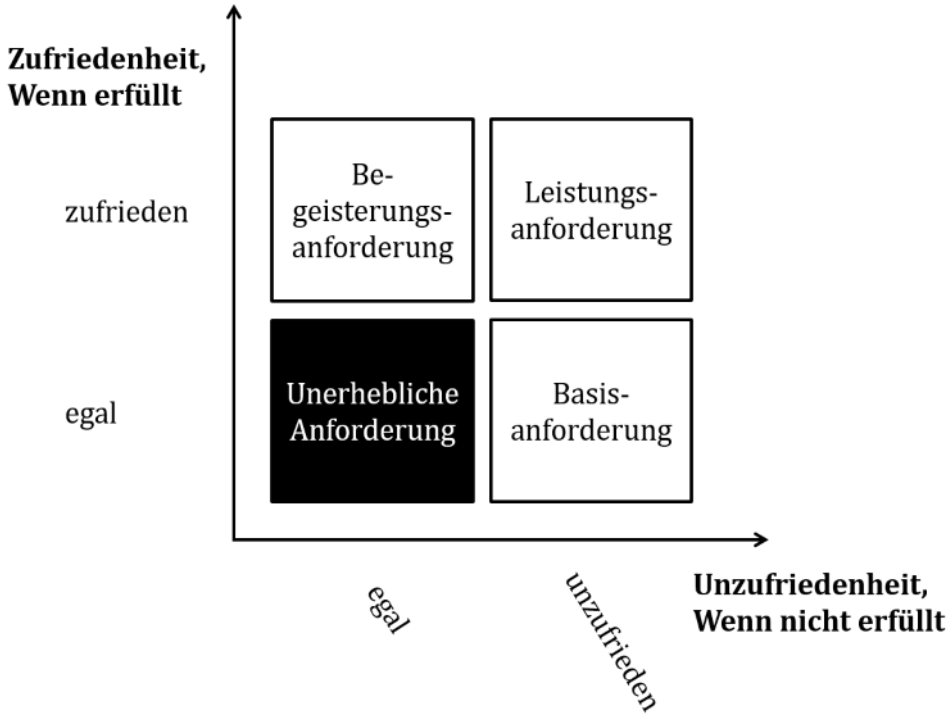


Abbildung 6: Matrix für eine Zwei-Kriterien-Priorisierung nach Kano

4.6 Analytische Priorisierungstechniken

Die Ad-hoc-Priorisierungstechniken haben den Vorteil, einfach einsetzbar und recht effizient zu sein. Allerdings sind ihre Ergebnisse subjektiv und oft später nicht mehr nachvollziehbar. Bei kritischen Entscheidungen oder auch im sicherheitskritischen Umfeld sind sie nicht optimal. Die analytischen Priorisierungstechniken erlauben dort eine neutralere und nachvollziehbarere Priorisierung. Wir stellen hier zwei Techniken vor:

- Wiegers'sche Priorisierungsmatrix
- Analytical Hierarchy Process (AHP)

4.6.1 Wiegers'sche Priorisierungsmatrix

Die Wiegers'sche Priorisierungsmatrix{ XE "Wiegers'sche Priorisierungsmatrix" } ist eine Priorisierungstechnik, bei der mehr als zwei Kriterien verwendet werden, um Anforderungen zu priorisieren. Diese Technik stellt den relativen Vorteil (bei ihrer Erfüllung) und den relativen

Nachteil (bei Nichterfüllung) jeder Anforderung den relativen Kosten und dem relativen Risiko dieser Anforderung gegenüber [WiBe2013]. Abbildung 7 zeigt ein Beispiel bzw. einen Auszug aus einer solchen Matrix für unser Online-Banking-System.

	Rel. Nutzen	Rel. Nachteil	Gesamt	Wert %	Rel. Kosten	Kosten %	Rel. Risiko	Risiko %	Priorität	Rang
Relatives Gewicht	1	1			1		3			
Anforderung										
Überweisung	9	9	18	4,0	6	3,8	9	4,5	0,231	3
Kontostand	9	9	18	4,0	4	2,5	8	4,0	0,276	1
Rückruf-Funktion	6	3	9	2,0	6	3,8	3	1,5	0,241	2
...										
Gesamt	240	210	450	100	160	100	200	100		

Abbildung 7: Wiegers'sche Priorisierungsmatrix für ein Beispiel.

Das Vorgehen zur Berechnung der Priorität folgt dabei dem folgenden Prozess:

1. Vorlage erstellen oder besorgen
2. Gewichtung der Priorisierungskriterien festlegen und eintragen: In unserem Beispiel des Online-Banking wiegen beispielsweise die Risiken mehr als die Kosten.
3. Liste der zu priorisierenden Anforderungen einfügen. Wie schon zuvor bemerkt, sollten diese Anforderungen auf derselben Detaillierungsebene stehen. Wir verwenden hier beispielhaft Anwendungsszenarien.
4. Jede Anforderung in Bezug auf Priorisierungskriterium „Nutzen" bewerten, den die Anforderung bei Erfüllung mit sich bringt, bei Wiegers auf einer Skala von 1 bis 9
5. Jede Anforderung in Bezug auf Priorisierungskriterium „Nachteil" bewerten, also den Nachteil, den es bringt, wenn die Anforderung nicht erfüllt ist, ebenfalls auf einer Skala von 1 bis 9
6. Den Gesamtwert der beiden berechnen, als gewichtete Summe, gewichtet nach den Gewichtungsfaktoren. Diese Summierung kann in einer Vorlage automatisch geschehen.
7. Den prozentualen Wert jeder Anforderung in Bezug auf die gesamte Anforderungsliste berechnen, also *Gesamtwert / Summe aller Gesamtwerte*. Auch dies kann automatisch geschehen.
8. Jede Anforderung in Bezug auf Priorisierungskriterium „Kosten" bewerten, wieder auf einer Skala von 1 bis 9
9. Berechnen des prozentualen Anteils der Kosten an den Gesamtkosten, also *Kosten / Summe aller Kosten*
10. Jede Anforderung in Bezug auf Priorisierungskriterium „Risiko" bewerten, wieder auf einer Skala von 1 bis 9
11. Berechnen des prozentualen Anteils des Risikos am Gesamtrisiko, also *Risiko / Summe aller Risiken*
12. Berechnen der Priorität jeder Anforderung nach der folgenden Formel: *Priorität = Wert % / (Kosten % x GewichtKosten + Risiko % x GewichtRisiko)*. Es handelt

sich hierbei um eine Priorisierung nach Nutzen-Kosten-Verhältnis, wobei das Risiko zu den Kosten aufgeschlagen wird. (Andere Priorisierungstechniken ziehen das Risiko vom Nutzen ab.)

13. Bestimmen des Rangs jeder Anforderung: Eine Anforderung hat einen umso höheren Rang je höher ihre Priorität.

Priorisierung der Anforderungen mit der Wiegers'schen Priorisierungsmatrix

In unserer Fallstudie hat der Requirements Manager Peter Reber entschieden, die Wiegers'sche Methode zu verwenden, weil im Banken-Umfeld Entscheidungen sorgfältig und gut begründet getroffen werden müssen. Zusätzlich zu der Wiegers'sche Matrix sollen auch die Begründungen für die jeweiligen Bewertungen dokumentiert werden, d.h. warum in welchem Feld wie viele Punkte vergeben wurden.

Die Schätzung der Werte erfolgt durch folgende Stakeholder: In Bezug auf Nutzen und Nachteil macht die Usability-Expertin ein Requirements Triage und sortiert alle diejenigen Anforderungen aus, die ganz wichtig und recht unwichtig sind und vergibt diesen Punkte. Den Nutzen und den Nachteil der restlichen Anforderungen bestimmt der Kundenbeirat. Zu den Kosten werden die Entwickler befragt. Diese bestimmen die Kosten gemeinsam durch Planning Poker. Das Risiko untersuchen die IT-Sicherheitsexperten. Dazu bestimmen sie zunächst anhand einer Risiko-Analyse, welche Risikoereignisse überhaupt in Bezug auf eine Anforderung auftreten können. Dann ermitteln sie die Wahrscheinlichkeit aufgrund ihrer Erfahrung durch AHP und den Schaden aufgrund einer Analyse von firmenspezifischen Statistiken und Kennzahlen.

Abbildung 7 zeigt das Ergebnis der Schätzungen bzw. einen Auszug davon. Man sieht hier, dass die relativ unwichtige, aber auch weniger teure und weniger riskante Rückruffunktion es auf den zweiten Platz geschafft hat, noch vor der Überweisung.

Tipp aus der Praxis: Das Ergebnis der Priorisierung hängt natürlich von den gewählten Priorisierungskriterien und der verwendeten Technik ab! Hätten Sie die obigen Anforderungen mit der Kano-Methode klassifiziert, wäre die Überweisung als Basisfunktion des Online-Bankings natürlich wichtiger gewesen als die Rückruffunktion. Wählen Sie darum die Priorisierungskriterien weise aus!

4.6.2 Analytical Hierarchy Process (AHP)

Der Analytical Hierarchy Process{ XE "Analytical Hierarchy Process" } AHP ist eine mathematisch raffinierte, theoretisch sehr interessante Methode. Um die Vorteile von AHP

nutzen zu können, ist es ratsam, ein Werkzeug zu verwenden, das die Methode unterstützt und die nötigen Berechnungen durchführt.

Die Grundidee der Methode ist jedoch einfach. Die Priorisierung wird für den Schätzer vereinfacht, indem er immer nur zwei Anforderungen paarweise miteinander vergleichen muss: Welche von beiden ist wichtiger (oder teurer oder riskanter) als die andere und um wie viel? Jedes Priorisierungskriterium ist hier möglich. Diese Entscheidung zwischen zwei Anforderungen ist leichter zu treffen als die Frage, welche von einer Liste von Anforderungen die wichtigste ist. Der Nachteil der Methode besteht allerdings darin, dass man jede Anforderung paarweise mit jeder anderen vergleichen muss. Dadurch kommt man bei *n* Anforderungen auf *n (n–1) /2* Vergleiche. Das erzeugt besonders viel Aufwand. Eine Beispielrechnung für den Aufwand folgt unten im Praxistipp. Die große Stärke, die keine andere Methode aufzuweisen hat, ist, dass die Methode Fehler der Schätzer ausgleichen kann. Wenn man von drei Anforderungen A, B und C bisher A für wichtiger gehalten hat als B und B wichtiger als C, dann muss A auch wichtiger als C sein. Hält man nun aber C für wichtiger als A, liegt offensichtlich ein Fehler vor. Wie gut und zuverlässig insgesamt die Bewertungen sind, kann die Methode anhand eines „*consistency ratio*" messen. Die Mathematik der Methode wurde vom Erfinder Saaty beschrieben [Saat1990]. Eine Zusammenfassung finden Sie bei Karlsson und Ryan [KaRy1997].

Uns interessiert hier vor allem das Vorgehen aus Schätzer–Sicht. Der Schätzer erhält jeweils zwei Anforderungen zur Auswahl und soll entscheiden, welche von beiden wichtiger ist. Dafür wird die Skala aus Abbildung 8 verwendet. Abgesehen davon, dass der Schätzer dies nun für sehr viele Anforderungspaare zu tun hat, ist die Methode nicht weiter schwierig für ihn. Die Auswertung übernimmt ein Werkzeug oder ein Experte, der anschließend die Prioritäten der Anforderungen ermittelt.

Skalenwert	Bedeutung
1	Anforderungen A und B sind gleich wichtig
3	A ist etwas wichtiger als B.
5	A ist sehr viel wichtiger als B.
7	A ist erheblich wichtiger als B.
9	A ist absolut dominierend über B.
2, 4, 6, 8	Dies sind Zwischenwerte

Abbildung 8: Skala des AHP für den Vergleich zweier Anforderungen.

Für die Unterstützung von AHP gibt es Open Source Werkzeuge, beispielsweise PriEsT [SMK2013], [PriEsT].

Tipp aus der Praxis: Die Wiegers'sche Matrix verlangt für jede Anforderung die Schätzung von vier verschiedenen Werten. Sie müssen damit rechnen, dass jede Schätzung 1–2 Minuten dauert. Wenn Sie also 100 Anforderungen in Ihrer Liste haben, bedeutet das 400–800 Minuten Aufwand, also 6,7 bis 13,3 Stunden, und das ohne Pausen.

Beim Planning Poker benötigt die Priorisierung einer Anforderung bei diszipliniertem Ablauf je fünf Minuten. 100 Anforderungen schafft man also in 500 Minuten, also 8,3 Stunden.

AHP schneidet am schlechtesten ab: Bei 100 Anforderungen muss man 100 x 99 / 2 = 4950 Vergleiche machen. Wenn jeder Vergleich eine halbe bis ganze Minute benötigt, kommt man auf 41 bis 82 Stunden Aufwand, also 5–10 Arbeitstage- pro Schätzer.

Sie sollten sich darum vorher gut überlegen, welche Anforderungen Sie priorisieren wollen und wie viele Personen dafür wirklich nötig sind. Die Ad-hoc Techniken kommen, da sie meist nur ein oder zwei Kriterien pro Anforderung schätzen, mit entsprechend weniger Aufwand aus, wenn man sich nicht in langen Diskussionen verliert. Rechnen wir 1–2 Minuten pro Anforderung, sind das bei 100 Anforderungen 1,6 bis 3,3 Stunden.

Die Anwendung der Wiegers'schen Matrix und der AHP-Methode werden nur bei Anforderungslisten mit maximal 30 Anforderungen empfohlen [WiBe2013], [Mois2002].

4.7 Kombination von Priorisierungstechniken

Verschiedene Priorisierungstechniken haben verschiedene Vor- und Nachteile. Die Ad-hoc-Techniken sind einfach anzuwenden, führen aber zu wenig nachvollziehbaren und auch nicht ganz objektiven Ergebnissen. Die analytischen Techniken sind zwar besser, skalieren jedoch schlecht. In der Wiegers'schen Priorisierungsmatrix sind pro Anforderung vier Schätzungen durchzuführen, und bei AHP verursacht die Priorisierung von doppelt so vielen Anforderungen nicht doppelt so viel Aufwand wie bei den meisten Techniken, sondern vierfach so viel Aufwand. Für längere Anforderungslisten entstehen so Aufwände von vielen Stunden oder sogar Tagen.

Da in vielen Projekten mit hunderten oder gar tausenden von Anforderungen gearbeitet wird, sind pragmatische Lösungen nötig. Zumeist ist es gar nicht nötig, die gesamte Anforderungsliste detailliert zu priorisieren. Man kann viel Zeit sparen bei beinahe gleicher Priorisierungsqualität, wenn man eine Ad-hoc-Technik und eine analytische Technik kombiniert.

In einer ersten Runde der Priorisierung kann beispielsweise eine Ad-Hoc Priorisierungstechnik angewendet werden, um die Anzahl der zu betrachtenden Anforderungen zu reduzieren. Handelt es sich beispielsweise darum, die wichtigsten Anforderungen für das nächste Release zu ermitteln, ist für die aktuell unwichtigeren Anforderungen keine besonders raffinierte Priorisierung nötig, da sie ohnehin erstmal zurückgestellt werden. Auch bei den offensichtlich dringenden Anforderungen ist keine weitere Differenzierung mehr nötig. Eine nähere Betrachtung lohnt sich aber bei den Anforderungen, die zunächst ungefähr gleich wichtig erscheinen, von denen aber die einen noch mit ins Release können, während andere zurückgelassen werden müssen. In dieser Anforderungsgruppe können Sie nun eine analytische Technik einsetzen, um die Trennlinie zwischen den Anforderungen zu ziehen, die ins Release dazu kommen und denjenigen, die nicht dazu kommen.

Im Fallbeispiel in Kapitel 4.6.1 haben wir bereits mögliche Kombinationen aus verschiedenen Priorisierungstechniken präsentiert. Welche Technik man wie verwendet, hängt davon ab,

welche Kriterien für die Priorisierung verwendet werden sollen, was das Ziel der Priorisierung ist (Top Ten ermitteln? Zwei Gruppen von Anforderungen gegeneinander abwägen?), wie viel Zeit man zur Verfügung hat, ob eine oder mehrere Personen befragt werden sollen (manche Methoden eignen sich weniger für die Entscheidungsfindung in einer Gruppe) und welche Kenntnisse die Stakeholder haben.

4.8 Inhalte für den RMP

Beim Erstellen des RMPs ist festzulegen, nach welchen Kriterien Anforderungen (in Bezug auf welche Entscheidung) priorisiert werden sollen, auch wann, von wem und mit welcher Technik. Es ist zu beachten, dass das Attributierungsschema diejenigen Attribute enthält, die diesen Priorisierungskriterien entsprechen. Nur so können die Prioritäten innerhalb des RM-Werkzeugs nicht nur in den Attributen dokumentiert werden, sondern auch ausgewertet werden, z.B. um für die Release- oder Iterationsplanung die wichtigsten Anforderungen auszufiltern.

4.9 Vertiefende Literatur

[Cohe2005] Mike Cohen: Agile Estimating and Planning, Prentice Hall International, 2005.

[Davi2005] Alan M. Davis: Just Enough Requirements Management – Where Software Development Meets Marketing. Dorset House Publishing, 2005.

[Ma2009] Qiao Ma: The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review. Master Thesis, AUT University, 2009, https://openrepository.aut.ac.nz/server/api/core/bitstreams/2f0f93a5-05b2-4688-9e07-6be9534cb3ca/content

5 Versions- und Änderungsmanagement

Das gesamte Leben ist von Veränderungen und neuen Anforderungen geprägt, die teilweise von außen und teilweise von innen heraus motiviert sind. Genauso kennen wir Änderungswünsche aus allen Projekten – von der Städteplanung bis zur Softwareentwicklung.

(Ver-)Änderungen sind per se nichts Schlechtes und passieren – unabhängig davon wie gut die ursprünglich abgenommene Vertragsgrundlage oder das abgenommene Anforderungs-Dokument auch waren. Speziell bei der Entwicklung von technischen Systemen erlebt man über die Phase der Anforderungserhebung hinweg einen starken (wenn auch nicht stetigen) Anstieg des Problem- und Lösungs-Bewusstseins durch über die Zeit gewonnene Erkenntnisse.

> „Requirements are rarely static. Although from the development management perspective, it is desirable to freeze a set of requirements permanently, it is rarely possible. Requirements that are likely to evolve should be identified and communicated to both acquirers and the technical community. A core subset of requirements may be frozen early. *The impact of proposed new requirements are evaluated to help ensure that the initial intent of the requirements baseline is maintained or that changes to the intent are understood and accepted by the acquirer." [ISO29148]*

Damit Sie der Änderungen Herr bleiben und Sie nicht von den Änderungen beherrscht oder gar überrollt werden, ist es für Sie als Requirements Manager besonders wichtig, auf den Umgang mit Änderungen vorbereitet zu sein. Planen Sie aus diesem Grunde bereits im RMP, wie Sie mit Änderungen im Rahmen der Anforderungserhebung sowie im späteren Projektverlauf umgehen wollen. Die folgenden Abschnitte erläutern die grundsätzlichen Konzepte, um sich im Dschungel der sich kontinuierlich ändernden Anforderungen und Anforderungs-Dokumente zurechtzufinden (Versionskontrolle). Sie erläutern auch welche Gründe zu Änderungen führen und wie diese durch einen Änderungsmanagement-Prozess umgesetzt werden können.

5.1 Versionierung von Anforderungen

Die Versionierung von Anforderungen ermöglicht es, die Entwicklung einer Anforderung über ihren gesamten Lebenszyklus zu verfolgen.

Das heißt, durch die Versionierung von Anforderungen haben wir zu jedem Zeitpunkt die Möglichkeit,

- ...Aussagen über die Änderungshäufigkeit zu treffen.
- ...die Evolution der einzelnen Anforderung(en) zu überprüfen.
- ...auf vorherige Anforderungs-Versionen zurückzugreifen.

Im direkten Zusammenhang mit der Versionierung von Anforderungen ist das Konfigurationsmanagement zu betrachten. Im Konfigurationsmanagement werden

bestimmte Sätze von Anforderungs-Versionen in eine Anforderungs-Konfiguration zusammengefasst (vgl. Kapitel 5.1.2).

Das Software-Konfigurationsmanagement ist die Disziplin zur Verfolgung und Steuerung der Evolution von Software. Es ist unerlässlich für die Entwicklung und Pflege großer, langlebiger Softwaresysteme [BSB2008].

In den folgenden Abschnitten wird beschrieben:

- wie eine Versionskontrolle von Anforderungen umgesetzt werden kann,
- was Anforderungskonfigurationen sind,
- was Anforderungs-Baselines sind,
- was bei der parallelen Weiterentwicklung von Anforderungen zu beachten ist.

5.1.1 Versionskontrolle für Anforderungen und Anforderungs-Dokumente

Die Versionskontrolle von Anforderungen bezeichnet den Vorgang, der es ermöglicht, über den Lebenszyklus eines Systems oder Produktes hinweg spezifische Entwicklungsstände von Anforderungen und Anforderungs-Dokumenten verfügbar zu halten.

Definition 5-1:

Versionskontrolle: Die Versionskontrolle (bzw. ein Versionskontrollsystem) dient der Dokumentation, Verwaltung und Wiederherstellung von Dokumenten, Dateien und einzelnen Artefakten (z.B. Anforderungen). Die Versionskontrolle ermöglicht es, Änderungen an Dokumenten und Artefakten nachvollziehen zu können und durchgeführte Änderungen zu revidieren, um zu alten Versionen zurückkehren zu können. Die Versionskontrolle ermöglicht demzufolge eine fortlaufende Betrachtung der Evolution eines Dokuments bzw. Artefakts über seine gesamte Lebenszeit.

Bevor wir allerdings auf die Versionierung und Versionskontrolle von Anforderungen eingehen, wollen wir einen kurzen Exkurs zu Zuständen von Anforderungen machen, da Zustände und Versionen zwar eng zusammenhängen und somit oft miteinander vermischt oder gar verwechselt werden – Zustände und Versionen aber zwei unterschiedliche Konzepte sind.

Zustände von Anforderungen

Definition 5-2:
Zustand{ XE "Status" } nach [RuSo2009]: „Zustände geben an, wie weit die Bearbeitung der Anforderung fortgeschritten ist. *Vergleicht man den Lebensweg einer Anforderung mit einem Projektplan, dann entsprechen die Zustände der Anforderung oft den Meilensteinen im Projektplan.“*

Wenn wir uns die Evolution einer einzelnen Anforderung oder einer Anforderungs-Spezifikation ansehen, so wird diese im Verlauf ihres Lebenszyklus unterschiedliche Zustände aufweisen können, z.B. startend von „angelegt“ mit der Erfassung der Anforderung, über „in Prüfung“, „freigegeben“ und so weiter, siehe Abbildung 9. Diese Zustände können bei einem einzelnen Anforderungs-Artefakt beispielsweise durch ein Attribut (z.B. Status) für das jeweilige Anforderungs-Artefakt (siehe Kapitel 3) festgehalten werden.

Bei Dokumenten findet man hingegen oftmals einen einleitenden Dokumententeil, der neben dem Titel, dem Autor, dem letzten Änderungsdatum und der Versionsnummer auch den Zustand (Status) des Dokuments enthält. Hierbei ist der Dokumentenstatus in der Regel abhängig vom Zustand der einzelnen Anforderungen des Dokuments.

Welche Zustände und Zustandsübergänge für Anforderungs-Artefakte oder Dokumente zulässig sind, können Sie individuell für ihr Projekt festlegen. Die benötigten Zustände und Zustandsübergänge sind dabei abhängig vom durchgeführten Projekt und RE-Prozess inkl. der geplanten Review-Zyklen.

Die folgende Abbildung 9 illustriert einen einfachen Zustandsautomaten, der mögliche Zustände und Zustandsübergänge für ein Anforderungs-Artefakt darstellt. So kann eine Anforderung beispielsweise in einer Version 0.1 angelegt werden, in den Status „in Prüfung“ überführt, freigegeben und sogar implementiert werden, ohne dass es jemals eine inhaltliche Änderung an der Anforderung gab, die eine neue Version erfordert hätte.

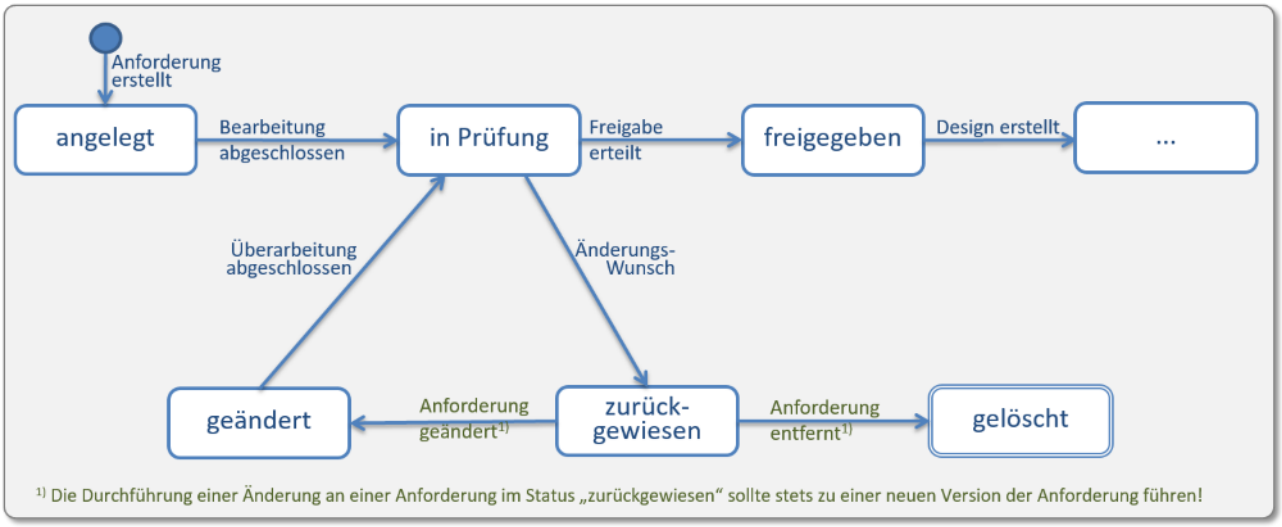


Abbildung 9: Zustände und Zustandsübergänge einer Anforderung

Versionen, Versionierung und Versionskontrolle

Im Vergleich zum Zustand einer Anforderung, welcher beispielsweise einen projektspezifischen Lebenszyklus der Anforderung darstellt, beschreibt eine Anforderungs-Version einen bestimmten inhaltlichen Stand einer Anforderung. So ist es durchaus möglich, dass eine Anforderung im Status „angelegt“ mehrere Versionsstände durchläuft, bevor diese in den Status „in Prüfung“ überführt wird. Gleiches gilt für eine zurückgewiesene Anforderung, welche unter Umständen in mehreren Iterationen geändert wird.

Definition 5-3:

{ XE "Version" }Eine *Version* ist ein zu einem bestimmten Zeitpunkt gültiger Inhaltsstand eines Anforderungs-Artefakts oder Dokuments. Über die Versionierung lässt sich die Historie eines Anforderungs-Artefakts oder -Dokuments lückenlos zurückverfolgen und auf eine frühere Version zurücksetzen. Inhaltliche Änderungen führen immer zu neuen Versionen.

Den Prozess der Erstellung neuer Versionen bezeichnen wir fortan als Versionierung. Eine Versionierung kann auf unterschiedlichen Ebenen erfolgen (z.B. auf Dokumentenebene oder auf Ebene der atomaren Anforderungen).

- Bei der Versionierung auf der Ebene von Dokumenten muss jede inhaltliche Veränderung des Dokuments (also z.B. die Änderung einer oder mehrerer Anforderungen innerhalb des Dokuments) zu einer neuen Version führen.
- Bei der Versionierung auf Anforderungsebene muss jede inhaltliche Veränderung eines Anforderungs-Artefakts zu einer neuen Version der Anforderung führen.

Bei der Versionierung ist grundsätzlich zu beachten, dass die „neue" Version die „alte" Version vollständig ersetzt. Falls Sie also eine bzw. mehrere textuelle Anforderungen durch eine modellbasierte Beschreibung (z.B. Aktivitätsdiagramm) beschreiben und die modellbasierte Beschreibung nur eine ergänzende (formalisierte) Sicht darstellt, sprechen wir nicht von einer neuen Version der ursprünglichen Anforderung. Hierbei handelt es sich vielmehr um eine ergänzende Beschreibung, welche parallel existieren darf und sollte. Um diese Abhängigkeit deutlich zu machen, können Verfolgbarkeits-Beziehungen genutzt werden – diese werden allerdings erst in Kapitel 6 eingeführt.

Durch Versionierung wird eine Versionskontrolle ermöglicht, die es den Requirements Manager beispielsweise erlaubt, unterschiedliche Dokumentationsstände (Versionen) miteinander zu vergleichen oder auf vorherige Dokumentationsstände (Versionen) zurückzuspringen (siehe **Definition 5-1 Versionskontrolle**). Zur Versionskontrolle gehören laut [WiBe2013] folgende Aktivitäten:

- **Definieren eines Schemas zur Kennzeichnung von Versionen**: Definieren Sie, nach welchem Schema Anforderungen und Anforderungs-Konfigurationen bzw. Dokumente versioniert werden sollen. Z.B.: Eine neue Version der Anforderung wird durch Hochzählen der Versionsnummer erzeugt – die Anforderungs-ID bleibt dabei unverändert!
- **Identifizieren von Versionen einzelner Anforderungen**: Legen Sie fest, wie Änderungen an einzelnen Anforderungen identifiziert werden können sollen. Legen Sie also fest, was festgehalten werden muss, um die Änderung zur letzten Anforderungs-Version ausreichend zu dokumentieren.
- **Identifizieren von Versionen für Anforderungs-Konfigurationen** (bzw. Dokumente): Legen Sie fest, wie Änderungen an Anforderungs-Konfigurationen identifiziert werden können sollen. Legen Sie also fest, was festgehalten werden muss, um die Änderungen zur letzten Dokumentenversion ausreichend zu dokumentieren.

Für die Versionierung von Anforderungen oder Dokumenten gibt es keine fest vorgeschriebene Vorgabe. Grundsätzlich kann man unterschiedliche Versionen über ganzzahlige Versionsnummern kennzeichnen (also: 1, 2, 3, ...). Empfehlenswert ist allerdings eine Versionierung auf Basis von Inkrementen, um bereits über die Versionsnummer einen ersten Indikator zu haben, ob es sich um eine grundsätzliche Änderung oder um eine marginale Anpassung (z.B. Rechtschreibkorrektur) handelt. Die Erhöhung des Inkrements steht dabei i.d.R. für eine marginale inhaltliche Anpassung und die Erhöhung um eine volle Version für eine umfassende inhaltliche Anpassung. Die Klassifizierung in marginale oder umfassende Änderungen sind natürlich eher subjektive Entscheidungen. Diese lassen sich auch mit einigen Konventionen objektivieren.

Versionierung von Anforderungs-Dokumenten

Bei der Versionierung von Dokumenten empfiehlt sich grundsätzlich die Verwendung eines Werkzeugs zur Versionsverwaltung (Version Control System). Zur manuellen Versionierung von Dokumenten ist eine Versionierungs-Kennzeichnung (z.B. auf Basis von Inkrementen) im Dateinamen sinnvoll. Hiermit kann für jede Überarbeitung am Dokument eine dedizierte Version erstellt werden.

Zur Dokumentation der Änderung ist es darüber hinaus wichtig, dass ein Dokument (auf den ersten Seiten) eine Dokumentenhistorie besitzt (vgl. Tabelle 4), um die durchgeführten Änderungen auf einem Blick erkennen zu können. Die Dokumentenhistorie sollte dabei mindestens folgende Informationen umfassen:

- die neue Versionsnummer des Dokuments
- das Datum, an dem die Änderung durchgeführt wurde
- den Bearbeiter, der die Änderung durchgeführt hat
- welche Änderungen durchgeführt wurden
- aus welchem Grund die Änderung vorgenommen wurde

Version	Datum	Name	Durchgeführte Änderung / Grund der Änderung
0.1	19.09.2014	Reber	Initiale Version
0.2	20.09.2014	Reber	Änderungen an Anforderung Req-0010, 0011, 0030, 0090
0.3	30.09.2014	Reber	Änderungen an der Priorität der Anforderungen Req-0010, 0011
1.0	02.10.2014	Reber	Version zum Ersten Review erstellt
1.1	15.10.2015	Reber	Änderungen auf Basis der Review-Ergebnisse eingepflegt an Req-0030, 0034, 0035, 0089, 0090

Tabelle 4: Beispiel für eine Dokumenten-Historie

Versionierung von Anforderungs-Artefakten

Bei der Versionierung von Anforderungen empfiehlt sich ein RM-Werkzeug. Nichtsdestotrotz kann eine Versionierung auch ohne Werkzeug durchgeführt werden.

Bei der Änderung von Anforderungen (sprich, bei der Erstellung einer neuen Anforderungs-Version) sind mindestens folgende Informationen zu dokumentieren, die die Änderung gegenüber der vorherigen Version beschreiben:

- die neue Versionsnummer der Anforderung (ganzzahlig oder als Inkrement)
- die durchgeführte Änderungsaktion im Vergleich zur letzten Baseline (z.B. gelöscht)
- die durchgeführte inhaltliche Änderung an der Anforderung
- den Änderungsgrund (d.h. was bzw. wer war ausschlaggebend für die Änderung)
- den Namen und die Rolle der Person, welche die Änderung durchgeführt hat
- den Zeitpunkt der Änderung (mit Datum + Uhrzeit)

Req-ID	Datum	Version	Name	Grund der Änderung	Aktion	Anforderung
Req-30	19.09.2014	1	Reber		Angelegt	Das System soll a
Req-30	20.09.2014	2	Reber	Änderungen aufgrund neuer Information vom Fachbereich	Geändert	Das System soll b
Req-30	15.10.2014	3	Reber	Änderung aufgrund Review von Max Müller	Geändert	Das System soll c
...						

Tabelle 5: Beispiel für eine Anforderungs-Versionierung

Hinweis: Bei der Verwendung eines RM-Werkzeugs werden einige dieser Informationen (z.B. neue Versionsnummer der Anforderung, die ändernde Person, der Zeitpunkt der Änderung) automatisch dokumentiert, ohne dass Sie dafür zusätzliche Zeit investieren müssen.

Vor der Zeit von Peter Reber wurden Anforderungen noch in Word- und Excel Dokumenten beschrieben. Dennoch wurde auch hier zumindest ein Mindestmaß an Attributierung und Versionierung beachtet. Das unten aufgeführte Beispiel zeigt einen Ausschnitt aus einem alten Anforderungs-Dokument. Die linke Spalte zeigt die Anforderungs-ID zusammen mit der Versionsnummer – sicherlich nicht die beste Art und Weise der Dokumentation – aber besser als nichts. Überarbeitete oder gelöschte Anforderungen wurden entsprechend mit einem neuen Status versehen und gekennzeichnet. Bei Anforderung BR_0040 sieht man zwei Versionen: Zum einen die zurückgewiesene v1 („Revised") und die aktuelle v2 („Modified"). Warum die Anforderung geändert wurde bzw. wer die Änderung veranlasst hat und wann diese durchgeführt wurde, ist hier nicht zu erkennen. Dennoch gibt es hier zumindest ein Minimum an Versionierung, welche zukünftig automatisiert und werkzeugbasiert durchgeführt werden soll, sobald Änderungen am Anforderungs-Artefakt durchgeführt werden.

Req-ID *(Mandatory)* *(e.g. R-001)*	Description of the requirement *(Mandatory)*	MoSCoW *(Mandatory)*	Requirement Owner (Dept. + optional Name) *(Mandatory)*	Benefit *(Optional)*
BR_0010	The user (customer, sales agent, customer care agent, partner, etc.) shall be able to query the calculated bandwidth at a dedicated address.	M	Ralf Muster	
~~BR_0020 descoped~~	~~The average realistic bandwidth for a dedicated address shall be represented in the iLocator~~	~~S~~	~~Ralf Muster~~	
BR_0030	The calculated bandwidth for a dedicated address shall be represented in GeoWorld	M	Ralf Muster	
~~BR_0040 changed~~	~~The bandwidth-check via "Coverage Map " shall provide different views on the available bandwidth~~	~~M~~	~~Ralf Muster~~	
BR_0040 (new)	The bandwidth-check via "Coverage Map" shall represent the calculated outdoor-bandwidth bitrates for each technology, e.g. • GSM as static value "<= 64kbis/s " • EDGE as static value "<=220kbit/s" • 3G (UMTS/HSDPA) bandwidth • 4G (LTE 800 / LTE2600) bandwidth	M	Ralf Muster	

Abbildung 10: Praxisbeispiel einer Anforderungs-Versionierung mit Word

Tipp aus der Praxis: Da RM-Werkzeuge leider noch nicht in allen Unternehmen Einzug erhalten haben, treffen wir im Projektalltag oftmals auf dokumentenbasierte Anforderungs-Spezifikationen (z.B. in Microsoft Word).
Eine Versionierung auf Anforderungsebene in der oben aufgeführten Art und Weise ist hiermit nicht ohne weiteres möglich. Sollten Sie in einer solchen Situation sein, so wenden Sie dieses Vorgehen zumindest auf Dokumentenebene an und nutzen Sie die Möglichkeiten des Überarbeiten-Modus bzw. kennzeichnen Sie Ihre Anforderungsänderungen eindeutig durch Streichungen und Kommentare (siehe Abbildung 10).
Natürlich ist das nicht nach Lehrbuch, aber immerhin wird hier so dargestellt, welche Anforderungen gelöscht und welche geändert wurden. In Dokumenten ist es überdies hilfreich, eine Dokumentenhistorie an den Anfang des Dokuments zu stellen, um dem Leser einen schnellen Überblick über die Historie des Dokuments und den darin enthaltenen Änderungen zu verschaffen (siehe Tabelle 4).

Zugriff auf aktuelle Versionen

Stellen Sie sicher, dass alle relevanten Projektteilnehmer auf die aktuell gültigen (freigegebenen) Anforderungs-Versionen zugreifen können – dies ist nicht zwingend gleichbedeutend mit dem aktuellsten Dokumentationsstand, der sich beispielsweise noch in Überarbeitung befindet und nicht freigegeben wurde. Die Bearbeiter der Anforderungen und der Requirements Manager müssen allerdings zu jedem Zeitpunkt in der Lage sein, auf die aktuellste Version einer Anforderung zugreifen zu können.

Wichtig ist darüber hinaus, dass Änderungen, die zu neuen Anforderungs-Versionen geführt haben, zu definierten Zeitpunkten aktiv an alle Stakeholder kommuniziert werden

[WiBe2013]. In der Regel wird dies zu Review-Terminen stattfinden, bei denen sog. Anforderungs-Konfigurationen (siehe Kapitel 5.1.2) zusammengestellt werden.

Hinweis: Die aktive Kommunikation an die Projektbeteiligten kann auch durch ein RM-Werkzeug übernommen werden, falls bspw. im Vorfeld definiert wurde, wer bei einer Änderung in Kenntnis zu setzen ist.

Umsetzung von Maßnahmen für eine Versionskontrolle

Als Requirements Manager müssen Sie bereits bei der Erstellung des RMP – also schon vor der Dokumentation der ersten Anforderung – festlegen, wie Sie in Ihrem Projekt eine Versionskontrolle für Anforderungen und Anforderungs-Dokumente umsetzen wollen.

Die Entscheidung, auf welcher Ebene – Dokumentenebene oder Anforderungsebene – eine Versionskontrolle erfolgen soll, obliegt dem Requirements Manager und ist vom Projektumfang abhängig. Tendenziell sollte bei komplexen Projekten mit hunderten von Anforderungen eine Versionskontrolle auf Anforderungsebene erfolgen, auch wenn der Aufwand hierfür deutlich höher sein wird. Er wird sich allerdings im Laufe des Projektes auszahlen. Über die Versionierung auf Anforderungsebene können Sie sicherstellen, dass Sie im gesamten Projektverlauf immer über dieselbe Anforderungs-Version sprechen. Sprich, Sie wissen, welche Anforderungs-Version in welcher Anforderungs-Konfiguration (z.B. für die Abnahme oder die Entwicklung) enthalten war und können somit explizit über bestimmte Anforderungs-Versionen sprechen bzw. diese verteilen.

Neben den Festlegungen, auf welcher Ebene eine Versionierung betrieben wird, und welche Informationen für neu angelegte Versionen zu dokumentieren sind, muss im RMP festgelegt werden, wer Änderungen auf welcher Ebene vornehmen darf.

Grundsätzlich sollte nur ein eingeschränkter Personenkreis berechtigt sein, Änderungen durchzuführen (vgl. [WiBe2013]). Diese Rollen und Rechte sind in einer Rollen- und Rechtematrix (z.B. RACI) zu dokumentieren (siehe auch [Oran2013], RACI Model).

Abgrenzung zum Änderungsmanagement

Bisher haben wir über die Umsetzung einer Versionskontrolle auf unterschiedlichen Ebenen gesprochen, um etwaige Veränderungen z.B. auf Anforderungsebene zu dokumentieren und nachvollziehen zu können. Die Gründe für solche Änderungen sind mannigfaltig.

Im Rahmen eines Projektes wollen wir hierzu zwei wesentliche Zeitpunkte unterscheiden:

- Änderungen, die im Rahmen des Requirements Engineering bis zu ihrer finalen Abnahme bzw. Freigabe der Anforderungs-Spezifikation auftreten
- Änderungen, die nach der finalen Abnahme bzw. Freigabe der Anforderungs-Spezifikation auftreten und somit eine nachträgliche Scope-Veränderung bedeuten

Änderungen der ersten Art können in der Regel direkt berücksichtigt werden und fließen als neue Version einer Anforderung in die Spezifikation ein, falls sie keine grundsätzliche Veränderung des Projektumfangs bedeuten. Änderungen der zweiten Art müssen immer über einen geregelten „Änderungsmanagement-Prozess" abgewickelt werden.

5.1.2 Anforderungs-Konfigurationen

Im Rahmen der Anforderungserhebung erstellen Sie zu bestimmten Zeitpunkten Anforderungs-Konfigurationen, um beispielsweise einen Review auf einen festgelegten und konsistenten Stand Ihrer Anforderungen durchführen zu lassen oder um hierfür eine Aufwandsabschätzung für die nachfolgenden Entwicklungsphasen zu erhalten.

Definition 5-4:
Anforderungs-Konfiguration{ XE "Anforderungskonfiguration" } nach [IREB2015]: „Eine Anforderungs-Konfiguration fasst eine definierte Menge logisch zusammengehöriger Anforderungen zusammen, wobei jede Anforderung maximal in einer Version in der Anforderungs-Konfiguration enthalten ist.“

Eine Anforderungs-Konfiguration ist also ein konkretes Set an Anforderungen (Anforderungs-Artefakten), das beispielsweise zu einem bestimmten Zeitpunkt zum Review bereitgestellt wird und eine bestimmte Version der Anforderungen beinhaltet. Die folgende Definition stellt darüber hinaus den Kommunikationsaspekt des Konfigurationsmanagements heraus, damit alle beteiligten Reviewer bzw. Nutzer der Version einen einheitlichen und konsistenten und zusammenhängenden Anforderungsstand erhalten.

Definition 5-5:
Konfigurationsmanagement{ XE "Konfigurationsmanagement" } aus ISO29148 „The purpose of the Configuration Management Process is to establish and maintain the integrity of all identified outputs of a project or process and make them available to concerned parties.“

Nach [PoRu2015] besitzen Anforderungs-Konfigurationen folgende Eigenschaften:

- **Sachlogischer Zusammenhang**: Die ausgewählten Anforderungs-Versionen einer Konfiguration hängen logisch zusammen und sind in ihrer Auswahl zweckgerichtet.
- **Konsistenz**: Die zusammengefassten Anforderungen und Anforderungsdokumente sind widerspruchsfrei und gehören sachlogisch zusammen.[1]

[1] In der Praxis werden oft auch Konfigurationen gebildet, die nicht inhaltlich konsistent sind. Diese Konfigurationen werden aus der Notwendigkeit heraus gebildet, den aktuellen Arbeitsstand einzufrieren, um später gegebenenfalls zurückgreifen zu können. Beispielsweise kann eine Konfiguration erzeugt werden, die den Startpunkt von Review-Aktivitäten dokumentiert.

- **Eindeutigkeit**: Die Konfiguration für die ausgewählten Anforderungs-Versionen verfügt über einen Bezeichner, durch den sie eindeutig identifiziert werden kann.
- **Unveränderlichkeit**: Die Konfiguration basiert auf einem bestimmten Versionsstand der Anforderungen. Änderungen an diesen Anforderungs-Versionen führen zu neuen Versionen, die in neuen Konfigurationen verwendet werden können.
- **Grundlage für Rücksetzen**: Konfigurationen bieten definierte Stände, um Anforderungen auf einen älteren konsistenten Anforderungsstand (Versionsstand) zurückzusetzen.

Hinweis Die Erstellung einer Anforderungs-Konfiguration kann beispielsweise als ein in sich konsistentes Anforderungs-Dokument mit ausgewählten Versionen von Anforderungs-Artefakten für eine geplante Projektphase betrachtet werden, welches geprüft und abgenommen werden soll. Hierbei müssen im Vergleich zur Anforderungs-Basislinie (siehe folgender Abschnitt) nicht zwingend nur stabile Anforderungs-Artefakte enthalten sein. Hierbei geht es eher um den sachlogischen Zusammenhang einer „Anforderungskomposition" (also Anforderungs-Konfiguration).

5.1.3 Die Anforderungs-Basislinie

Der Begriff Anforderungs-Basislinie (Anforderungs-Baseline) sollte Ihnen bereits aus dem CPRE Foundation Level bekannt sein. Baselines stellen in der Regel einen „eingefrorenen" Dokumentationsstand dar, der beispielsweise zur Erreichung bestimmter Meilensteine (Übergabe einer Spezifikation zur Kostenschätzung) erstellt wird.

Definition 5-6:

Anforderungs-Basislinie: Anforderungs-Basislinien sind ausgewählte, formal geprüfte und freigegebene Anforderungs-Konfigurationen, die stabile Anforderungs-Artefakte umfassen und oftmals einen fest definierten Entwicklungs- und Auslieferungsstand für ein Produkt widerspiegeln (z.B. für ein bestimmtes Produktrelease).

Anforderungs-Basislinien sind also in der Regel nach außen hin sichtbar, wohingegen einfache Anforderungs-Konfigurationen eher zu internen Zwecken genutzt werden (vgl. [WiBe2013] und [Pohl2010]).

Definition 5-7:

Release-Management{ XE "Release-Management" } nach [BSB2008]: „Das Release-Management beschäftigt sich mit der Bündelung von Anforderungen für ein Produkt, der Zeitplanung für die Herstellung und letztendlich für die Auslieferung eines fertigen Systems. […] Für ein Release werden üblicherweise alle aktuellen Konfigurationselemente unter einem gemeinsamen Bezeichner geführt („label“) und aus der so erzeugten Konfiguration dann die Software erstellt.“

Die als Anforderungs-Basislinie festgelegte Anforderungs-Konfiguration sollte nur Anforderungen enthalten, welche für eine bestimmte Version des Produkts (z.B. Release) geplant und stabil sind und nicht für solche, die zu diesem Zeitpunkt erst vorgeschlagen wurden oder noch in Bearbeitung bzw. Abstimmung sind [WiBe2013]. Achten Sie bei der Auswahl von Anforderungs-Versionen für eine Anforderungs-Basislinie also auf deren Zustand bzw. Status (siehe auch Kapitel 3).

Anforderungs-Basislinien unterstützen drei wesentliche Tätigkeiten im Entwicklungsprozess (vgl. [Pohl2010]):

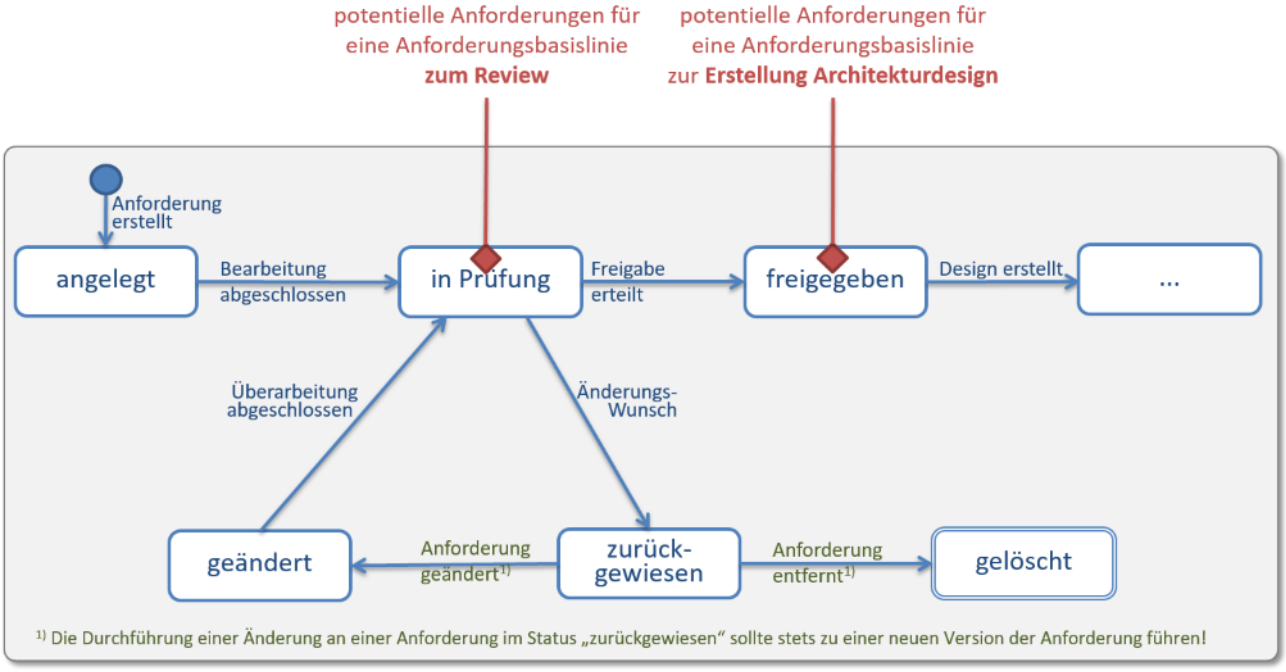
- Sie bilden die **Grundlage für das Planen von Auslieferungsstufen (Releases)**, da sie eine für den Kunden sichtbare Konfiguration von stabilen Anforderungs-Versionen darstellen.
- Sie dienen der **Abschätzung des Realisierungsaufwands** eines bestimmten Auslieferungsstands.
- Sie ermöglichen den **Vergleich mit Konkurrenzprodukten** am Markt mit dem festgelegten Auslieferungsstand.

Ein geeigneter Zeitpunkt zum Erstellen einer Anforderungs-Basislinie kann das Erreichen eines Meilensteins, z.B. die Beauftragung des Architekturdesigns bzw. der Implementierung, sein (vgl. [WiBe2013]). Meilensteine für Anforderungs-Basislinien aber auch für Konfigurationen sind in der Regel durch das Projekt oder den Entwicklungsprozess vorgegeben.

Abbildung 11 skizziert hierzu zwei beispielhafte Meilensteine *„zum Review“* und *„Erstellung Architekturdesign“*. Für den ersten Meilenstein können auch Versionen von Anforderungs-Artefakten verwendet werden, die im Zustand „in Prüfung“ sind. Für den zweiten Meilenstein sollen hingegen nur Versionen von Anforderungs-Artefakten verwendet werden, die im Zustand „freigegeben“ sind.

Hinweis: Definieren Sie in Ihrem RMP, zu welchem Zweck Anforderungs-Basislinien erstellt werden sollen, wer Anforderungs-Basislinien erstellen darf und nicht zuletzt, nach welchen Kriterien die Auswahl von Anforderungs-Artefakten für eine Anforderungs-Basislinie zu erfolgen hat.

Betrachten Sie die in der Anforderungs-Basislinie enthaltenen Anforderungs-Artefakte als abgenommene und beauftragte Spezifikation. Die hierin enthaltenen Anforderungen können nur noch über ein geordnetes Änderungsmanagement angepasst werden.



Zustände und Zustandsübergänge eines Anforderungsartefakt im Rahmen der Anforderungserhebung

Abbildung 11: Mögliche Meilensteine für Anforderungs-Konfigurationen und Anforderungs-Basislinien

5.1.4 Verzweigung von Anforderungen

Der Begriff „*Branching*{ XE "Branching" }“ (von englisch *branch* = Zweig) stammt aus dem Konfigurationsmanagement und erlaubt die parallele Entwicklung von Systemen in unterschiedlichen Entwicklungszweigen. Zweige werden beispielsweise im Rahmen von fest geplanten Releases genutzt, um auf einem Zweig mit der Weiterentwicklung für das folgende Release zu starten, während auf dem parallelen „Produktions-Zweig“ des bereits ausgelieferten Systems nur Bug-Fixing bzw. Minimaländerungen durchgeführt werden. Die beiden Entwicklungs-Zweige werden dann in der Regel vor dem nächsten Hauptrelease wieder zusammengeführt (englisch „gemerged“), damit beispielsweise die bereits behobenen Fehler im Produktions-Zweig mit dem neuen Release nicht wieder in Produktion genommen werden.

Auch wenn der Begriff des „*Branching*" im Ursprung aus dem Konfigurationsmanagement stammt und eher umsetzungslastig ist, können wir dieses Konzept ebenfalls im RM wiederfinden, da die Verzweigungen in den Anforderungssträngen die Verzweigungen in den Entwicklungssträngen widerspiegeln.

Anders als die Versionierung von Anforderungen, erlaubt die Verzweigung von Anforderungen die parallele Gültigkeit mehrerer Anforderungs-Versionen zum selben Zeitpunkt. Der Mechanismus der Verzweigung wird genutzt, um beispielsweise kleine und dringende Änderungen parallel zu einer laufenden Entwicklung durchführen zu können.

Um einen Anforderungs-Zweig{ XE "Anforderungs-Zweig" } zu erstellen, wird eine gültige Anforderungs-Konfiguration ausgewählt (z.B. die letzte Anforderungs-Basislinie), auf welcher der neue Anforderungs-Zweig aufbauen soll. Diese Konfiguration wird in einen neuen Anforderungs-Zweig kopiert, geändert, versioniert und zu einer neuen Anforderungs-Konfiguration zusammengefasst. Stellen Sie sich einen Anforderungs-Zweig beispielsweise als Kopie einer ausgewählten Dokumentenversion vor, an der parallel gearbeitet werden kann. Wesentlich ist hier, dass dieselbe Anforderung in beiden Zweigen parallel existieren darf – es also je Zweig eine gültige Version eines Anforderungs-Artefaktes gibt.

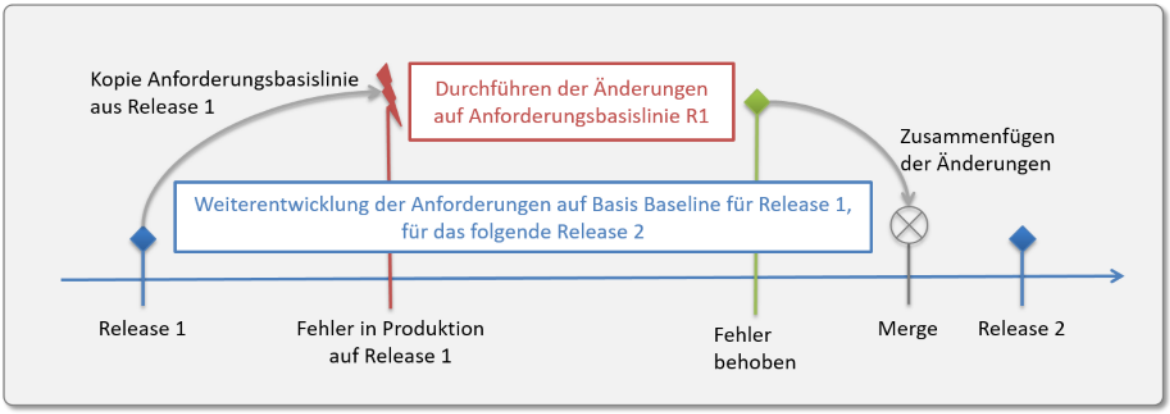


Abbildung 12: Branching und Merging von Anforderungs-Konfigurationen

Nachdem die abgezweigte Anforderungs-Konfiguration erfolgreich umgesetzt wurde, werden die beiden Zweige zu einem späteren Zeitpunkt – in der Regel vor einem neuen Release – wieder zusammengeführt (Merge). Ab diesem Zeitpunkt gibt es wieder nur noch eine Version der jeweiligen Anforderungs-Artefakte, damit die folgenden Änderungen nur noch an einer Version durchgeführt werden müssen, siehe Beispiel in Abbildung 12.

Die Anforderungs-Verzweigung stellt neben der Verfeinerung und Versionierung von Anforderungen eine zusätzliche Dimension der Komplexität im Umgang mit Anforderungen im RM dar, welche sparsam und bewusst verwendet werden sollte. Sonst kann die unkontrollierte Anwendung von Anforderungs-Verzweigungen zu mehr Chaos als Nutzen führen.

Probleme, die in Verbindung mit Anforderungs-Verzweigungen auftreten, sind u.a., dass:

- Anforderungs-Zweige die eindeutige Identifizierung von Anforderungen erschweren.

- Anforderungs-Zweige neben Versionen und Verfeinerungen zusätzlich die Komplexität des RE und RM erhöhen.
- Anforderungs-Zweige redundante Anforderungsinformationen erzeugen, welche parallel gepflegt und langfristig wieder zusammengeführt werden müssen (Merge).

In Einzelfällen kann es durchaus vorkommen, dass in Anforderungs-Zweigen entstandene Anforderungs-Versionen bewusst nicht mehr zusammengeführt werden und die beiden Anforderungs-Versionen fortlaufend parallel existieren sollen. An dieser Stelle spricht man dann allerdings nicht mehr von derselben Version in unterschiedlichen Anforderungs-Zweigen, sondern von Varianten (vgl. Kapitel 7).

Diese Varianten werden dann mit unterschiedlichen Anforderungs-IDs weitergeführt, damit die Anforderungsvarianten weiterhin eindeutig identifiziert werden können.

Tipp aus der Praxis: Nicht nur im RE und RM wird die Komplexität der Anforderungsverwaltung durch Anforderungs-Zweige erhöht, auch in den folgenden Phasen der Softwareentwicklung führen Parallelentwicklungen zu Herausforderungen, da für jeden Entwicklungszweig eigene Entwicklungs- und Testumgebungen sowie Mannschaften vorhanden sein müssen. Kommt es dann durch Softwarefehler zu Verzögerungen bei der Inbetriebnahme von Verzweigungen (Branches), kann sich dies sogar auf die Abnahme und Inbetriebnahme von Folgereleases auswirken. Man sollte also die Anzahl der Anforderungs-Zweige klein halten. Grundsätzlich findet man in Unternehmen oftmals einen parallelen Entwicklungs-Zweig vor, der fürs Bugfixing und kleine Änderungen verwendet wird.

5.2 Änderungsmanagement für Anforderungen

IEEE 29148 beschreibt die Natur der Änderungen von Anforderungen mit folgenden Worten: „Whatever the cause of requirements changes, it is important to recognize the inevitability of change and adopt measures to mitigate the effects of change. *Change has to be managed by ensuring that proposed changes go through a defined impact assessment, review, and approval process, and by applying careful requirements tracing and version management. Hence, the requirements engineering process is not merely a front-end task, but spans the life cycle.* In a typical project the activities of the requirements management evolve over time from elicitation to change management." ISO29148

Hinweis: Seien Sie auf Änderungen gefasst und planen Sie diese mit ein. Etablieren Sie einen einfachen und effektiven Änderungsprozess. Je länger ein Projekt läuft, desto höher ist die Wahrscheinlichkeit von Änderungen an Ihren Anforderungen. Ein grober Richtwert sind 1-5 % Änderungen pro Monat (siehe auch [Eber2012], [WiBe2013]).

Der geplante Umgang mit Änderungen also ist eine wesentliche Aufgabe im RM. Wichtig ist hierbei zu akzeptieren, dass Änderungen die Regel sind und keine Ausnahme darstellen.

Bei der Änderung von Anforderungen wollen wir zwei wesentliche Änderungszeitpunkte voneinander unterscheiden, da diese in der Regel auch unterschiedlich behandelt werden.

- Erstens die Evolution der Anforderung bis zu ihrer finalen Abnahme bzw. Freigabe für das Architekturdesign oder die Implementierung. Dies ist in der Regel ein Zeitintervall vor der ersten Anforderungs-Basislinie. Im Rahmen dieser Phase der Anforderungserhebung, -Analyse, -Verhandlung, ist es normal, dass Änderungen an Anforderungen ohne einen separaten Änderungsmanagement-Prozess durchgeführt werden. Beachten Sie hierfür die Regeln der Versionierung.
- Zweitens die Evolution der Anforderung nach der finalen Abnahme bzw. im Rahmen der Designerstellung, Implementierung oder gar im Betrieb. Auch diese Änderungen sind normal und können von außen (z.B. Gesetzesänderungen) oder von innen (z.B. neue Strategien) getrieben sein. Diese Änderungen sollten Sie über einen entsprechenden Änderungsmanagement-Prozess bearbeiten! Denn diese Änderungen sind in der Regel jene, die im Vorfeld weder vom zeitlichen noch monetären Aufwand her abgeschätzt wurden und über eine Auswirkungsanalyse neu bewertet werden müssen.

Definition 5-8:

Änderungsmanagement{ XE "Änderungsmanagement" } nach [BSB2008]:

„Das Änderungsmanagement regelt die Fortentwicklung des Produkts, indem es vor allem die Änderungswünsche an das Produkt sowie deren Abwicklung überwacht. Durch das Änderungsmanagement wird der Lebenszyklus aller Änderungswünsche über die Schritte Entstehen, Bewerten, Realisieren, Testen und Abnehmen hinweg überwacht.“

5.2.1 Ursachen, Quellen und Zeitpunkte von Anforderungsänderungen

Gründe für Anforderungsänderungen sind mannigfaltig. Anforderungen an ein (Software-) System unterliegen im Rahmen seines Lebenszyklus Änderungen. Diese Änderungen können von unterschiedlichen Personen bzw. Rollen, aus unterschiedlichen Entwicklungsphasen sowie zu unterschiedlichen Projekt- und Lebenszyklusphasen aufgebracht werden.

In einem ersten Schritt ist es hilfreich zu wissen, woher Änderungen an Anforderungen kommen und was die Ursachen und Quellen von Anforderungsänderungen sind. [RuSo2009] unterscheiden folgende **Quellen** für Änderungen:

- **Incident Management** (technische Hotline des Systems): Hier laufen Störungen aus dem technischen Systembetrieb sowie des Systemnutzers auf, die behoben werden müssen. Im Rahmen der Analyse können sich hieraus Änderungen ergeben.
- **Fachbereich und Produkt-Management**: Durch diese Personengruppen werden in aller Regel neue Anforderungen für das System geboren, die die Benutzung verbessern oder neue Facetten abbilden.

- **Entwickler**: Hier werden in der Regel Änderungswünsche zur technischen Umsetzung definiert. Diese Änderungen haben keinen direkten Einfluss auf die Nutzerfunktionalität eines Systems.
- **Tester**: Hier werden in der Regel Änderungen definiert, um im System existierende Fehler (durch fehlerhafte oder lückenhafte Anforderungen) zu beheben.

Ursachen für Anforderungsänderungen nach [Pohl2010] sind bspw.:

- **Fehler im laufenden Systembetrieb**: Änderungen aufgrund von fehlerhaften Systemverhalten, die durch den Nutzer oder Applikations-Betrieb als Incident gemeldet werden. Diese Anforderungsänderungen resultieren aus fehlerhaften oder fehlenden Anforderungen und nicht auf einer falschen Umsetzung der Anforderung.
- **Kontextänderungen**: Änderungen, die sich aufgrund sich veränderter Randbedingungen im Systemkontext ergeben. Diese Änderungen können aus allen Kontextfacetten stammen (Nutzungsfacette, Gegenstandsfacette, IT-Systemfacette oder der Entwicklungsfacette). Diese Anforderungsänderungen resultieren aus einer sich verändernden Welt und werden über den Fachbereich, das Produktmarketing oder die Entwicklung eingereicht.

Die aufgeführten Ursachen haben keinen Anspruch auf Vollständigkeit. Sie sollen Ihnen vor allem einen Einblick verschaffen, warum Änderungen passieren und aus welchen Gründen und Quellen Änderungen stammen können. Machen Sie sich also möglichst frühzeitig Gedanken darüber, welche Quellen und Gründe von Änderungen Sie erwarten.

Hinweis: Im Rahmen der Analyse einer durchzuführenden Änderung müssen neben der Änderung an der eigentlichen Anforderung ebenfalls Auswirkungen auf direkt und indirekt abhängige Anforderungen und weitere Entwicklungs-Artefakte berücksichtigt werden (vgl. Kapitel 6). Natürlich sind Sie als Requirements Manager nicht direkt verantwortlich für das Architekturdesign, den Test und die Entwicklung, aber wenn die Änderung in „Ihren" Anforderungen stattfindet, sind die weiteren Rollen über diese Änderungen zu informieren, sodass eine ganzheitliche Bewertung der Änderung stattfinden kann.

Änderungen können zu unterschiedlichen **Zeitpunkten** während des gesamten Projekts und Lebenszyklus eines (Software-)Systems auftreten, z.B.:

- **Im Rahmen der Anforderungserhebung**: Das Einfügen einer neuen Anforderung führt zu einer Anpassung einer bestehenden Anforderung, da sich der Systemkontext geändert hat.
- **Im Rahmen des Architekturdesigns**: Eine Architekturentscheidung für die System-architektur bedingt, dass eine zuvor durch Hardware abgedeckte Funktion aus Kostengründen durch zusätzliche Funktionalität in der Software realisiert werden soll.
- **Im Rahmen der Implementierung:** Die Implementierung einer Anforderung zeigt Performanz-Probleme, die wiederum nur durch eine Anpassung der eigentlichen Anforderung gelöst werden können.
- **Im Rahmen des Softwaretests**: Ein Testergebnis zeigt, dass eine Anforderung nicht entsprechend der Spezifikation umgesetzt wurde, die Implementierung aber eine

bessere Lösung bietet, welche beibehalten werden soll. Hier ist die Anforderung entsprechend zu aktualisieren.

- **Im Rahmen des Abnahmetests**: Bei der Abnahme stellt sich heraus, dass der Kunde die Lieferung nicht abnimmt, da er sich die Umsetzung der Anforderungen anders vorgestellt hatte, diese aber nur unzureichend dokumentiert hat. Die Anforderungen müssen detailliert und die entsprechenden Entwicklungs-Artefakte überarbeitet werden.
- **Im Betrieb des Systems**: Bei der Nutzung einer Software kann sich herausstellen, dass umgesetzte Funktionalitäten Lücken für die Bearbeitung des Geschäftsprozesses aufweisen und somit neue Anforderungen erhoben und bestehende Anforderungen geändert werden müssen.

Die Dimensionen der Anforderungsänderungen – unterschiedliche Quellen, unterschiedliche Ursachen und unterschiedliche Zeitpunkte – machen das Änderungsmanagement zu einer komplexen Aufgabe, die nicht ad-hoc und durch Zuruf funktionieren kann, sondern einen dedizierten Änderungsmanagement-Prozess benötigt (siehe Abschnitt 5.3).

5.2.2 Arten von Änderungen an Anforderungen

Wie Sie sich vorstellen können und sicherlich schon am eigenen Leibe erfahren durften, ist eine Änderung nicht gleich einer Änderung. Wenn Sie Ihr Fahrzeug mit einem Automatikgetriebe bestellt haben, Sie im Rahmen der Auslieferung aber feststellen, dass das Fahrzeug ein Schaltgetriebe hat, ergibt sich hieraus eine Reklamation (anders ausgedrückt: ein Änderungswunsch).

Diese Art der Änderung hat allerdings für Sie – wie für den Fahrzeuglieferanten – einen anderen Stellenwert als die Änderung, die Sie 6 Wochen nach Ihrer Bestellung aufgebracht haben, dass Sie anstelle einer Außenlackierung in „Silbergraumetallic" lieber das „Spacegraumetallic" wollten.

Für das Änderungsmanagement sollte man im Umgang mit Änderungen wissen, welche Arten von Änderungen existieren, um eine Strategie zu entwickeln, wie mit unterschiedlichen Arten von Änderungen umgegangen werden soll. Im CPRE Foundation Level [IREB2015] wird folgende Klassifikation von Änderungen vorgeschlagen:

- **Korrektive Änderungen**: Eine Änderung ist korrektiv, falls sie auf ein Fehlverhalten im Betrieb bzw. beim ausgelieferten Produkt zurückzuführen ist, dessen Ursache auf Fehlern in den Anforderungen beruht.
- **Adaptive Änderungen**: Eine Änderung ist adaptiv, falls sie auf neue Randbedingungen, Erkenntnisse bzw. Kontextveränderung zurückzuführen ist. Diese Art der Änderung kommt in der Regel von außerhalb des Projektes (z.B. der Gesetzgebung).
- **Ausnahmeänderungen**: Eine Änderung ist eine Ausnahmeänderung, falls sie auf ein schadhaftes Verhalten zurückzuführen ist bzw. zu einem schadhaften Verhalten führen würde. Diese Änderung ist schnellstmöglich umzusetzen, um den Schaden zu begrenzen. Diese Art der Änderung kann sowohl korrektiv als auch adaptiver Natur sein.

Für die Umsetzung einer Anforderungsänderung müssen Änderungsanträge gestellt werden die durch ein Änderungsmanagement bewertet und bearbeitet werden (siehe Kapitel 5.2). Ein Änderungsantrag umfasst unter anderem die gewünschten inhaltlichen Änderungen an bestehenden Anforderungen (sprich, an der aktuellen Anforderungs-Basislinie).

Hinweis: Nicht jede Änderung führt zu einer Anpassung der Anforderungen. So führen bspw. Softwarefehler nicht zur Änderung der eigentlichen Anforderungen, sondern ausschließlich zur Änderung bzw. Korrektur der Implementierung in Bezug auf die (korrekten) Anforderungen. Das eingangs gewählte Beispiel mit der falschen Getriebeart lässt sich aus Kundensicht eindeutig als „Bug" einstufen, da die Anforderung des Kunden („Automatikgetriebe") klar festgehalten war.

Diese Änderungen können folgendermaßen charakterisiert werden:

- Die Änderung erfordert die **Integration einer neuen** Anforderung (in der Regel eine Scope-Erweiterung).
- Die Änderung erfordert das **Löschen einer bestehenden** Anforderung (in der Regel eine Scope-Reduktion).
- Die Änderung erfordert das **Ändern einer bestehenden** Anforderung, durch Ergänzung, Reduktion oder inhaltliche Veränderung (Scope-Änderung).

Beachten Sie für Ihren Änderungsmanagement-Prozess (vgl. Kapitel 5.2), dass es unterschiedliche Arten von Änderungen gibt, die Änderungen in „Ihren" Anforderungen erfordern. Legen Sie fest, von welchen Änderungsanfragen Sie in Ihrem Projektterminus sprechen wollen, von wem diese Änderungsanfragen kommen dürfen und wie die jeweilige Anfrage auszusehen hat. So kann beispielsweise ein Änderungsantrag zur Behebung eines Defect als korrektive Änderung anders aussehen als ein Änderungsauftrag für eine Innovation (also eine adaptive Änderung).

Für sein Projekt legt Peter Reber fest, von welchen Arten von Änderungen er im Projekt sprechen möchte, was er darunter versteht und wer diese Änderungsanfragen in welcher Form stellen darf. Hierbei nutzt Peter die oben genannte Klassifizierung, verwendet allerdings eigene Bezeichnungen, die sich im Unternehmen etabliert haben.

Spec Fehler: *Ein „Spec Fehler" gehört in die Klasse der korrektiven Änderungen und beschreibt einen Fehler im Produkt, der auf eine fehlerhafte Beschreibung in der Anforderungs-Spezifikation zurückzuführen ist.* Diese Änderungen sind ausschließlich *über den IT-Service Desk* zu kanalisieren und werden *über das Ticket-System als Änderungsantrag* gemeldet.

Scope Change: *Ein „Scope Change" gehört zur Klasse der adaptiven Änderungen und beschreibt neue Anforderungen an das System, aus Benutzer-, Unternehmens- oder gesetzlicher Sicht.* Diese Änderungen kommen in der Regel *über das Produktmarketing* und sind über das *Template für Change Requests* zu dokumentieren.

Tuning-Anfrage: Eine *„Tuning-Anfrage" gehört zur Klasse der adaptiven Änderungen und beschreibt neue technologische Anforderungen an das System zur Verbesserung der Betreibbarkeit*. Diese Änderungen werden in der Regel *über den IT-Betrieb* gestellt und sind über das *Template für IT-Change Requests* zu dokumentieren.

5.2.3 Analyse und Dokumentation der Stabilität von Anforderungen

Für das Vorankommen im Projekt müssen Sie Ihre Anforderungen in einem vorgegebenen Zeitrahmen finalisieren, um Ihr Projekt *in Time*, *in Cost* und *in Quality* abschließen können. Damit der Auftraggeber möglichst schnell Erfolge sieht und von seiner Investition profitieren kann, wird oftmals ein phasen- bzw. releasebasierter Ansatz gewählt, bei dem das gewünschte Produkt schrittweise in Produktion genommen wird. Hierzu ist es allerdings notwendig, dass Sie wissen, welche Anforderungen bereits abgestimmt und stabil sind, damit Sie diese in die Entwicklung übergeben können (vgl. Abbildung 11 in Kapitel 5.1.3).

Für diese Auswahl sollten Anforderungen nach ihrer Stabilität und damit hinsichtlich der Wahrscheinlichkeit, ob die aktuelle Version der Anforderung noch verändert wird, klassifiziert werden. Eine solche Klassifikation zur Auswahl von Anforderungen für eine bestimmte Phase kann einerseits allein durch die Bewertung der Stabilität erfolgen (siehe Kapitel 3) oder aber durch eine entsprechende Priorisierungstechnik (siehe Kapitel 4), welche neben der Stabilität noch weitere Aspekte, z.B. erwarteter Nutzen aus der Anforderung, mit in die Bewertung einbezieht. Die Berücksichtigung der Anforderungsstabilität sollte bei der Auswahl für ein Ziel-Release auf jeden Fall immer in die Bewertung einfließen, da sie unter anderem für die Einschätzung des Risikos relevant ist, eine ausgewählte Anforderungs-Konfiguration (Anforderungs-Basislinie) zur Umsetzung freizugeben.

Tipp aus der Praxis: Wie bereits eingangs erwähnt, kann nach der Phase der Anforderungserhebung und Dokumentation mit einer Änderungsquote von 1-5 % pro Monat gerechnet werden. Sprich: Wenn Sie 1.000 Projektanforderungen haben, ist es nicht unüblich, dass sich 20 Anforderungen pro Monat ändern.
Sollten sich in Ihrem Projekt, nach der Freigabe der Anforderungen, mehr als 10 % der Anforderungen pro Monat ändern, sollten Sie sich gemeinsam mit dem Auftraggeber ernsthafte Gedanken über das eigentliche Projektziel machen, um eine schleichende Scope-Erweiterung zu unterbinden.

An dieser Stelle stellen Sie sich zu Recht die Frage: Wie wähle ich eine solche Anforderungs-Konfiguration aus bzw. wie kann ich feststellen, dass ein Set von Anforderungen (eine Anforderungs-Konfiguration) so weit ist, um eine Anforderungs-Basislinie zu erstellen und in die Entwicklung zu übergeben, ohne dass kurze Zeit später die ersten Änderungen an diesen Anforderungen aufgebracht werden? Die Antwort ist: Dies kann keiner genau vorhersagen.

Aber auch wenn Sie kein Hellseher sind, können Sie über die Änderungswahrscheinlichkeit Ihrer Anforderungen eine Aussage treffen. Die folgenden Regeln (Heuristiken) helfen Ihnen

mit begrenztem Wissen und unvollständigen Informationen, in kurzer Zeit zu einer Bewertung über die Wahrscheinlichkeit der Änderung von Anforderungsgruppen zu gelangen (vgl. [VanL2009]).

- Anforderungsgruppen, die demselben Ziel dienen und grundsätzlich eine hohe Stabilität (gemessen an der Änderungshäufigkeit) aufweisen, haben eine geringere Änderungswahrscheinlichkeit als einzelne Anforderungen.
- Ziele sind in der Regel stabiler als lösungsorientierte Anforderungen.
- Funktionale Anforderungen, welche die Kernziele erfüllen, sind in der Regel stabiler als Qualitätsanforderungen.
- Funktionale Anforderungen, die in der Menge der Anforderungen wiederkehrend (als Verschmelzung, Erweiterung oder Variante) auftauchen, sind in der Regel als stabile Anforderungen zu betrachten.
- Anforderungen, die alternative Auswahlmöglichkeiten beschreiben, sind unter besonderer Vorsicht zu betrachten und in der Regel weniger stabil als oben genannte, da Entscheidungen oft auf unvollständigem Wissen und Annahmen beruhen.
- Anforderungen, welche zwingend einer Variante oder Erweiterung des Systems zugeordnet sind, sind stabiler als noch nicht zugeordnete Anforderungen.
- Anforderungen, die bis vor Kurzem häufig geändert wurden, sind voraussichtlich noch nicht als stabile Anforderung anzusehen.

Legen Sie also für sich und Ihr Team fest, nach welchen Kriterien Anforderungs-Basislinien erstellt werden: Das heißt, welche Anforderungen in eine Anforderungs-Basislinie einfließen dürfen (im Sinne von festgelegten Bewertungskriterien). Sorgen Sie bereits frühzeitig dafür, dass entsprechende Attribute zur Dokumentation des Anforderungs-Zustandes, der Stabilität, der Dringlichkeit, etc. angelegt (vgl. Kapitel 3) und vor allem gepflegt werden, damit Sie zu jeder Zeit in der Lage sind, die richtigen Anforderungen für stabile Anforderungs-Basislinien auszuwählen.

5.3 Änderungsmanagement-Prozess

Nach ITIL sorgt das Änderungsmanagement{ XE "Änderungsmanagement" } (Change Management{ XE "Änderungsmanagement" }) für eine kontrollierte Durchführung von Änderungen innerhalb der IT-Infrastruktur. Ziel des Änderungsmanagement-Prozesses ist es, durch eine gute Abstimmung innerhalb der Organisation festzulegen, welche Änderungen erforderlich sind und wie diese mit möglichst geringen Folgen für die IT Services umzusetzen sind (Übersetzung in Anlehnung an [Oran2013]).

Dies erreicht der Änderungsmanagementprozess durch die Definition von Aktivitäten, Verantwortlichkeiten und notwendigen Artefakten, die eine klare Vorgehensweise für die Abwicklung von Änderungswünschen an Anforderungen beschreibt.

In den meisten Änderungsmanagement-Prozessen spielt das Change Control Board{ XE "Change Control Board" } CCB eine wichtige Rolle im Änderungsprozess. In ITIL heißt es auch CAB = Change Advisory Board{ XE "Change Advisory Board" }. [WiBe2013] beschreibt das Change Control Board (Änderungskomitee) als{ XE "Änderungskomitee" } eine Gruppe von Personen mit unterschiedlichen Interessen (z.B. Projektmanager, Entwickler, Tester, IT-

Betrieb, Helpdesk), die für jeden Änderungsantrag entscheidet, ob und wann dieser umgesetzt werden soll.

Das CCB trifft seine Entscheidung darüber, ob ein Änderungsantrag aufgrund der durchgeführten Auswirkungsanalyse angenommen, abgelehnt oder verschoben wird (vgl. [WiBe2013]). Dabei ist das Ziel, die Auswirkung (den Effekt), den eine Änderung auf alle direkt und indirekt betroffenen Systeme und Prozesse hat, zu identifizieren.

Die IT der Beispiel-Bank, bei der Peter Reber angestellt ist, arbeitet nach einem unternehmensbasierten Projektprozess. Aus diesem Grund hat Peter Reber eine gute Basis zur Etablierung eines Änderungsmanagement-Prozesses und zur Definition der Schnittstellen zu zuliefernden und umsetzenden Prozessen.

Im folgenden Modell hat er die Schnittstellen zum Änderungsmanagement-Prozess (Change Management) skizziert. In der Abbildung ist erkennbar, dass Probleme, die beim Kunden und im IT Betrieb identifiziert werden, zuerst durch das Problem Management bewertet werden, bevor ein Änderungsantrag gestellt wird. Das Change Management als Prozess wird durch das Change Control Board (CCB) umgesetzt. Mitglieder des CCB sind bspw. der Projektleiter, Vertreter von Anwendern und IT sowie Peter Reber als Requirements Manager. Neben dem Problem Management bekommt das Change Management Änderungsanträge aus den Fachbereichen (z.B. das Produktmarketing, die Rechtsabteilung). Durch das CCB akzeptierte und umgesetzte Änderungen werden als Umsetzungsauftrag bzw. Implementierungsauftrag an das Release und Deployment Management übergeben. Peter Rebers Aufgabe wiederum ist es, vor dem CCB-Meeting alle nötigen Informationen von den Experten einzuholen, wie Kostenschätzung, Wichtigkeit der Änderung, Auswirkungen auf Usability und Sicherheit.

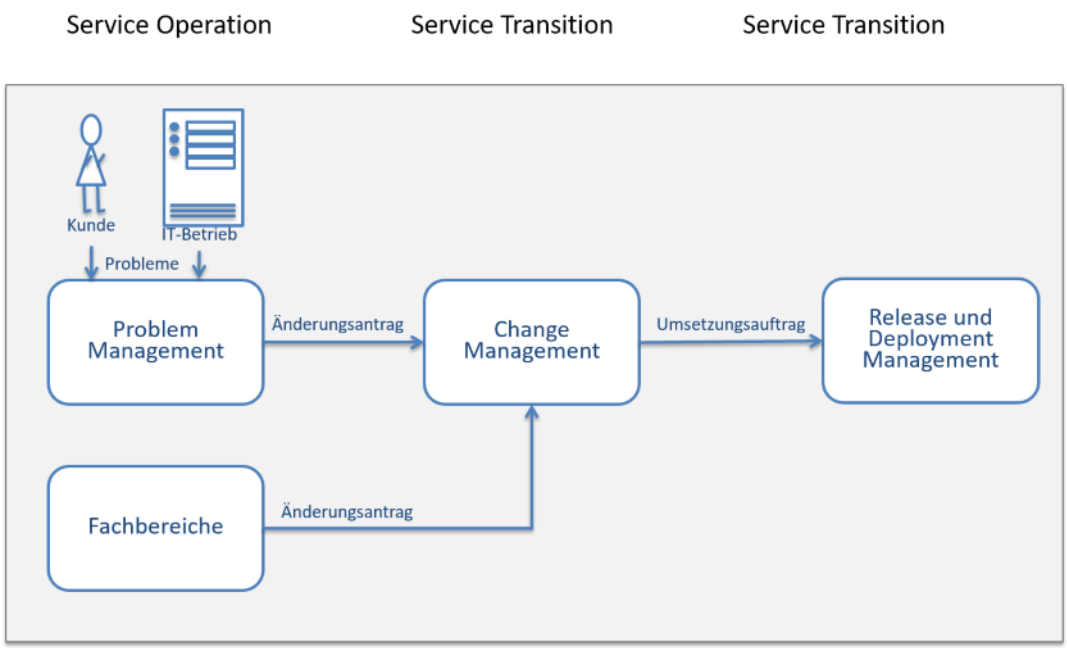


Abbildung 13: Beispiel Schnittstellen zum Change Management

Konzentrieren wir uns im Folgenden auf den Änderungsprozess selbst. Damit Sie Änderungen zielgerichtet integrieren können, planen Sie einen einfachen und effizienten Änderungsprozess für Ihr Projekt. [WiBe2013] geben hierzu ein paar nützliche Tipps:

- Definieren Sie das Ziel des Änderungsmanagement-Prozesses.
- Definieren Sie Rollen und Verantwortlichkeiten im Änderungsmanagement-Prozess.
- Definieren Sie die Eingangskriterien für Änderungsanträge.
- Definieren Sie eindeutige Zustände und Zustandsübergänge, die ein Änderungsantrag durchlaufen kann.
- Definieren Sie einen „schlanken" Änderungsmanagement-Prozess.
- Definieren Sie Ausgangskriterien für den Prozess.
- Legen Sie fest, welches Reporting über Änderungen aufgesetzt werden soll.

Vorschläge für Änderungsmanagement-Prozesse finden Sie bereits im CPRE Foundation Level bzw. in vielen weiteren Literaturquellen: [PoRu2015], [PMI2013], [VanL2009], [WiBe2013].

Welcher Prozess für Ihr Projekt am besten geeignet ist, kann aufgrund der vielfältigen Eigenschaften und Unterschiede der Prozesse nicht pauschal beantwortet werden. Eine Auswahl muss vor allem zu den im Unternehmen gelebten Prozessen passen und Akzeptanz finden. Im Grunde gibt es allerdings keine fundamentale Unterscheidung in den grundlegenden Aktivitäten eines Änderungsmanagement-Prozesses.

Den Auslöser für ein Änderungsmanagement stellt immer der Eingang eines Änderungsantrages dar (weitere Details zum Änderungsantrag siehe Kapitel 5.3.1). Die wesentlichen Aktivitäten im Rahmen eines Änderungsmanagementprozesses lassen sich folgendermaßen zusammenfassen:

- **Schritt 1 – Erstellen des Änderungsantrages**
- **Schritt 2 – formale Prüfung des Antrages:** Im Rahmen dieser Prüfung wird überprüft, ob der Änderungsantrag den festgelegten Eingangskriterien entspricht.
- **Schritt 3 – Klassifikation des Änderungsantrages**: Im Rahmen der Klassifikation von Änderungsanträgen wird festgestellt, ob es sich um eine korrektive, eine adaptive oder eine Ausnahmeänderung handelt. Dabei ist der Requirements Manager in die Bewertung eingebunden, um festzustellen, welche Ursache eine Änderung hat.
- **Schritt 4 – Auswirkungsanalyse der Änderung:** Die Auswirkungsanalyse hat zum Ziel, die Auswirkungen der Änderungen abzuschätzen und zu dokumentieren. Diese Auswirkungen müssen nicht nur für andere Anforderungen, sondern auch für weitere Artefakte (Architektur, Quell-Code, Testfälle, Schulungsunterlagen) bewertet werden. Nutzen Sie hierzu die dokumentierte Verfolgbarkeits-Information (siehe Kapitel 6). Ziel ist es, den notwendigen Anpassungsaufwand für die geforderten Änderungen zu ermitteln.
- **Schritt 5 – Entscheidung über die Umsetzung des Änderungsantrages:** Die Ergebnisse der Auswirkungsanalyse werden vom Change Control Board genutzt, um festzulegen, ob ein Änderungsantrag angenommen oder abgelehnt wird. Es ist nicht in jedem Fall sinnvoll, einen Änderungsantrag zu akzeptieren und umzusetzen. Gründe für eine mögliche Ablehnung eines Änderungsantrages sind z.B.:
  - Die Änderung ist zu aufwändig und steht in keinem gerechtfertigten Verhältnis zu dem notwendigen Aufwand für die Realisierung oder erwarteten Nutzen.
  - Die gewünschte Änderung steht im Widerspruch zu anderen Anforderungen.
  - Die Umsetzung der Änderung würde zu einem zu hohen Risiko bezüglich der Stabilität des betrachteten (Software-)Systems führen.
  - Die Änderung ist nicht durch einen Vertrag abgedeckt.

  Aus Gründen der Nachvollziehbarkeit und zur Abstimmung mit den beteiligten Stakeholdern ist es zwingend notwendig, die Entscheidungen des Change-Control Boards zu dokumentieren.
- **Schritt 6 – Priorisierung der Änderungsanträge:** Die angenommenen Änderungsanträge werden durch das Change-Control-Board priorisiert (z.B. nach Kosten und Nutzen bei adaptiven Änderungen, oder nach Häufigkeit und Auswirkung des Fehlers bei korrektiven Änderungen) (Siehe auch Kapitel 4).
- **Schritt 7 – Einplanung der Änderungsanträge zur Umsetzung:** Die angenommenen Änderungsanträge werden zur Umsetzung eingeplant, z.B. durch ein Projekt, Release etc. und anschließend umgesetzt.

Im Anschluss an den Änderungsmanagement-Prozess startet die eigentliche Änderung. Diese wird entweder durch ein laufendes oder neues Projekt umgesetzt. Die Umsetzungsverantwortung liegt in der Regel weiterhin im Änderungsmanagement.

Für das RM ist an dieser Stelle relevant, dass die notwendigen Änderungen an den Anforderungs-Artefakten sorgfältig durchgeführt werden und sich die Anforderungs-Spezifikation nach der Änderung wieder in einem konsistenten Zustand befindet.

Nutzen Sie zur Durchführung der Änderung die vorhandene Verfolgbarkeit, um alle zu ändernden Artefakte zu identifizieren. Bedenken Sie bei der Änderung der Anforderungs-Artefakte, dass Sie für die geänderten Anforderungen:

- Eine neue Version erzeugen (z.B. V07, am 12.08.2014)
- Den Status der neuen Anforderungs-Version aktualisieren (z.B. gelöscht oder geändert)
- Die Art der Änderung dokumentieren (z.B. korrektive Änderung)
- Den Grund der Änderung dokumentieren (z.B. Anforderung obsolet durch CR-1287)
- Die bestehenden Verfolgbarkeits-Beziehungen aktualisieren

Hinweis: Schaffen Sie darüber hinaus einen Nachweis, was, warum geändert wurde, damit auch andere noch nachvollziehen können, warum eine Änderung durchgeführt wurde bzw. zu welchen Änderungen ein bestimmter Änderungsantrag geführt hat. Betrachten Sie den Änderungsantrag als neues Artefakt und erzeugen Sie neue Verfolgbarkeits-Beziehungen zwischen dem Änderungsantrag und den geänderten Anforderungs-Artefakten.

5.3.1 Der Änderungsantrag

Änderungsanforderungen{ XE "Änderungsanforderung" } bzw. Anforderungsänderungen{ XE "Anforderungsänderung" } werden durch einen Änderungsantrag{ XE "Änderungsantrag" } (engl. Change Request{ XE "Änderungsantrag" } (CR)) beschrieben. Im Rahmen Ihres RMP sollten Sie eine Vorlage für einen Änderungsantrag definieren. Eine Vorlage für einen Änderungsantrag kann beispielsweise wie die Schablone in Tabelle 6 aussehen. Je nach Unternehmen und Projekt können dies aber auch vollständige Dokumentenvorlagen sein. Achten Sie unabhängig von der Form darauf, dass alle für den Änderungsantrag relevanten Attribute in der Vorlage enthalten sind. Als Grundlage können Sie die in Tabelle 6 vorgeschlagenen Attribute für einen Änderungsantrag verwenden.

Inhalt	Beschreibung
Projektname	Bezeichnung des Projekts, für welches die angefragte Änderung gilt
Antragsnummer	Fortlaufende Nummer der Änderungsanträge innerhalb eines Projektes
Titel	Titel der gewünschten Änderung
Datum	Datum des Änderungsantrags
Antragsteller	Name des Antragstellers
Ursprung	Quelle bzw. Ursprung der Änderung (z.B. Marketing, Management, Kunde, Test)
Fachliche Verantwortung	Name oder Abteilung für die fachliche Verantwortung der Ursprungsfunktionalität

Inhalt	Beschreibung
Änderungsart	Art der Änderungsanfrage (z.B. Defect, Innovation, Tuning)
Status	Aktueller Status der Änderungsanfrage (z.B. bewertet, akzeptiert, abgelehnt)
Priorität des Antragstellers	Priorität der Änderung aus der Sicht des Antragstellers
Priorität der Umsetzung	Priorität der Änderung aus der Sicht des Änderungsgremiums
Prüfer der Änderungsanfrage	Name der Person, welche die Durchführung (inkl. Auswirkung) der Änderung prüft
Aktualisierungsdatum	Datum der letzten Aktualisierung der Änderungsanfrage
Version	Versionsnummer des Änderungsdokuments
Release	Release-Zuordnung, zu dem die Änderung umgesetzt werden soll
Spezifikationsaufwand	Prognostizierter Spezifikations-Aufwand der Änderung
Umsetzungsaufwand	Prognostizierter Implementierungs-Aufwand der Änderung
Beschreibung der Änderung	Beschreibung der durchzuführenden Änderung(en)
Kommentare	Kommentare zum Änderungsauftrag

Tabelle 6: Attribute für einen Änderungsantrag (in Anlehnung an [Poh12010])

5.4 Inhalte für den RMP

Dokumentieren Sie in Ihrem RMP, wie Sie in Ihrem Projekt Anforderungen und Dokumente versionieren wollen. Legen Sie fest, welche Zustände eine Anforderung einnehmen darf, wie die Zustandsübergänge zu erfolgen haben und wer den Zustand von Anforderungs-Artefakten ändern darf (vgl. Abbildung 9). Legen Sie darüber hinaus fest, auf welcher Basis eine Anforderungs-Basislinie erstellt wird und was es für den folgenden Anforderungsmanagement-Prozess bedeutet, sobald eine Anforderungs-Basislinie erstellt wurde, z.B. nach einer Anforderungs-Basislinie werden Änderungen nur noch über einen Änderungsmanagement-Prozess akzeptiert. Legen Sie grundsätzlich im RMP fest, wie Sie mit Änderungen im Projekt umgehen wollen, wie diese zu dokumentieren sind, ob es ein Änderungsgremium gibt, wie dieses besetzt wird usw.

Über den RMP haben Sie die Möglichkeit, alle Stakeholder explizit über das geplante methodische Vorgehen zu informieren, sodass der von Ihnen erarbeitete Prozess auch gelebt wird. Zusätzlich bietet ein RMP für nachträglich in das Projekt kommende Teilnehmer die Möglichkeit, sich schnell in die organisatorischen und methodischen Prozesse einzuarbeiten.

5.5 Vertiefende Literatur

[Eber2012] C. Ebert: Systematisches Requirements Engineering. Dpunkt, 4. Auflage, 2012.

[Pohl2010] K. Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. Springer, 2010.

[RuSo2009] C. Rupp & die SOPHISTen: Requirements-Engineering und -Management, Hanser, 5. aktualisierte und erweiterte Auflage, 2009. Kapitel 15

[VanL2009] A. van Lamsweerde: Requirements Engineering – from System Goals to UML Models to Software Specifications. John Wiley and Sons, 2009.

[WiBe2013] K. Wiegers and J. Beatty: Software Requirements, 3rd edition. Microsoft Press, 2013.

[BSB2008] Christoph Bommer, Markus Spindler, Volkert Barr: Softwarewartung – Grundlagen, Management und Wartungstechniken. Dpunkt.verlag, 2008

6 Verfolgbarkeit von Anforderungen

„The overall objective of traceability management is to support consistency maintenance in the presence of changes, by ensuring that the impact of changes is easily localizable for change assessment and propagation." [VanL2009]

6.1 Gründe für die Verfolgbarkeit von Anforderungen

Wie Sie bereits im Foundation Level CPRE [IREB2015] gelernt haben, hat das Thema Verfolgbarkeit{ XE "Verfolgbarkeit" } eine wesentliche Bedeutung für das RM. Die Umsetzung von Verfolgbarkeit ermöglicht es unter anderem:

- Abhängigkeiten zwischen Anforderungs-Artefakten zu erkennen
- Abhängigkeiten zwischen Anforderungs-Artefakten und sonstigen Entwicklungs- bzw. Qualitätssicherungsartefakten zu erkennen
- den Nachweis der Umsetzung und Qualitätssicherung einer Anforderung zu erbringen
- Notwendige Änderungen im Rahmen des Änderungsmanagements zu analysieren und durchzuführen

Die Umsetzung von Verfolgbarkeit bedeutet im Wesentlichen die Pflege von Referenzen bzw. Verweisen zur Dokumentation von Beziehungen zwischen unterschiedlichen Anforderungs-Artefakten sowie zu vorhergehenden (z.B. Geschäftsziele) und zu nachfolgenden Artefakten (z.B. Testfälle).

Bevor wir fortfahren, wollen wir kurz auf die unterschiedlichen Begrifflichkeiten für Verfolgbarkeit von Anforderungen in der zugrundeliegenden Fachliteratur eingehen. In der Literatur findet man für „Verfolgbarkeit" verschiedene Begriffe: Verfolgbarkeit, Traceability, Anforderungs-Verfolgbarkeit etc. In dieser Lehreinheit werden wir den Begriff Verfolgbarkeit verwenden, wenn wir nicht auf eine spezifische Referenz in der Literatur eingehen.

6.1.1 Was versteht man unter Verfolgbarkeit von Anforderungen?

Definition 6-1:

Traceability{ XE "Verfolgbarkeit" } nach IREB: Verfolgbarkeit ist die Fähigkeit eine Anforderung (1) zurück zu ihrem Ursprung (Stakeholder, Dokumente, Begründungen etc.), (2) vorwärts bis hin zum Architekturdesign und Code-Artefakten, sowie (3) zu anderen Anforderungen von denen diese Anforderung abhängt, zu verfolgen.

Wie die obige Definition bereits beschreibt, geht es bei Verfolgbarkeit um die Fähigkeit, die Abhängigkeiten zwischen Anforderungen untereinander sowie deren Abhängigkeit zu vorhergehenden und nachfolgenden Artefakten nachvollziehen zu können.

In der folgenden Definition wird zudem explizit auf die Verfolgbarkeit eines Anforderungs- bzw. Entwicklungs-Artefakts über seinen gesamten Entwicklungs- oder Lebenszyklus eingegangen.

Definition 6-2:

Verfolgbarkeit{ XE "Verfolgbarkeit" } nach [RuSo2009]:

„Traceability (Verfolgbarkeit) ist die Fähigkeit, Verbindungen und Abhängigkeiten zwischen Informationen, welche während der Entwicklung, Erstellung, Wartung und Weiterentwicklung eines Systems anfallen, jederzeit nachvollziehen zu können."

Wenn wir im Folgenden von Verfolgbarkeit von Anforderungen sprechen, verstehen wir darunter die Fähigkeit, Anforderungen untereinander sowie deren Abhängigkeit zu vorhergehenden und nachfolgenden Artefakten durch dokumentierte Verfolgbarkeits-Beziehungen über ihren gesamten Entwicklungs- oder Lebenszyklus nachvollziehen zu können.

6.1.2 Warum Verfolgbarkeit zwischen Anforderungen und anderen Entwicklungs-Artefakten wichtig ist

Die Verfolgbarkeit von Anforderungen ist in der Regel kein Projektziel, sondern eher ein Mittel zum Zweck. In der Literatur finden sich eine Reihe von Gründen, die die Verfolgbarkeit zwischen Artefakten motivieren, vgl. [HJD2011], [IREB2015], [WiBe2013], [VanL2009]:

- Belegbarkeit, wie Ziele und Anforderungen erreicht werden sollen.
- Nachweisbarkeit, warum, ob und wie eine Anforderung umgesetzt wurde
- Identifikation von unnötigen Anforderungen und Eigenschaften im System (Goldrandlösungen)
- Identifikation von fehlenden Artefakten (z.B. fehlende Testfälle)
- Vereinfachung der Zurechenbarkeit von Entwicklungsaufwänden zu Anforderungen
- Unterstützung bei der Wiederverwendung von Artefakten
- Unterstützung in Wartung, Pflege und Weiterentwicklung eines Systems

Die Verfolgbarkeit von Anforderungen unterstützt die Beantwortung wichtiger Fragen im Projektalltag – beispielsweise welche Auswirkung die Änderung bestimmter Anforderungen hat, wie hoch der zu erwartende Umsetzungsaufwand ist oder wie eine Anforderung umgesetzt bzw. getestet wurde.

Durch das Vorhandensein von Verfolgbarkeit werden Sie als Requirements Manager insbesondere bei den folgenden vier Analysen unterstützt (vgl. [HJD2011], [PMI2013]):

- **Auswirkungsanalyse{** XE "Auswirkungsanalyse" **}:** Analyse, welche Artefakte durch eine Änderung (Reduktion oder Erweiterung des Umfangs) beeinflusst werden (vgl. Änderungsmanagement-Prozess)

- **Herkunftsanalyse{** XE "Quellenanalyse" **}**: Analyse, warum ein bestimmtes Artefakt (z.B. Anforderung) existiert, um bspw. unnötige Anforderungen zu erkennen und zu vermeiden
- **Abdeckungsanalyse{** XE "Abdeckungs-Analyse" **}**: Analyse, ob bei der Erfassung der Anforderungen bzw. folgender Entwicklungs-Artefakte alles berücksichtigt wurde, sodass das gewünschte Produkt vollständig erfasst, entwickelt und getestet werden kann
- **Leistungswertanalyse** (*Earned-Value-Analyse*): Analyse zur Ermittlung des Arbeitsfortschritts (sog. Leistungswert), um diesen gegen den ursprünglichen Projektplan zu vergleichen und bei Bedarf geeignete Maßnahmen einzuleiten (vgl. auch Kapitel 8)

Ferner ist die Verfolgbarkeit zwischen Anforderungen und weiteren Artefakten (z.B. Geschäftsprozesse, Gesetzestexten, Testfällen) wesentlich, um bestimmte Reifegrade für Referenzmodelle (z.B. CMMI), Standards / Richtlinien (z.B. ISO12207) oder gesetzliche Vorschriften (z.B. SOX) zu erfüllen.

6.2 Unterschiedliche Verfolgbarkeits-Betrachtungen

[VanL2009] beschreibt Verfolgbarkeit wie folgt: *„...traceability relies on the existence of links between items that we can follow backwards, towards source items, and forwards, towards target items..."* – also die Fähigkeit, zwischen einem Vorgänger- und Nachfolger-Artefakt zu navigieren.

[GoFi1994] unterscheiden die Verfolgbarkeit aus der Perspektive der Anforderungs-Spezifikation in:

- **Pre-Requirements-Specification-Traceability{** XE "Pre-Requirements-Specification-Traceability" **}** ist die Verfolgbarkeit von Anforderungen zu ihrem Ursprung also bspw. zu den vorgelagerten Zielen und Visionen oder sonstigen Anforderungsquellen aus dem Systemkontext, z.B. Verweis auf vorhandene Geschäftsregeln oder Stakeholder.
- **Post-Requirements-Specification-Traceability{** XE "Post-Requirements-Specification-Traceability" **}** ist die Verfolgbarkeit von Anforderungen zu nachfolgenden Entwicklungs-Artefakten, z.B. Systemarchitektur-Komponenten, Code-Fragmenten, Testfällen.

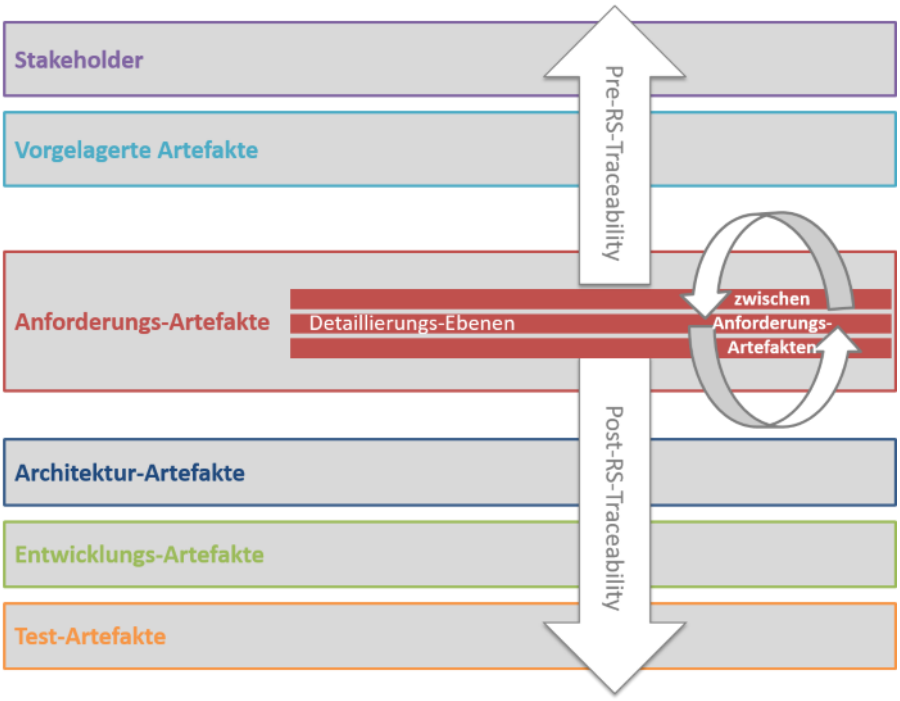


Abbildung 14: Erweiterte Pre- und Post RS Traceability

[Pohl2010] zeigt die erweiterte Sichtweise der Pre- und Post-Requirements-Specification-Traceability um die Verfolgbarkeit zwischen Anforderungs-Artefakten untereinander (also beispielsweise logische Abhängigkeiten zwischen zwei funktionalen Anforderungen) und spricht hierbei von der **erweiterten Pre- und Post Requirements-Specification-Traceability**.

Abbildung 14 illustriert die erweiterte Pre- und Post-RS-Traceability. Wobei sich die Erweiterung auf die Verfolgbarkeit zwischen den Anforderungs-Artefakten konzentriert.

In Kapitel 2.1.3 haben wir davon gesprochen, dass Anforderungen in der Praxis eng mit Architekturentscheidungen verknüpft sind (Twin-Peaks-Model). Zur strukturierten Dokumentation Ihrer Anforderungen haben wir empfohlen, unterschiedliche Detaillierungsebenen einzuführen. Grundsätzlich wird diese Betrachtung zwar durch die Verfolgbarkeit zwischen Anforderungen adressiert – allerdings wird hier nicht explizit unterschieden, ob es sich um bspw. logische Beziehungen zwischen Anforderungen auf einer Ebene oder um eine Detaillierung auf einer tieferen Ebene handelt. Wir wollen diese Unterscheidung hier allerdings explizit in unsere Betrachtung einbeziehen.

Für unsere weitere Betrachtung von Anforderungs-Verfolgbarkeit unterscheiden wir fortan die folgenden **Dimensionen der Verfolgbarkeit**:

- **Verfolgbarkeit zwischen Anforderungen auf derselben Detaillierungsebene:** Diese Art der Verfolgbarkeit beschreibt beispielsweise inhaltliche Abhängigkeiten zwischen funktionalen Anforderungen.
- **Verfolgbarkeit zwischen Anforderungen auf unterschiedlichen Detaillierungsebenen:** Diese Art der Verfolgbarkeit beschreibt beispielsweise Detaillierungen von gesetzlichen Anforderungen zu Systemanforderungen (vgl. Kapitel 2.1.3)
- **Verfolgbarkeit zwischen Versionen von Anforderungen**: Diese Art beschreibt die Verfolgbarkeit der Evolution einer Anforderung über die Zeit. Speziell bei dieser Betrachtung ist, dass es zu jedem Zeitpunkt nur eine gültige Version gibt.
- **Vorwärts-Verfolgbarkeit von Anforderungen zu nachgelagerten Entwicklungs-Artefakten:** Diese Art der Verfolgbarkeit beschreibt beispielsweise Abhängigkeiten, welche die Umsetzung / Realisierung der Anforderung bis hin zur System-Komponente oder zum Testfall dokumentieren.
- **Rückwärts-Verfolgbarkeit von Anforderungen zu vorgelagerten Artefakten:** Diese Art der Verfolgbarkeit beschreibt die Begründung bzw. den Ursprung einer Anforderung.

6.3 Beziehungs-Typen für Verfolgbarkeits-Beziehungen

„Traceability links are thus aimed at localizing items, their origin, rationale and impact. To enable item tracing, such links must be made explicit and documented." [VanL2009]

Dieser Textausschnitt aus [VanL2009] sagt aus, dass Verfolgbarkeits-Beziehungen zwischen voneinander abhängigen Artefakten explizit zu dokumentieren sind, damit diese Abhängigkeiten zu einem späteren Zeitpunkt nachvollzogen werden können.

Weiterhin beschreibt den Grundsatz der Verfolgbarkeit wie folgt: *„In a production chain, an item is traceable if we can fully figure out where the item comes from, why it comes from there, and where it goes to – that is, what it will be used for and how it will be used"*.

Um zu ermöglichen, dass Anforderungs-Artefakte zu ihrem Ursprung sowie zu ihren folgenden Entwicklungs-Artefakten verfolgbar sind und dass durch die Verfolgbarkeits-Beziehungen klar wird, warum diese Abhängigkeit besteht, sind unterschiedliche Typen von Verfolgbarkeits-Beziehungen notwendig.

Natürlich ließen sich Verfolgbarkeits-Beziehungen grundsätzlich durch einen einzigen Beziehungs-Typ beschreiben z.B. „hängt_ab_von".

Hier wäre allerdings bei der Verwendung der dokumentierten Verfolgbarkeits-Beziehungen nicht klar, was der eigentliche Grund dieser Beziehung ist: Sprich, drückt diese Verfolgbarkeits-Beziehung aus, dass eine logische Abhängigkeit zwischen zwei Anforderungen existiert, drückt diese aus, dass eine Anforderung durch eine andere Anforderung detailliert wird oder sagt diese sogar aus, dass sich zwei Anforderungen gegenseitig ausschließen, da sie unterschiedliche Varianten darstellen?

Es fehlt der Hintergrund der Verfolgbarkeits-Beziehung, die eine spätere Nutzung der dokumentierten Verfolgbarkeits-Beziehungen, beispielsweise für Auswirkungsanalysen, wertvoll macht.

[VanL2009] schreibt hierzu: *„The more specialized the dependency, the more specific the reason for it, the more accurate the link, the easier its correct establishment and the more accurate its analysis for multiple uses in traceability management."* Bei der Verwendung und Festlegung von Verfolgbarkeits-Beziehungen gibt es in der Literatur allerdings keine einheitliche Festlegung oder Empfehlung zur Verwendung von Beziehungs-Typen.

6.3.1 Klassen von Beziehungs-Typen für Verfolgbarkeit

[Pohl2010] bildet fünf Klassen von Beziehungs-Typen für die Dokumentation von Verfolgbarkeit, welche je nach Verfolgbarkeitsziel verwendet werden können:

Bedingung: Unter der Klasse „**Bedingung**" werden Verfolgbarkeits-Beziehungen zusammengefasst, um inhaltliche Abhängigkeiten zwischen zwei Artefakten zu beschreiben. Folgende Beziehungs-Typen fallen beispielsweise in diese Klasse:

- *Einschränkung*: Diese Beziehung drückt aus, dass zwischen einem Quell-Artefakt und Ziel-Artefakt eine Einschränkung besteht.
- *Vorbedingung*: Diese Beziehung drückt aus, dass ein Quell-Artefakt eine Vorbedingung für ein Ziel-Artefakt darstellt; sprich, dass eine Anforderung die Vorbedingung für die Erfüllung der anderen darstellt.

Inhalt: Unter der Klasse „**Inhalt**" werden Verfolgbarkeits-Beziehungen zusammengefasst, die inhaltliche Vergleiche zwischen zwei Artefakten anstellen. Folgende Beziehungs-Typen fallen beispielsweise in diese Klasse:

- *Gleichheit*: Diese Beziehung drückt aus, dass ein Quell-Artefakt und Ziel-Artefakt inhaltlich identisch sind.
- *Widerspruch*: Diese Beziehung drückt aus, dass Quell-Artefakt und Ziel-Artefakt im gegenseitigen Widerspruch stehen, der zu einer logischen bzw. inhaltlichen Inkonsistenz führt.
- *Konflikt*: Diese Beziehung drückt aus, dass ein Quell-Artefakt mit einem Ziel-Artefakt in Konflikt steht. Dieser Konflikt muss allerdings nicht zu einem Widerspruch führen, sondern behindert nur die Realisierung des Ziel-Artefakts.

Dokumentation: Unter der Klasse „**Dokumentation**" werden Verfolgbarkeitsbeziehungen zusammengefasst, die weitere Informationen zu einem Artefakt geben. Folgende Beziehungs-Typen fallen beispielsweise in diese Klasse:

- *Beispiel_für*: Diese Beziehung drückt aus, dass ein Quell-Artefakt ein Beispiel für ein Ziel-Artefakt darstellt, z.B. ein Szenario für eine lösungsorientierte Anforderung.
- *Testfall_für*: Diese Beziehung drückt aus, dass ein Quell-Artefakt ein Testfall für ein Ziel-Artefakt ist, z.B. Testfall für eine lösungsorientierte Anforderung.
- *Verantwortlich_für*: Diese Beziehung drückt aus, dass die Person bzw. Rolle eines Quell-Artefakts verantwortlich für Ziel-Artefakt ist, z.B., ist die Rolle „Kundenbetreuung" verantwortlich für das Szenario „Konto stornieren".

- *Hintergrund*: Diese Beziehung drückt aus, dass ein Quell-Artefakt eine Hintergrund-Information für ein Ziel-Artefakt liefert, z.B. Unternehmensrichtlinie für Sicherheitsanforderungen liefert den Hintergrund für die Anforderung zur Kundenauthentifizierung.

Abstraktion: Unter der Klasse „**Abstraktion**" werden Verfolgbarkeitsbeziehungen zusammengefasst, die Abstraktionsbeziehungen zwischen zwei Artefakten beschreiben. Folgende Beziehungs-Typen fallen beispielsweise in diese Klasse:

- *Klassifikation*: Diese Beziehung drückt aus, dass ein Quell-Artefakt eine Klassifikation für ein Ziel-Artefakt liefert, z.B. Szenario „Kontostand abrufen" gehört zur Klasse der administrativen Szenarien.
- *Aggregation*: Diese Beziehung drückt aus, dass ein Quell-Artefakt eine Aggregation über mehrere Ziel-Artefakte liefert, z.B. das Szenario „Kunde authentifizieren" ist eine Aggregation über „Kundenlogin" und „mobile TAN".
- *Generalisierung*: Diese Beziehung drückt aus, dass ein Quell-Artefakt eine Generalisierung für ein Ziel-Artefakt liefert, z.B. Szenario „Buchungen für die letzten 30 Tage abrufen" und „Buchungen für Zeitraum abrufen" werden unter „Buchungen abrufen" zusammengefasst.

Evolution: Unter der Klasse „**Evolution"** werden Verfolgbarkeitsbeziehungen zusammengefasst, welche die Art der Weiterentwicklung einer Anforderung beschreiben (z.B. erfüllt, verfeinert, ersetzt, erweitert). Folgende Beziehungs-Typen fallen beispielsweise in diese Klasse:

- *Ist_Grundlage_für*: Diese Beziehung drückt aus, dass ein Quell-Artefakt eine Grundlage für ein Ziel-Artefakt geliefert hat, z.B. ist die Verwendung von mobilen Telefonen Grundlage für die Qualitätsanforderung „Verwendung eines mobilen TAN Verfahrens".
- *Formalisiert*: Diese Beziehung drückt aus, dass ein Quell-Artefakt eine Formalisierung für ein Ziel-Artefakt liefert, z.B. ein Aktivitätsdiagramm formalisiert eine textuelle Szenario-Beschreibung.
- *Verfeinert*: Diese Beziehung drückt aus, dass ein Quell-Artefakt ein Ziel-Artefakt verfeinert, z.B. eine funktionale Anforderung „Kunde muss sich über ein gültiges Passwort autorisieren" wird verfeinert durch eine Qualitätsanforderung „Ein gültiges Passwort muss alphanumerisch und 8 – 20 Zeichen lang sein".

Welche dieser oder weiterer in der Fachliteratur existierenden Beziehungs-Typen für Ihr Entwicklungsprojekt tatsächlich sinnvoll und notwendig sind, kann leider pauschal nicht beantwortet werden. Wichtig für Ihren RMP (also für die Planung Ihres RE-Prozesses) ist allerdings, dass Verfolgbarkeits-Beziehungs-Typen entsprechend ihres Verfolgbarkeitsziels ausgewählt und verwendet werden (vgl. 6.1).

Tipp: Lassen Sie sich nicht verleiten, alle in der Literatur möglichen Beziehungs-Typen zu verwenden. – Halten Sie die Anzahl der unterschiedlichen Typen so klein wie möglich, um Ihr Verfolgbarkeitsziel zu erfüllen. Eine große Anzahl von Beziehungs-Typen erlaubt zwar die größtmögliche Flexibilität und Genauigkeit, erfordert aber auch einen deutlich höheren Aufwand. Wichtig ist, dass Sie bereits vor dem Start der Anforderungs-Spezifikation eine Festlegung treffen, welche Beziehungs-Typen in Ihrem Projekt verwendet werden sollen.

Zur Auswahl der relevanten Beziehungs-Typen in Bezug auf Ihr Verfolgbarkeitsziel können beispielsweise die in Kapitel 6.2 eingeführten Verfolgbarkeits-Dimensionen unterstützen. Falls Sie also nur beweisen müssen, wie eine Anforderung umgesetzt und getestet wird, so ist beispielsweise die Betrachtung der **Verfolgbarkeits-Beziehungen zur Dokumentation der Vorwärts-Verfolgbarkeit von Anforderungs-Artefakten zu nachgelagerten Entwicklungs-Artefakten** ausreichend.

6.3.2 Dimensionen und Beziehungs-Typen

In diesem Abschnitt zeigen wir anhand von ein paar beispielhaften Zuordnungen, welche Beziehungs-Typen für welche Verfolgbarkeits-Dimension genutzt werden können. Alle Beziehungs-Typen sind hierbei aus der Perspektive des Anforderungs-Artefakts benannt.

Typen von Verfolgbarkeits-Beziehungen zur Dokumentation der Vorwärts-Verfolgbarkeit von Anforderungs-Artefakten zu nachgelagerten Entwicklungs-Artefakten.

- Wird_getestet_durch: Dieser Typ dokumentiert, dass ein Anforderungs-Artefakt durch einen bestimmten Testfall verifiziert wird. Diese Beziehung wird in der Regel durch den Testmanager gepflegt, der den Testfall erstellt.
- Wird_realisiert_durch: Dieser Typ dokumentiert, dass eine Anforderung durch eine bestimmte Software- bzw. Systemkomponente realisiert bzw. abgebildet wird. Diese Beziehung wird in der Regel durch den Systemarchitekten gepflegt, der das Architektur-Design-Artefakt erstellt.
- Wird_implementiert_durch: Dieser Typ dokumentiert, dass eine Anforderung bspw. durch eine bestimmte Funktion, Klasse, Komponente etc. implementiert wird. Diese Beziehung wird in der Regel durch den Entwickler gepflegt.

Typen von Verfolgbarkeits-Beziehungen zur Dokumentation der Rückwärts-Verfolgbarkeit von Anforderungs-Artefakten und vorgelagerten Artefakten.

- Erfüllt: Dieser Typ dokumentiert, dass eine Anforderung für ein vorgelagertes Artefakt (z.B. einen Geschäftsprozess) zur Erfüllung beitragen wird. Diese Beziehung wird in der Regel durch den Requirements Engineer erstellt.
- Schließt_aus: Dieser Typ dokumentiert, dass eine Anforderung die Erfüllung eines vorgelagerten Artefakts (z.B. Geschäftsziel) ausschließt. Diese Beziehung wird in der Regel durch den Requirements Engineer erstellt.
- Steht_in_Konflikt_zu: Dieser Typ dokumentiert, dass eine Anforderung in Konflikt mit einem vorgelagerten Artefakt (z.B. einer gesetzlichen Forderung) steht. Konflikt heißt hierbei, dass die Umsetzung der System-Anforderung die Erfüllung der gesetzlichen

Forderung einschränkt – aber nicht ausschließt. Diese Beziehung wird in der Regel durch den Requirements Engineer erstellt.

- Wird_erklärt_durch: Dieser Typ dokumentiert, dass es für eine Anforderung weitere nicht in der Anforderung enthaltene Hintergrundinformationen gibt. Diese Beziehung wird in der Regel durch den Requirements Engineer erstellt (z.B. von einer Benutzer-Anforderung zu einer gesetzlichen Vorschrift zum Umgang mit SEPA Mandaten).

Typen von Verfolgbarkeits-Beziehungen zur Dokumentation der Verfolgbarkeit zwischen Anforderungs-Artefakten auf einer Detaillierungsebene.

- *Ist_abhängig_von:* Dieser Typ dokumentiert, dass eine Anforderung technisch, logisch oder inhaltlich abhängig von der Erfüllung einer weiteren Anforderung ist (z.B. zwischen einer Anforderung zum Erstellen einer Banküberweisung und der Anforderung zur visuellen Darstellung des Ablaufs).
- *Ist_Beispiel_für:* zur Dokumentation, dass eine Anforderung ein Beispiel für eine andere Anforderung darstellt. Dieser Beziehungs-Typ kann bspw. dazu genutzt werden, um zwischen lösungsorientierten funktionalen Anforderung oder einer Qualitätsanforderung und einem beschreibenden Szenario oder Mock-Up eine Beziehung zu beschreiben.
- *Steht_in_Konflikt_zu:* Dieser Typ dokumentiert, dass zwei Anforderungen miteinander im Konflikt stehen und die Umsetzung der einen Anforderung die Erfüllung der anderen Anforderung einschränkt – allerdings nicht ausschließt. Hieraus lassen sich Einschränkungen ableiten, die im Rahmen eines Projektes beschrieben werden müssen.
- *Steht_im_Widerspruch_zu:* zur Dokumentation, dass sich zwei Anforderungen inhaltlich widersprechen und sich somit in einer konsistenten Lösung ausschließen. Tatsächlich können diese Anforderungen dennoch beide gefordert sein, weil sie in unterschiedlichen Produkten umgesetzt werden sollen. Falls nicht beide gefordert sind, macht dieser Beziehungs-Typ Widersprüche explizit, die aufgelöst werden müssen.
- *Ist_Variante_für:* zur Dokumentation, dass eine Anforderung eine Variante einer anderen darstellt, welche beispielsweise als alternative Lösungsvariante geschätzt werden soll. (**Hinweis**: Eine Alternative ist die explizite Modellierung von Variabilität über die Merkmals-Modellierung, siehe Kapitel 7.3.)

Typen von Verfolgbarkeits-Beziehungen zur Dokumentation der Verfolgbarkeit zwischen Anforderungs-Artefakten auf unterschiedlichen Detaillierungsebenen.

- *Formalisiert*: Zur Dokumentation, dass eine mathematische Beschreibung eine informale Anforderung (z.B. textuelle Geschäftsregel) formalisiert. Dies kann auch die Formalisierung zwischen einer Szenarien-Beschreibung in Prosa und einer vorlagenbasierten Anwendungsfall-Beschreibung sein. Details zu Modellierung von Anforderungen finden Sie im IREB Certified Professional for Requirements Engineering Advanced Level "Requirements Modeling" CHQW2022.
- *Detailliert*: Zur Dokumentation, dass eine oder mehrere Anforderungen auf einer tieferen Detaillierungsebene (z.B. System-Anforderung) eine Anforderung auf

höherer Detaillierungsebene (z.B. Benutzer-Anforderung) dahingehend erweitert (detailliert), dass alle relevanten Aspekte zur Umsetzung beschrieben wurden.

6.4 Darstellungsformen für Verfolgbarkeits-Beziehungen

Zur Dokumentation bzw. Umsetzung von Verfolgbarkeit werden Verfolgbarkeits-Beziehungen genutzt, um die Beziehungen zwischen Artefakten zu dokumentieren (z.B. **Anforderung** *wird getestet durch* **Testfall** oder **textuelle Anforderung** *wird formalisiert durch* **Anforderungsmodell**).

Zwischen welchen Artefakten Verfolgbarkeit zu dokumentieren ist bzw. welche Typen von Verfolgbarkeits-Beziehungen zu verwenden sind, wird durch das Ziel, welches mit der Verfolgbarkeit erreicht werden soll, definiert. Hierbei muss, je nach Verfolgbarkeitsziel, nicht zwingend jede der oben genannten Verfolgbarkeits-Dimensionen berücksichtigt werden. Soll beispielsweise durch die Verfolgbarkeit sichergestellt werden, dass alle Business-Anforderungen in einem Projekt durch System-Anforderungen abgedeckt wurden, bzw. dass eine System-Anforderung mindestens einer Business-Anforderung dient, dann ist unter Umständen bereits eine einfache bidirektionale Verfolgbarkeits-Beziehung vom Typ *„wird umgesetzt durch"* zwischen diesen Artefakten ausreichend. Muss allerdings nach einem bestimmten Sicherheitsstandard (z.B. in der Luft- und Raumfahrt) Verfolgbarkeit realisiert werden, so ist bspw. eine durchgängige Verfolgbarkeit vom Ursprung der Anforderung bis hin zu Code-Artefakten und Testartefakten erforderlich.

6.4.1 Implizite und explizite Dokumentation von Verfolgbarkeit

Verfolgbarkeit kann entweder implizit oder explizit dokumentiert werden.

- **Implizite Dokumentation von Verfolgbarkeit{** XE "Implizite Dokumentation von Verfolgbarkeit" **}**: Eine implizite Verfolgbarkeit kann beispielsweise über Namenskonventionen, Dokumentenstrukturen, Glossare, Referenzen etc. erreicht werden.
- **Explizite Dokumentation von Verfolgbarkeit{** XE "Explizite Dokumentation von Verfolgbarkeit" **}**: Die explizite Verfolgbarkeit wird über definierte und bewusst gesetzte Verfolgbarkeits-Beziehungen zwischen voneinander abhängigen Artefakten erreicht (vgl. Kapitel 6.4.3).

Unter impliziter Verfolgbarkeit verstehen wir die Möglichkeit, Beziehungen zwischen Anforderungen untereinander bzw. zu vorhergehenden und nachfolgenden Artefakten über strukturelle oder stilistische Konventionen erkennen zu können.

Eine implizite Verfolgbarkeit kann durch gleiche Dokumentenstrukturen (z.B. in Lasten- / Pflichtenheft und Testkonzept) erreicht werden. So kann beispielsweise bei einer Strukturierung nach der benutzerzentrierten Funktionalität über Lasten- / Pflichtenheft und Testkonzept erreicht werden, dass man zumindest über verschiedene Entwicklungsphasen hinweg ersehen kann, wie ein Set an Anforderungen (zu einer Funktionalität) aus dem Lastenheft umgesetzt und qualitätsgesichert wird.

Innerhalb einer Spezifikation, können ähnlich wie innerhalb eines Buches Beziehungen (bzw. Abhängigkeiten) zu vorherigen und nachfolgenden Kapiteln, Definitionen, Abbildungen etc. beschrieben werden.

So ist es grundsätzlich auch innerhalb einer Spezifikation möglich, zumindest auf grobgranularer Ebene Verfolgbarkeit zu ermöglichen. Innerhalb einer Spezifikation können beispielsweise Referenzen von Benutzer-Anforderungen zu Qualitätsanforderungen oder zu Anforderungen an die Benutzeroberfläche dokumentiert werden.

Werden darüber hinaus noch gleiche Begriffe (bzw. Prozesswörter) mittels Glossar definiert und durchgängig verwendet, so kann neben dem Verweis auf ein Kapitel auch die entsprechende Stelle in der Spezifikation gefunden werden, auf die eigentlich Bezug genommen wird.

Dennoch ist die implizite Verfolgbarkeits-Dokumentation kein ausreichender Ansatz, um Verfolgbarkeit von Anforderungen im Sinne unseres Verständnisses (vgl. Kapitel 6.1.1) zu ermöglichen.

Wir sehen die Strukturierung (und damit die implizite Dokumentation) aus diesem Grunde nicht als Ersatz, um Verfolgbarkeit zu dokumentieren, sondern als Ergänzung, um Verfolgbarkeit innerhalb und zwischen unterschiedlichen Spezifikationen für den Leser dieser Dokumente zu ermöglichen.

Allerdings ist die explizite Dokumentation von Verfolgbarkeit auch kein Ersatz für eine gut strukturierte, lesbare und verständliche Anforderungs-Spezifikation. Wir möchten sogar so weit gehen und sagen, dass eine verständliche Struktur niemals für die explizite Dokumentation von Verfolgbarkeit ausgelassen werden darf. Schließlich ist eine Spezifikation dafür gedacht, von Menschen gelesen und verstanden zu werden! Verfolgbarkeit hingegen ist mehr ein Mittel zum Zweck, um beispielsweise Umsetzungsnachweise zu erbringen oder die Auswirkung von Änderungen zu analysieren (vgl. Kapitel 6.1.2).

6.4.2 Bidirektionale und unidirektionale Verfolgbarkeits-Beziehungen

Für die Umsetzung von Verfolgbarkeits-Beziehungen kann man zwischen unidirektionalen (gerichteten) und bidirektionalen (nicht gerichteten) Beziehungs-Typen unterscheiden.

- **Unidirektionale Verfolgbarkeits-Beziehungen**: Erlauben die Verfolgbarkeit von einem Artefakt zu einem anderen, aber nicht umgekehrt. So erlaubt die Referenz von einer Testanforderung zu einer System-Anforderung beispielweise zu überprüfen, warum die Testanforderung existiert bzw. wovon diese abhängt. Es kann allerdings von der System-Anforderung kein eindeutiger Verweis auf eine Testanforderung gefunden werden. Diese Art der Beziehungen ist häufig in dokumentenbasierten Techniken zu finden, bei denen Beziehungen manuell bspw. durch textuelle Referenzen gepflegt werden und entweder auf das Vorgänger- oder Nachfolge-Artefakt verweisen. Bei der Dokumentationsrichtung ist wichtig zu beachten, dass auf das Artefakt verwiesen wird, zu dem eine Abhängigkeit besteht.

- **Bidirektionale Verfolgbarkeits-Beziehungen**: Erlauben die Verfolgbarkeit von einem Artefakt zu einem anderen und umgekehrt. Anders als bei der unidirektionalen Beziehung ist es hier möglich, zwischen den Artefakten zu navigieren, z.B. von der Anforderung aus zum Testfall (z.B. durch eine textuelle Referenz auf einen Testfall) und umgekehrt, vom Testfall zur entsprechenden Anforderung, die mit diesem Testfall überprüft werden soll. Diese Art der Beziehungen erlaubt die Betrachtung des Vorgänger- und Nachfolge-Artefaktes (Pre- und Post- Requirements-Specification Tracebility). In RM-Werkzeugen werden bidirektionale Beziehungen in der Regel automatisch erzeugt, sobald eine Verfolgbarkeits-Beziehung erstellt wird. So unterstützt das Werkzeug eine Navigation oder Auswirkungsanalyse in beiden Richtungen. Bei rein textuellen Referenzen ist hingegen die explizite Pflege je beteiligtem Artefakt notwendig.

Hinweis: In der Praxis treffen wir, vor allem bei dokumentenbasierten Spezifikationen, allerdings sehr häufig auf unidirektionale Verfolgbarkeits-Beziehungen, bei der beispielsweise eine System-Anforderung ausschließlich auf eine Geschäfts-Anforderung verweist, aber bei der Geschäfts-Anforderung kein Verweis auf die nachfolgenden Artefakte besteht.

6.4.3 Darstellungsformen für Verfolgbarkeits-Beziehungen

Für die Dokumentation (das Nutzbarmachen) von Verfolgbarkeit ist ein gewisser Aufwand innerhalb des Projektes einzukalkulieren. Dieser Aufwand ist abhängig vom Verfolgbarkeitsziel (Vorwärts-, Rückwärts-Verfolgbarkeit, Verfolgbarkeit zwischen Anforderungs-Artefakten), von der Anzahl der zu berücksichtigen Beziehungs-Typen (siehe Kapitel 6.3), von der Anzahl der Anforderungen im Projekt und nicht zuletzt von der gewählten Darstellungsform.

Zur Dokumentation von Verfolgbarkeit gibt es verschiedene Darstellungsformen. In diesem Abschnitt stellen wir Ihnen die gängigsten Formen vor, vgl. [Pohl2010], [RuSo2009], [VanL2009].

6.4.3.1 Textuelle Referenzen

Die Dokumentation von textuellen Referenzen ist die einfachste Möglichkeit, um Verfolgbarkeits-Beziehungen zwischen Artefakten zu implementieren. Hierbei beschreibt die Beziehung den Beziehungs-Typ sowie eine eindeutige Kennung des Artefakts, auf das die Beziehung verweist (z.B. [TC_0021 testet à FR_3131]). Diese Art der Darstellung hat den entscheidenden Vorteil, dass diese unabhängig von einem RM-Werkzeug genutzt werden kann und leicht verständlich ist. Sie wird in der Regel direkt beim Artefakt dokumentiert, also bspw. beim Testfall der Verweis auf die Anforderung.

Die Dokumentation kann entweder im Anforderungstext selbst (Abbildung 15) oder über dafür vorgesehene Attribute (z.B. „Referenzen auf Testfälle" und „Referenzen Anforderung") umgesetzt werden, siehe Abbildung 16.

Anforderungs-Identifikation	Anforderungsbeschreibung	Priorität	...
FR_3131	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ...wird_getestet_durch TC_0021 ...wird_formalisiert durch FR_0016	Hoch	

Abbildung 15: Verfolgbarkeit durch textuelle Referenzen im Anforderungstext

Referenz auf Testfall	Referenz auf Anforderung	Anforderungs-Identifikation	Anforderungsbeschreibung	Priorität	...
TC_0021	FR_0012 FR_0013 FR_0016	FR_3131	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea ...	Hoch	
TC_0021	---	FR_3132	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea ...	Mittel	
TC_0150	FR_0020	FR_3133	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed ...	Hoch	

Abbildung 16: Verfolgbarkeit durch textuelle Referenzen mittels eigener Attribute

6.4.3.2 Hyperlinks

Im Gegensatz zu rein textuellen Referenzen erlauben Hyperlinks die direkte Navigation zum Ziel-Artefakt. Hyperlinks werden vom Ausgangs-Artefakt zum Ziel-Artefakt erstellt (z.B. von der Anforderung zum Testfall). Bidirektionale Beziehungen lassen sich durch entsprechende Kreuzreferenzen erzeugen.

Im Vergleich zu einfachen textuellen Referenzen hat der Einsatz von Hyperlinks den entscheidenden Vorteil, dass man direkt zu den referenzierten Artefakten „springen" kann (siehe Beispiel in folgender Abbildung 17). Das Beispiel zeigt einen Hyperlink von der funktionalen Anforderung FR_3132 auf den Testfall TC_0021 (als Vorwärts-Verfolgbarkeit von der Anforderung zum Testfall). Das Beispiel zeigt die Umsetzung einer bidirektionalen Verfolgbarkeits-Beziehung: von der Anforderung zum Testfall und vom Testfall zum ursprünglichen Anforderungs-Artefakt bzw. zu den beiden Anforderungs-Artefakten (FR_3131 und FR_3132).

Referenz auf Testfall	Referenz auf Anforderung	Anforderungs-Identifikation	Anforderungsbeschreibung	Priorität	...
TC_0021	FR_0012 FR_0013 FR_0016	FR_3131	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea ...	Hoch	
TC_0021	---	FR_3132	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea ...	Mittel	
TC_0150	FR_0020	FR_3133	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed ...	Hoch	

Referenz auf Anforderung	Testfall-Identifikation	Anforderungsbeschreibung	Priorität	...
FR_3131 FR_3132	TC_0021	Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea ...	Mittel	

Abbildung 17: Verfolgbarkeit durch Hyperlinks

Hinweis: Die Verwendung von Hyperlinks ist in der Regel meist nur innerhalb eines Werkzeugs bzw. zwischen den Werkzeugen desselben Anbieters möglich.

6.4.3.3 Verfolgbarkeitsmatrizen

Durch Verfolgbarkeits-Matrizen{ XE "Verfolgbarkeitsmatrix" } werden Verfolgbarkeits-Beziehungen durch Referenzen in den Zellen einer Matrix dargestellt. In der Horizontalen wird pro Zeile ein Ausgangs-Artefakt dokumentiert. In der Vertikalen wird pro Spalte ein Ziel-Artefakt dokumentiert, sodass in der daraus resultierenden Matrix je Zelle die Beziehung vom Ausgangs-Artefakt zum Ziel-Artefakt dokumentiert werden kann. Diese Art der Darstellung erlaubt eine abstrakte Darstellung der Abhängigkeiten zwischen zwei Artefakt-Typen in einer Matrix.

Verfolgbarkeits-Matrizen werden oftmals dazu verwendet, um jeweils genau einen Beziehungs-Typ (z.B. erfüllt) zwischen Ausgangs- und Ziel-Artefakt zu dokumentieren (siehe Abbildung 18).

Die Verfolgbarkeits-Beziehung wird dann beispielsweise als einfaches „x" in der jeweiligen Zelle dokumentiert. In diesem Beispiel eine Rückwärtsverfolgbarkeit vom Testfall (TC) zur Anforderung (FR), gepflegt vom Ersteller des Testfalls.

Im gezeigten Beispiel testet der Testfall TC_10 die funktionale Anforderung FR_0010; Testfall TC_20 testet die funktionale Anforderung FR_0011; die Testfälle TC_30 und TC_40 testen die funktionale Anforderung FR_0020 und der Testfall TC_40 testet darüber hinaus die funktionale Anforderung FR_0030. Hier haben wir also eine N-zu-M-Beziehung.

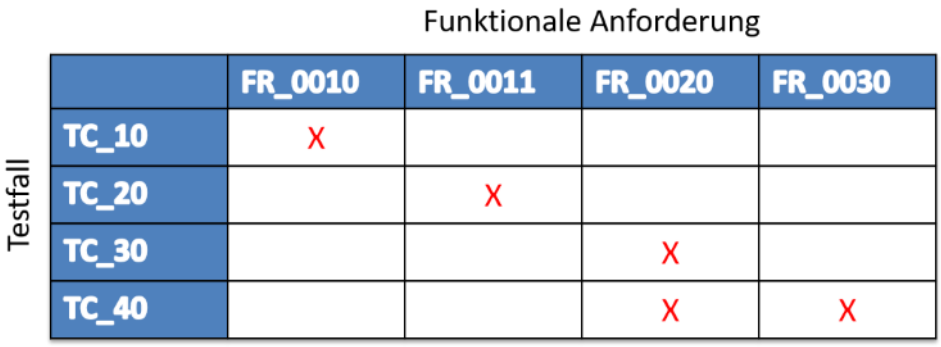


Abbildung 18: Verfolgbarkeits-Matrix mit einem Beziehungs-Typ (FR = functional requirement, TC = test case)

Sollen unterschiedliche Beziehungs-Typen zwischen zwei Artefakten dokumentiert werden (z.B. zwischen Anforderungen auf einer Detaillierungsebene), dann können in den Zellen auch die jeweiligen Beziehungs-Typen dokumentiert werden (siehe Abbildung 19).

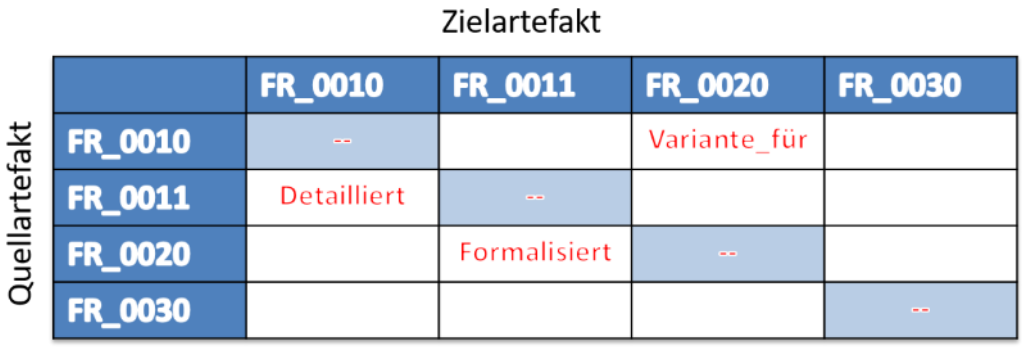


Abbildung 19: Verfolgbarkeits-Matrix mit mehreren Beziehungs-Typen (FR = functional requirement, TC = test case)

Die Abbildung zeigt ein Beispiel für die Verwendung unterschiedlicher Beziehungs-Typen in einer Verfolgbarkeits-Matrix. Zu lesen ist die Matrix von der Zeile (Quell-Artefakt) zur Spalte (Ziel-Artefakt), also: FR_0011 „Detailliert" FR_0010; FR_0020 „Formalisiert" FR_0011 und FR_0010 ist eine „Variante_für" FR_0020.

RM-Werkzeuge wie z.B. DOORS erstellen Verfolgbarkeits-Matrizen automatisiert auf Basis der zuvor angelegten Verfolgbarkeits-Beziehungen zwischen Artefakten. In der Praxis werden diese Matrizen allerdings schnell sehr groß und sind auf Grund ihrer Größe schwierig zu pflegen und kaum zu lesen.

6.4.3.4 Verfolgbarkeits-Tabellen

Im Gegensatz zu Verfolgbarkeits-Matrizen bieten Verfolgbarkeits-Tabellen{ XE "Verfolgbarkeitstabelle" } die Möglichkeit, Verfolgbarkeits-Beziehungen zwischen allen Artefakten auf unterschiedlichen Detaillierungsebenen zu beschreiben.

Sie bieten damit ein mächtiges Hilfsmittel, um die Verfolgbarkeit von Zielen über Use Cases und funktionale Anforderungen bis hin zu Testfällen zu dokumentieren. Verfolgbarkeits-Tabellen können unabhängig von einem speziellen RM-Werkzeug verwendet werden, um die Verfolgbarkeit zwischen Artefakten zu dokumentieren, die darüber hinaus in unterschiedlichen Werkzeugen (z.B. Rational Rose, Visual Paradigm, Quality Center) und Office Applikationen (z.B. Word, Excel) dokumentiert sind.

Vorwärtsverfolgbarkeit aus Sicht einer Geschäfts-Anforderung →

Geschäfts-Anforderung	Use Case	Funktionale Anforderung	System Anforderung	Designartefakt	Testfall
BR_0010	UC_10	FR_0012 FR_0013 FR_0016	CRM_0011 DWH_0010 Billing_0020	GUI_0081	TC_0021 TC_0022 TC_0025
...	...	...	...	...	...
BR_0099	UC_60	FR_0020	CRM_0011 CRM_0020		TC_0060 TC_0150

Abbildung 20: Verfolgbarkeits-Tabelle (BR = Business Requirement, UC = Use Case, FR = Functional Requirement, CRM = Customer Relationship Management, DWH = Data Warehouse, GUI = Graphical User Interface / grafische Benutzeroberfläche, TC = test case / Testfall)

Abbildung 20 zeigt, welche Artefakte mit einer Geschäfts-Anforderung (hier die Quelle) in Beziehung stehen. So hat BR_0010 eine Verfolgbarkeits-Beziehung zu UC_10; zu den funktionalen Anforderungen FR_0012, FR_0013, FR_0016; zu den System-Anforderungen CRM_0011, DWH_0010, Billing_0020; dem Architektur-Design-Artefakt GUI_0081 sowie den Testfällen TC_0021, TC_0022, TC_0025. Nicht erkennbar ist in diesem Beispiel der zugrundeliegende Beziehungs-Typ zwischen diesen Artefakten. Da sich die Verfolgbarkeits-Beziehung allerdings immer auf das eine Quell-Artefakt bezieht, wäre eine entsprechende Erweiterung um den Beziehungs-Typ beim jeweiligen Ziel-Artefakt denkbar.

So könnte man z.B. durch eine Ergänzung bei FR_0012 beschreiben, dass die Geschäfts-Anforderung BR_0010 durch die funktionale Anforderung FR_0012 verfeinert wird „*wird verfeinert durch:* FR_0012". Über diese Erweiterung hat man sogar die Möglichkeit, je Quell-Ziel-Beziehung unterschiedliche Beziehungs-Typen zu verwenden (siehe Abbildung 21).

Leserichtung der Beziehungen aus Sicht einer Geschäfts-Anforderung

Geschäfts-Anforderung	Use Case	Funktionale Anforderung	...
BR_0010	*...wird abgebildet durch:* UC_10	*...wird verfeinert durch*: FR_0012 *...wird verfeinert durch:* FR_0013 *...steht in Konflikt zu:* FR_0016	
...	...	...	
BR_0099	*...wird abgebildet durch:* UC_60	*...wird detailliert durch:* FR_0020	

Abbildung 21: Verfolgbarkeits-Tabelle mit Beziehungs-Typen

6.4.3.5 Verfolgbarkeitsgraphen

Eine weitere Darstellungsart für Verfolgbarkeit sind Verfolgbarkeits-Graphen{ XE "Verfolgbarkeits-Graph" }. In einem Verfolgbarkeits-Graphen stellen die Knoten des Graphen die relevanten Artefakte dar, und die Kanten repräsentieren die Beziehungen zwischen den Artefakten. Um die unterschiedlichen Entwicklungs-Artefakte (z.B. Szenario, Anforderung, Testfall) und Beziehungs-Typen (z.B. verfeinert, implementiert, testet) auf einen Blick voneinander unterscheiden zu können, empfiehlt es sich, eine entsprechende Notationsform für die Knoten- und Kantentypen festzulegen. Der Einsatz von Verfolgbarkeits-Graphen ist allerdings nur zu empfehlen, falls diese Graphen werkzeugbasiert auf Basis der Artefakte und Beziehungen automatisiert erstellt werden können. Die manuelle Erstellung und Pflege solcher Graphen wird in der Realität zu aufwendig sein. Grundsätzlich bieten Verfolgbarkeits-Graphen allerdings eine gut verständliche Art, die Abhängigkeiten zu überprüfen und zwischen den unterschiedlichen Artefakten zu navigieren. Ähnlich wie in Verfolgbarkeits-Matrizen oder Tabellen fehlen hier allerdings die eigentlichen Artefakte – weshalb der Kontext der Verfolgbarkeits-Beziehung verloren geht. Die folgende Abbildung zeigt einen beispielhaften Verfolgbarkeits-Graphen (Abbildung 22).

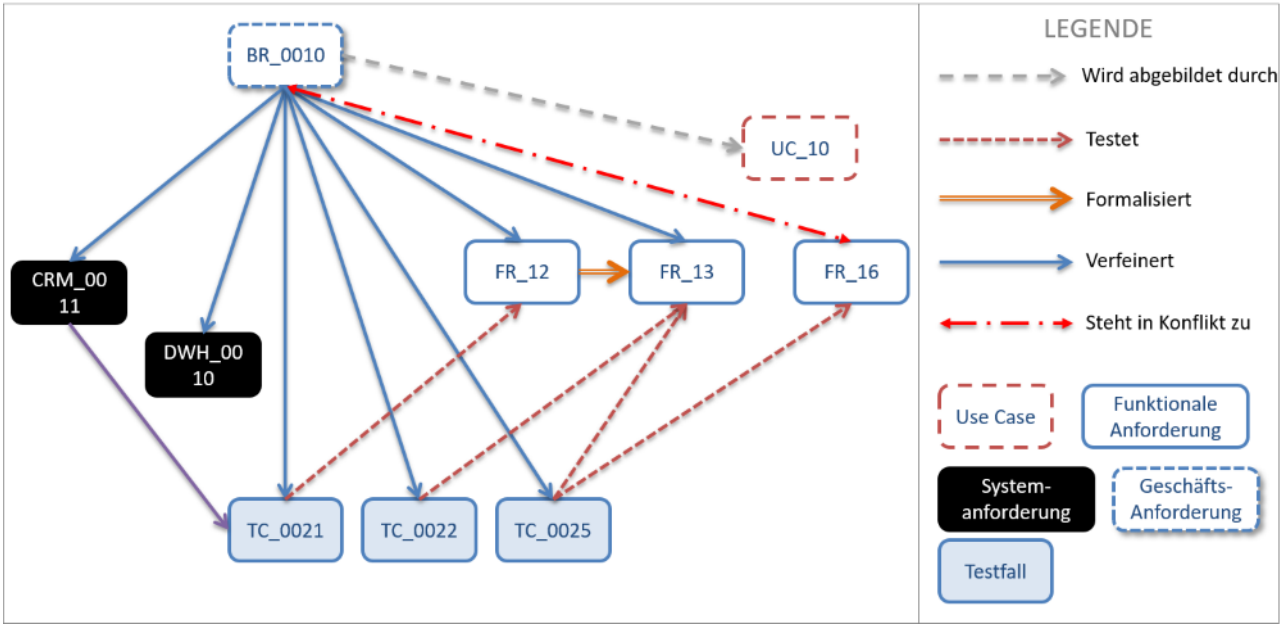


Abbildung 22: Verfolgbarkeits-Graph

Die Abbildung zeigt Verfolgbarkeits-Beziehungen zwischen unterschiedlichen Entwicklungs-Artefakten als Knoten (Geschäfts-Anforderungen, Use Cases, funktionalen Anforderungen, System-Anforderungen und Testfällen) und unterschiedlichen Beziehungs-Typen als Kanten (wird abgebildet durch; testet; formalisiert; verfeinert; steht in Konflikt zu).

Wie in der Abbildung ersichtlich, bieten Verfolgbarkeits-Graphen eine graphische Möglichkeit, Beziehungen zwischen unterschiedlichen Artefakten darzustellen. Beim Einsatz dieser Graphen ist allerdings darauf zu achten, dass die Anzahl der Artefakte und Beziehungen nicht zu hoch gewählt wird. Dieses Beispiel ist mit der Anzahl der unterschiedlichen Artefakte (5) und Beziehungen (5) bereits an der Grenze der Verständlichkeit angelangt. Schließlich sollen diese graphischen Darstellungen dazu genutzt werden, um zu erkennen, zwischen welchen Artefakten Abhängigkeiten existieren. Daher wird in der Praxis sogar oft nur auf einen Beziehungs-Typ reduziert.

Diese Abhängigkeiten sind meist komplex genug, sodass man nicht zwingend zusätzliche Komplexität durch die Anzahl der dargestellten Artefakte und Beziehungen mit in das Model bringt. Bei der Verwendung von Werkzeugen lassen sich aber über Filter bestimmte Artefakte oder Beziehungs-Typen ein- und ausblenden, sodass Sie immer nur mit der nötigen Komplexität „konfrontiert" werden.

6.4.3.6 Vergleich der unterschiedlichen Darstellungsformen für Verfolgbarkeit

Die folgende Tabelle (Tabelle 7) stellt die bis hierher besprochenen Darstellungsformen gegenüber und nennt deren Vor- und Nachteile. Tabelle 7 klassifiziert die unterschiedlichen Darstellungsformen in Inline-Darstellung und orthogonale Darstellung. Unter Inline Dokumentation wurden die Darstellungsformen „textuelle Referenzen" und „Hyperlinks" einsortiert, da hier die Verfolgbarkeits-Beziehungen in direkter Verbindung zur

Anforderungs-Spezifikation stehen – also im Kontext dargestellt werden. Bei der orthogonalen Dokumentation durch Verfolgbarkeits-Matrizen, Verfolgbarkeits-Tabellen und Verfolgbarkeits-Graphen wird in der Regel das Wissen um die Beziehungen getrennt von der Anforderungs-Spezifikation dargestellt, da diese Beschreibungen i.d.R. von den Artefakten selbst abstrahieren.

Darstellungsform	Positiv	Negativ	Gut geeignet für
Inline Dokumentation von Verfolgbarkeit			
Textuelle Referenzen	Werkzeugunabhängig und übergreifend umsetzbar Beziehung ist beim Artefakt als Klartext erkennbar	Verfolgbarkeits-Analysen sind sehr aufwendig	...die Darstellung von Verfolgbarkeit in papierbasierten textuellen Spezifikationen
Hyperlinks	Beziehung ist beim Artefakt als Klartext erkennbar Einfache Navigierbarkeit zwischen Artefakten, um direkte Abhängigkeiten zu erkennen	Verfolgbarkeit zwischen verschiedenen Werkzeugen nicht ohne weiteres möglich	...die Darstellung von Verfolgbarkeit in elektronischen Spezifikationen
Orthogonale Dokumentation von Verfolgbarkeit			
Verfolgbarkeits-Matrizen	Abhängigkeit zwischen zwei Artefakten ist schnell und einfach erkennbar	Manuelle Erstellung von Verfolgbarkeits-Matrizen ist aufwändig und führt zu vielen großen, nur schwach gefüllten Matrizen	...die Darstellung zur Verfolgbarkeit nur eines einzigen Beziehungs-Typen zwischen zwei bestimmten Artefakt-Typen (z.B. Use Cases und Anforderungen)

Darstellungsform	Positiv	Negativ	Gut geeignet für
Verfolgbarkeits-Tabellen	Werkzeugunabhängig umsetzbar Ermöglicht übersichtliche Darstellung der erweiterten Pre- & Post RS Traceability. Erlaubt vielfältige Verfolgbarkeits-Analysen	Hohe Komplexität bei der Erstellung	...die Darstellung von Verfolgbarkeit zwischen textuellen und modellbasierten Artefakten in unterschiedlichen Dokumenten / Werkzeugen
Verfolgbarkeits-Graphen	Graphische Darstellung von Traceability ermöglicht „abstrakte" Darstellung von Verfolgbarkeits-Beziehungen zwischen Artefakten	Einsatz nur mit entsprechender Werkzeugunterstützung realisierbar	...die Darstellung komplexer Verfolgbarkeit zwischen Artefakten in einem RM-Werkzeug

Tabelle 7: Darstellungsformen von Verfolgbarkeits-Beziehungen

6.5 Erstellung einer Strategie zur projektspezifischen Verfolgbarkeit

Die Erstellung und Verwendung von Verfolgbarkeit in einem Projekt muss bewusst geplant werden. Es ist in der Regel nicht zielführend, sämtliche möglichen Beziehungen zwischen allen Artefakten zu dokumentieren. Vielmehr sollte man sich zu Projektbeginn Gedanken machen, wozu Verfolgbarkeit in diesem Projekt notwendig ist und an welchen Stellen welche Art von Verfolgbarkeit benötigt wird, um dieses Ziel zu erfüllen. Zur Erstellung einer Verfolgbarkeits-Strategie{ XE "Verfolgbarkeits-Strategie" } sind folgende Fragen zu beantworten.

- Verfolgbarkeitsziel: Warum oder wozu muss Verfolgbarkeit in diesem Projekt umgesetzt werden? Diese Frage muss durch das bereits mehrfach erwähnte Verfolgbarkeitsziel beantwortet werden.
- Nutzungsstrategie: Wofür soll die dokumentierte Verfolgbarkeit genutzt werden? Diese Frage muss durch eine Nutzungsstrategie beantwortet werden.
- Aufzeichnungs-Strategie: Wer ist für die Dokumentation der Verfolgbarkeit verantwortlich? Diese Frage muss durch eine Aufzeichnungs-Strategie beantwortet werden.
- Projektspezifisches Verfolgbarkeits-Modell: Wie und zwischen welchen Artefakten soll Verfolgbarkeit dokumentiert werden? Diese Frage muss durch eine Strategie zur Dokumentation von Verfolgbarkeit beantwortet werden.

Hinweis: Neben der Beantwortung dieser Fragen, ist vor allem darauf zu achten, dass die Strategie bei allen Beteiligten bekannt ist, von allen Beteiligten verstanden und vor allem akzeptiert wird. Ansonsten wird jede noch so ausgefeilte Strategie im Projektalltag verpuffen.

6.5.1 Das Verfolgbarkeitsziel

Das Verfolgbarkeitsziel soll die Frage beantworten, aus welchem Grund Verfolgbarkeit im Rahmen des jeweiligen Projektes erforderlich ist bzw. etabliert werden soll. Gründe für die Notwendigkeit von Verfolgbarkeit können entweder von außen motiviert (z.B. um Standards zu erfüllen) oder von innen aus dem Projekt heraus motiviert sein (z.B. um Änderungsanfragen schneller und korrekter bearbeiten zu können).

Von außen motivierte Ziele für Verfolgbarkeit sind beispielsweise:

- Durch das Unternehmen vorgegebene Richtlinien oder Entwicklungsstandards, um Zertifizierungen zu erfüllen: z.B. CMMI (Capability Maturity Model Integration des SEI) [SEI1999], [SEI2010]; ISO 9000 / ISO 9001 [ISO9000] ISO 12207 [ISO12207]
- Durch Gesetze oder Verordnungen vorgeschriebene rechtliche Vorgaben in bestimmten Märkten und Domänen: z.B. SOX (Sarbanes-Oxley Act) [USCo2002]
- Durch bestimmte Domänen vorgegebene Richtlinien: z.B. IEC DIN EN 61508 [DIN61508], Department of Defense DOD-STD-2167A

Von innen heraus motivierte (projektgetrieben) Ziele sind beispielsweise:

- Die Unterstützung der Nachweisbarkeit gegenüber dem Auftraggeber: z.B. warum eine Anforderung umgesetzt wurde, wie eine Anforderung umgesetzt wurde, bzw. dass eine Anforderung umgesetzt wurde
- Die Qualitätssicherung von Spezifikationen durch Identifikation von unnötigen Anforderungen in einer Spezifikation (ohne Quelle) oder fehlenden Testfällen
- Die Unterstützung bei der Wartung, Pflege und Weiterentwicklung eines Systems, z.B. durch Identifikation der zu ändernden Anforderungen und Folgeartefakte

6.5.2 Die Nutzungsstrategie

Die Nutzungsstrategie soll die Frage beantworten, wie die aufgezeichnete Verfolgbarkeits-Information genutzt werden soll.

Die Nutzungsstrategie legt fest, wie die dokumentierten Verfolgbarkeits-Informationen durch das Team genutzt werden sollen. Eine Nutzungsstrategie kann sich beispielsweise auf die Auswirkungsanalyse beziehen, bei der die Verfolgbarkeits-Beziehungen dazu verwendet werden, herauszufinden, auf welche Anforderungs- und Folgeartefakte sich eine Änderung auswirkt. Die Nutzungsstrategie legt auch fest, wer über welche Artefakt-Typen und Beziehungs-Typen welche Analysen durchführen darf bzw. soll.

Die Auswirkungsanalyse wird in der Regel durch den Requirements Manager durchgeführt. Hierbei prüft er auf Basis der zu ändernden Anforderungs-Artefakte und der

dokumentierten Verfolgbarkeits-Beziehungen, welche weiteren Ziele, Anforderungen, Architektur-Design-Artefakte, Testfälle etc. von der Änderung betroffen sind.

Bei einer Testabdeckungs-Analyse aus Anforderungs-Sicht hingegen konzentriert man sich auf Anforderungs-Artefakte und Testfälle, um zu prüfen, ob alle Anforderungen durch Testfälle abgedeckt sind. Die Durchführung dieser Analyse kann entweder durch den Requirements Manager oder durch einen Testmanager erfolgen.

Mögliche Verwendungen von Verfolgbarkeits-Informationen, die in eine Nutzungsstrategie einfließen, sind:

- **Auswirkungsanalyse**: Verwendung der Verfolgbarkeit, um das Änderungsausmaß in Anforderungen und Folgeartefakten zu identifizieren
- **Testabdeckungs-Analyse**: Verwendung der Verfolgbarkeit, um fehlende Testabdeckung von Anforderungen zu identifizieren
- **Wiederverwendbarkeit**: Verwendung der Verfolgbarkeit, um wiederverwendbare Artefakte zu identifizieren
- **Änderungshäufigkeit**: Verwendung der Verfolgbarkeit, um die Häufigkeit und die Hintergründe der Anforderungsänderungen zu identifizieren
- **Umsetzungsbelegbarkeit**: Verwendung der Verfolgbarkeit, um die Umsetzung von Anforderungen belegen zu können

Hinweis: Sie legen die Nutzungsstrategie in der Regel auf Basis der Ziele fest. Überlegen Sie sich hier, wozu Sie die Verfolgbarkeits-Information nutzen wollen, wer diese nutzen soll und welche Beziehungs-Typen und Artefakte für diese Verwendung relevant sind.

6.5.3 Die Aufzeichnungs-Strategie

Die Aufzeichnungs-Strategie soll die Frage beantworten, wer die notwendigen Verfolgbarkeits-Beziehungen umsetzt und aktuell hält. Die Aufzeichnungs-Strategie legt die Verantwortlichkeit für die Dokumentation von Verfolgbarkeits-Beziehungen fest. In der Aufzeichnungs-Strategie wird je Beziehungs-Typ zwischen zwei Artefakt-Typen festgehalten, wer diese Beziehung zu welchem Zeitpunkt zu pflegen hat.

Eine Aufzeichnungs-Strategie kann beispielsweise die von[HJD2011] oder [WiBe2013] vorgeschlagene chronologische Dokumentation von Verfolgbarkeits-Beziehungen sein. Hierbei wird die Beziehung zwischen zwei Artefakten erstellt, sobald das neue Artefakt erzeugt wird (z.B. der Beziehungs-Typ *„detailliert"* einer Benutzer-Anforderung zu einer Geschäfts-Anforderung wird durch den Business Analyst gepflegt, sobald eine Benutzer-Anforderung für eine Geschäfts-Anforderung erstellt wird).

Dies hat den Vorteil, dass eine eindeutige Verantwortung für das Setzen von Verfolgbarkeits-Beziehungen existiert und dass Verfolgbarkeits-Beziehungen mit dem Anlegen eines Artefaktes erzeugt werden können.

Wäre beispielsweise ein Business Analyst dafür verantwortlich, die Verfolgbarkeits-Beziehungen zwischen Testfällen und Anforderungen zu pflegen, könnte dies erst erfolgen,

sobald die Testfälle erzeugt wurden. Zudem müsste der Business Analyst Annahmen darüber treffen, welche Testfälle die Umsetzung welcher Anforderungen überprüfen sollen. Hier hätten wir also doppelten Aufwand und eine entsprechende Fehleranfälligkeit, weshalb wir ebenfalls die chronologische Dokumentation von Verfolgbarkeits-Beziehungen empfehlen. Sodass Sie „nur" festlegen müssen, welche Person bzw. Rolle in ihrem Projekt für die von Ihnen – im projektspezifischen Verfolgbarkeits-Modell – festgelegten Beziehungs-Typen zu pflegen hat.

6.5.4 Das projektspezifische Verfolgbarkeits-Modell

Durch das projektspezifische Verfolgbarkeits-Modell soll die Frage beantwortet werden, wie (d.h. durch welche Darstellungsform, vgl. Kapitel 6.4) und zwischen welchen Artefakt-Typen Verfolgbarkeit dokumentiert werden soll. Bevor Sie also Verfolgbarkeits-Beziehungen dokumentieren, ist es notwendig, dass Sie sich bereits vor der Dokumentation Ihrer Anforderungen im Klaren sind, zwischen welchen Artefakt-Typen Sie Verfolgbarkeit dokumentieren wollen bzw. müssen (vgl. 6.5.1). Diese Festlegungen beschreiben Sie entweder textuell, als eigenständiges Informationsmodell (Beispiel in Abbildung 23) oder ergänzend in Ihrem RIM.

Ein projektspezifisches Verfolgbarkeits-Modell beschreibt die zulässigen Beziehungs-Typen zwischen den relevanten Anforderungs-Artefakt-Typen. Darüber hinaus beschreibt es, wie – also durch welche Darstellungsform – Verfolgbarkeit zu dokumentieren ist (vgl. Kapitel 6.4.3). Die Erstellung und Verwendung eines projektspezifischen Verfolgbarkeits-Modells wird in Kapitel 6.6 beschrieben.

6.6 Projektspezifische Verfolgbarkeits-Modelle erstellen und verwenden

Durch die Festlegung der Dokumentations-Strategie – also dem Verfolgbarkeits-Modell mit seinen zulässigen Artefakt-Typen und der zulässigen Beziehungs-Typen und der Darstellungsform – kann allen Projektbeteiligten eindeutig dargestellt werden, welche Artefakt-Typen und Beziehungs-Typen existieren und wie diese zu pflegen sind (vgl. [Pohl1996], [Pohl2010], [MGP2009], [MJZC2013]). Wer diese zu welchem Zeitpunkt zu pflegen hat, wird mit der Aufzeichnungs-Strategie (vgl. Kapitel 6.4.3) festgelegt.

Hinweis: Zur tatsächlichen Umsetzung und Verwendung eines für das Projekt oder Unternehmen erarbeiteten projektspezifischen Verfolgbarkeits-Modells empfiehlt sich in der Regel ein Requirements-Management-Werkzeug, welches die entsprechenden Artefakte, die erlaubten Beziehungen und die entsprechenden Stakeholder-Rollen abbildet. Selbstverständlich sind alle methodischen Konstrukte auch mit herkömmlichen Office-Anwendungen umsetzbar – hier fehlt es aber oftmals an der Möglichkeit, manuell gesetzte Verfolgbarkeits-Beziehungen automatisch zu analysieren bzw. notwendige Auswirkungsanalysen zu erstellen.

6.6.1 Erstellung eines projektspezifischen Verfolgbarkeits-Modells

In einem projektspezifischen Verfolgbarkeits-Modell wird festgelegt, welche Beziehungs-Typen (z.B. *wird_verfeinert_durch*; *wird_getestet_durch*) zwischen welchen Artefakt-Typen (z.B. Anforderungen und Testfällen) existieren sollen bzw. dürfen.

Im Folgenden wird ein exemplarischer Prozess zur Definition eines projektspezifischen Verfolgbarkeits-Modells beschrieben.

- **Auswahl eines Referenzschemas**:
  Im ersten Schritt sollte geprüft werden, ob man ein bereits vorhandenes Verfolgbarkeitsmodell wiederverwenden und anpassen kann. Eine effektive Möglichkeit, ein projektspezifisches Verfolgbarkeits-Modell zu definieren, liegt in der Wiederverwendung eines bestehenden Verfolgbarkeits-Modells aus einem ähnlichen Projekt oder eines unternehmensweiten Verfolgbarkeits-Modells. Ein solches Verfolgbarkeits-Modell kann als Basis für die Definition des projektspezifischen Verfolgbarkeits-Modells dienen und wird in der Regel eine Vielzahl der zu definierenden Artefakte und Abhängigkeiten bereits enthalten.
- **Auswahl der Artefakt-Typen**:
  Im zweiten Schritt wird festgelegt, zwischen welchen Artefakten grundsätzlich eine Verfolgbarkeit gewährleistet werden soll, um das im Rahmen der Verfolgbarkeits-Strategie festgelegte Ziel und die Nutzungsszenarien zu unterstützen, z.B. Verfolgbarkeit zwischen Use Case und funktionaler Anforderung sowie zwischen Anforderung und Testfall.
- **Definition von zulässigen Beziehungs-Typen zwischen den Artefakten**:
  Hier ist festzulegen, welche Verfolgbarkeits-Beziehungen (vgl. Kapitel 6.3) zwischen zwei Artefakten erlaubt sind, z.B. gültige Beziehung zwischen Anforderung und Testfall: „wird validiert durch".
- **Festlegung der Anzahl der Verfolgbarkeits-Beziehungen**:
  Hier wird festgelegt, wie viele Beziehungen zwischen den realen Artefakten (auf Instanz-Ebene des Verfolgbarkeits-Modells) mindestens erwartet werden, z.B. jede Anforderung benötigt eine Verfolgbarkeits-Beziehung zu einem Testfall.
- **Definition der Abhängigkeit zwischen Artefakten:**
  Hier ist festzulegen, welches Artefakt von einem anderen Artefakt abhängig ist, z.B. der Testfall ist abhängig vom Inhalt der Anforderung. Bei der Verwendung von unidirektionalen Beziehungen ist dabei auf die Referenzierung zu achten (vgl. Kapitel 6.4.2).

Das beispielhafte Verfolgbarkeits-Modell (Abbildung 23) stellt dar, zwischen welchen Artefakt-Typen welche Verfolgbarkeitstypen zulässig sind. Eine Anforderung kann beispielsweise eine weitere Anforderung oder ein Geschäftsziel detaillieren. Eine Anforderung wird durch ein Design-Element realisiert. Eine Anforderung wird durch einen Testfall getestet. In der Instanz dieses Modells wäre es beispielsweise nicht erlaubt, dass ein Testfall über den Beziehungs-Typ *detailliert* mit einem Geschäftsziel verbunden wird.

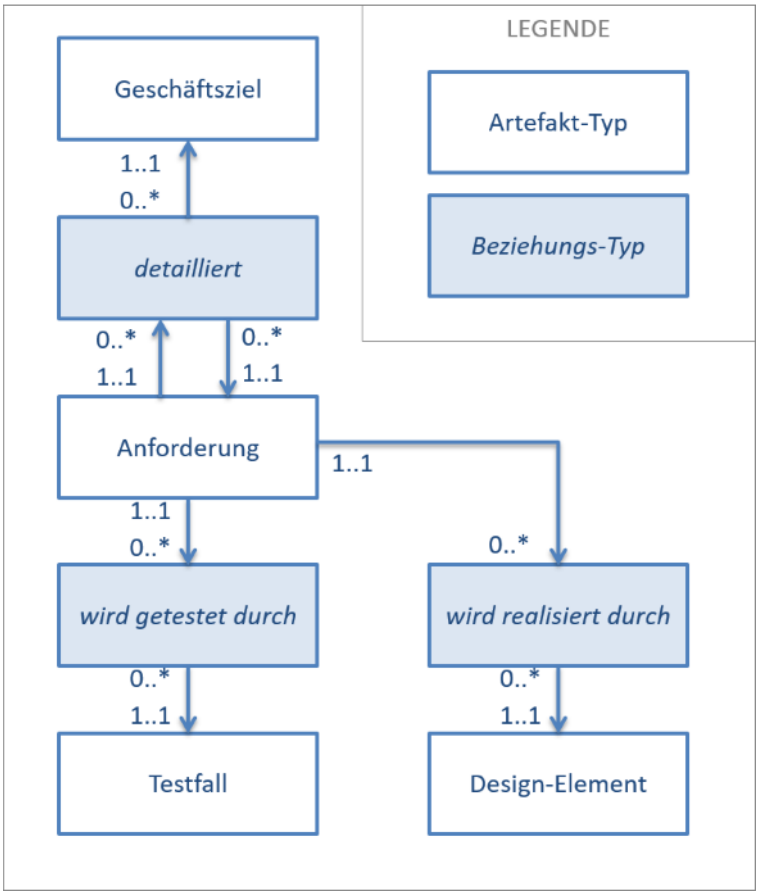


Abbildung 23: Beispiel eines einfachen Verfolgbarkeits-Modells

Mittels dieser Festlegung können Sie für alle Beteiligten ein klares Bild schaffen, zwischen welchen Artefakt-Typen Verfolgbarkeit realisiert werden soll und was die gültigen Beziehungs-Typen sind. Dieses Modell kann und wird in der Praxis deutlich detaillierter und umfangreicher als das aufgeführte Beispiel sein, da oftmals mehr Artefakt-Typen und Beziehungs-Typen existieren.

Hinweis: Zur Erstellung von konkreter Verfolgbarkeit auf Basis eines projektspezifischen Verfolgbarkeits-Modells werden die benötigten Modellelemente und Beziehungen des Verfolgbarkeits-Modells (Informationsmodell) instanziiert und entsprechend der im Verfolgbarkeits-Modell festgelegten Artefakte und Beziehungen dokumentiert.

Das Verfolgbarkeits-Modell kann entweder in das RIM (Anforderungs-Informationsmodell, siehe Kapitel 2) integriert oder als eigenständiges Informationsmodell erstellt werden.

Für ein eigenständiges Verfolgbarkeits-Modell spricht vor allem die Fokussierung auf die relevanten Artefakte im Rahmen der Erstellung einer Verfolgbarkeits-Strategie (z.B. bei der Festlegung von Verantwortlichkeiten). Für ein gemeinsames Informationsmodell spricht allerdings die zentrale Pflege von bspw. Artefakt-Bezeichnungen, neuen Artefakten etc. bei etwaigen Änderungen.

6.6.2 Verwendung eines projektspezifischen Verfolgbarkeits-Modells

Neben der Festlegung von Artefakten und Verfolgbarkeits-Beziehungen, z.B. in einem Informationsmodell, sind für die Umsetzung und Verwendung eines projektspezifischen Verfolgbarkeits-Modells weitere Aspekte zu beachten:

Festlegung der Darstellungsform

Nachdem festgelegt wurde, welche Beziehungen zwischen welchen Artefakten dokumentiert werden sollen, ist zu klären, in welcher Darstellungsform Verfolgbarkeits-Beziehungen dokumentiert werden sollen. Die Auswahl der Darstellungsform für Verfolgbarkeits-Beziehungen wird hierbei in der Regel von der Darstellungsform der Artefakte beeinflusst – sprich, falls die Anforderungs-Artefakte rein textuell erfasst werden, wird man Verfolgbarkeits-Beziehungen vermutlich ebenfalls als textuelle Referenzen bzw. Hyperlinks dokumentieren und nicht über Verfolgbarkeits-Graphen. (vgl. Darstellungsformen, Kapitel 6.4.3).

Bereitstellung einer Erfassungsunterstützung

Die Erfassung von Verfolgbarkeits-Beziehungen zwischen Artefakten stellt einen zusätzlichen Aufwand dar, der in der Regel anderen Stakeholdern zugutekommt (z.B. Projektleiter). Aus diesem Grund ist es sehr hilfreich, wenn die Dokumentation von Verfolgbarkeits-Beziehungen soweit möglich unterstützt wird. Diese Unterstützung kann zum einen durch RM-Werkzeuge – oder aber auch durch bestimmte Eigenprogrammierungen in bspw. Wordmakros unterstützt werden.

Erstellen eines Abgleichs von Werkzeug- zu Projekt-Artefakten

Bei der Verwendung eines RM-Werkzeugs benötigt man in der Regel eine Übersetzung in die vorhandene Terminologie des Werkzeugs. In diesem Schritt werden die Bezeichner der im Modell definierten Artefakte und Beziehungstypen zu den durch das Werkzeug angebotenen Bezeichnern in Beziehung gebracht und eindeutig referenziert. Falls das Werkzeug beispielsweise nur einen Artefakt-Typ „Anforderung" anbietet, das Verfolgbarkeits-Modell allerdings in „Benutzeranforderung" und „Systemanforderung" unterscheidet, dann ist hier eine entsprechende Zuordnung und ggf. eine zusätzliche Attributierung notwendig, die eine spätere Unterscheidung ermöglicht.

Peter Reber sieht sich nun vor der Herausforderung, eine Verfolgbarkeits-Strategie für sein Projekt zu entwickeln. Hierzu hat er für sich und sein Team folgende Festlegung getroffen:

1. Verfolgbarkeitsziel

Die Verwendung von Verfolgbarkeit ist bei Peter Reber durch zwei Dinge getrieben: (a) möchte die Softwareentwicklungseinheit den nächsten CMMI Level erreichen und (b) möchte Peter gern eine Möglichkeit der Nachweisbarkeit haben, dass nur Anforderungen umgesetzt wurden, die durch das Management gewünscht sind.

In der Vergangenheit gab es hierzu öfters unnötige Diskussionen zum Aufwand, da das Management den Eindruck hatte, dass die IT nur Dinge umsetzt, die keiner benötigt, und deshalb immer alles so teuer wird.

2. Nutzungsstrategie

Peter möchte die Verfolgbarkeit im Wesentlichen zu folgenden Zwecken nutzen

1. Belegbarkeit, dass nur das entwickelt wird, was direkt und indirekt durch Geschäfts-Anforderungen für das Projekt begründet ist.
2. Belegbarkeit, dass für alle Anforderungen ein Testfall vorgesehen wurde.
3. Analyse von Änderungsauswirkungen auf vorhandene Anforderungen.

(zu 1) Zur Belegbarkeit, dass nur Anforderungen umgesetzt wurden, die im Projekt vom Management in der Anforderungs-Spezifikation angefordert wurden. Diese Auswertung soll über einen dedizierten Beziehungs-Typ vom Entwicklungs-Artefakt zur Anforderung realisiert werden. Die Auswertung, dass keine Entwicklungs-Artefakte ohne dedizierte Anforderung auf Geschäftsebene bestehen, wird durch den Entwickler erstellt. Diese Darstellung soll alle Entwicklungs-Artefakte mit den dazugehörigen Anforderungs-Artefakten enthalten. Sollte die Auswertung Entwicklungs-Artefakte enthalten, die keiner Anforderung auf Geschäftsebene zugeordnet werden können, so sind diese Artefakte mit dem Designer und Entwickler zu klären, um zu verhindern, dass unnötige und ungewünschte Funktionalität umgesetzt wird.

(zu 2) Zur Überprüfung der Testabdeckung soll eine Auswertung über einen dedizierten Beziehungs-Typ von der Anforderung zum Testartefakt genutzt werden, um sicherzustellen, dass alle Anforderungen durch Testfälle berücksichtigt wurden.

Diese Auswertung wird durch den Requirements Engineer erstellt und soll alle Anforderungs-Artefakte und die dazugehörigen Testfälle enthalten. Falls Anforderungen nicht mit einem Testfall in Beziehung stehen, so hat hier eine Prüfung mit dem Testmanager zu erfolgen.

(zu 3) Um Anforderungsänderungen durch eine gezielte Änderungsanalyse zu unterstützen, sollen drei dedizierte Beziehungs-Typen eingeführt werden (1) zwischen Geschäfts- und Benutzer-Anforderungen, (2) zwischen Benutzer- und System-Anforderungen und (3) zwischen den Anforderungen untereinander, zur Dokumentation logischer und inhaltlicher Abhängigkeiten. Diese Auswertung wird bei eingehenden Änderungen durch den Requirements Manager (Peter selbst) angestoßen. Das Ergebnis soll alle vor- und nachgelagerten Artefakte für eine ausgewählte Menge an Anforderungen darstellen. Hierbei soll zu jedem Anforderungs-Artefakt erkenntlich werden, welche vor- und welche nachgelagerten Artefakte zu der Anforderung in

Bezug stehen. Diese Auswertung hilft anschließend bei der Bewertung, welchen Impact diese Änderung tatsächlich auf die vor- und nachgelagerten Anforderungs-Artefakte hat. Das heißt, hierdurch wird bewertbar, welcher Aufwand (im Sinne von Personentagen und Kosten) durch die Änderung entsteht und ob dabei besondere Schwierigkeiten (z.B. Architekturänderungen etc.) zu erwarten sind.

3. Aufzeichnungs-Strategie

Beziehungs-Typen zwischen Entwicklungs-Artefakten und Anforderungs-Artefakten (System-Anforderungen) sollen jeweils vom Entwickler gepflegt werden, sobald eine Funktionalität für eine Anforderung umgesetzt wird.

Beziehungs-Typen zwischen Testfällen und Anforderungs-Artefakten sind vom Testmanager zu pflegen, sobald ein Testfall für eine Anforderung erstellt wird. Die Verfolgbarkeit ist als bidirektionale Beziehung zu dokumentieren, weshalb der Testmanager eingeschränkten schreibenden Zugriff auf die Anforderungen benötigt, um die Vorwärtsbeziehung zum Testfall zu erfassen.

Beziehungen zwischen den unterschiedlichen Anforderungs-Artefakten sind durch den Requirements Engineer und Business Analysten zu pflegen, sobald eine neue Anforderung erstellt wird, die (a) eine Detaillierung darstellt oder (b) eine logische bzw. inhaltliche Abhängigkeit zu einer bestehenden Anforderung hat und diese in irgendeiner Art und Weise beeinflusst.

4. Erstellung eines Verfolgbarkeits-Modells (Dokumentations-Strategie)

Zur Umsetzung der Verfolgbarkeit über unterschiedliche Dokumentationswerkzeuge und Dokumente sollen textuelle Referenzen mit dafür vorgesehenen Attributen (vgl. Abbildung 16) und Verfolgbarkeits-Matrizen verwendet werden.

Welche Verfolgbarkeits-Beziehungen zwischen welchen Artefakten zu pflegen sind, hat Peter Reber im folgenden Verfolgbarkeits-Modell dokumentiert.

Im Projekt gibt es 3 Ebenen (Klassen) von Anforderungen: Geschäfts-Anforderungen, Benutzer-Anforderungen und System-Anforderungen.

Geschäfts-Anforderungen können dabei entweder durch Benutzer-Anforderungen oder direkt durch System-Anforderungen detailliert werden. Benutzer-Anforderungen werden immer durch mindestens eine System-Anforderung detailliert. Anforderungen untereinander (siehe abstrakte Klasse ‚Anforderung') können über einen *beeinflusst*-Beziehungs-Typ miteinander in Beziehung stehen, falls sie voneinander logisch oder inhaltlich abhängig sind. Weitere Detaillierungen des Inhaltes sind derzeit nicht vorgesehen. Jedes Anforderungs-Artefakt wird durch einen Testfall getestet und schlussendlich wird jede System-Anforderung durch ein Entwicklungs-Artefakt umgesetzt. Es dürfen KEINE Entwicklungs-Artefakte existieren, die keine System-Anforderung realisieren. Es dürfen allerdings Testfälle existieren, die nicht zu

neuen Anforderungen rationiert wurden, d.h. nicht direkt der Überprüfung dieser Anforderung dienen – dies sind beispielsweise Regressionstestfälle.

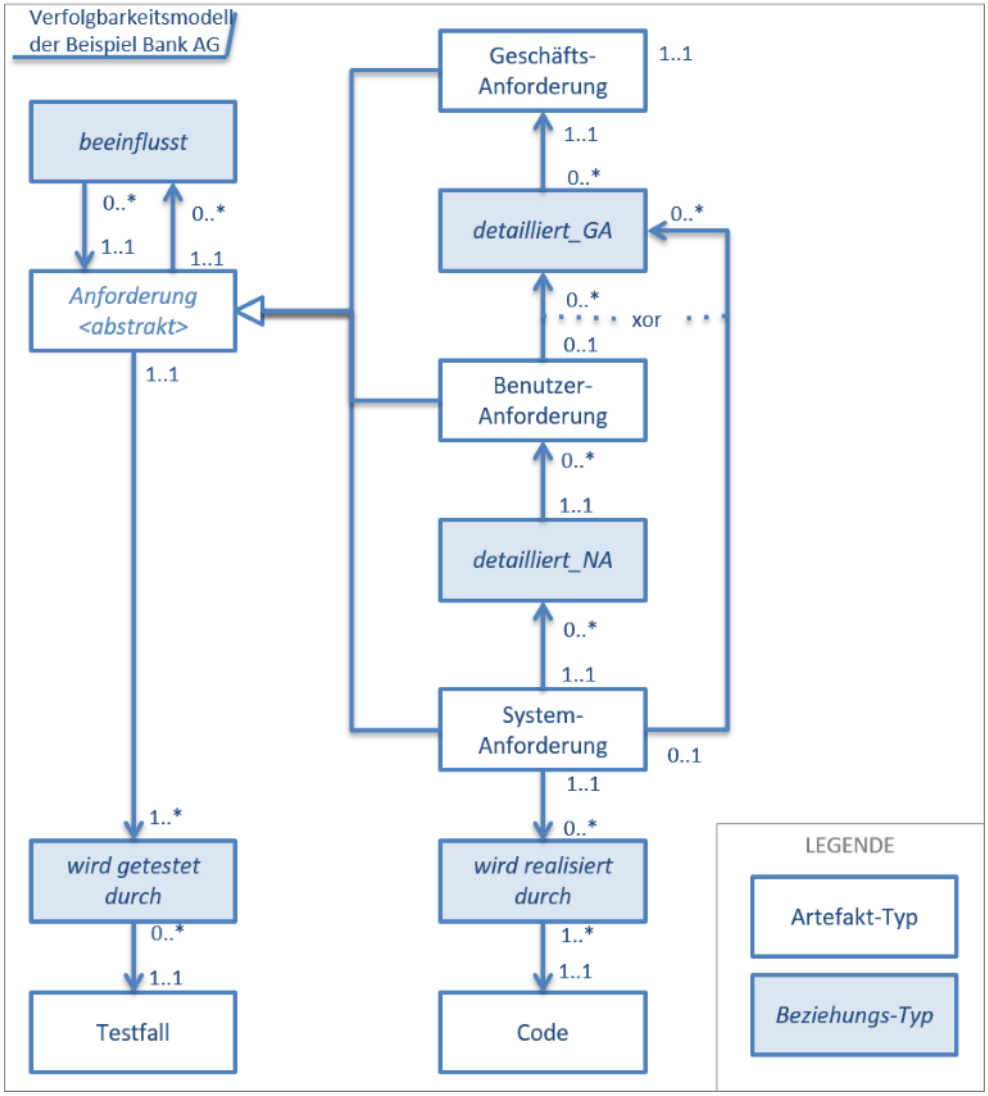


Abbildung 24: Verfolgbarkeits-Modell der Beispiel Bank AG

6.7 Maße zur Bewertung von umgesetzter Verfolgbarkeit

Im vorherigen Abschnitt haben wir betrachtet, wie Sie Verfolgbarkeit dokumentieren können und wie Sie eine Verfolgbarkeits-Strategie für Ihr Projekt aufstellen. – Allerdings muss die von Ihnen eingeführte Verfolgbarkeits-Strategie auch gelebt werden und nicht nur auf dem Papier existieren. Es stellt sich also im Laufe des RM irgendwann die Frage, ob die aufgestellte Verfolgbarkeits-Strategie befolgt wird bzw. wurde und wie vollständig Verfolgbarkeits-Beziehungen zwischen den Artefakten tatsächlich dokumentiert wurden.

Führen Sie hierzu eine Überprüfung durch, um zum einen die Qualität der aktuellen Dokumentation in Bezug auf die Verfolgbarkeit sicherzustellen und zum anderen, um Probleme in der Verfolgbarkeits-Strategie zu identifizieren. Eine Überprüfung der

Verfolgbarkeits-Informationen gibt Aufschluss über die Qualität der aktuellen Dokumentation.

Die folgenden Maße können Sie dabei unterstützen, die Vollständigkeit und Qualität der Verfolgbarkeits-Beziehungen zu prüfen:

- Verhältnis der Anzahl von korrekten Verfolgbarkeits-Beziehungen (z.B. wurde der richtige Beziehungs-Typ verwendet, existiert das referenzierte Artefakt noch) zur Gesamtzahl von Verfolgbarkeits-Beziehungen (Korrektheit)
- Verhältnis der Anzahl der vorhandenen Verfolgbarkeitsbeziehungen zur Gesamtanzahl der benötigten Verfolgbarkeitsbeziehungen (Vollständigkeit)
- Verhältnis der Anzahl von Anforderungen mit Verfolgbarkeitsbeziehung zur Gesamtzahl von Anforderungen (Dichte)
- Verhältnis der Anzahl von Testfällen mit Verfolgbarkeits-Beziehung zu Anforderungen zur Gesamtzahl von Testfällen (Rückwärtsverfolgbarkeit Testfall zu Anforderung)
- Verhältnis der Anzahl von Anforderungen mit Verfolgbarkeits-Beziehung zu einem Testfall im Verhältnis zur Gesamtzahl von Anforderungen
- Verhältnis der Anzahl Dokumente, auf die korrekt verwiesen wird, zur Gesamtzahl von Dokumenten (z.B. existiert das Dokument im angegebenen Verzeichnis)

Hinweis: Beachten Sie, dass speziell die Prüfungen auf Korrektheit nicht vollständig automatisiert werden können. Speziell inhaltliche Prüfungen erfordern eine menschliche Begutachtung. Automatisierte Prüfungen können beispielsweise für das Vorhandensein von Artefakten oder Dokumenten eingesetzt werden. Darüber hinaus wären eingeschränkte Aussagen über die Korrektheit von Beziehungs-Typen möglich, falls die Überprüfung der genutzten Beziehungen auf dem erstellten Verfolgbarkeits-Modell beruht und z.B. zwischen zwei Anforderungen ein Beziehungs-Typ erkannt wird, der nur zwischen Testfällen und Anforderungen gesetzt werden darf.

Eine geringe Anzahl von Verfolgbarkeits-Beziehungen im Verhältnis zur Anzahl der Artefakte lässt vermuten, dass die Beziehungen nicht konsequent und vollständig gepflegt wurden. Andererseits lässt eine geringe Anzahl korrekter Beziehungen im Verhältnis zur Gesamtanzahl von Beziehungen vermuten, dass Beziehungen nachlässig gepflegt wurden oder dass etwaige Änderungen nicht konsequent an allen betroffenen Artefakten durchgeführt wurden.

Jede Abweichung kann unterschiedliche Gründe haben, die es dann zu erörtern gilt. Legen Sie beispielsweise einen Schwellwert je Maß fest, den Sie erreichen wollen. Sollte dieser Schwellwert unterschritten werden, sollten Sie überprüfen, warum.

Prüfen Sie darüber hinaus auf Basis Ihrer Nutzungsstrategie, ob Sie zu geeigneten Ergebnissen kommen. Falls Sie nicht zu Ihren Ergebnissen kommen, kann dies mindestens zwei Gründe haben: (1) die Aufzeichnungs- und Dokumentations-Strategie wurde nur unzureichend verfolgt oder (2) die Dokumentations-Strategie war nicht weitreichend genug, um Ihre Nutzungsstrategie zu erfüllen.

Hinweis: Gehen Sie der Frage nach und überprüfen Sie, ob ihre Verfolgbarkeits-Strategie tatsächlich gelebt wird oder ob sie nur eine Ideologische Festlegung war. Falls Sie herausfinden, dass die Verfolgbarkeits-Strategie nicht oder nur unzureichend verfolgt wird, finden Sie heraus, warum, und versuchen Sie die Hürden (zu kompliziert, nicht verstanden, zu zeitaufwendig, keine Werkzeugunterstützung, etc.) aus dem Weg zu räumen.

Mögliche Gründe für die fehlende bzw. fehlerhafte Dokumentation von Verfolgbarkeit sind:

- Die Notwendigkeit der Dokumentation von Verfolgbarkeit ist innerhalb des Teams nicht bewusst (Nutzen wurde eventuell nicht verstanden).
- Fehlende bzw. nicht verstandene Verfolgbarkeits-Strategie innerhalb des Teams
- Zeitliche Rahmenbedingungen im Projekt erlauben keine Dokumentation von Verfolgbarkeit
- Es liegt innerhalb des Projekt-Teams kein abgestimmtes und akzeptiertes Verfolgbarkeits-Modell vor
- Unzureichende Werkzeugunterstützung bei der Erfassung von Verfolgbarkeits-Beziehungen

6.8 Herausforderungen bei der Verfolgbarkeit zwischen textuellen und modellbasierten Artefakten

Die Verfolgbarkeit zwischen textuellen Artefakten (z.B. funktionale Anforderungen) und modellbasierten Artefakten (z.B. Aktivitäten in UML-Aktivitätsdiagrammen) bzw. zwischen modellbasierten Artefakten untereinander ist nur mit hohem Aufwand zu erreichen und wird aus diesem Grund in der Praxis nicht so häufig gelebt.

Gründe sind in der Regel die fehlende Integration zwischen modellbasierten und textbasierten RE- und RM-Werkzeugen sowie die fehlende eindeutige (zumindest sichtbare) Referenz bei Modellelementen (z.B. Verweis von einer textuellen Anforderung auf eine Klasse in einem UML Klassendiagramm). Natürlich hat diese Klasse irgendwo werkzeugintern oder auch in den Eigenschaften einen eindeutigen Identifikator, dieser ist aber für einen Nutzer schwierig herauszufinden. Auch wenn die heutigen Werkzeuge noch keine vollständige bzw. leistungsfähige Unterstützung bieten, um Modellartefakte mit textuellen Anforderungs-Artefakten zu verknüpfen, gibt es Möglichkeiten, eine Verfolgbarkeit über diese unterschiedlichen Artefakte herzustellen. Mögliche Lösungen bestehen darin, dass man sich an dieser Stelle entweder mit eigenen Identifikatoren in den Bezeichnern hilft oder man über Glossare eindeutige textuelle Bezeichner schafft, auf die verwiesen werden kann. Abbildung 25 zeigt ein Beispiel für eine werkzeugunabhängige Umsetzung von Verfolgbarkeit zwischen einem Use Case Modell und textuellen Anforderungen.

Funktionale Anforderung	Use Case	System Anforderung	GUI Element	Testfall
FR_0012	UC_10	CRM_0011 CRM_0020 DWH_0010 Billing_0020	GUI_0081	TC_0021 TC_0022 TC_0025
FR_0013	UC_11	CRM_0011 CRM_0020		TC_0060 TC_0150
FR_0020	UC_20			

FR_ID	Funktionale Anforderung	Priorität
FR_0012	Der Name des Kunden muss vor der Erfassung gegen die Bestandskundendaten geprüft werden.	Hoch
FR_0013	Die Bankverbindung des Kunden muss gegen den SEPA HUB validiert werden	Hoch
...	...	...

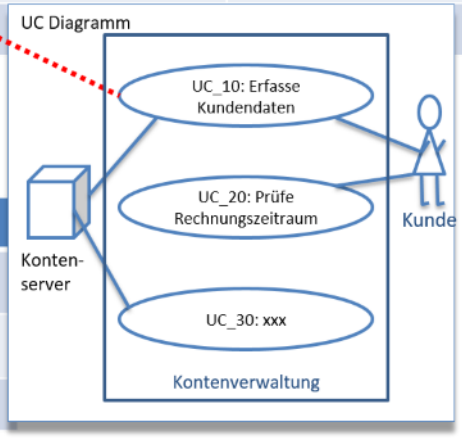
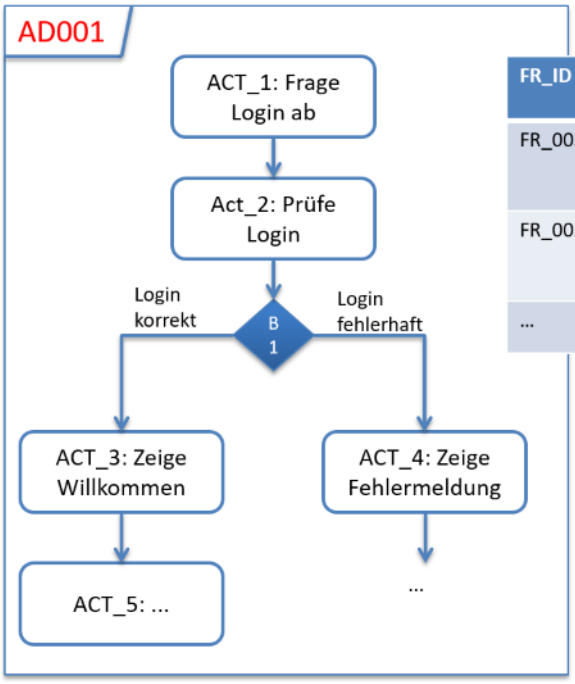


Abbildung 25: Verfolgbarkeit zwischen textuellen und modellbasierten Artefakten

Abbildung 25 zeigt ein Beispiel einer Verfolgbarkeits-Tabelle. Hier sieht man, dass zwischen der funktionalen Anforderung FR_0012 und dem Use Case UC_10 eine Verfolgbarkeits-Beziehung existiert. Die Verfolgbarkeits-Tabelle könnte man grundsätzlich noch um den Beziehungs-Typ erweitern (siehe Kapitel 6.4.3.4).



FR_ID	Funktionale Anforderung	Priorität	Aktivitäts-diagramm
FR_0033	Das System soll die SSL-Loginseite für Kunden anzeigen, damit der Kunde seine Autorisierung durchführen kann	Hoch	AD001, ACT_1
FR_0034	Das System muss die vom Kunden eingegebenen Logindaten gegen die Benutzerdatenbank prüfen.	Hoch	AD001, ACT_2
...	...	...	AD001, B1

Abbildung 26: Verfolgbarkeit zwischen textuellen und modellbasierten Artefakten

Die obige Abbildung 26 zeigt ein weiteres Beispiel für die Verfolgbarkeit zwischen textuellen und modellbasierten Artefakten. In diesem Beispiel existiert eine Verfolgbarkeits-Beziehung von textuellen Anforderungen zu Aktivitäten eines Aktivitätsdiagramms. Speziell bei Aktivitätsdiagrammen kann diese Art der Verfolgbarkeit verwendet werden, um die einzelnen Aktivitäten und Bedingungen besser zu beschreiben. Im Beispiel wurde hierzu jede Aktivität mit einem Identifikator (z.B. ACT_00xx) vor ihrem eigentlichen Namen beschrieben. Die textuelle Anforderung referenziert hier (als unidirektionale Beziehung) über eine textuelle Referenz auf das Aktivitätsdiagramm und die entsprechende Aktivität. Die Darstellung von bidirektionalen Beziehungen ist zwar ebenfalls möglich, erschwert aber in der Regel die Lesbarkeit solcher Modelle, sodass Sie hier abwägen müssen, was wichtiger ist – die bidirektionale Verfolgbarkeit oder die Lesbarkeit der Modelle.

Tipp aus der Praxis: Einige Modellierungs-Werkzeuge unterstützen die Realisierung von Verfolgbarkeit zwischen Modellen und textuellen Artefakten über Wortmuster bzw. Glossare.

6.9 Inhalte für den RMP

Dokumentieren Sie die von Ihnen festgelegte Verfolgbarkeits-Strategie inklusive des Verfolgbarkeits-Modells in Ihrem RMP (vgl. Fallbeispiel in Kapitel 6.6). An dieser Stelle ist weniger wichtig, wie (also in welcher Form) Sie die Dinge in ihren RMP integrieren, als die Tatsache, dass Sie Ihre Überlegungen und Festlegungen, wie Sie in Ihrem Projekt Verfolgbarkeit erfassen, darstellen und nutzen wollen, in Ihrem RMP festhalten. Nur so haben Sie eine Möglichkeit, diese Konzepte vor dem Start mit allen beteiligten Stakeholdern abzustimmen und gemeinsam zu beschließen.

Darüber hinaus ermöglicht die explizite Dokumentation Ihrer Verfolgbarkeits-Strategie in einem RMP, dass nachträglich in das Projekt kommende Teilnehmer sich schnell einarbeiten können und die organisatorischen und methodischen Vorgaben klar nachlesen können.

6.10 Vertiefende Literatur

[GoFi1994] O.C.Z. Gotel and A.C.W Finkelstein: An Analysis of the Requirements Traceability Problem. In Proceedings of IEEE International Conference on Requirements Engineering, 1994.

[HJD2011] E. Hull, K. Jackson, und J. Dick: Requirements Engineering. Springer, 3rd Ed, 2011.

[MGP2009] P. Mäder, O. Gotel, and I. Philippow: Getting Back to Basics: Promoting the Use of a Traceability Information Model in Practice. In: Proceedings of 5th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE2009), Vancouver, Canada, May 2009.

[MJZC2013] P. Mäder, P.L. Jones, Y. Zhang, and J. Cleland-Huang: Strategic Traceability for Safety-Critical Projects. In: IEEE Software, Volume 30, Issue 3, May / June 2013.

[Pohl2010] K. Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. Springer, 2010.

[VanL2009] A. van Lamsweerde: Requirements Engineering – from System Goals to UML Models to Software Specifications. John Wiley and Sons, 2009.

7 Variantenmanagement für Anforderungen

Bevor wir auf das Thema Variantenmanagement{ XE "Varianten-management" } im Kontext des Anforderungsmanagements eingehen und Ihnen beschreiben, wie man Variabilität in Anforderungen dokumentiert, werden wir ein paar Begrifflichkeiten aus der Produktlinien-Entwicklung erläutern.

Zuerst möchten wir die Begrifflichkeiten „Produktfamilie" und „Produktlinie" voneinander trennen, um das Verständnis für Produktlinien zu schärfen.

Definition 7-1:

Produktfamilie{ XE "Produktfamilie" }: Eine Produktfamilie ist eine Menge von komplementär miteinander verbundenen Produkten, welche die Anforderungen eines gemeinsamen Anwendungsbereichs (z.B. Office Suiten) abdecken. Diese Produkte werden in der Regel so gestaltet, dass sie sich gegenseitig ergänzen, vgl. [Gabl2014a].

Definition 7-2:

Produktlinie{ XE "Produktlinie, Produktfamilie" }: Eine Produktlinie fasst verschiedene Varianten eines Produktes zusammen. Die unterschiedlichen Produkte haben in der Regel eine substitutive Beziehung und unterscheiden sich beispielsweise in Funktionsumfang und Preis (z.B. Apple iPhones). Die Produkte einer Produktlinie sind in der Regel so definiert, dass jedes dieser Produkte individuelle Kundenwünsche erfüllt, vgl. [Gabl2014b].

Eine **Produktlinie** umfasst also eine Menge sich unterscheidender konkreter Produkte, die sich alle eine gemeinsame Basis teilen (die sog. Gemeinsamkeiten). Neben diesen Gemeinsamkeiten hat eine Produktlinie einen definierten variablen Anteil, der es ermöglicht, unterschiedliche Produkte zu erstellen (die sogenannte Variabilität der Produktlinie). Über die definierten Gemeinsamkeiten und die Variabilität der Produktlinie können so unterschiedliche Produkte erzeugt werden. Eine Produktlinie kann dabei Hardware und Softwareanteile umfassen, die als Gemeinsamkeiten oder Variabilität definiert wurden und in unterschiedlichen Produkten verwendet werden können.

Ein Anforderungspool ist eine Menge von Anforderungen, die mehr beinhaltet als die Menge der Anforderungen an ein konkretes Produkt. Ein Anforderungspool kann auch Anforderungen beinhalten, die aktuell in keinem Produkt Berücksichtigung finden.

Die Produktlinien-Entwicklung unterscheidet zwei verschiedene Prozesse:

- das *Domänen-Engineering* (englisch: Domain Engineering): Im Domänen-Engineering werden die Gemeinsamkeiten und die Variabilität aus bereits bestehenden Produktvarianten identifiziert und daraus ein Modell der Produktlinie entwickelt.
- das *Applikations-Engineering* (englisch: Application Engineering): Hier wird das Produktlinienmodell produktspezifisch angepasst und so Produktvarianten erzeugt.

Definition 7-3:

Software-Produktlinien (nach [ClNo2007]): "A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."

Variabilität{ XE "Variabilität" } ist ein Begriff, der häufig im Zusammenhang mit Produktlinien zu finden ist (vgl. [PBL2005], [Pohl2010]). Variabilität ermöglicht die Spezifikation (und damit die Umsetzung) unterschiedlicher Produkte durch die Definition von Variations-Punkten und Varianten, ohne dass für jedes Produkt eine eigene Spezifikation zu erstellen ist.

Beispiel Gemeinsamkeiten und Variabilität einer Produktlinie: Der Apple iPad kann als Produktlinie verstanden werden. Gemeinsamkeiten von iPads sind z.B. die Gehäuse, die Displays, die Prozessoren. Das Erstellen konkreter Produktvarianten (z.B. iPad, 64GB, schwarz, mit WiFi und 4G) wird durch die **Variations-Punkte** „unterschiedliche Farben", „unterschiedliche Speichergrößen" etc. erreicht. Hierbei wird die Variabilität durch Variations-Punkte und Varianten beschrieben.

Variations-Punkte{ XE "Variationspunkt" } sind Stellen (z.B. in der Spezifikation), die eine Auswahl konkreter Varianten erlauben bzw. erfordern.

Beispiel für Variations-Punkte: Die Variations-Punkte „iPad Speicher" und „iPad Farbe" stellen Variations-Punkte dar, die durch unterschiedliche Varianten konkretisiert werden.

Definition 7-4:

Variations-Punkt: Ein Variationspunkt beschreibt, wo - an welcher Stelle - innerhalb einer Produktlinie die Anforderungen variieren.

Varianten sind konkrete Ausprägungen von Artefakten (z.B. Anforderungen oder Eigenschaften des Produkts) in Bezug auf einen Variations-Punkt.

Beispiel für Varianten (eines Variations-Punktes): Der Variations-Punkt „iPhone Speicher" hat folgende Varianten: „8GB", „16GB", „32GB", „64GB".

Definition 7-5:

Variante{ XE "Variante" }: Varianten beschreiben zwei oder mehr mögliche (zulässige) Ausprägungen der Anforderungen an einem Variationspunkt (z.B. 7 Zoll, 10 Zoll, 12 Zoll Display).

Wenn wir im Folgenden von Variabilität sprechen, meinen wir immer die Unterschiede zwischen verschiedenen Produkten, also die Varianten, die zum selben Zeitpunkt in einer Produktlinie (aus der unterschiedliche Produkte abgeleitet werden können) gültig sind. Die Veränderung von Anforderungs-Artefakten über die Zeit verstehen wir nicht als Variabilität, sondern als Versionierung (vgl. Kapitel 5).

In der Produktlinien-Entwicklung wird in der Regel zwischen der Domänenentwicklung und der Applikations-Entwicklung unterschieden (vgl. [PBL2005]). Die Domänenentwicklung erstellt die wiederverwendbaren Artefakte als Gemeinsamkeiten und Variabilität der Produktlinie. Die Applikations-Entwicklung erstellt individuelle Produkte auf Basis der wiederverwendbaren Artefakte.

Hierbei enthält jedes Produkt die gesamten Gemeinsamkeiten und eine Auswahl von Varianten. Damit im Rahmen der Wiederverwendung lauffähige und konsistente Produkte abgeleitet werden können, sind speziell für die Auswahl von Varianten entsprechende Auswahl- und Kombinations-Regeln zu berücksichtigen. Hierzu werden in der Domänenentwicklung entsprechende Regeln (Abhängigkeiten) für die Kombinierbarkeit und Ableitung von konkreten Produkten definiert (siehe Kapitel 7.1).

Auch wenn Sie nicht das Ziel haben, eine Produktlinien-Entwicklung zu betreiben, kann der Einsatz von Variabilität für Sie eine interessante Möglichkeit darstellen, um:

- unterschiedliche Varianten in Ihrer Anforderungs-Spezifikation für eine Anforderung zu beschreiben, die das Ziel des Auftraggebers mit unterschiedlicher Güte erfüllen. Oftmals ist sich der Auftraggeber in der Anforderungserhebung noch unsicher, ob er lieber Lösung A (z.B. Navigation mit Sprachführung) oder Lösung B (z.B. Navigation mit Sprach- und Bildführung) möchte. Seine Entscheidung möchte er häufig vom

Aufwand oder der Umsetzungszeit abhängig machen. Aus diesem Grund ist es hier, selbst in der Standardproduktentwicklung, schon mal erforderlich, unterschiedliche Varianten zu spezifizieren.

- optionale Anforderungen innerhalb Ihrer Anforderungs-Spezifikation zu beschreiben, die ggf. als Zusatzanforderung(en) berücksichtigt werden könnten und vor der Realisierung abgeschätzt werden sollen. Die Gründe für solche optionalen Anforderungen sind oft analog den gerade aufgeführten.
- über Variations-Punkte und Varianten unterschiedliche Installations- und Konfigurationsmöglichkeiten einer Applikation gezielt dokumentieren zu können. Hier ist nicht von einer Produktlinie die Rede, sondern eher von Variabilität im Sinne von Konfigurationsmöglichkeiten.
- eine gezielte Wiederverwendung von Anforderungen in ähnlichen Projekten zu ermöglichen.
- ähnliche Produktvarianten zu entwickeln, die entweder vor der Umsetzung oder erst mit der Auslieferung bzw. mit der Lizensierung zu einer konkreten Produktvariante werden.

Tipp aus der Praxis: Diese Notwendigkeit zur Dokumentation von Varianten als Alternativen und Optionen treffen wir in der Realität immer dann an, sobald sich ein Stakeholder nicht konkret entscheiden kann, was er konkret möchte und er seine Entscheidung beispielsweise vom Aufwand abhängig machen möchte. Genau hier haben Sie über den Mechanismus der Variabilität die Möglichkeit, den folgenden Entwicklungsphasen für die Aufwandsschätzung zu signalisieren, dass die als Varianten ausgewiesenen Artefakte gesondert berücksichtigt werden müssen.

7.1 Einsatz von Varianten für Anforderungen

Wie bereits erläutert, beziehen sich Varianten immer auf Variations-Punkte. In einem Anforderungs-Dokument finden sich in der Regel eine Vielzahl von Variations-Punkten und Varianten – selbst, wenn wir uns nicht in der Produktlinien-Entwicklung befinden und die Variabilität nicht explizit dokumentiert wurde.

Variabilität kann implizit oder explizit dokumentiert werden. Bei der impliziten Dokumentation muss man aus der Formulierung der Anforderung heraus erkennen, dass verschiedene Produktausprägungen (Varianten) möglich sind.

Bei der **impliziten** Dokumentation wird beispielsweise durch das Wort „oder" signalisiert, dass verschiedene Produktausprägungen möglich sind (vgl. Abbildung 27). Allerdings ist das Wort „oder" kein zuverlässiger Indikator für einen Variations-Punkt, da dieses ebenfalls sehr häufig in logischen Bedingungen verwendet wird. Auch andere Schlüsselbegriffe wie „sowohl ... als auch" sind hierfür i.d.R. nicht eindeutig genug.

FR_ID	Funktionale Anforderung	Priorität	Status
FR_0010	Zur Höhenmessung soll in der GPS-Wanderuhr entweder eine barometrische Höhenmessung oder eine GPS-basierte Höhenmessung zum Einsatz kommen	Hoch	Abgenommen
...	...	...	...

Abbildung 27: Beispiel einer impliziten Dokumentation von Variabilität

Bei der **expliziten** Dokumentation von Variabilität werden Variations-Punkte und Varianten entweder in die Anforderungs-Spezifikation integriert oder orthogonal zu den Anforderungs-Artefakten (d.h. in einem separaten Modell) erstellt. Im Falle textueller Anforderungen finden sich bei einer integrierten Dokumentation sowohl die Variations-Punkte als auch die möglichen Varianten im Anforderungstext explizit ausgewiesen wieder (Abbildung 28).

FR_ID	Funktionale Anforderung	Priorität	Status
FR_0010	Zur <Variationspunkt> Höhenmessung </Variationspunkt> soll in der GPS-Wanderuhr entweder eine <Variante1> barometrische Höhenmessung </Variante1> oder eine <Variante2>GPS-basierte Höhenmessung </Variante2> zum Einsatz kommen	Hoch	Abgenommen
...	...	...	...

Abbildung 28: Beispiel einer integriert expliziten Dokumentation von Variabilität

Zur expliziten Dokumentation von Variabilität in einem orthogonalen Modell kann beispielsweise folgende Notation verwendet werden (siehe Abbildung 29, vgl. [PBL2005] oder [Pohl2010]).

Im Falle einer orthogonalen Dokumentation bleibt die textuelle Anforderung unangetastet. Die Dokumentation von Variations-Punkt und Varianten erfolgt in einem eigenen Modell, und die Variations-Punkte und Varianten werden zu den dazugehörigen Anforderungs-Artefakten in Beziehung gesetzt, z.B. über eine Verfolgbarkeits-Beziehung (siehe Kapitel 6).

Für die spätere Ableitung von konkreten Produkten im Rahmen der Applikations-Entwicklung, ist bereits bei der Dokumentation von Variabilität zu berücksichtigen, dass nicht alle Varianten frei miteinander kombiniert werden können. Es gibt hier klare Regeln, welche Varianten an einem Variations-Punkt kombiniert werden dürfen bzw. müssen und welche Varianten über Variations-Punkte hinweg kombiniert werden dürfen bzw. müssen, und welche nicht.

Das orthogonale Modell (Abbildung 29) gibt hier beispielsweise schon an, welche Regeln bei der Auswahl der Varianten „barometrische Höhenmessung" und „GPS-basierte

Höhenmessung“ zu beachten sind. In diesem Beispiel darf beispielsweise nur eine der beiden Varianten für ein konkretes Produkt ausgewählt werden.

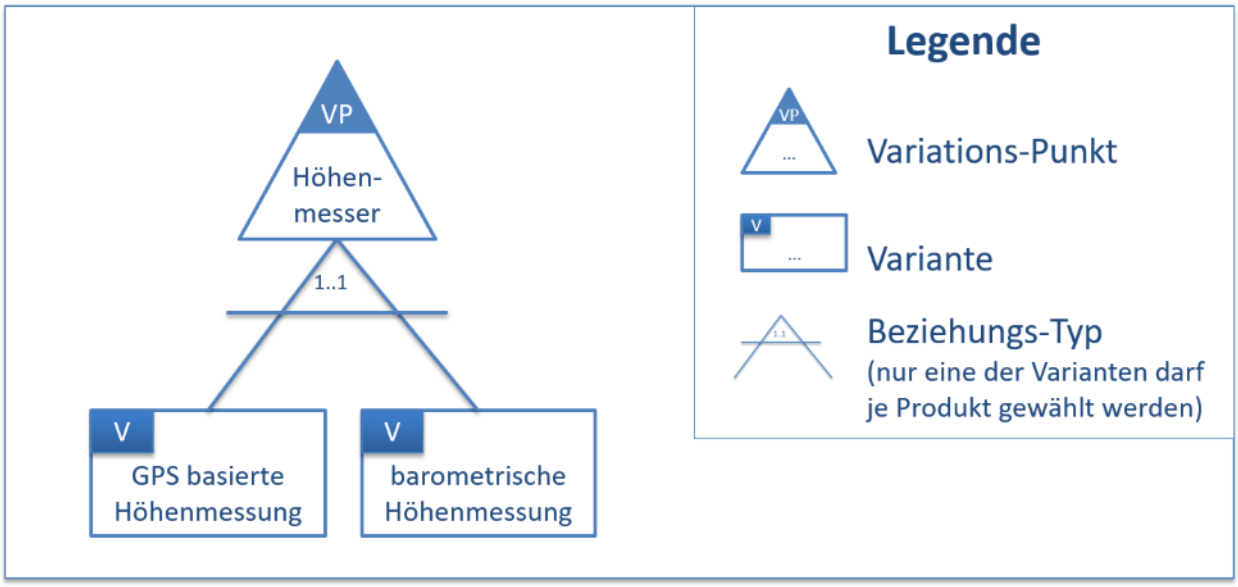


Abbildung 29: Beispiel explizite Dokumentation von Variabilität als orthogonales Modell

Das folgende Variabilitäts-Modell (vgl. [PBL2005]) beschreibt, welche Abhängigkeiten zwischen Variations-Punkten und Varianten existieren können. Die Variabilitätsabhängigkeit beschreibt, wie die Varianten eines Variations-Punktes miteinander kombiniert werden können. Hierzu gibt es folgende Beziehungs-Typen:

- **Alternative Beziehungen**: Um auszudrücken, dass an einem Variations-Punkt entweder Variante_1 (GPS-basierte Höhenmessung) oder Variante_2 (barometrische Höhenmessung) gewählt werden muss (vgl. Abbildung 29).
- **Optionale Beziehungen**: Um auszudrücken, dass eine Variante an einem Variations-Punkt gewählt werden darf, z.B. Speicherung der zurückgelegten Höhenmeter.
- **Pflichtbeziehungen**: Um auszudrücken, dass eine Variante an einem Variations-Punkt gewählt werden muss (das heißt ein Pflichtbestandteil an diesem Variations-Punkt ist), z.B. Einstellbarkeit metrisches und englisches Maßsystem zur Höhenmessung.

Die Beziehungsabhängigkeit beschreibt, wie Varianten oder Variations-Punkte untereinander kombiniert werden können. Hierzu gibt es folgende geläufige Beziehungs-Typen:

- **Benötigt**: Um auszudrücken, dass beispielsweise die Auswahl einer Variante eine andere Variante erfordert, um in einem konkreten Produkt realisiert werden zu können. So benötigt die Variante zur barometrischen Höhenmessung beispielsweise einen Luftdrucksensor zusätzlich zum GPS Modul.
- **Schließt-aus**: Um auszudrücken, dass beispielsweise eine Variante durch die Auswahl einer anderen Variante ausgeschlossen wird, da sich die Varianten für ein Produkt gegenseitig ausschließen. So schließt beispielsweise die Auswahl der GPS-basierten Höhenmessung die Auswahl der barometrischen Wettervorhersage aus, da diese wie

die barometrische Höhenmessung nur über einen Luftdrucksensor realisiert werden kann.

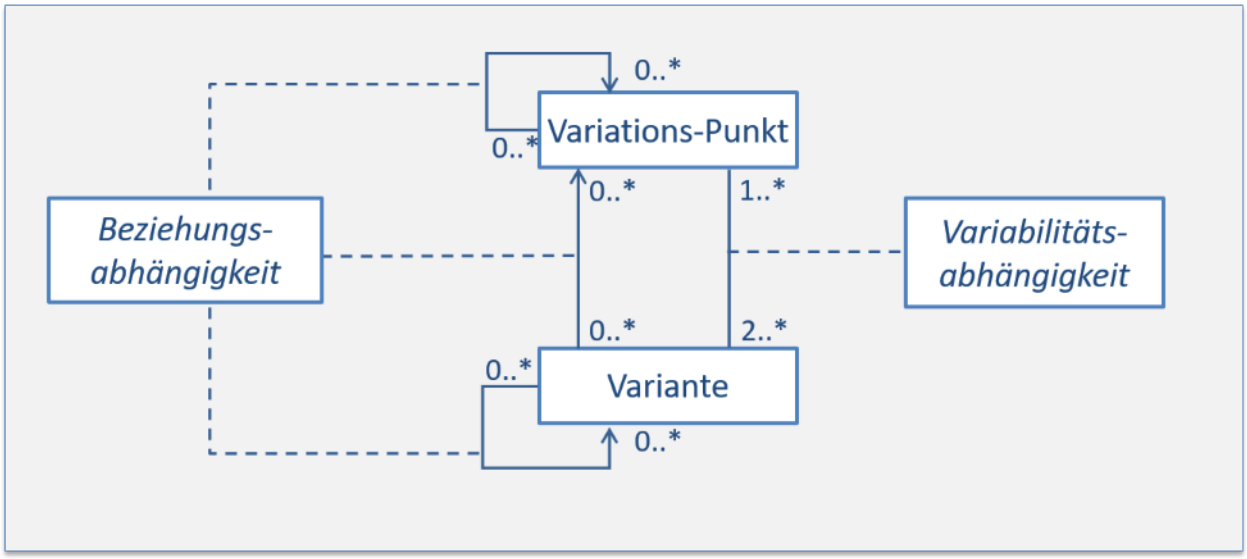


Abbildung 30: Variabilitäts-Modell

Unter Berücksichtigung der Variabilitäts- und Beziehungsabhängigkeiten können unterschiedliche Produkte für die Umsetzung definiert werden. Der Zeitpunkt, an dem die Variabilität aufgelöst wird – also der Zeitpunkt, zu dem konkrete Varianten ausgewählt werden müssen, um ein konkretes Produkt zu haben – nennt man Bindezeitpunkt. Nach [CHW1998] kann Variabilität entweder vor der Entwicklung (also vor der Erstellung des Produkts), mit der Realisierung (also bei der Implementierung), bei der Erstellung der Software (also mit der Kompilation), mit der Erstinstallation oder sogar erst zur Laufzeit gebunden werden.

Je später Varianten gebunden werden, um ein Produkt zu konkretisieren, umso mehr verschwimmt der Begriff Variabilität mit dem Begriff der Konfiguration. Folgende Beispiele sollen dies verdeutlichen:

- **Bindezeitpunkt einer Variante{** XE "Bindezeitpunkt einer Variante" **} vor der Produktrealisierung:**
  Das heißt, vor der Umsetzung entscheidet sich beispielsweise ein Kunde für eine bestimmte Produktausprägung (Variante). Eine spätere Änderung auf eine andere Variante ist für diese gebundene Variante nicht mehr möglich. Zum Beispiel Kunde möchte eine Wanderuhr mit barometrischer Höhenmessung haben.
- **Bindezeitpunkt einer Variante während der Erstinstallation**:
  Das heißt, mit der Installation bzw. Inbetriebnahme entscheidet sich beispielweise der Kunde für eine bestimmte Produktausprägung (Variante). Eine Änderung der gewählten Variante zur Laufzeit ist hier nicht mehr möglich.
- **Bindezeitpunkt einer Variante während der Laufzeit**:
  Das heißt, zu einem beliebigen Zeitpunkt während der Laufzeit hat beispielsweise der Kunde die Möglichkeit, sich eine bestimmte Produktausprägung (Variante) freischalten zu lassen. Zum Beispiel nachträglicher Erwerb von Funktionalität, die es

einem orientierungslosen Wanderer ermöglicht, mit einer Uhr zurück zum Ausgangspunkt zu navigieren.

- **Bindezeitpunkt während der Laufzeit als Konfiguration**:
  Ähnlich wie beim letzten Aspekt hat beispielsweise ein Kunde während der Laufzeit die Möglichkeit, Änderungen an seinem Produkt vorzunehmen, z.B. Farbauswahl für das Display (monochrom / Farbe), Sprachauswahl (Deutsch, Englisch, Französisch, Portugiesisch).

Die Möglichkeit der Dokumentation von Variations-Punkten und Varianten in Anforderungen können wir uns nicht nur im Rahmen der Produktlinien-Entwicklung zu Nutzen machen. Die Verwendung von Variabilität hilft uns ebenfalls echte Anforderungsvarianten, bei denen sich die Stakeholder noch nicht auf eine konkrete Anforderung verständigen konnten, sauber zu dokumentieren und durch die folgenden Phasen bewerten und schätzen zu lassen. Darüber hinaus können durch Varianten und Variations-Punkte ebenfalls die wählbaren Konfigurationseinstellungen z.B. Änderung der Sprache – die nicht zwingend auf eine Produktlinien-Entwicklung zurückzuführen sind – besser verständlich dokumentieren.

In der Produktlinien-Entwicklung im Allgemeinen und im Requirements Management im Speziellen hat die explizite Dokumentation von Variabilität folgende Vorteile [Pohl2010]:

- **Kommunikation:**
  Die explizite Dokumentation von Variations-Punkten und Varianten unterstützt die Kommunikation mit den Stakeholdern, da hierdurch einfach ersichtlich wird, an welcher Stelle welche Varianten unter welchen Bedingungen ausgewählt werden können.
- **Entscheidungsunterstützung:**
  Die explizite Dokumentation führt einerseits bei der Dokumentation von Variabilität zu bewussteren Entscheidungen, an welchen Stellen Variabilität vorgesehen werden soll. Andererseits unterstützt sie bei der Verwendung von Variabilität, um konkrete Varianten für ein gegebenes Produkt auszuwählen.
- **Verfolgbarkeit**:
  Durch eine explizite Dokumentation von Variabilität inklusive der Beziehungen zu den jeweiligen Anforderungs-Artefakten können die im Rahmen einer Änderung abhängigen Anforderungsvarianten ermittelt und bei Bedarf angepasst werden. Durch die orthogonale Dokumentation von Variabilität bekommen wir also die notwendige Verfolgbarkeit für Anforderungsvarianten.

7.2 Formen expliziter Dokumentation von Varianten und deren Bewertung

Wie bereits eingangs erwähnt, wird Variabilität in der Praxis häufig direkt in den Anforderungen formuliert. Diese Formen setzen die in Kapitel 7.1 eingeführten Konzepte wie Variations-Punkt, Variante, Variabilitätsabhängigkeiten, Beziehungsabhängigkeiten und die Dokumentation von Bindungszeitpunkten sehr unterschiedlich um.

In der Praxis finden sich eine Reihe unterschiedlicher textueller Darstellungsformen, von denen wir in den folgenden Abschnitten folgende Stellvertreter näher betrachten (vgl. [Bout2011]).

- Textuelle Zuordnung von Anforderungen zu konkreten Produkten
- Explizite Zuordnung von Anforderungen zu konkreten Produkten
- Explizite Zuordnung von Anforderungen zu konkreten Produktmerkmalen
- Indirekte Zuordnung von Anforderungen zu Merkmalen zu konkreten Produkten

Abschließend werden wir diese Darstellungsformen in Bezug auf die in Kapitel 7.1 vorgestellten Konzepte zur Variabilität analysieren und Ihnen zusätzliche Kriterien zur Bewertung unterschiedlicher Darstellungsformen für Variabilität vorstellen.

Definition 7-6:

Merkmal: Ein Merkmal ist eine für den Anwender sichtbare Eigenschaft oder Qualität eines Systems.

7.2.1 Textuelle Zuordnung von Anforderungen zu konkreten Produkten

Eine Form Varianten zu dokumentieren, ist die Dokumentation der Varianten in einzelnen Anforderungen mit der textuellen Zuordnung, für welches Produkt die jeweilige Variante gültig ist (Abbildung 31).

FR_ID	Funktionale Anforderung	Priorität	Status
FR_0010	Zur Höhenmessung soll in der GPS-Wanderuhr „Premium“ eine barometrische Höhenmessung zum Einsatz kommen.	Hoch	Abgenommen
FR_0011	Zur Höhenmessung soll in der GPS-Wanderuhr „Basis“ eine GPS-basierte Höhenmessung zum Einsatz kommen.	Hoch	Abgenommen

Abbildung 31: Beispiel textuelle Zuordnung zu konkreten Produkten

In diesem Beispiel sehen wir zwei Anforderungen in leicht unterschiedlichen Ausprägungen. FR_0010 sagt aus, dass für die Premium GPS-Wanderuhr eine barometrische Höhenmessung zum Einsatz kommen soll. FR_0011 sagt aus, dass für die Basis GPS-Wanderuhr eine GPS-basierte Höhenmessung zum Einsatz kommen soll. Durch die Produktnamen „Premium“ und „Basis“ wird in der Anforderungsvariante beschrieben, welche Variante in welchem Produkt zum Einsatz kommen soll. Dies stellt bereits einen großen Mehrwert zur impliziten Darstellung von Variabilität in Anforderungen dar – denken Sie zurück an unser Beispiel in Abbildung 27.

7.2.2 Explizite Zuordnung von Anforderungen zu konkreten Produkten

Eine weitere Möglichkeit, Varianten zu dokumentieren, besteht in der expliziten Zuordnung von Anforderungsvarianten zu konkreten Produkten, vgl. Abbildung 32.

FR_ID	Funktionale Anforderung	Basis-Modell	Premium-Modell	Priorität	Status
FR_0010	Zur Höhenmessung soll eine barometrische Höhenmessung zum Einsatz kommen.	--	X	Hoch	Abgenommen
FR_0011	Zur Höhenmessung soll eine GPS-basierte Höhenmessung zum Einsatz kommen.	X	--	Hoch	Abgenommen

Abbildung 32: Beispiel explizite Zuordnung zu konkreten Produkten

In diesem Beispiel werden die Anforderungsvarianten nicht über eine textuelle Nennung im Anforderungstext, sondern explizit über ein eigenes Attribut dem jeweiligen Produkt zugewiesen. So ist bereits auf einem ersten Blick erkennbar, dass jede der beiden Anforderungsvarianten nur in einem bestimmten Produkt gilt. Hier ist die Zuordnung zu Produkten (Produktvarianten) über je ein Produktattribut dargestellt. Die jeweilige Anforderungsvariante wird mit einem „X" dem jeweiligen Produkt zugeordnet (z.B. FR_0010 zum Produkt Premium-Modell). Selbstverständlich kann die explizite Zuordnung zu Produkten auch in anderer Form erfolgen, z.B. über ein einzelnes Attribut „Produkt" mit den jeweiligen Produkten als Werten des Attributs. Die konkrete Umsetzung ist dabei abhängig von der Anzahl der möglichen Produkte sowie von der Zuordnung der Varianten.

Sind bestimmte Anforderungsvarianten bspw. für mehrere Produkte gültig, so ist die Zuordnung über einzelne Attribute pro Produkt wahrscheinlich besser als über Attributwerte.

7.2.3 Explizite Zuordnung von Anforderungen zu konkreten Produktmerkmalen

In der Realität führt die Zuordnung zu konkreten Produkten oft zu einer großen Anzahl von Produkten. Das liegt daran, dass sich konkrete Produkte oftmals über mehrere Dimensionen bzw. Produktmerkmale definieren, z.B. Segmente (Basis und Premium), Märkte (Europa, USA, Asien), Kundengruppen (Wanderer, Läufer, Radfahrer, Golfer). Wenn wir davon ausgehen, dass alle Ausprägungen frei miteinander kombinierbar sind, kommen wir auf 2 x 3 x 4 = 24 Produkte.

FR_ID	Funktionale Anforderung	Kunden-gruppe	Modell	Priorität	Status
FR_0010	Zur Höhenmessung soll eine barometrische Höhenmessung zum Einsatz kommen.	Wanderer, Läufer	Premium	Hoch	Abgenommen
FR_0011	Zur Höhenmessung soll eine GPS-basierte Höhenmessung zum Einsatz kommen.	Wanderer, Radfahrer, Läufer	Basis	Hoch	Abgenommen
FR_0012	Zur Höhenmessung soll eine GPS-basierte Höhenmessung zum Einsatz kommen.	Golfer	Premium, Basis	Hoch	Abgenommen

Abbildung 33: Explizite Zuordnung zu konkreten Produktmerkmalen

Im Beispiel (Abbildung 33) zeigen wir die Möglichkeit der expliziten Zuordnung von Anforderungsvarianten zu Produktmerkmalen, also z.B. zu Modell und Kundengruppe.

Konkrete Produkte wären beispielsweise die Kombinationen Premium-Wanderer und Premium-Läufer. Diesen beiden Produkten ist die Anforderung FR_0010 „barometrische Höhenmessung" zugeordnet. Für die Kundengruppe Golfer wird hingegen sowohl für das Premium- als auch für das Basismodell nur die „GPS-basierte Höhenmessung" angeboten.

Hinweis zur Zuordnung: Die Anforderungen FR_0011 und FR_0012 beschreiben inhaltlich dieselbe Anforderung. Als Anforderungs-Artefakt unterscheiden sie sich lediglich in der Zuordnung zu Kundengruppe und Modell. Da die Golfer-Uhr unabhängig vom Modell nur über die barometrische Höhenmessung verfügen soll, wurde die Anforderung FR_0011 dupliziert, da eine eindeutige Zuordnung ansonsten nicht möglich gewesen wäre. Hätten wir die Attribute Kundengruppe bei FR_0011 um den Wert „Golfer" und das Attribut Modell um den Wert „Premium" ergänzt, so wäre FR_0011 für ungewollte Produkte (z.B. Läufer-Premium) gültig.

7.2.4 Indirekte Zuordnung von Anforderungen zu Produkten über Merkmale

Eine weitere Möglichkeit, Anforderungsvarianten zu konkreten Produkten zuzuordnen, ist die Zuordnung von Anforderungen zu Merkmalen. Merkmale sind hier besondere Eigenschaften der Anforderungen, welche die Variabilität beschreiben. In dem unten aufgeführten Beispiel (Abbildung 34) ist das „Lederarmband" ein Merkmal der dazugehörigen Anforderung FR_0030. Merkmale abstrahieren hier von der gesamten Anforderung und betrachten im Wesentlichen die für den Benutzer sichtbare Eigenschaft (vgl. Kapitel 7.3).

Im unten gezeigten Beispiel wird die Anforderung einem Merkmal zugeordnet, z.B. FR_0011 dem Merkmal „GPS-basierte Höhenmessung". Diese Merkmale sind beispielsweise auch oft auf Produktverpackungen oder ähnlichem zu finden, um dem potenziellen Kunden zu zeigen, welche Merkmale das Produkt aufweist. Die zweite Tabelle unten in Abbildung 34 zeigt die Zuordnung von Merkmalen zu konkreten Produkten.

So hat beispielsweise die GPS-Wanderuhr Premium eine barometrische Höhenmessung und ein Lederarmband, die GPS-Wanderuhr Basis hingegen eine GPS-basierte Höhenmessung

und ein Textilarmband. Anforderungen, denen kein Merkmal zugewiesen wurde (z.B. FR_0070), gelten für alle abgeleiteten Produkte – sprich, sie gehören zu den gemeinsamen Anforderungen (bzw. Gemeinsamkeiten) über alle Produkte.

FR_ID	Anforderung	Merkmal	...
FR_0010	Zur Höhenmessung soll eine barometrische Höhenmessung zum Einsatz kommen.	barometrische Höhenmessung	...
FR_0011	Zur Höhenmessung soll eine GPS-basierte Höhenmessung zum Einsatz kommen.	GPS-basierte Höhenmessung	...
FR_0030	Als Armband soll ein Lederarmband zum Einsatz kommen.	Lederarmband	...
FR_0031	Als Armband soll ein Kunststoffarmband zum Einsatz kommen.	Kunststoffarmband	...
FR_0032	Als Armband soll ein Textilarmband zum Einsatz kommen.	Textilarmband	...
FR_0070	Die Uhr soll die aktuelle Tageszeit im 24-Stundenformat anzeigen.		---

Produkt	Merkmal
GPS-Wanderuhr Premium	barometrische Höhenmessung + Lederarmband
GPS-Wanderuhr Basis	GPS-basierte Höhenmessung + Textilarmband
GPS-Laufuhr Premium	barometrische Höhenmessung + Kunststoffarmband
GPS-Laufuhr Basis	GPS-basierte Höhenmessung + Kunststoffarmband
...	

Abbildung 34: Beispiel Zuordnung von Anforderungen zu Merkmalen zu Produkt-Konfigurationen

7.2.5 Vergleich der Darstellungsformen

Für den Vergleich der Darstellungsformen werden wir die in Kapitel 7.1 eingeführten Aspekte zur Abbildung von Variabilität nutzen und uns folgende Fragen stellen:

- Werden Variations–Punkte und Varianten unterschieden bzw. sind diese erkennbar?
- Werden Abhängigkeiten (Variabilitätsabhängigkeit, Beziehungsabhängigkeit) für die zulässigen Variantenkombinationen abgebildet bzw. sind diese erkennbar?
- Werden unterschiedliche Bindezeitpunkte für Varianten berücksichtigt?

Die folgende Abbildung (Abbildung 35) stellt die drei oben genannten Aspekte mit den vier vorgestellten Darstellungsformen gegenüber. Die Zeilen stellen die Kriterien dar und die Spalten stellen die Darstellungsformen dar.

	Textuelle Zuordnung von Anforderungen zu konkreten Produkten	Explizite Zuordnung von Anforderungen zu konkreten Produkten	Explizite Zuordnung von Anforderungen zu konkreten Produktmerkmalen	Indirekte Zuordnung von Anforderungen zu Merkmalen zu konkreten Produkten
Variations-Punkte und Varianten	In keinem der vorgestellten Dokumentationsformen werden Variations-Punkte explizit modelliert. Selbst die implizite Herleitung von Variations-Punkten ist nicht einfach. Varianten werden prinzipiell als separate Anforderungen beschrieben.			
Variabilitäts- und Beziehungs-Abhängigkeiten	Abhängigkeiten zwischen Varianten sind weder explizit dokumentiert, noch indirekt erkennbar			Abhängigkeiten zwischen Varianten sind nicht explizit dokumentiert. Allerdings werden gültige Varianten-Kombinationen über die Produktzuordnung dargestellt. Zudem könnte man über die Merkmale darauf schließen, dass sich ähnliche Merkmale ausschließen.
Bindezeitpunkte	Alle vorgestellten Dokumentationsformen beschränken sich nur auf einen Bindezeitpunkt. In unsern Beispielen wurde immer der Bindezeitpunkt „Entwicklung“ betrachtet.			

Abbildung 35: Analyse der Darstellungsformen

Für die Bewertung einer konkret eingesetzten Darstellungsform für Variabilität sind darüber hinaus, für die praktische Anwendung der Darstellungsform in Ihren Projekten, folgende Kriterien von Relevanz [Bout2011]:

- **Vermittelbarkeit**: Wie einfach kann die gewählte Darstellungsform fachfremdem Personal geschult werden?
- **Skalierbarkeit**: Wie leicht kann die gewählte Darstellungsform auch für eine größere Anzahl von Produkten sinnvoll eingesetzt werden?
- **Erweiterbarkeit**: Wie hoch ist der Aufwand, um ein neues Produkt aus bestehenden und neuen Anforderungsvarianten zu konfigurieren?
- **Migrierbarkeit**: Inwieweit können bestehende Anforderungs-Dokumentationen ohne explizite Variabilitätsinformation in Richtung der gewählten Darstellungsform weiterentwickelt werden?
- **Überprüfbarkeit**: Inwieweit können Fehlkonfigurationen in der gewählten Darstellungsform automatisiert identifiziert werden?
- **Vergleichbarkeit**: Inwieweit können die Anforderungen verschiedener Produkte einfach miteinander verglichen werden?
- **Änderbarkeit**: Wie einfach können bestehende Anforderungen für ein einzelnes Produkt geändert werden ohne Auswirkung auf andere Produkte der Produktlinie?

7.3 Merkmals-Modellierung

Anders als die orthogonale Modellierung von Variabilität (vgl. [Pohl2010], [BLP2004]) ist die Merkmals-Modellierung (engl. Feature Modeling{ XE "Merkmals-Modellierung" }{ XE "Merkmals-Modellierung" }) eine integrierte Modellierung von Variabilität, bei der sowohl die gemeinsamen Produktmerkmale als auch die Varianten mit deren Abhängigkeiten in einem Merkmals-Modell beschrieben werden.

Der bekannteste Ansatz zur Merkmals-Modellierung stammt von [KCHN1990] und wurde mit FODA (*Feature Oriented Domain Analysis*) eingeführt. Der ursprüngliche Ansatz der Merkmals-Modelle wurde über die Jahre hinweg leicht modifiziert und weiterentwickelt [KKLK1998], [KLD2002], [SHT2006].

Analog zu unserer Definition 7-6 „Merkmal" lautet die ursprüngliche Definition nach [KCHN1990] zu Feature (= Merkmal).

Definition 7-7:

Feature{ XE "Feature" } [KCHN1990]: *„a feature is a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or system".*

7.3.1 Erstellen von Merkmals-Modellen

In einem Merkmals-Modell werden die gemeinsamen sowie variablen Merkmale einer Produktlinie inklusive ihrer Abhängigkeiten beschrieben. Ein Merkmals-Modell kann tabellarisch oder modellbasiert dokumentiert werden. Typischerweise werden Merkmals-Modelle als grafisches Modell (Merkmals-Diagramm) dargestellt. Merkmals-Diagramme haben ihren Ursprung in Und-/Oder-Bäumen. In Merkmals-Modellen sind Variations-Punkte und Varianten optisch nicht eindeutig voneinander zu unterscheiden (vgl. Abbildung 36). Merkmale sind je nach Betrachtung Eltern-Merkmal oder Kind-Merkmal und aus diesem Grund entweder Variations-Punkt oder Variante. Die untersten Blattelemente können eindeutig als Varianten identifiziert werden. Variations-Punkte hingegen sind alle Nicht-Blatt-Elemente des Baums.

Die Beschreibungselemente eines Merkmals-Diagramms lassen sich in die folgenden drei Kategorien einteilen:

- Basis-Elemente (vgl. FODA [KCHN1990])
- Erweiterte Elemente (vgl. [CzEi2000])
- Kardinalitätsbasierte Elemente

Das folgende Modell beschreibt ein Meta-Modell für die Merkmals-Modellierung. Das Modell zeigt zum einen die Verfeinerungs-Beziehung zwischen Eltern- und Kind-Merkmalen (Basis-Elemente) und zum anderen die Abhängigkeits-Beziehungen zwischen Merkmalen (Erweiterte Elemente).

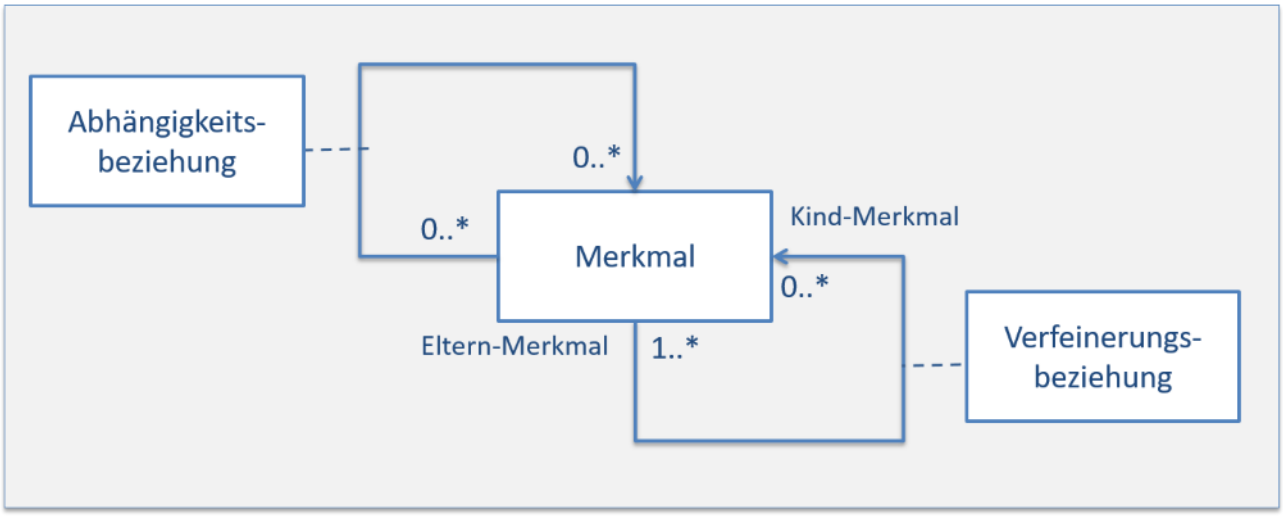


Abbildung 36: Merkmals-Meta-Model

Mit den **Basiselementen** eines Merkmals-Modells werden Eltern-Merkmale sowie deren Kinder beschrieben. Mittels der Verfeinerungs-Beziehung wird beschrieben, welche Merkmale in Konfiguration eines konkreten Produktes zwingend einfließen müssen und was bei der Auswahl von variablen Merkmalen zu berücksichtigen ist. Kinder-Merkmale können zu Eltern-Merkmalen folgende Beziehungen aufweisen:

- **Verpflichtend**: Das Kind-Merkmal ist für ein konkretes Produkt zwingend notwendig.
- **Optional**: Das Kind-Merkmal kann für ein konkretes Produkt ausgewählt werden.
- **Oder**: Mindestens eines der Kind-Merkmale einer Gruppe muss ausgewählt werden, um ein konkretes Produkt erstellen zu können.
- **Alternative**: Genau eines der Kind-Merkmale einer Gruppe muss ausgewählt werden, um ein konkretes Produkt erstellen zu können.

Durch die **erweiternden Elemente** lässt sich (ähnlich wie in Variabilitäts-Modellen) festlegen, welche zusätzlichen Abhängigkeiten bei der Auswahl von Merkmalen zu berücksichtigen sind. Die gängigsten **Abhängigkeits-Beziehungen** sind:

- **Benötigt**: Die Auswahl von Merkmal A impliziert die Auswahl von Merkmal B.
- **Schließt aus**: Die Merkmale A und B können nicht im gleichen Produkt enthalten sein.

Die im Folgenden verwendete Notation für Merkmals-Modelle basiert auf [CzEi2000]. Abbildung 37 beschreibt ein Merkmals-Modell, in welchem die Basiselemente zum Einsatz kommen. Das Modell beschreibt das bereits zuvor genutzte Beispiel „GPS Wanderuhr". Das Modell zeigt das als Eltern-Merkmal dargestellte Produkt „GPS Wanderuhr" sowie drei direkte Kind-Merkmale. Das Merkmal „Wettervorhersage" ist ein optionales Merkmal für die GPS Wanderuhr, ausgedrückt über die Verbindung mit dem leeren Kreis. Die beiden Merkmale „Entfernungsmessung" und „Höhenmessung" sind Pflichtmerkmale für die GPS Wanderuhr. Dies wird durch die Beziehung mit dem ausgefüllten Kreis zwischen Eltern- und Kind-Merkmal ausgedrückt. Das Merkmal „Höhenmessung" hat wiederum Verfeinerungs-Beziehungen zu zwei weiteren Kind-Merkmalen „barometrische Höhenmessung" und „GPS-basierte Höhenmessung".

Diese beiden Kind-Merkmale sind über eine Alternativ-Beziehung mit dem Elternknoten verbunden, sodass nur eine der beiden Merkmale in eine Produkt-Konfiguration einfließen darf. Alternativ-Beziehungs-Typen werden über einen leeren Bogen über alle Kind-Merkmale, von denen nur eines gewählt werden darf, dargestellt. Oder-Beziehungen, die die Auswahl mehrerer Kind-Merkmale zulassen, werden hingegen über einen ausgefüllten Bogen dargestellt.

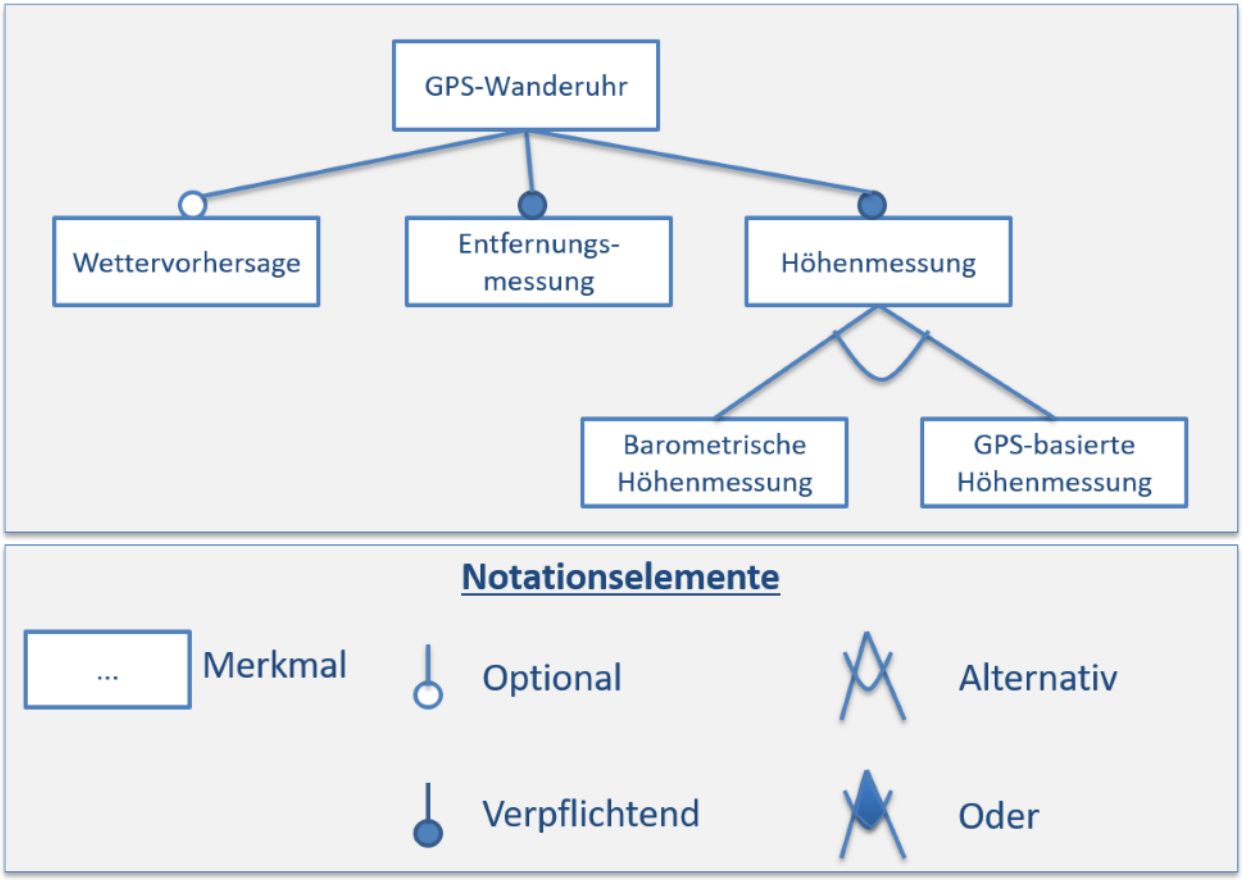


Abbildung 37: Beispiel Merkmals-Modell mit Basiselementen

Abbildung 37 zeigt eine normalisierte Form eines Merkmals-Models. Grundsätzlich können die einfachen Verfeinerungs-Beziehungen „Optional" und „Verpflichtend" mit den Gruppen-Verfeinerungs-Beziehungen „Alternativ" und „Oder" kombiniert werden (vgl. anhand von zwei weiteren Beispielen in Abbildung 38). Auch wenn die drei Modelle auf den ersten Blick unterschiedlich erscheinen, ist die Bedeutung aller Modelle hier identisch. Die Gruppen-Verfeinerungs-Beziehungen „Oder" und „Alternativ" haben hier eine höhere Wertigkeit als die Einfach-Verfeinerungs-Beziehungen „Pflicht" und „Oder".

In Abbildung 38 sieht man anhand der beiden Beispiele, dass die Alternativ-Beziehung eine höhere Priorität hat als „Pflicht" und „Oder" Beziehung zwischen Eltern- und Kind-Merkmal. Hier kann unabhängig von der einfachen Eltern-Kind-Beziehung (Optional oder Pflicht) nur eines der beiden Kind-Merkmale „GPS-basierte" oder „Barometrische" gewählt werden. Die einfache Verfeinerungs-Beziehung am Kind-Merkmal ist hier zu vernachlässigen.

Aus diesem Grund findet man bei Gruppen-Verfeinerungs-Beziehungen in der Regel nur die einfache Verknüpfung zum Kind-Merkmal (vgl. Abbildung 37).

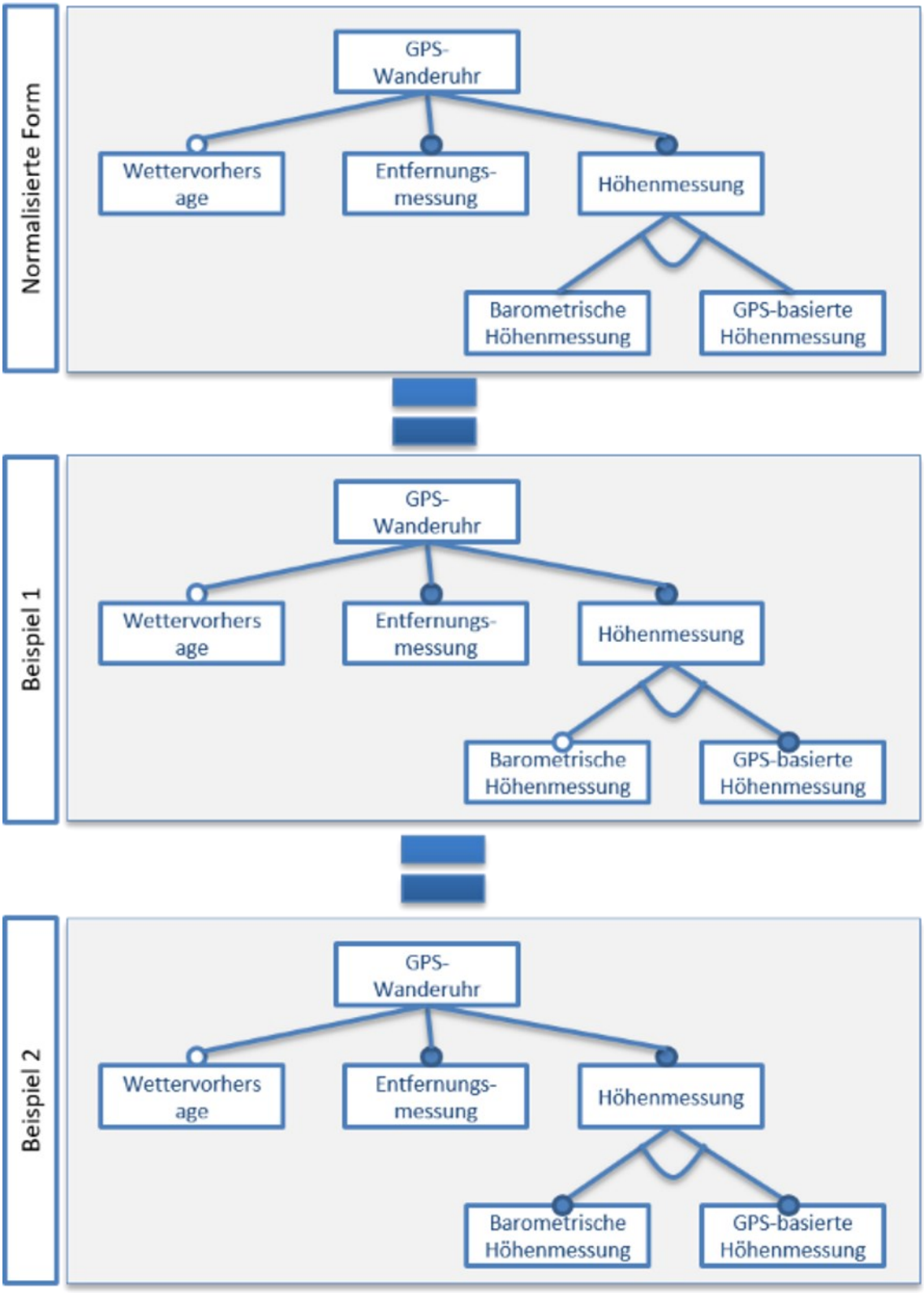


Abbildung 38: Beispiel Kombination "Alternativ" mit "Optional" und "Pflicht" Beziehungen

Merke: die Gruppen-Beziehungs-Typen „Alternativ" und „Oder" haben eine höhere Priorität für die Auswahl, sodass die einfachen Eltern-Kind-Beziehungen „Optional" und „Verpflichtend" für die Kinder innerhalb einer Gruppe nicht berücksichtigt werden.

Aufbauend auf das obige Beispiel zeigt das Beispiel in Abbildung 39, wie die erweiternden Elemente (hier Abhängigkeits-Beziehungen) in einem Merkmals-Modell grafisch dargestellt werden. Hierzu wurde das Modell um eine Benötigt-Beziehung zwischen dem Merkmal „Wettervorhersage" und „barometrische Höhenmessung" erweitert. Dieser Beziehungs-Typ sagt aus, dass bei der Auswahl des optionalen Merkmals „Wettervorhersage" ebenfalls das alternative Merkmal „barometrische Höhenmessung" gewählt werden muss.

Die Benötigt-Beziehung wird durch einen gestrichelten Pfeil mit Pfeilspitze auf das benötigte Merkmalelement dargestellt. Der Beziehungs-Typ „schließt aus" wird über einen gestrichelten Pfeil mit geschlossener Pfeilspitze in Richtung beider Merkmale dargestellt.

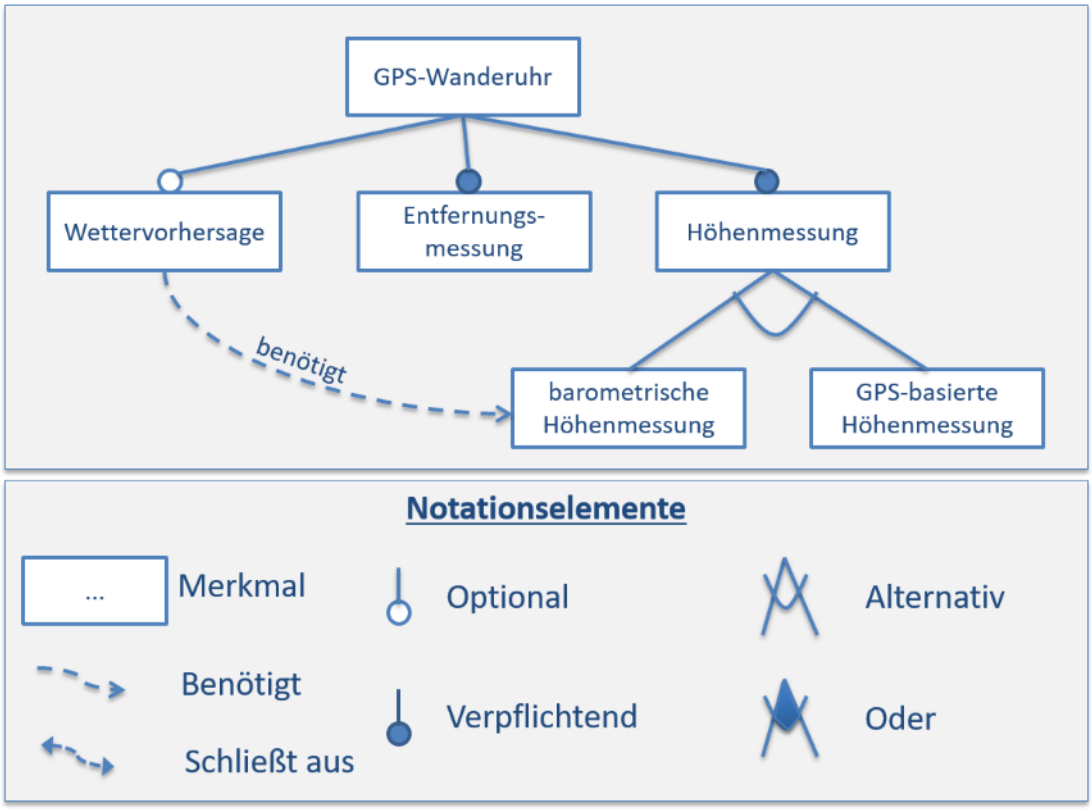


Abbildung 39: Beispiel Merkmals-Modell mit erweiterten Elementen

Mit kardinalitätsbasierten Elementen können die **Verfeinerungs-Beziehungen** der Basis-Elemente weiter präzisiert werden, indem Notationen wie [min, max] an den Eltern-Kind-Beziehungen annotiert werden. Hierdurch kann beispielsweise ausgedrückt werden, dass bei einer Oder-Auswahl nicht alle, sondern maximal zwei Kind-Merkmale gewählt werden dürfen (z.B. nur zwei der 14 Sprachen dürfen ausgewählt werden). Hierzu würde man den Oder-Beziehungs-Typ und den Alternativ-Beziehungs-Typ um die gewünschte Kardinalität erweitern.

7.3.2 Ableitung von Produkt-Konfigurationen aus Merkmals-Modellen

Für die Erstellung konkreter Produkte müssen die variablen Merkmale eines Merkmals-Modells zu einem bestimmten Zeitpunkt gebunden werden (vgl. Kapitel 7.1). Zur Bestimmung, wie viele unterschiedliche Produkte aus einem Merkmals-Modell abgeleitet werden können, ist das Modell umgangssprachlich „auszumultiplizieren". Hierzu werden ausgehend von der Wurzel (also vom obersten Eltern-Merkmal) alle möglichen Produkt-Konfigurationen festgelegt, die auf Basis der Verfeinerungs- und Abhängigkeits-Beziehungen möglich sind.

Beispiel: Produkt-Konfigurationen eines Merkmals-Modells
Die folgenden beiden Beispiele sollen auf Basis der oben eingeführten Merkmals-Modelle darstellen, welche Produkt-Konfigurationen die Modelle zulassen. Das Beispiel in Abbildung 37 erlaubt vier unterschiedliche Produkte:
– **Produkt 1:** GPS-Wanderuhr + Wettervorhersage + Entfernungsmesser + Höhenmessung + barometrische Höhenmessung
– **Produkt 2:** GPS-Wanderuhr + Entfernungsmesser + Höhenmessung + barometrische Höhenmessung
– **Produkt 3:** GPS-Wanderuhr + Wettervorhersage + Entfernungsmesser + Höhenmessung + GPS basierte Höhenmessung
– **Produkt 4:** GPS-Wanderuhr + Entfernungsmesser + Höhenmessung + GPS basierte Höhenmessung

Das Beispiel in Abbildung 39 erlaubt dagegen nur drei unterschiedliche Produkte, da die Benötigt-Beziehung zwischen „Wettervorhersage" und „barometrischer Höhenmessung" die Kombination mit der „GPS-basierten Höhenmessung" – also Produkt 3 – ausschießt.
– **Produkt 1:** GPS-Wanderuhr + Wettervorhersage + Entfernungsmesser + Höhenmessung + barometrische Höhenmessung
– **Produkt 2:** GPS-Wanderuhr + Entfernungsmesser + Höhenmessung + barometrische Höhenmessung
– **Produkt 4:** GPS-Wanderuhr + Entfernungsmesser + Höhenmessung + GPS basierte Höhenmessung

7.3.3 Identifikation von Merkmalen

Merkmale werden in der Regel nicht „auf der grünen Wiese" definiert, sondern sind auf Basis von bestehenden Systemdokumentationen, Anforderungs-Dokumenten etc. zu identifizieren und zu definieren. [BoHo2011] beschreiben einen semi-automatisierbaren Ansatz zur Identifikation von Merkmalen aus bestehenden Spezifikationen in vier Schritten.

- **Schritt 1: Suche nach Substantiven**: Als Ausgangspunkt bei der Identifikation von Merkmalen werden Anforderungstexte nach Substantiven durchsucht.
- **Schritt 2: Normalisierung**: Im folgenden Schritt werden die gefundenen Substantive normalisiert, d.h. sprachlich bereinigt und in die Grundform gebracht (z.B. werden Substantive vom Plural in den Singular überführt).

- **Schritt 3: Entfernen von Duplikaten**: Im dritten Schritt werden aus der Liste der normalisierten Substantive die Duplikate entfernt.
- **Schritt 4: Entfernen von Stopp-Wörtern:** Zum Schluss werden allgemeine Substantive, die mit dem Produkt als solches nichts zu tun haben, aus der Liste entfernt (z.B. Wörter, die sich mit vertraglichen oder allgemeinen Entwicklungs-aspekten im Projekt beschäftigen), sodass als Ergebnis eine Liste mit Merkmalskandidaten herauskommt.

Bei diesem Ansatz geht es in erster Linie darum, Sie dabei zu unterstützen, aus bereits vorhandenen, textuellen Anforderungen mögliche Variations-Punkte und Varianten zu identifizieren, um auf dieser Basis ein Merkmals-Modell zu erstellen. Dies ist vor allem dann hilfreich, falls Sie bereits Dokumente haben, in denen Sie sich regelmäßig – aber ohne explizit dokumentierte Variabilität – der Wiederverwendung bedienen und noch kein Variabilitäts-Modell oder Merkmals-Modell für ihre neue Produktlinie definiert haben.

Beispiel: Identifikation von Merkmalen aus einem Anforderungstext
Eingang für die Analyse soll die folgende Anforderung sein:
R_1020: Als Armband für die GPS-Wanderuhr soll der Kunde zwischen einem Metallarmband, einem Textilarmband oder einem Lederarmband wählen können.
– Mit Schritt 1 (Suche nach Substantiven) würden folgende Substantive identifiziert werden: Armband, GPS-Wanderuhr, Kunde, Metallarmband, Textilarmband, Lederarmband.
– Durch Schritt 2 und 3 wird in unserem Beispiel keine Veränderung stattfinden, da wir nur eine singuläre Anforderung betrachten.
– Mit der Anwendung von Schritt 4 (Entfernen von Stoppwörtern) würde der Begriff Kunde aus der Liste entfernt werden. Kunde kann hier als **Stopp-Wort** identifiziert werden, da dieses Substantiv keine Eigenschaft des Produktes beschreibt – auch wenn der Kunde derjenige ist, der später mal das Produkt kaufen soll.

Ausgang der Analyse sind folgende Merkmalskandidaten: *Armband, GPS-Wanderuhr, Metallarmband, Textilarmband, Lederarmband*.

Basierend auf diesen Substantiv-Listen kann ein Experte in der Regel schnell potenzielle Merkmale bzw. Variations-Punkte und Varianten identifizieren. Ein wesentlicher Nachteil dieser Vorgehensweise ist allerdings, dass insbesondere Variations-Punkte, die textuell nicht explizit erwähnt sind, nicht identifiziert werden können. Zudem müssen die korrekten Beziehungs-Typen zwischen Eltern und Kind-Merkmalen in der Regel aus der Anforderung selbst analysiert werden. Variations-Punkte (also Eltern-Merkmale) und die Beziehungen zu den Kind-Merkmalen können oft identifiziert werden, falls der fachliche Experte bei unterschiedlichen Varianten beharrlich nach dem Grund (also dem „Warum") der unterschiedlichen Varianten fragt. So ergibt sich beispielsweise auf Nachfragen, warum die Uhr mal ein Textilarmband, mal ein Lederarmband und ein anderes Mal ein Metallarmband hat, der Grund, dass unterschiedliche Kunden unterschiedliche Armbänder bevorzugen.

Demnach ist Armband der entsprechende Variations-Punkt (bzw. das Eltern-Merkmal) und die konkreten Armband-Ausprägungen die Varianten (bzw. die Kind-Merkmale).

7.3.4 Werkzeugunterstützung

Soll Variabilität explizit dokumentiert werden, ist dies ohne den Einsatz von speziellen Werkzeugen nur schwerlich möglich. Natürlich können Sie Merkmals-Modelle als Und-/Oder-Bäume mit vorhandenen Modellierungs-Werkzeugen erstellen – nur unterstützen diese in der Regel keine Beziehungs-Typen für Variabilität oder das Ableiten von Produkt-Konfigurationen.

Auf dem Markt gibt es allerdings eine Reihe von Werkzeugen, die es erlauben:

- Merkmals-Modelle zu erstellen
- Produkt-Konfigurationen zu bilden
- Produkt-Konfigurationen auf ihre Zulässigkeit zu prüfen

In der Regel haben diese Werkzeuge Schnittstellen zu anderen Modellierungs- bzw. RM-Werkzeugen, in denen die eigentlichen Entwicklungs-Artefakte (z.B. Anforderungen) liegen. So wird es ermöglicht, dass Variabilitäts-Modelle oder Merkmals-Modelle mit anderen Entwicklungs-Artefakten in Beziehung gesetzt werden, um die Verfolgbarkeit zwischen den unterschiedlichen Modellen herstellen zu können.

7.4 Inhalte für den RMP

Die in diesem Kapitel vorgestellten Aspekte der Variabilitäts-Modellierung können bei Bedarf in den RMP einfließen, falls Sie entweder Varianten darstellen wollen oder Sie sogar eine echte Produktlinie entwickeln. Definieren Sie in Ihrem RMP, wie Sie Variabilität – also Variations-Punkte, Varianten und deren Abhängigkeiten – in Ihren Anforderungen dokumentieren wollen. Dies kann beispielsweise eine Dokumentation im Text, als orthogonales Modell oder Merkmals-Modell sein (z.B. Abbildung 39). Wichtig an dieser Stelle ist, dass Sie explizit festlegen, wie Variabilität zu dokumentieren ist (z.B. durch Merkmals-Modelle), damit Sie dies vor dem Start mit allen beteiligten Stakeholdern abstimmen und beschließen können.

7.5 Vertiefende Literatur

[Bout2011] E. Boutkova: Experience with Variability Management in Requirement Specifications. In: D.E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid (eds): Software Product Lines – 15th International Conference (SPLC), München, 2013, S. 303-312.

[BoHo2011] E. Boutkova and F. Houdek: Semi-automatic identification of features in requirement specifications. In: Proceedings of the 19th International Requirements Engineering Conference, Trento, Italy, September 2011.

[BLP2004] Bühne, S.; Lauenroth, K.; Pohl, K.: Why is it not Sufficient to Model Requirements Variability with Feature Models. In: Aoyama, M.; Houdek, F.; Shigematsu, T. (Hrsg.) Proceedings of Workshop: Automotive Requirements Engineering (AURE04). IEEE Computer Society Press, Los Alamitos 2004.

[CHW1998] J. Coplien, D. Hoffmann, and D. Weiss: Commonality and Variability in Software Engineering. In: IEEE Software, Volume 15, Issue 6, 1998.

[ClNo2007] P. Clements and L. Northrop: Software Product Lines: Practices and Patterns. Addison Wesley, Boston, 6. Auflage, 2007.

[CzEi2000] K. Czarnecki and U.W. Eisenecker: Generative Programming: Methods, Tools, and Applications. Addison Wesley, 2000.

[KCHN1990] C. Kang, S. Cohen, J. Hess, W. Novak, und A. Peterson: Feature-Oriented Domain Analysis (FODA) – Feasibility Study. Software Engineering Institute, 1990.

[KKLK1998] K. Kang, S. Kim, J. Lee, K. Kim, E. Shin und M. Huh, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," Annals of Software Engineering. vol. 5, 1998, pp. 143–168.

[KLD2002] K. Kang, J. Lee, P. Donohoe: Feature-Oriented Product Line Engineering. IEEE Software 19(4): 58–65 (2002).

[PBL2005] K. Pohl, G. Böckle, F. van der Linden: Software Product Line Engineering – Foundations, Principles, and Techniques. Springer, 2005.

[Pohl2010] K. Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. Springer, 2010.

[SHT2006] P.-Y. Schobbens, P. Heymans, J.C. Trigaux: Feature Diagrams: A Survey and a Formal Semantics. In: Proceedings of the 14th International Requirements Engineering Conference (RE'06), September 2006.

8 Berichtswesen im Requirements Management

8.1 Ziele und Nutzen des Berichtswesens im RM

Berichte sind Teil des Projekt- und Unternehmenscontrollings. Sie dienen dazu, dass Informationen über Projekte oder über Organisationseinheiten zusammengetragen und für bestimmte Zielgruppen passend aufbereitet werden, um deren Informationsbedarf zu decken.

Das Berichtswesen{ XE "Berichtswesen" } ist beispielsweise definiert als „die Erstellung und Weiterleitung von funktionsübergreifenden Berichten im Sinne von geordneter Zusammenstellung von Nachrichten ausschließlich für das Management" [Zieg1998]. Eine andere Definition stellt die Erstellung und Zielsetzung des Berichtswesens besonders heraus: „Man kann unter ihm alle Personen, Einrichtungen, Regelungen, Daten und Prozesse verstehen, mit denen Berichte erstellt und weitergegeben werden. Dabei stellen Berichte unter einer übergeordneten Zielsetzung, einem Unterrichtungszweck, zusammengefasste Informationen dar." [Küpp2005].

Berichte sind die konkrete, technische Umsetzung von Sichten, also „ein Auszug aus einem Artefakt, der nur die Inhalte enthält, die momentan interessieren" (siehe: Kapitel 3).

Mit Hilfe von Berichten ist es möglich zu erfahren, wie viel in einem Projekt bereits gearbeitet wurde und in welcher Qualität. Diese Informationen dienen der Projektkontrolle und der Qualitätssicherung. Konkreter unterstützen sie:

- die Kenntnis über den Status des Projektverlaufs,
- Transparenz über den Projektverlauf für das Management und das Team selbst,
- frühzeitiges Erkennen von Abweichungen des Ist-Verlaufs vom Soll-Verlauf,
- wichtige Managemententscheidungen möglichst frühzeitig verlässlich treffen zu können (z.B. ob der Liefertermin verschoben werden muss und wenn ja, um wie viel),
- eine reduzierte, aufs Wesentliche fokussierte Sicht auf die relevanten Daten.

Definition 8-1:

Das *Berichtswesen im Requirements Management (RM)* ist das Sammeln, Auswerten und Darstellen von Informationen über Anforderungen oder den RE-Prozess. Die in Berichten enthaltenen Informationen dienen neben dem reinen Informieren auch als Grundlage für Projektentscheidungen sowie zur Steuerung des RE-Prozesses.

Definition 8-2:

Ein *Bericht* ist ein Dokument, das eine oder mehrere Sichten für einen bestimmten Stakeholder und einen bestimmten Zweck zusammenfasst.

Uns interessiert im Zusammenhang mit dem RM vor allem, welchen Beitrag der Requirements Manager zum Berichtswesen leistet. Insbesondere zeigen wir auf, wie anforderungsbasiertes Projektcontrolling funktionieren kann, d.h. die Beobachtung des Projektfortschritts anhand der Informationen im RM-Werkzeug.

Die Voraussetzung für die Erstellung eines solchen Berichts ist natürlich, dass das RM-Werkzeug die entsprechenden Informationen enthält. Diese wiederum sind meist in Form von Attributen hinterlegt, die genau die für das Berichtswesen benötigte Werteliste besitzen.

Bereits bei der Erstellung des Attributierungsschemas muss darum an die Berichte gedacht werden, die erstellt werden sollen. Der Zweck der Attribute liegt ja ohnehin nicht in ihrer Erhebung, sondern in ihrer Auswertung.

Interessant sind für das Berichtswesen sowohl Produkt- als auch Prozesskennzahlen. Also sowohl Maße für Umfang und Qualität des bisher entstandenen Produktentwicklungsergebnisses als auch für den Status und die Qualität des Entwicklungsprozesses: *„The purpose of the Measurement Process is to collect, analyze, and report data relating to the products developed and processes implemented within the organization, to support effective management of the processes, and to objectively demonstrate the quality of the products."* ISO29148.

Eine Schwierigkeit des Berichtswesens besteht darin, dass diejenigen Personen, welche die nötigen Informationen ins RM-Werkzeug einpflegen sollen, nicht identisch sind mit denen, die diese Informationen nutzen. Daraus folgt nicht nur, dass die Projektmitglieder mangels eigenem Nutzen wenig intrinsische Motivation mitbringen, die Daten einzugeben, sondern sogar ein Interesse daran haben können, den wahren Stand des Projektes nach oben zur kontrollierenden Instanz hin zu beschönigen.

Ziele und Stakeholder des Berichtswesens

Bei der Weiterentwicklung des Online-Bankings kommt es vor allem darauf an, die neuen Releases ordentlich und störungsfrei einzuführen. Es muss frühzeitig (mindestens drei Wochen zuvor) eine Vielzahl an Betroffenen darüber informiert werden, zu welchem Zeitpunkt welche Funktionalitäten eingeführt werden. Dies sind beispielsweise das Management, das Rechenzentrum, die Kundenbetreuer im Call Center und die Mitarbeiter in den Bankfilialen. Zu dem versprochenen Zeitpunkt müssen die jeweiligen Funktionalitäten fehlerfrei funktionieren.

Wichtig im Berichtswesen ist daher eine Statusverfolgung für das nächste Release, das frühzeitig Abweichungen vom Zeitplan feststellt, und außerdem

auch sicherstellt, dass alle ggf. implementierten Fehler bis zur Auslieferung entdeckt und behoben sind.

Hier wiegen Termintreue und Qualität mehr als Kosten und Lieferumfang. Im Zweifel würde eher eine Funktionalität weggelassen, als fehlerhaft ausgeliefert werden. Und es würden eher zusätzliche Kosten in Form von Entwicklertagen investiert, als den Termin zu verschieben.

Gleichzeitig hat auch das interne Controlling, das projektübergreifend den Kostenfluss kontrolliert, ein Auge auf dieses Projekt, und möchte wissen, welche Kosten pro Monat verursacht wurden. Dabei interessiert vor allem, ob das bewilligte Budget unter- oder überschritten wurde bzw. werden wird.

Der Projektleiter ist derjenige, der den Status des Projektes und dessen Qualität mit Hilfe der Berichte auswertet und seine Schlussfolgerungen an die anderen Stakeholder kommuniziert. Für die Controlling-Abteilung, seinen Abteilungsleiter und das Projektteam erstellt er einen ausführlichen Statusbericht.

Für die restlichen Stakeholder hat er einen Newsletter eingerichtet, der wöchentlich in gekürzter Form über den geplanten Releasetermin und die voraussichtlich enthaltenen Funktionalitäten informiert. Es handelt sich dabei um eine selektive Auswahl der Informationen, die sein ausführlicherer Statusbericht enthält.

Die Daten für diese Berichte, die sich auf die Anforderungen und den RE-Prozess beziehen, stammen aus dem RM-Werkzeug, das den Status der Anforderungen über deren gesamten Lebenszyklus verwaltet. Darum ist Peter Reber für den Projektleiter derjenige, der die anforderungsbezogenen Inhalte für den Statusbericht liefert. Weitere Daten stammen aus der Entwicklung und der Qualitätssicherung, die jeweils andere Werkzeuge für die Verwaltung ihrer Inhalte und Erzeugung von Statusinformationen verwenden.

8.2 Etablierung eines Berichtswesens im RM

8.2.1 Schnittstellen

Das RM ist eng verzahnt mit dem Projektmanagement, Produktmanagement und dem Qualitätsmanagement. Somit bestehen diese Schnittstellen auch innerhalb des Berichtswesens. Deshalb ist es sinnvoll, das Berichtswesen dieser drei Bereiche und deren Daten zu koordinieren. Das Projektmanagement wird sicherlich einer der wichtigsten Berichtsempfänger des Berichtswesens über das RM sein, aber auch das Qualitätsmanagement sollte betrachtet werden. Gegebenenfalls ergibt es sogar in einigen Unternehmen Sinn, Berichte themenübergreifend sowohl über das RM als auch das Qualitätsmanagement zu generieren. Dies sollte in den einzelnen Situationen jeweils überprüft werden.

8.2.2 Inhalte eines Berichts

Berichte können formlos in einem E-Mail-Text verschickt werden. Oft gibt es jedoch Vorlagen dafür, sodass jeder Bericht dieselbe Struktur aufweist. Das macht die Berichte einfach und effizient zu lesen und zu erstellen: Dieselbe Information steht immer an derselben Stelle auf der Seite. Besonders praktisch ist es für den Autor, wenn der Bericht automatisch aus dem Werkzeug heraus erzeugt werden kann, in dem ohnehin die nötigen Informationen verwaltet werden, wenn also das RM-Werkzeug Statusberichte erstellt.

Berichte sind wichtige Projektdokumente und als solche nachvollziehbar abzulegen. Hier gelten alle Regeln der guten Dokumentenverwaltung. Am besten haben die Berichtsdateien sprechende und eindeutige Namen, die auch das Datum der Erstellung enthalten in einer Form, welche bei alphanumerischer Sortierung zu einer chronologischen Sortierung führt. Beispielsweise status_20140817.docx. In manchen Projektkrisen, Konflikten und Streitfällen stellen die Berichte wertvolle Dokumentationen dar über den Verlauf des Projektes und den Informationsfluss, d.h.: wer hat wen wann worüber informiert?

Ein Bericht enthält nicht nur eine oder mehrere Sichten sowie Kennzahlen, sondern dokumentiert auch seinen eigenen Erstellungs- und Genehmigungsprozess. Außerdem muss eindeutig sein, worauf sich der Bericht bezieht, z.B. auf welches Projekt und welchen Berichtszeitraum.

Manchmal gibt es verschiedene Berichte über denselben Inhalt, aber mit verschiedenem Zweck, die verschiedene Titel tragen, die meist auch Aufschluss über Zielgruppe und Zweck geben, z.B. der Statusbericht oder das Management Summary.

Natürlich enthält jeder Berichtstyp andere Informationen, d.h. eine andere Sicht. Er soll nur das enthalten, was die Empfänger benötigen, um ihren Informationsbedarf zu decken.

8.2.2.1 Kennzahlen im RM

Kennzahlen sind ein wichtiger Bestandteil von Berichten.

„You can't control what you can't measure" (Du kannst nicht kontrollieren, was du nicht messen kannst) lautet DeMarcos [DeMa1982] viel zitierter Satz. 27 Jahre später [DeMa2009] diskutiert DeMarco, dass nicht jedes Projekt gleich viel Kontrolle braucht, dass sich nicht alles kontrollieren lässt, dass Kontrolle auch nicht alles ist und nicht die wichtigste Managementaufgabe. Menschenführung z.B. sei ebenfalls sehr wichtig. Das stimmt ja auch, aber es bleibt dabei: Messung erleichtert die Kontrolle. Konkrete Zahlen und harte Fakten ergänzen oder korrigieren unsere intuitiven Eindrücke beim ingenieurmäßigen Bearbeiten und Verwalten eines Projektes. Dies gilt genauso speziell für das Verwalten der Anforderungen und deren Änderungen, also das RM.

Ebert [Eber2012] definiert ein Maß als:

- „(1) Eine formale, präzise, reproduzierbare, objektive Zuordnung einer Zahl oder eines Symbols zu einem Objekt, um eine spezifische Eigenschaft zu charakterisieren.
- (2) Mathematisch: Abbildung M eines empirischen Systems C und seiner Relationen R in ein numerisches System M.

- (3) Die Nutzung (Erhebung, Analyse, Bewertung) eines Maßes. Beispiele: Maß für ein Produkt (z.B. Fehler, Dauer, Planabweichung) oder einen Prozess (z.B. Fehlerkosten, Effizienz, Effektivität)."

Man unterscheidet zwischen Produkt- und Prozesskennzahlen.

Definition 8-3:

Die *Produktkennzahl* misst Umfang oder Qualität des zu erstellenden Produktes zu einem bestimmten Zeitpunkt.

Da wir hier die Anforderungs-Brille aufsetzen, messen wir Umfang und Qualität des Produktes anforderungsbezogen, also z.B.: „Welche Anforderungen sind für das nächste Release vorgesehen?" oder: „Wie viele Anforderungen sind bereit zur Auslieferung?".

Definition 8-4:

Die *Prozesskennzahl* misst Fortschritt oder Qualität des Arbeitsprozesses.

Auch hier interessiert uns besonders die Anforderungs-Perspektive, also der Fortschritt des Entwicklungsprozesses in Bezug auf die Anforderungen („Wie viele Anforderungen sind bereits vollständig spezifiziert?") und besonders des RE-Prozesses („Welcher Anteil der aktuell bekannten Anforderungen wurden bereits überprüft?").

Ebert [Eber2012] unterscheidet drei Arten von Maßen im RE:

- „Fortschritt (z.B. Zahl der spezifizierten, realisierten oder getesteten Anforderungen)
- Anforderungsqualität (z.B. Zahl der Fehler aus Anforderungs-Dokumenten)
- Modellsemantik (z.B. Abdeckungsgrad der Anforderungen durch das Analysemodell)"

Den Kennzahltyp „Modellsemantik" könnte man auch als „Verfolgbarkeitsmaß" bezeichnen, da sie die Vollständigkeit der Traceability zwischen zwei verschiedenen Anforderungs-Spezifikationen misst. Sieht man die Anforderungen als Teil des Produkts, handelt es sich auch hierbei um eine Produktkennzahl.

Die ISO 29148 [ISO29148] schreibt über Requirements-Kennzahlen:

„Requirements engineering as a discipline benefits from measuring requirements in both the process and product contexts. More than one measure may be needed to provide the insight into the information needs for the requirements. Practice has consistently proven various useful measures, including:

Requirements volatility – In the process context, requirements volatility can indicate an organization's requirements engineering process will not converge a collection of requirements into a well-formed set. In the product context, a high volatility value can indicate

risk early by stakeholders failing to reach consensus on system requirements, putting significant risk on subsequent activities in the life cycle.

Other useful requirements measures include:

- *Requirements trends*
- *Requirements change rate and backlog*
- *Requirements verification*
- *Requirements validation and*
- *TBD and TBR closure progress per plan."*

TBD steht hierbei für „*to be determined*" (noch festzulegen) und TBR for „*to be resolved*" (noch zu lösen) oder „*to be revised*" (noch zu prüfen / überarbeiten) und markiert damit noch offene Punkte direkt in der Anforderungs-Spezifikation.

Als anforderungsbezogene Kennzahlen zur Statusverfolgung des Projektes empfehlen verschiedene Autoren die folgenden:

- Status der Anforderungen, der über den Zeitverlauf dargestellt werden kann, z.B. die Anzahl oder der Anteil der Anforderungen, die vereinbart sind, entwickelt oder abgeschlossen sind, vgl. Abbildung 42
- Änderungsrate = Anteil der Anforderungen, die sich in einem Zeitraum geändert haben, gemessen am Gesamtumfang des Projektes, misst die Stabilität des Projekts bzw. seiner Anforderungen
- Fehlerraten = Anzahl der Fehler pro Einheit (z.B. Fehler pro 1000 Anforderungen), misst das Ergebnis der Anforderungs-Inspektion oder Software-Tests
- Attributierungsgrad der Anforderungen: Diese Kennzahl misst, ob die Attribute der Anforderungen vollständig befüllt sind, der Zielwert ist 100 %
- Ausmaß der Verlinkung zwischen verschiedenen Artefakten
- Anforderungsabdeckung: Prozentualer Anteil aller Anforderungen, die durch mindestens einen Testfall validiert wurden [SpLi2007]
- Testabdeckung: Kriterium zur Messung der Vollständigkeit der ausgeführten Tests [SpLi2007]
- Velocity = Anzahl an Anforderungen, die bei der iterativen Entwicklung in einer Iteration umgesetzt werden können
- Durchlaufdauer eines Änderungsantrags von der Beantragung bis zur Genehmigung

Um die obigen Kennzahlen zu ermitteln, benötigen wir verdichtende Sichten, die Summen und Teilsummen berechnen oder prozentuale Anteile.

8.2.2.2 Übliche Inhalte von Berichten

Die üblichen Inhalte von regelmäßigen Berichten im Rahmen des RM, z.B. an die Projektleitung, sind die folgenden:

Projektname: Der Bericht muss konkret angeben, auf welches Projekt er sich bezieht. (Handelt es sich um den Bericht einer Organisationseinheit, z.B. Abteilung, dann steht hier statt einem Projektnamen der Name der Abteilung.)

Datum der Berichtserstellung: Die im Bericht dargestellten Inhalte ändern sich täglich oder sogar stündlich. Darum ist es wichtig anzugeben, zu welchem Zeitpunkt die Daten extrahiert wurden, d.h. auf welchem Informationsstand der Bericht beruht.

Versionsnummer: Gibt es mehrere Versionen des Berichts, etwa weil jemand noch eine Ergänzung gemacht hatte, so muss die neue Version eine neue Versionsnummer tragen für die Eindeutigkeit des Berichts und zur besseren Nachvollziehbarkeit von Änderungen.

Berichtszeitraum: Berichte können sich auf Tage, Wochen, Monate, Jahre oder jedes andere Zeitintervall beziehen. Wöchentliche und monatliche Berichte sind am häufigsten, aber in kritischen Projektphasen können Berichte auch täglich oder halbtäglich erstellt werden. Natürlich macht es bei der Interpretation der Berichtsinhalte einen Unterschied, ob sie sich auf das beziehen, was innerhalb von einer Woche oder innerhalb eines Monats geleistet wurde.

Ersteller und Empfänger: Ein Bericht hat einen Ersteller (Autor) und Empfänger (Verteiler). Unter den Empfängern kann man auch noch unterscheiden zwischen den Personen, die ihn nur zur Information erhalten, und denen, die ihn genehmigen müssen. Die Namen dieser Personen sind üblicherweise auf dem Bericht genannt und somit dokumentiert.

Freigabestatus: Benötigt der Bericht eine Freigabe, so ist dieser Status hier zu vermerken. Es kann sein, dass der Bericht in verschiedenen Freigabestatus unterschiedliche Inhalte enthält.
Gesamtstatus: Gleich am Anfang des Berichts möchte – insbesondere der eilige Leser – einen Überblick erhalten, wie kritisch das Projekt ist. Gerade vielbeschäftigte Manager lesen den Bericht nur, wenn das Projekt kritisch ist. Berichte über Projekte, die nach Plan verlaufen, enthalten für den Vorgesetzten keinerlei Informationswert, da seine Unterstützung nicht benötig wird. Beliebt sind Ampelskalen mit folgenden Bedeutungen der Farben:

- **Grün**: Das Projekt verläuft nach Plan. Kein akutes Problem, kein Handlungsbedarf.
- **Gelb**: Das Projekt verläuft nicht nach Plan. Das Projektteam kann voraussichtlich die Probleme selbst lösen. Es müssen jedoch Maßnahmen ergriffen werden.
- **Rot**: Das Projekt ist gefährdet und das Projektteam kann die Probleme nicht (mehr) selbst lösen. Dringende Unterstützung von oben oder außerhalb ist nötig sowie Entscheidung wie beispielsweise eine Verschiebung des Liefertermins, Budgeterhöhung, Bildung einer Task Force.

Zusätzlich können auch noch Status für einzelne Projektteile (z.B. Phasen oder Arbeitspakete) angegeben werden oder für einzelne Aspekte (z.B. Kosten, Termintreue, Fortschritt, Qualität, Risiko). Außerdem interessiert, insbesondere kurz vor dem Liefertermin, welche der geplanten Anforderungen bereits lieferbereit sind und welche nicht. Hier können Sie auch dokumentieren, welcher Meilenstein bereits erreicht wurde.

Earned-Value-Analyse: Die Earned-Value-Analyse (EVA) ist in Anhang C näher beschrieben. Als Grundlage für die EVA gibt der Bericht folgende Kennzahlen des Projektes an:

- Budget (Budget at Completion PC): Das für das gesamte Projekt verfügbare Budget. Das Budget wird in € (oder einer anderen Währung) oder in Zeiteinheiten (Personentagen bzw. Personenmonaten) angegeben. Es können hier beide Angaben gemacht werden oder Sie verwenden durchgängig nur eine der beiden Kennzahlen.

- Geplanter Fertigstellungsgrad: Hier wird in % angegeben, zu welchem Anteil das Projekt zum aktuellen Zeitpunkt fertig gestellt sein sollte. Er wird berechnet als der Quotient aus dem geplanten Arbeitsvolumen und dem Gesamtvolumen des Projektes. Hier kann ebenfalls entweder eine Währung oder eine Zeiteinheit verwendet werden. Arbeitsvolumen und Gesamtvolumen müssen natürlich in derselben Einheit gemessen werden.
- Fertigstellungsgrad: Hier wird in % angegeben, zu welchem Anteil das Projekt zum aktuellen Zeitpunkt tatsächlich fertig gestellt wurde.
- Bisherige Kosten bzw. Aufwand: Hier wird in € oder einer Zeiteinheit angegeben, welche Kosten bisher entstanden sind.
- Kostenindex: Dies ist der Quotient aus bisherigen Kosten und dem Gesamtbudget des Projektes in %.

Mit Hilfe dieser Zahlen wissen Sie, welcher Teil des Ergebnisses bisher fertig gestellt wurde und welcher Anteil des Budgets dafür verbraucht wurde. Daraus lässt sich berechnen, ob das Projekt im Zeitplan liegt und ob es effizient arbeitet, d.h. das erzeugte Ergebnis zum verbrauchten Budget passt. Und hierauf wiederum lassen sich Prognosen erstellen, ob das Projekt pünktlich und innerhalb des Budgets abgeschlossen werden kann. Daraus ergibt sich dann der Status des Projektes. Eine detaillierte Beschreibung zur Earned-Value-Analyse finden Sie im Anhang C. Auch diese Kennzahlen können nach Arbeitspaketen oder Anforderungen aufgeschlüsselt werden. Meist ist dies jedoch nicht nötig.

Weitere Kennzahlen*:* Weitere Kennzahlen können die Qualität der Projektergebnisse beschreiben, u.a. auch der Anforderungen, beispielsweise die Anzahl der in der Inspektion gefundenen Fehler oder der Anteil der Anforderungen, bei denen noch nicht alle Attribute gepflegt sind. Auch weitere Ergebnisse der Qualitätssicherung sind interessant, z.B. die Testabdeckung, Fehlerdichte des Codes und Anzahl der noch offenen Fehler oder der noch offenen schweren Fehler. Weitere Beispiele für Kennzahlen finden Sie in Kapitel 8.2.1.

Die Daten, die für die Berichte benötigt werden, müssen natürlich im Requirements-Information-Model (siehe Kapitel 2) bzw. Attributierungsschema (siehe Kapitel 3) vorhanden sein.

Bewertung und Prognosen: Über die Zahlen hinaus benötigt der Bericht immer auch eine Bewertung durch den Berichtersteller. Die Empfänger können nicht unbedingt beurteilen, ob ein bestimmter Wert für dieses spezielle Projekt oder den aktuellen Zeitpunkt in Ordnung ist oder nicht. Diese Bewertung stellt darum einen wesentlichen Bestandteil des Berichtes dar. Prognosen sind ein Teil dieser Bewertung. Damit nicht jeder Leser selbst die Prognosen berechnen und Schlussfolgerungen ziehen muss, sollte der Berichtautor Prognosen erstellen. Wurde bisher beispielsweise für einen Fertigstellungsgrad von 20 % des Arbeitsvolumens 25 % des Budgets verbraucht, benötigt dies eine Begründung, die wiederum für eine Prognose wichtig ist. Handelt es sich bei den 5 % Kostenüberschreitung um eine einmalige Sache, z.B. unerwartete Kosten, die anfangs einmalig auftraten, und wurde seither nach Plan gearbeitet? Dann kann man auch für den Rest des Projektes hoffen, dass 10 % des Budgets zu 10 % Fertigstellungsgrad führen und die Kosten am Ende 105 % des Budgets betragen? Geht die Kostenüberschreitung jedoch darauf zurück, dass man sich bei der Kostenschätzung vertan hat oder unvorhergesehene Schwierigkeiten die Arbeiten

aufwendiger machen, so steht zu befürchten, dass dies auch für den Rest des Projektes gilt. Dann werden die restlichen 80 % des Projektes wohl weitere 100 % des Budgets verbrauchen, sodass das Projekt am Ende 125 % des geplanten Budgets kosten wird. Auch beim zeitlichen Verzug lässt dessen Ursache darauf schließen, ob diese sich noch hereinholen lässt oder der Endtermin verschoben muss und wie weit.

Besondere Ereignisse: Die Zahlen sagen nichts darüber aus, ob während des Berichtszeitraums etwas Besonderes vorgefallen ist. Dies können Planabweichungen, eingetretene Risiken oder umfangreiche Änderungsanträge sein. Diese sollten hier noch in Textform angegeben werden.

Offene Punkte: Was bleibt offen? Was ist als nächstes zu tun, durch wen und bis wann? Hier stehen normalerweise nur die nächsten Aufgaben, außerplanmäßige Aufgaben und Entscheidungen, die dringend getroffen werden müssen.

Grafische Darstellungen: Zusätzlich zu den trockenen Zahlen sind grafische Darstellungen beliebt, die auf einen Blick eine Übersicht geben. Im Folgenden sind einige Beispiele aufgeführt.

Eine farbige Darstellung des *Status als Ampel*: Besonders, wenn die Status mehrerer Elemente (z.B. Arbeitspakete) dargestellt werden wie in Abbildung 40, dann führt eine grafische Ampel zu einer höheren Übersichtlichkeit. Vergleichen Sie in Abbildung 40 den tabellarischen Teil (a) mit der Ampeldarstellung (b).

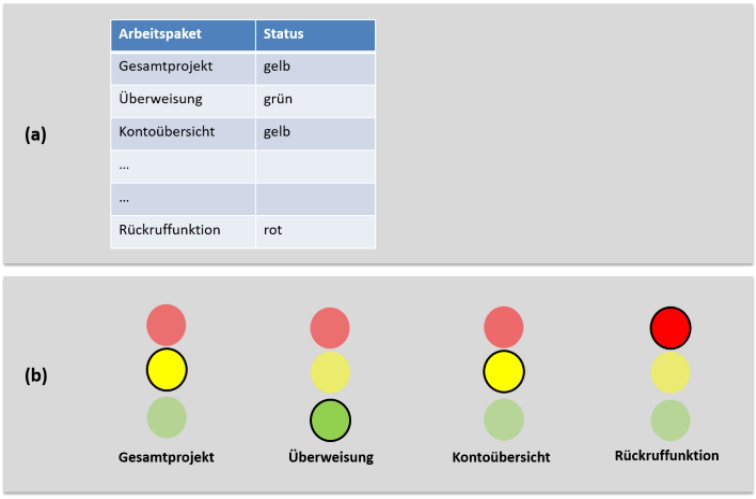


Abbildung 40: Status des Gesamtprojekts und der Anforderungen als Tabelle (a) und als Ampel (b)

Zeit-Diagramme für die Kosten, den Fertigstellungsgrad, die Meilensteintermine und andere Projektkennzahlen: Trägt man diese Kennzahlen wie in Abbildung 41 über die Zeit auf, so erhält man einen guten Überblick über deren zeitliche Entwicklung. Dieser Überblick wiederum erlaubt Prognosen über die zeitliche Entwicklung des Projektes in der Zukunft. Ein Projekt, in dem kontinuierlich 2 % des Projektes pro Woche abgearbeitet wurden, wird ohne radikale Änderungen vermutlich auch weiterhin nur 2 % pro Woche schaffen, allen Hoffnungen auf Wunder zum Trotz. Wenn nun nach der dreizehnten Projektwoche 26 % des Fertigstellungsgrads mit 28 % des Budgets erreicht sind, dann hat das Projekt etwas zu viel Budget verbraucht.

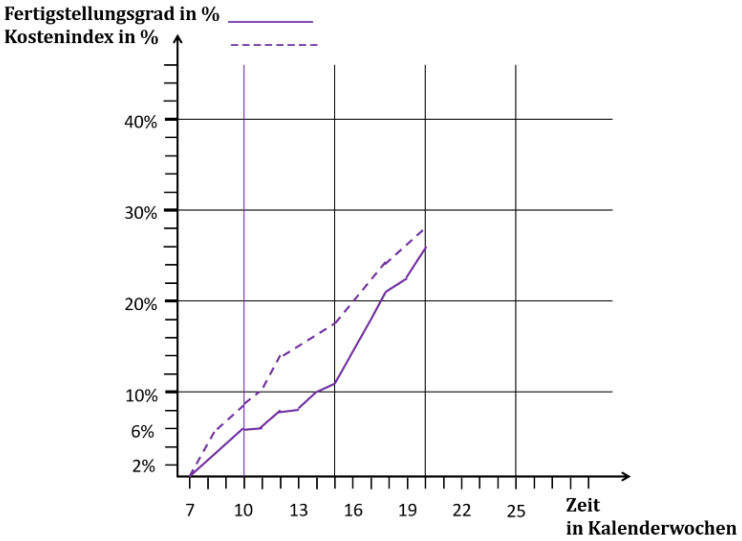


Abbildung 41: Zeit-Diagramm für den Fertigstellungsgrad und Kostenindex eines Projektes.

Abbildung 42 stellt die Status-Entwicklung der Anforderungen über die Zeit auf. Eine solche Grafik lässt sich sehr leicht in einem Tabellenkalkulationsprogramm erstellen, wenn die Daten tabellarisch vorliegen. In dieser Darstellung stecken viele Informationen: Aus der Gesamtzahl der Anforderungen (=Gesamthöhe des Balkens) ist ersichtlich, dass in diesem Projekt in den ersten Wochen sehr viele Anforderungen erhoben wurden und später nur noch wenige neu dazu kamen. Es gibt also kaum „Requirements Creep" (=schleichende Erhöhung des Projektumfangs). Mit der Vereinbarung der Anforderungen wurde in Kalenderwoche 12 begonnen und dann zügig innerhalb weniger Wochen die nötigen Entscheidungen getroffen. Insgesamt zeigt dieses Diagramm einen sehr zufriedenstellenden Projektverlauf.

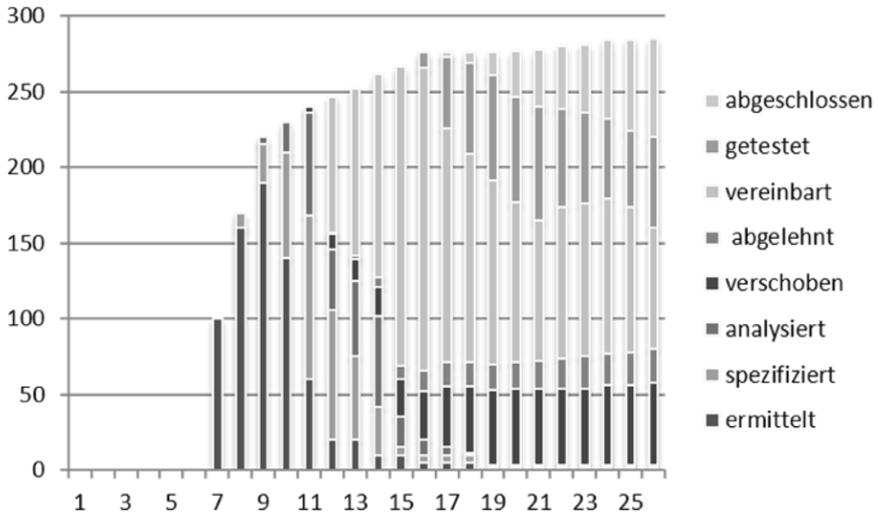


Abbildung 42: Zeit-Diagramm für die Status der Anforderungen. Waagrecht (x-Achse) aufgetragen ist die Zeit in Kalenderwochen, senkrecht (y-Achse) die Anzahl der Anforderungen. Denkbar wäre auch, auf der y-Achse den Aufwand anzuzeigen, d.h. jede Anforderung mit ihrem geschätzten Aufwand zu gewichten.

Inhalte der Statusberichte

Wie bereits gesagt soll es zwei Berichte geben: Den Statusbericht an Controlling, Abteilungsleiter und Projektteam, sowie eine gekürzte Form für den Newsletter an die übrigen Projekt-Stakeholder.

Der Statusbericht soll wöchentlich erstellt werden, kurz vor dem Liefertermin des Releases wenn nötig täglich. Die wichtigsten Projektgrößen sind hierbei:

- Gesamtstatus des Releases: Dieser wird gelb, sobald eine der Anforderungen voraussichtlich doch nicht geliefert werden kann: er wird rot gesetzt, wenn unverzichtbare Anforderungen nicht geliefert werden können und darum der Go-Life-Termin verschoben werden muss
- Bisher im Release verbrauchte Kosten in € und Kostenindex in %
- Fertigstellungsgrad in Bezug auf das nächste geplante Release in %, auch verglichen mit dem geplanten Fertigstellungsgrad
- Status der Anforderungen, als Balkendiagramm aufgetragen über die Zeit, so wie in Abbildung 42
- Anzahl noch offener Fehler als Maß für die Qualität, aufgetragen über die Zeit
- Prognosen über die Termintreue bzw. den voraussichtlichen Liefertermin

Zusätzlich enthält er natürlich die organisatorischen Informationen wie Projektname, Datum, Versionsnummer, Berichtszeitraum, Ersteller und Empfänger.

Der Newsletter enthält – neben den organisatorischen Daten – den Status des Releases, den geplanten Release-Termin, den Fertigstellungsgrad verglichen mit dem geplanten Fertigstellungsgrad und die Prognose für den Liefertermin.

Tipp aus der Praxis: Weniger ist mehr! Halten Sie die Anzahl der zu erstellenden Berichte und der darin enthaltenen Inhalte so gering wie möglich. Jedes Feld erzeugt bei jeder neuen Erstellung Aufwand beim Autor und den Empfängern. Und zu viel unnötige Information kann den Blick auf das Wesentliche sogar trüben.

8.2.3 Tipps für die Entwicklung und Anwendung des Berichtswesens

In der Entwicklung und Anwendung des (anforderungsbasierten) Berichtswesens gibt es einige praktische Herausforderungen:

- **Fokussierung auf das Wesentliche:** Auch wenn die Stakeholder und der Nutzen des Berichtswesens bekannt sind, so besteht die Kunst doch darin, sich auf die

wesentlichen Inhalte zu konzentrieren. Dabei helfen der Berichtsdefinitionsprozess in Kapitel 8.2.4 und die GQM-Methode, die wir in Kapitel 8.2.5 beschreiben.

- **Abstimmung:** Die für den Bericht nötigen Informationen müssen im Requirements-Information-Model (vgl. Kapitel 2) und Attributierungsschema (vgl. Kapitel 3) vorgesehen sein. Da das nachträgliche Ändern von Informationsmodell und Attributierungsschema schwierig ist und das Einführen eines neuen Attributes eine aufwendige Nachpflege der Inhalte verlangt, sollten frühzeitig, noch vor Projekt- oder Arbeitsbeginn, die Datenmodelle des RM geklärt werden. Hilfreich ist es dabei, Referenzmodelle zu verwenden, die bereits zuvor miteinander abgestimmt wurden.
- **Datenerfassung:** Diejenigen, welche die Daten zu erfassen haben, sind nicht dieselben, welche den Informationsbedarf haben und den Bericht erstellen oder lesen. Die Datenerfasser haben also keine intrinsische Motivation dafür, die Daten einzupflegen. Umso wichtiger ist es, dass die Datenerfassung in ihre täglichen Arbeitsprozesse gut integriert ist und klar ist, wer wann welche Daten einzutragen hat.
- **Datenqualität:** Das reine Vorhandensein der Attribute führt noch nicht selbstverständlich dazu, dass auch alle Inhalte gepflegt, aktuell und richtig sind. Während es im Sinne eines effizienten Arbeitsprozesses nicht sinnvoll ist, zu viele Pflichtfelder einzuführen, zumal manche Informationen beim Anlegen einer Anforderung noch nicht vorhanden sind, wäre es für das Berichtswesen wichtig, dass die Attribute gepflegt sind. Fehlende Inhalte führen zu unvollständigen Aussagen der Berichte. In Kapitel 8.2.7 beschreiben wir, wie Sie die Datenqualität sicherstellen oder im Bericht fehlende Daten berücksichtigen.

8.2.4 Berichtsdefinitionsprozess

Für die Definition der anforderungsbasierten Berichte ist ein Abgleich mit dem Attributierungsschema (vgl. Kapitel 3) notwendig. Für diese Aufgabe ist der Requirements Manager verantwortlich. Falls er den Abgleich selbst nicht durchführen kann, delegiert er diese Aufgabe an eine geeignete Person.

Laut ISO 15288 ([ISO15288], 6.3.7.3 a) 1) bis 4)) definiert man ein Mess- bzw. Berichtswesen entlang dieser Schritte:

1. Die Eigenschaften der Organisation beschreiben, die für die Messung relevant sind
2. Informationsbedürfnisse identifizieren und priorisieren
3. Kennzahlen auswählen und dokumentieren, die diese Informationsbedürfnisse erfüllen
4. Vorgehensweisen für die Datensammlung, -analyse und das Berichtswesen definieren

In den folgenden Abschnitten teilen wir diese vier Schritte noch weiter auf.

Eigenschaften der Organisation

Für das Berichtswesen relevante Eigenschaften der Organisation sind insbesondere ihr Organigramm und das Anforderungs-Informationsmodell einschließlich

Attributierungsschema. Auch die Eigenschaften des Projektes gehören hier dazu: Umfang, Zeitplan, Stakeholder.

Informationsbedarf identifizieren: Identifikation von Berichtsempfängern

Alle Stakeholder des Projektes sind potenzielle Berichtsempfänger. Es können jedoch auch Personen Berichtsempfänger sein, die (noch) nicht auf der Stakeholder-Liste stehen.

Informationsbedarf identifizieren: Ziele und/oder Risiken im Projekt ermitteln

Die Berichtsempfänger möchten den Bericht verwenden, um ihre Informations-Ziele zu erreichen. Diese müssen nun – am besten von diesen selbst – ermittelt werden. Diese Ziele bestehen oft darin, Projektrisiken zu verringern oder den Risikoeintritt rechtzeitig zu erfahren, um Maßnahmen einleiten zu können.

Beispiele für Ziele oder Risiken in der Produktentwicklung sind:
- Der geplante Lieferzeitpunkt oder Meilensteintermine müssen eingehalten werden (Ziel).
- Das Budget muss eingehalten werden (Ziel).
- Wichtige geforderte Funktionen können nicht bereitgestellt werden (Risiko).
- Das Budget zur Erhebung der unterschiedlichen Stakeholder-Anforderungen wird nicht ausreichen (Risiko).
- Die geforderte Qualität kann nicht geliefert werden (Risiko).

Informationsbedürfnisse priorisieren

Das Hauptziel des Projektes ist immer noch die Produktentwicklung und nicht die Berichterstellung. Darum muss sich der Bericht auf die wichtigsten Informationsbedürfnisse der wichtigsten Berichtsempfänger konzentrieren.

Es ist gut denkbar, für die wichtigsten Berichtsempfänger einen maßgeschneiderten, optimierten Bericht zu erstellen und den weniger wichtigen Berichtsempfängern denselben Bericht oder einen Auszug daraus zuzusenden, selbst wenn er für ihre Ziele nicht optimal ist, sondern gerade ausreichend.

Bei der Priorisierung der Berichtsempfänger und deren Informationsbedarf spielen Kriterien eine Rolle wie Position des Berichtsempfängers in der Hierarchie und Erfolgskritikalität des Informationsbedarfs, d.h. wie wichtig ist es für den Projekterfolg, dass dieser Informationsbedarf erfüllt wird.

Kennzahlen auswählen, Berichtsinhalte festlegen

In Abschnitt 8.2.2.1 und Abschnitt 8.2.2.2 wurden einige Kennzahlen und Berichtsinhalte vorgeschlagen, die in der Literatur empfohlen und oft erhoben werden. Diese können als Beispiele und (unvollständige) Checkliste dienen. Welche Inhalte aber tatsächlich in einem bestimmten Bericht enthalten sein sollen, hängt vom Informationsbedarf und weiteren Faktoren ab. Wie man vom Informationsbedarf zur Kennzahl kommt, die ihn unterstützt, beschreibt die GQM-Methode (siehe Abschnitt 8.2.5).

Es gibt zwei wichtige Kriterien für die Auswahl der Berichtsinhalte und Kennzahlen:

1. Die Informationsbedürfnisse werden erfüllt.
2. Die Daten sind leicht verfügbar.

Die ISO 29148 ISO29148 rät in Bezug auf Datenverfügbarkeit, sich auf diejenigen Daten und Kennzahlen zu fokussieren, welche ohnehin erhoben werden, und diese als Checkliste für die zu erhebenden Daten zu verwenden: "It is good practice to choose measures for which data are readily available through the life cycle. *The data collection can then be integrated into the requirements related processes to obtain the data and insight on a regular basis as the requirements engineering proceeds. It is also good practice to review the analyzed requirements related measures collectively, looking for predictive trends and projections that can aid risk management.*"

Beide Kriterien – Informationsbedarf und Datenverfügbarkeit – muss man gegeneinander abwägen. Der Bericht darf nur Daten enthalten, die einen vorhandenen Informationsbedarf erfüllen. Natürlich ergibt es keinen Sinn, Daten allein aus dem Grund in einen Bericht hinein zu nehmen, weil sie leicht verfügbar sind, wenn sie keiner braucht. Umgekehrt kann es aber sinnvoll sein, Kennzahlen nicht in den Bericht aufzunehmen, obwohl sie nützlich wären, wenn ihre Erhebung schwierig und unverhältnismäßig aufwändig ist.

Bei der Auswahl der zu berichtenden Daten gehen Sie am besten von den Daten aus, die Sie ohnehin haben, und prüfen, welche davon einen Informationsbedarf erfüllen. Anschließend prüfen Sie, ob jeder Informationsbedarf erfüllt ist und falls nicht, welche Daten eventuell zusätzlich noch erhoben werden müssen.

Denkbar ist, dass in verschiedenen Projektphasen verschiedene Informationsbedürfnisse bestehen oder unterschiedliche Daten verfügbar sind oder sein sollen.

Vorgehensweisen für die Datensammlung, -analyse und das Berichtswesen definieren

Die Datensammlung erfolgt idealerweise während der normalen Projektarbeiten, ist in die normalen Arbeitsabläufe integriert und verursacht keinen Extraaufwand. Die Vorgehensweise zur Datensammlung behandeln wir später in Kapitel 8.2.6 noch.

Für die Vorgehensweise der Datenanalyse und das Berichtswesen ist zu klären:

- **Generierungszyklus**: Wie oft ist der Bericht zu erstellen? Eventuell gibt es auch bestimmte Zeitpunkte, zu denen ein Status des Projekts ermittelt werden soll, z.B. die Meilensteine.
- **Werkzeuge**: Mit Hilfe welcher Hilfsmittel wird der Bericht erstellt?
- **Berichtersteller**: Wer erstellt den Bericht? Grundsätzlich ist es möglich, dass mehrere RM-Prozessbeteiligte Inhalte zu einem Bericht bereitstellen können, wie auch mehrere Prozessbeteiligte Berichte empfangen können.
- **Berichtsform**: In welcher Form wird der Bericht (Format und Vorlage) erstellt und wie verteilt?

Die Berichtserstellung kann manuell durch einen definierten Berichtsersteller, durch den Requirements Manager oder auch automatisch durch ein RM-Werkzeug umgesetzt werden. Die Art der Umsetzung hängt von der Reife der Werkzeugumgebung ab, inwieweit das

Thema RM im Unternehmen etabliert wurde und von der Wichtigkeit des Berichtswesens im Unternehmen. Die automatische Generierung von Berichten stellt insbesondere bei umfangreichen und bei regelmäßig zu generierenden Berichten und umfangreichen Daten eine besondere Rolle dar, da dadurch der Generierungsaufwand für die Berichte und die Fehlerwahrscheinlichkeiten in den Berichten merklich reduziert werden.

Die Darstellung des Berichtes wird außer durch die Werkzeuge auch durch im jeweiligen Unternehmen vorherrschende Standards, Gewohnheiten und Erwartungen beeinflusst.

8.2.5 Goal-Question-Metric-Methode

Bei der Berichtserstellung ist die „Goal-Question-Metric" (GQM) [BaWe1984], [Basi1992], [BCR] eine mögliche Methode um sicherzustellen, dass keine unnötigen oder – besser gesagt – nur zielführende Kennzahlen für Berichte bzw. Berichtsinhalte definiert werden. Die GQM ist eine systematische Vorgehensweise zur Identifikation solcher Kennzahlen. Hierbei wird eine geeignete Kennzahl durch das Beantworten der folgenden Fragen identifiziert:

- Welches Ziel soll durch die Messung erreicht werden? (Goal)
- Was soll gemessen werden bzw. welche Fragen soll die Messung beantworten? (Question)
- Welche Kennzahl(en) sind in der Lage, die notwendigen Eigenschaften zu beschreiben? (Metric)

Bei der Anwendung der GQM-Methode auf das Berichtswesen starten wir mit den Berichtsempfängern und deren Informationsbedarf (=Ziel). Welche Frage soll der Bericht beantworten und welche Kennzahl ist hierfür geeignet?

GQM für die Termintreue

Ziel: In unserem Beispielprojekt interessiert ganz besonders die Termintreue.

Frage: Wann wird das neue Release produktiv gehen?

Kennzahl: Der voraussichtliche Liefertermin ist die Kennzahl, die hier besonders relevant ist. Ermittelt wird er durch die Earned-Value-Analyse, die wiederum die Erhebung mehrerer weiterer Kennzahlen nötig macht.

Es ist auch denkbar, dass ein Informationsbedarf zu mehreren Fragen führt oder zur Beantwortung einer Frage mehrere Kennzahlen nötig sind.

Besteht beispielsweise das Ziel in einer hohen Kundenzufriedenheit, so entsteht die Kundenzufriedenheit nicht nur durch einen einzigen Faktor, sondern vermutlich durch eine Mischung aus harten und weichen Faktoren. Abbildung 43 zeigt ein komplexeres Beispiel, bei dem die zunächst hergeleiteten Kennzahlen wiederum als Ziel interpretiert und weiter analysiert werden.

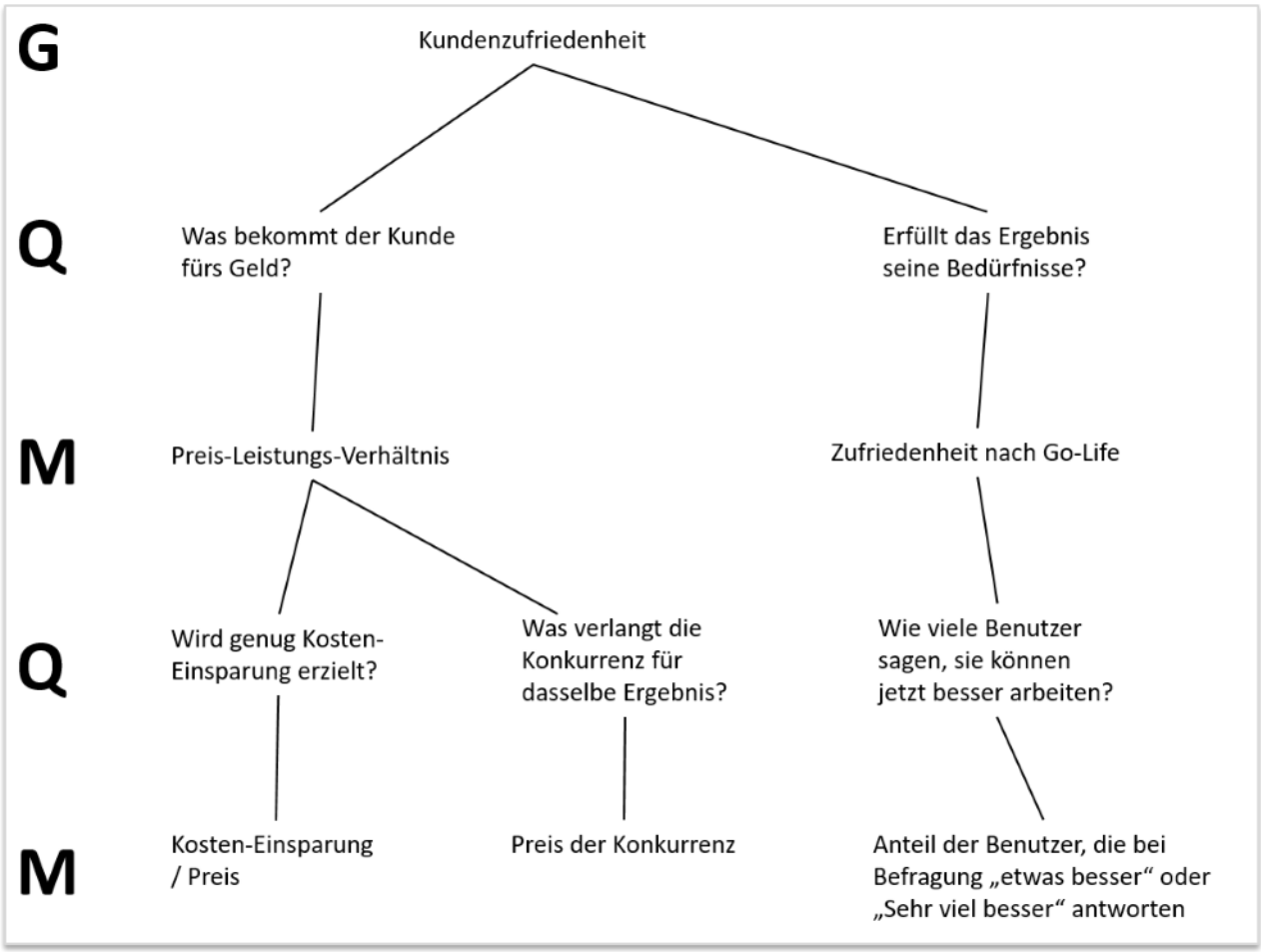


Abbildung 43: Beispiel für die Anwendung der GQM-Methode.

8.2.6 Datenerfassung

Das Durchführen der Datenerfassung für das Berichtswesen umfasst die folgenden Aufgaben ([ISO15288], 6.3.7.3 b):

1. Vorgehensweisen für die Datenerzeugung, -sammlung, -analyse und das Berichtswesen in die relevanten Prozesse integrieren: Es muss also innerhalb der Beschreibung der Arbeitsprozesse festgelegt werden, wer wann welche Daten für den Bericht erfasst. Dies gilt u.a. für den RE-Prozess (vgl. Kapitel 9). Der früheste Zeitpunkt der Datenerfassung ist natürlich der, in dem diese Daten entstehen, z.B. Aufwandsschätzungen oder tatsächliche Aufwände. Es kann aber auch sein, dass Daten zu bestimmten Zeitpunkten aus einem anderen System importiert oder aggregiert werden, weil sie dort bereits vorhanden sind. Auch der Zeitpunkt der Datenanalyse und Berichtserstellung muss definiert werden, um sicherzustellen, dass regelmäßig aktuelle und richtige Informationen bei den Stakeholdern ankommen. Geplant werden muss auch die Prüfung der Datenqualität, die für die anforderungsbezogenen Daten in den Aufgabenbereich des Requirements Managers fällt. Dabei prüft er die Daten auf Vollständigkeit, Plausibilität und Qualität.
2. Daten sammeln, speichern und prüfen: Wie unter Punkt (1) geplant, werden die Daten dann gesammelt und deren Qualität geprüft. Gegebenenfalls kann oder soll der

Requirements Manager sicherstellen, dass dies auch tatsächlich geschieht. Stimmt die Qualität nicht, dann sorgt der Requirements Manager dafür, dass die Wissensinhaber ihr Wissen in das RM-Werkzeug einpflegen und so für die Berichterstellung zur Verfügung stellen.

3. Daten analysieren und Informationen erzeugen: Wie unter Punkt (1) geplant, werden zu den definierten Zeitpunkten die Sichten und Berichte erzeugt.
4. Ergebnisse dokumentieren und an die Benutzer kommunizieren: Die Berichte werden dokumentiert (z.B. als Datei mit Zeitstempel abgelegt) und an die betroffenen Stakeholder verteilt. Die Dokumentation kann hilfreich werden, wenn man später den Verlauf des Projektstatus und den Wissensstand nachvollziehen möchte.

8.2.7 Prüfen der Datenqualität

Der Berichtsempfänger erhält die vorhandenen Informationen in Form von Berichten, um daraus seine Maßnahmen und/oder Entscheidungen abzuleiten. Darum sollte die Datenqualität stimmen, weil sonst Fehlentscheidungen die Folge sein können. Es sollte ein Informationsaustausch der beteiligten Personen stattfinden, um die Qualität des Berichtswesens zu erhöhen. Sowohl der Berichtsersteller als auch der Berichtsempfänger tragen hierfür die Verantwortung.

„Das Informations- und Berichtswesen sollte nicht einseitig von den Mitarbeitern zum Projektleiter verlaufen. Vielmehr sollte der Projektleiter selber sein Wissen und seine Informationen den beteiligten Mitarbeitern zur Verfügung stellen. Informationsdefizite führen nicht nur zu unkoordinierten Tätigkeiten, sondern wirken sich auch negativ auf Mitarbeitermotivation aus" [KuSt2001].

Der Requirements Manager ist dafür verantwortlich, die Datenqualität zu prüfen. Dabei gibt es zwei Kriterien zu untersuchen: die Vollständigkeit und die Qualität.

Es ist relativ leicht, die Vollständigkeit der Daten zu prüfen. Sind beispielsweise alle Attribute mit Namen „Aufwand" eingepflegt, dann sind sie vollständig. Lassen sich durch Filtern oder Sortieren nach diesem Attribut Anforderungen finden, bei denen dieses Attribut leer ist, sind die Daten unvollständig. Natürlich kann es sein, dass es auch in Ordnung ist, wenn einzelne Daten fehlen. Beispielsweise kann eine Anforderung, die sich noch in der Klärung befindet, natürlich keine Werte im Feld „tatsächlicher Aufwand" enthalten, weil noch kein Aufwand angefallen ist. Oder bei der Ermittlung des Fertigstellungsgrads des aktuellen Release sind die Status der Anforderungen, die für spätere Releases zurückgestellt wurden, völlig irrelevant. Darum sind Kriterien für die Vollständigkeit der Daten zu definieren.

Schwieriger ist es, die Qualität der Inhalte zu beurteilen. Auch hierfür sind Kriterien zu definieren. Manchmal stecken diese bereits im Attributierungsschema, das z.B. nicht plausible oder widersprüchliche Attributkombinationen verbieten kann (siehe Kapitel 3). Dann ist die Erfassung nicht plausibler Daten erst gar nicht möglich. Manchmal ist es jedoch technisch nicht machbar, solche Attributkombinationen bei der Erhebung zu verhindern. Dann muss man sie mit Hilfe von passenden Sichten identifizieren.

Wenn fehlende oder nicht plausible Daten gefunden werden, stellt sich die Frage, wer bis wann diese Daten korrigieren soll bzw. kann. Grundsätzlich ist der Attributverantwortliche

dafür zuständig, die Inhalte der Attribute entweder selbst nachzupflegen oder dafür zu sorgen, dass es ein anderer tut. Die Dringlichkeit hängt von der Dringlichkeit des Informationsbedarfs ab. Es kann sowohl in einer schnellen Aktion eine ganze Liste von Daten nachgepflegt werden, als auch angewiesen werden, dass bei der nächsten Bearbeitung einer Anforderung deren Daten zu korrigieren sind.

8.3 Risiken und Probleme bei der Anwendung des Berichtswesens

In der Praxis treten praktische Schwierigkeiten bei der Erhebung und Auswertung der Daten auf, die dazu führen, dass Berichte die Realität nicht adäquat abbilden. Da aufgrund der Berichte wichtige Managemententscheidungen getroffen werden sollen, kann ein lückenhafter oder sogar absichtlich geschönter Bericht weitreichende Folgen nach sich ziehen.

Auswertung der Daten: komprimiertes Abbild der Realität

Ein Bericht stellt immer ein stark komprimiertes Modell der Realität dar, in dem Gleichartiges zu Kategorien zusammengefasst wird und unwesentliche Details weggelassen werden. Es ist sehr schwer, dies so zu tun, dass jederzeit jede zukünftig auftretende Frage gut beantwortet werden kann.

Darum muss man sich der Oberflächlichkeit eines Berichts immer bewusst sein. Insbesondere sollte man vermeiden, falsche Schlüsse aus den vorhandenen Daten zu ziehen. So erlaubt z. B. ein Bericht, der eine 99%ige Verfolgbarkeit aller Anforderungen im Projekt ausweist, noch keine Aussage über den Fortschritt des Projektes oder die Qualität der Verlinkungen. Gerade die noch nicht verlinkten Anforderungen könnten die wichtigsten bzw. aufwandintensivsten Anforderungen sein, die maßgeblich zum Projekterfolg beitragen. Dieser Problematik sollte man sich bei einer Reduktion der Komplexität auf Kennzahlen immer bewusst sein. Oft sind nur sehr grobe Aussagen und Schlussfolgerungen möglich.

Werden wirklich verlässliche Daten benötigt, muss die Frage richtig gestellt und die richtige Kennzahl ausgewertet werden. Die Herleitung der richtigen Kennzahl kann durch die GQM-Methode (siehe Kapitel 8.2.5) unterstützt werden. Gegebenenfalls müssen noch weitere Daten erhoben werden.

Datenqualität

Fehlende Daten sind meist leicht zu entdecken. Weniger leicht ist es, die Qualität der Daten zu beurteilen: Stimmen die Daten mit der Realität überein? Sind sie aktuell? Messen sie genau das, was sie sollen oder wurde z.B. im Attribut „Aufwand" nur der Implementierungsaufwand erhoben, obwohl auch der Testaufwand mit berücksichtigt werden soll? Ist die Kritikalität tatsächlich das Ergebnis einer Expertenbefragung oder vorläufig gesetzt worden?

Unentdeckte, aber auch bekannte Mängel in der Datenqualität führen dazu, dass der Bericht die Wirklichkeit des Projektes nicht richtig wiedergibt. Aufgrund dieser falschen Daten können nur schwer richtige Managemententscheidungen getroffen werden. Und selbst wenn die mangelnde Datenqualität bekannt ist, sind Entscheidungen schwer zu treffen.

Eine schlechte Datenqualität entsteht oft dadurch, dass die Beteiligten die Datenpflege unwichtig nehmen, weil sie selbst wenig Nutzen davon haben. Manchmal haben sie im Gegenteil sogar ein Interesse daran, die Daten zu schönen, oder zumindest ein Interesse daran, Zeit bei der Datenpflege zu sparen, indem sie keine sorgfältigen Analysen durchführen, sondern hastig plausibel erscheinende Daten eingeben.

Schlechte Datenqualität kann aber auch dadurch entstehen, dass nicht alle Beteiligten dieselbe Vorstellung haben. In der agilen Entwicklung (vgl. Kapitel 10) ist die „definition of done" ein wichtiges Diskussionsthema. Es muss eindeutig definiert sein, wann eine Anforderung als erledigt gewertet ist. Mögliche Kriterien dafür, dass eine Anforderung umgesetzt ist, sind z.B.: Der Code ist erstellt, Unit Tests sind erstellt und erfolgreich durchgelaufen, die Dokumentation ist angepasst und die Codekonvention ist eingehalten.

Gerade, wenn die Daten absichtlich schlecht gepflegt werden, sind Qualitätsmängel schwer zu entdecken. Denn dann trägt der Datenpfleger Sorge, dass die Daten, wenn schon nicht richtig, so doch plausibel sind.

Darum sollte der Datenerhebungs- und Datenanalyseprozess Schritte zur Qualitätssicherung enthalten. Es sind vielfältige Plausibilitätsprüfungen denkbar, bei denen verschiedene Daten miteinander verglichen werden. Bei der Earned-Value-Analyse, wo Projektfortschritt und Budgetverbrauch miteinander verglichen werden, können außer echten Projektproblemen durchaus auch Datenerhebungsprobleme identifiziert werden. Passen Projektfortschritt und Budgetverbrauch nicht zusammen, wäre darum der erste Schritt, die Korrektheit der Daten zu prüfen. Entsprechend kann man auch andere Daten quer prüfen, z.B. den Status der Anforderungen mit ihrem Erhebungsdatum vergleichen. Ist eine Anforderung schon lange erhoben, aber immer noch in einem frühen Status, dann ist sie entweder vergessen worden zu besprechen, oder es wurde einfach nur ihr Status nicht entsprechend ihrem Bearbeitungsstand aktualisiert.

Bewertung der Leistung konkreter Mitarbeiter

In Deutschland stehen die Mitarbeiter und deren Daten unter gesetzlichem Schutz. Insbesondere soll vermieden werden, dass die Leistung des einzelnen Mitarbeiters gemessen wird. Soll trotzdem ein solcher Bericht aufgesetzt werden, z.B. um die Auslastung der Mitarbeiter zu erheben und bei Bedarf umzuverteilen, dann ist ein solcher Bericht mit dem Betriebsrat abzustimmen.

Darum werden Daten besser anforderungs-, projekt- oder teambezogen erhoben. Personenbeziehbare Daten sollten nur erhoben werden, wenn sie unbedingt nötig sind. Zumeist sind sie dies jedoch nicht.

Datenschutzbestimmungen

Geltende Datenschutzbestimmungen grundsätzlicher wie auch unternehmensspezifischer Art müssen bei der Definition und der Umsetzung des Berichtswesens eingehalten werden. Werden personenbezogene Daten von Beteiligten eingegeben und ohne deren Wissen im Unternehmen in Form von Berichten weiterkommuniziert, kann dies zu Problemen führen. In diesem Zusammenhang ist es wichtig, mit den Datenautoren klar zu vereinbaren, wer welche Daten im Rahmen von welchen zu tätigenden Entscheidungen erhält. Grundsätzlich sollten

personenbezogene und personenbeziehbare Daten sparsam verwendet werden, also schon gar nicht erst eingegeben werden. Auch bei der Definition von Sichten sollten Sie darauf achten, dass keine Aussagen über einzelne Personen getroffen werden können, um nicht unbeabsichtigt Regeln des Datenschutzes zu verletzen.

Inflationäres Berichtswesen

Steigt die Menge an Berichtsinformationen stetig an, könnte dies auch dazu führen, dass die Berichtsempfänger diese Daten aus zeitlichen Gründen nicht mehr verarbeiten können und dadurch wichtige Entscheidungen nicht mehr fundiert getroffen werden.

Weniger ist darum mehr! Fokussieren Sie sich auf die wirklich nötigen Informationen. Dies kann auch bedeuten, dass verschiedene Zielgruppen unterschiedliche Berichte erhalten, in denen nur bestimmte Aspekte dargestellt werden oder in verschiedener Detailtiefe dargestellt sind.

8.4 Inhalte für den RMP

Der RMP legt fest, welche (anforderungsbezogenen) Berichte wann erstellt werden. Für jeden Bericht sind der Berichtsempfänger und das Ziel des Berichts dokumentiert, z.B. tabellarisch. Die Herleitung von Berichtsinhalten aus Zielen kann als Goal-Question-Metric-Baum grafisch dargestellt werden (wie in Abbildung 43). Es ist auch festgelegt, welche Inhalte der Bericht enthält und wie diese aus welchen Attributen ermittelt oder berechnet werden und wie sie dargestellt werden (z.B. mit welcher grafischen Darstellung). Die Festlegung kann auch in Form einer Berichtsvorlage oder Sicht dokumentiert sein.

8.5 Vertiefende Literatur

[DeMa1982] Tom DeMarco: Controlling Software Projects: Management, Measurement, and Estimation. Prentice Hall/Yourdon Press, 1982

Earned-Value-Analyse eher für Anfänger:

[Wann2013a] Roland Wanner: Earned Value Management: Die wichtigsten Methoden und Werkzeuge für ein wirkungsvolles Projektcontrolling. CreateSpace Independent Publishing Platform, 2013.

Earned-Value-Analyse eher für Fortgeschrittene:

[Wann2013b] Roland Wanner: Earned Value Management: So machen Sie Ihr Projektcontrolling noch effektiver. CreateSpace Independent Publishing Platform, 3. Auflage, 2013

9 Management von Requirements Engineering Prozessen

9.1 Requirements Engineering als Prozess

Ein Prozess besteht aus voneinander abhängigen Aktivitäten, die jeweils Input in Output transformieren [ISO9000]. Jede Aktivität ist eindeutig ihrer verantwortlichen Organisationseinheit, z.B. einer Rolle, zugeordnet. Das RE und RM als Teil des Entwicklungs-Prozesses kann als Prozess betrachtet werden.

Unter dem RE-Prozess{ XE "Requirements Engineering Prozess" } versteht man einen systematischen Prozess für die Entwicklung von Anforderungen durch einen iterativen, kooperativen Prozess der Ermittlung, Dokumentation, Prüfung und Abstimmung sowie Verwaltung der Anforderungen (nach [LoKa1995]).

Zu diesem RE-Prozess gehören die folgenden vier Typen von Aktivitäten [IREB2015]:

- Anforderungen ermitteln
- Anforderungen dokumentieren
- Anforderungen prüfen und abstimmen
- Anforderungen verwalten

In jedem Projekt gibt es mehrere Ermittlungs-Aktivitäten, wie beispielsweise Workshops und Meetings mit Stakeholdern, die Analyse von Dokumenten und des Altsystems und so weiter. Ebenso gibt es auch für die anderen RE-Aktivitätstypen mehrere Einzelaktivitäten.

Der RE-Prozess hat als Eingangsinformation die Bedürfnisse und Ideen der Stakeholder. Außerdem spielen auch der Status quo vor dem Projektstart (z.B. das Altsystem) und Konkurrenzprodukte eine Rolle. Das Ergebnis des RE-Prozesses ist eine validierte, konfliktfreie, konsistente, priorisierte, qualitätsgesicherte Anforderungs-Spezifikation, die als verlässliche Grundlage für die weiteren Projektarbeiten dienen kann.

Im Allgemeinen haben die vier Aktivitätstypen – deren Vorgehen und Methoden im CPRE Foundation Level [IREB2015] definiert sind – die folgenden Eingaben und Ergebnisse, die natürlich verschieden aussehen können, insbesondere wenn firmenspezifische Vorgaben oder Standards eingehalten werden müssen – oder entsprechend den gegebenen Randbedingungen (siehe Tabelle 8).

Diese vier Typen von Aktivitäten sind immer durchzuführen, egal ob ausdrücklich dokumentiert oder implizit. Sie müssen nicht und können auch nicht sequenziell hintereinander durchgeführt werden, sondern können iterativ, inkrementell oder parallel verlaufen. Auf irgendeine Art ermittelt man immer Anforderungen, und wenn es durch informelle Gespräche ist. Auch eine Dokumentation gibt es meistens, schlimmstenfalls chronologisch über zahlreiche Besprechungsnotizen verteilt. Es wäre natürlich nicht konform zu den Empfehlungen des IREB, Anforderungen nur implizit zu dokumentieren. Standards und Firmenrichtlinien verlangen eine unterschiedliche Durchführung dieser Aktivitäten und machen unterschiedliche Vorgaben hinsichtlich der zu erstellenden Dokumente.

Aktivitätstyp	Eingabe	Ergebnis
Anforderungen ermitteln	Stakeholder und deren Bedürfnisse und Ideen gegebenenfalls: ein vorhandenes Altsystem und dessen Dokumentation; Konkurrenzprodukte	Mündliche und schriftliche Anforderungen einschließlich der System-Vision
Anforderungen dokumentieren	Mündliche und schriftliche Anforderungen	Schriftliche Anforderungs-Spezifikation – textuell oder modellbasiert oder beides
Anforderungen prüfen und abstimmen	Schriftliche Anforderungs-Spezifikation	Eine validierte, konfliktfreie, konsistente, priorisierte, qualitätsgesicherte Anforderungs-Spezifikation
Anforderungen verwalten	Schriftliche Anforderungs-Spezifikation + Änderungsanträge	Eine stets aktuelle, validierte, konfliktfreie, konsistente, priorisierte, qualitätsgesicherte Anforderungs-Spezifikation Aufbereitung der Anforderungen für einzelne Stakeholder-Gruppen

Tabelle 8: Vier Aktivitätstypen beim RE sowie deren Input und Output.

Die Ergebnisse des RE-Prozesses müssen Qualitätskriterien in drei voneinander unabhängigen Dimensionen erfüllen: in der Dimension der Spezifikation, Darstellung und Übereinstimmung (englisch: specification, representation and agreement dimension) [Pohl2010]. Die Anforderungen sollen im Zeitverlauf innerhalb dieser Dimensionen reifer werden, obgleich es gleichzeitig keinen stetigen Anstieg auf allen Dimensionen geben muss. So kann ein Zuwachs auf der Spezifikations-Dimension (z.B. Formalisierung) auch zu einem Rückschritt auf der Übereinstimmungs-Dimension führen, da durch die Formalisierung neue Widersprüche ans Licht gekommen sind.

- **Spezifikation:** Diese Dimension beschreibt die Vollständigkeit der Spezifikation bzw. die Vollständigkeit des Verständnisses der Anforderungen. Zu Anfang des RE-Prozesses sind die Anforderungen vage und unklar (englisch: opaque). Mit Fortschreiten des Prozesses werden die Anforderungen vollständiger im Sinne von einer vollständigen Abdeckung des zu lösenden Problems sowie einer Beschreibung, die detailliert genug ist, um richtig verstanden zu werden. Verschiedene Standards geben Richtlinien dafür vor, welche Anforderungen die Anforderungen erfüllen müssen, damit sie als vollständig gelten können. Jedoch ist es nicht möglich, die Vollständigkeit von Anforderungen zu beweisen.
- **Darstellung:** Hier variiert die Skala von informell bis formal. Als informelle Darstellung gelten Skizzen, Freitext und Prototypen. Als semi-formal gelten graphisch

dargestellte Modelle, beispielsweise Klassendiagramme, Zustandsmaschinen, Anwendungsfalldiagramme oder Datenflussdiagramme. Auch tabellarisch dargestellte Anwendungsfälle, welche strikt einer vorgegebenen syntaktischen Struktur folgen, sind semi-formal. Formale Spezifikationen beschreiben Anforderungen mit Hilfe von Logiksprachen und formaler Semantik völlig eindeutig. Auch die Erstellung einer formalen Spezifikation beginnt üblicherweise mit informellen Darstellungsformen.

- **Zustimmung:** Während des RE-Prozesses ist das Herstellen von Zustimmung ein weiteres Ziel. Man bewegt sich hierbei auf der Übereinstimmungs-Dimension von der persönlichen Sicht zur gemeinsamen Sicht auf die Anforderungen.

Die Anforderungs-Spezifikation soll während des RE-Prozesses auf allen drei Dimensionen optimiert werden. Dabei tragen die Aktivitäten der Ermittlung vor allem zur Verbesserung in Richtung der Spezifikations-Dimension bei, das Dokumentieren zur Darstellung und die Prüfung und Abstimmung zur Verbesserung in der Abstimmungs-Dimension. Die Anforderungsverwaltung zielt darauf ab, das Qualitätsniveau in allen drei Dimensionen selbst bei Änderungen zu erhalten.

Verschiedene Standards (vgl. Kapitel 1.5) machen Vorschläge, wie der RE-Prozess bzw. der Entwicklungsprozess gestaltet werden kann. Sie sind jedoch nur Blaupausen, die jeweils an die Gegebenheiten in der Firma angepasst werden müssen. Durch Tailoring können Prozessparameter, Rollen, Aktivitäten und Ergebnistypen an die vorhandenen Bedürfnisse angepasst werden.

Vorgaben zum RE-Prozess in der Beispiel-Bank

Peter Reber hält sich als zertifizierter CPRE-Profi natürlich an den IREB-Standard. Dieser gibt jedoch nicht vor, wie der RE-Prozess im Detail durchgeführt werden soll. Im Gegenteil zeigt dieser Standard eher die Bandbreite der Wahlmöglichkeiten auf und unterstützt ein passgenaues Maßschneidern (englisch: Tailoring) des Prozesses.

Ein Muss sind die vier Aktivitätstypen:

1. Anforderungen ermitteln
2. Anforderungen dokumentieren
3. Anforderungen prüfen und abstimmen
4. Anforderungen verwalten

Mit welchen Methoden man diese durchführen kann und nach welchen Kriterien man die richtige Methode auswählt, ist im CPRE Foundation Level und dem jeweiligen Modul beschrieben. Wir haben ja bereits zuvor in diesem Buch die Anforderungs-Landschaft für das Fallstudien-Projekt festgelegt.

Zu definieren wäre noch, in welcher Reihenfolge wer die geplanten Aktivitäten wie durchführt. Diese Parameter des RE-Prozesses diskutieren wir im folgenden Unterkapitel 9.2.

9.2 Parameter des Requirements Engineering Prozesses

Der RE-Prozess kann selbst bei Verwendung derselben Ermittlungs-, Abstimmungs- und Dokumentationsmethoden ganz verschieden gestaltet werden und muss sich insbesondere den gegebenen Randbedingungen anpassen, z.B. Projektgröße und Kompetenz der beteiligten Personen. Bei aller Vielfalt existierender RE-Prozesse in verschiedenen Vorgehensmodellen gibt es nur eine übersichtliche Anzahl an Prozess-Parametern, die man bei der Wahl bzw. Anpassung des RE-Prozesses variieren kann:

- Zeitverlauf der Ermittlung, d.h. upfront versus iterativ
- Detaillierungstiefe der Dokumentation, d.h. der unterste Detailgrad, auf dem spezifiziert wird (mit den beiden Extremen schwergewichtige versus leichtgewichtige Spezifikation)
- Eingliederung von Änderungen, konkret: Änderungsantrag versus Product Backlog
- Verteilung der Verantwortlichkeit

Diese Parameter sollten Sie an die gegebenen Randbedingungen anpassen. Solche Randbedingungen sind:

- Größe des Projektes
- Handelt es sich um eine Neuimplementierung oder eine geringfügige Weiterentwicklung, Verbesserung oder Variation eines bestehenden, reifen Systems oder Produkts?
- Ist das System sicherheitskritisch?
- Wurde ein Festpreis vereinbart oder nicht?
- Steht ein seit Jahren zusammenarbeitendes stabiles Team zur Verfügung?
- Verfügbarkeit von Personen und deren Qualifikation

9.2.1 Zeitverlauf der Ermittlung (upfront oder iterativ)

Die Anforderungen können entweder komplett am Projektanfang (englisch: „upfront{ XE "Upfront Anforderungsermittlung" }“) erhoben werden oder iterativ (iteratives RE{ XE "Iteratives Requirements Engineering" }): Im ersteren Fall (*upfront*) erstellt man zu Projektbeginn eine Anforderungs-Spezifikation (z.B. ein Lastenheft), die den kompletten geplanten Projektumfang vollständig beschreibt, zumindest auf der obersten Detaillierungsebene der Anforderungen. Beim iterativen RE verfolgt man nicht den Anspruch, die Anforderungen oder auch nur den Projektumfang (Projekt-Scope) anfangs komplett festzulegen, sondern sieht die Anforderungs-Dokumentation (z.B. das Product Backlog) als eine vorläufige Liste. Anforderungen können jederzeit dazu kommen oder sich ändern, auch während der Implementierung. Achtung: Es besteht ein Unterschied zwischen iterativem RE und iterativer Entwicklung. So ist es denkbar, zunächst upfront eine vollständige Anforderungs-Spezifikation zu erstellen und später die Anforderungen in iterativer Entwicklung umzusetzen.

Handelt es sich bei dem Projekt um die geringfügige Weiterentwicklung, Verbesserung oder Variation eines bestehenden, reifen Systems oder Produkts oder um ein kleines Projekt, dann ist eher zu erwarten, dass die Anforderungen für das Gesamtprojekt stabil definiert werden

können und wenig Überraschungen zu erwarten sind. Hier ist die upfront Anforderungserhebung möglich und sinnvoll.

Handelt es sich jedoch um ein sehr innovatives Projekt mit vielen Unwägbarkeiten, um ein Großprojekt, um ein volatiles Umfeld, wechselnde, unentschlossene oder einander widersprechende Stakeholder oder führen andere Risikofaktoren dazu, dass eine verlässliche Upfront-Spezifikation nicht möglich ist, dient das iterative RE zur Risikoverringerung. Das iterative RE verlangt aber auch die regelmäßige Beteiligung zumindest der wichtigsten Stakeholder. Ist das nicht möglich, sondern nur ein einmaliger Ermittlungsworkshop oder eine initiale Ermittlungsphase, dann müssen die Anforderungen upfront ermittelt werden. Eine Upfront-Spezifikation ist aber auch dann nötig, wenn sie unter erschwerten Bedingungen erstellt werden muss, wenn es sich um ein Festpreisprojekt handelt (sodass der Projektumfang frühzeitig festgelegt werden muss) oder um ein sicherheitskritisches System, für das in der Gesamtschau eine Sicherheitsanalyse durchgeführt werden muss, oder um eine Technologie die nur schwer änderbar und erweiterbar ist, also spontan hinzukommende Anforderungen nur noch schwer berücksichtigt werden können.

9.2.2 Detaillierungstiefe der Anforderungs-Dokumentation

Die Detaillierungstiefe der Dokumentation bzw. Spezifikation kann fließend variieren zwischen schwergewichtigen{ XE "Schwergewichtiges Requirements Engineering" } und leichtgewichtigen Anforderungen{ XE "Leichtgewichtiges Requirements Engineering" }: Die schwergewichtige Spezifikation beschreibt alle Anforderungen auf mehreren Detaillierungsebenen im Detail, einschließlich aller ihrer Attribute und Verfolgbarkeits-Beziehungen. Dadurch wird die Spezifikation sehr umfangreich. In der agilen Entwicklung erzeugt man üblicherweise leichtgewichtige Spezifikationen mit wenigen Detaillierungsebenen. Hier spezifiziert man Anforderungen nur so umfangreich wie nötig und nicht früher als nötig.

Wann bestimmte Informationen nötig sind, hängt vom Vorgehensmodell ab. Was benötigt wird, hängt von den Stakeholdern, deren Bedürfnissen und Vorkenntnissen ab. Eine projektspezifische Stakeholder-Analyse unterstützt dabei, zu definieren, wie detailliert die Requirements-Spezifikation sein muss. So erfüllt die Spezifikation u.a. den Zweck, dass der Entwickler versteht, was die Stakeholder wünschen. Die Details der Implementierung werden bei der leichtgewichtigen Spezifikation entweder dem Entwickler überlassen (insbesondere, wenn er sich mit der Fachdomäne sehr gut auskennt), mündlich diskutiert ohne sie zu dokumentieren oder anhand eines Prototypen verfeinert.

Die leichtgewichtige Anforderungs-Spezifikation beschreibt Anforderungen beispielsweise als User Stories. Anforderungen werden erst dann im Detail spezifiziert, wenn ihre Implementierung unmittelbar bevorsteht. Auch wenn üblicherweise upfront schwergewichtig spezifiziert wird (z.B. im Wasserfallmodell und V-Modell XT) und iterativ leichtgewichtig (wie in Scrum und anderen agilen Methoden, vgl. Kapitel 10), so sind die beiden Parameter Zeitverlauf und Detaillierungstiefe doch unabhängig voneinander. Man könnte sowohl

upfront eine leichtgewichtige Spezifikation erstellen als auch iterativ eine schwergewichtige (wie im Rational Unified Process).

Ob leicht- oder schwergewichtig, das hängt weniger von der Projektgröße oder der Vertragsform ab, sondern vor allem vom Informations- und Dokumentationsbedarf des spezifischen Projektes und dessen Stakeholder.

Sicherheitskritische Systems werden üblicherweise allein schon wegen gesetzlicher Vorgaben schwergewichtig und vollständig spezifiziert. Grundsätzlich lässt sich durch eine leichtgewichtige Spezifikation unnötiger Aufwand einsparen, wenn die Informationen, die man in der schwergewichtigen Spezifikation zusätzlich dokumentiert hätte, auch undokumentiert allen Stakeholdern zur Verfügung stehen, also bei kleinen oder schon lange zusammenarbeitenden Teams, Entwicklern mit sehr guter Domänen- und Kundenkenntnis und bei der Weiterentwicklung eines existierenden Systems.

Aber auch dann kann leichtgewichtig vorgegangen werden, wenn die schnelle Erstellung und Änderung eines Prototypen technisch leicht machbar ist und die Anforderungen anhand des Prototypen verfeinert werden.

Die Detaillierungstiefe wird im Requirements-Information-Model (Kapitel 2) festgelegt.

9.2.3 Änderungsmanagement: Eingliederung von Änderungen (Änderungsantrag versus Product Backlog)

Anforderungen ändern sich während des Projektes. Manche RE-Prozesse integrieren neue oder geänderte Anforderungen als Änderungsanträge in die Anforderungs-Spezifikation und den Entwicklungsprozess. Dabei handelt es sich meist um Projekte mit Festpreis und Upfront-Anforderungs-Spezifikation, d.h. organisatorisch und juristisch ist die Definition des Projektumfangs und die Anforderungserhebung zu einem bestimmten Zeitpunkt abgeschlossen. Spätere Änderungen sind rechtlich gesehen eine Auftragsänderung. Der Änderungsantrag{ XE "Änderungsantrag" } (englisch: Change Request{ XE "Änderungsantrag" }) bedeutet rechtlich eine Neubeauftragung.

Üblicherweise wird bereits im Vertrag des Projekts festgelegt, wie Änderungsanträge zu handhaben sind. Sie durchlaufen dann ein vereinfachtes Genehmigungsverfahren mit den Schritten Analyse (der Anforderungen und deren Nutzen), Auswirkungsanalyse / Impact Analyse (d.h. Analyse der Änderungen am System, deren Kosten und Risiken), Entscheidung durch das Change Control Board und dann die Umsetzung. Beschrieben wird der Änderungsantrag oft in einer Änderungsantrag-Vorlage, die ihm eine eindeutige Nummer und Namen zuweist, das zu lösende Problem und die vorgeschlagene Lösung beschreibt, Kosten, Nutzen und Risiken quantifiziert und den Status verwaltet (beantragt, angenommen, abgelehnt, verschoben, umgesetzt) (vgl. Kapitel 5).

Im iterativen RE werden dagegen Anforderungen im Product Backlog gesammelt und alle Anforderungen – die älteren und die neuen – gleichwertig behandelt. Dies wird dadurch ermöglicht, dass man sich nie auf einen definierten Systemumfang festlegt. Der Vorteil dieses Vorgehens liegt in der Flexibilität. Neue, wichtige Anforderungen können leicht ins Projekt integriert werden. Diese Flexibilität bringt allerdings auch den Nachteil mit sich, dass

man nur schwer festlegen kann, worin der Lieferumfang letztlich bestehen wird. Die Anzahl der Anforderungen auf dem Product Backlog kann sich ständig erhöhen, schlimmstenfalls sogar schneller als sie umgesetzt werden.

Trotzdem ist es nicht zwingend nötig, dass durch eine Upfront-Anforderungs-Spezifikation spätere Anforderungen als Änderungsanträge zu behandeln sind. Es wäre grundsätzlich denkbar, auch die upfront erstellte Anforderungs-Spezifikation später anzupassen, ohne Änderungen als Änderungsantrag zu erfassen und zu genehmigen. Änderungen an den Anforderungs-Artefakten müssen natürlich dokumentiert werden und nachvollziehbar sein.

Umgekehrt könnte man auch beim iterativen RE eine Genehmigung für neue Anforderungen einholen. Entscheidend ist hier vor allem die Vertragsform. Beim Werkvertrag kann eine neue Anforderung nur im beiderseitigen Einvernehmen ins Projekt integriert werden, was einen mehr oder weniger umfangreichen Genehmigungsprozess nötig macht. Ansonsten ist es eine Vereinbarungssache zwischen Auftraggeber und Auftragnehmer, wie mit Änderungen umgegangen werden soll.

Aus Sicht des RM macht es jedenfalls Sinn, die Änderungsanforderungen früher oder später in die existierende Anforderungs-Spezifikation zu integrieren, um jederzeit eine aktuelle Spezifikation des geplanten Systems verfügbar zu haben.

Es sollte also nicht zwei separate Spezifikationen geben, z.B. die Anforderungs-Spezifikation auf dem Stand des Projektbeginns plus eine Liste chronologisch sortierter Änderungsanträge. Eine solche Darstellungsform erfüllt nicht die Modifizierbarkeitsanforderung an Anforderungs-Spezifikationen. Aus Sicht des RM stellt sich also vor allem die Frage, durch wen, wann und in welcher Form Änderungsanforderungen in die Spezifikation integriert werden. Dabei ist in jedem Fall eine gute Abstimmung zwischen Projektmanagement und RM nötig. So liefert das RM den Entscheidern wichtige Informationen über den voraussichtlichen Einfluss einer Änderung (zu Auswirkungsanalysen siehe Kapitel 6.5.2 zur Nutzungsstrategie der Verfolgbarkeit).

9.2.4 Verteilung der Verantwortlichkeit

Für das RE kann eine einzelne Rolle (z.B. der „Requirements Manager“) verantwortlich sein in dem Sinne, dass er den RE-Prozess plant, steuert und verbessert. Die Aktivitäten des Requirements Prozesses führt er entweder selbst durch oder sorgt für deren Durchführung. Es kann aber auch ein ganzes Team oder mehrere Rollen geben, die für das RE verantwortlich sind, entweder für verschiedene Aktivitäten oder verschiedene Inhalte (z.B. funktionale Anforderungen versus Usability-Anforderungen). Das RE kann aber auch in den Entwicklungsprozess eng integriert sein, ohne dass ein separater RE-Prozess oder eine RE-Rolle existiert. Dann führt das Entwicklungsteam die RE-Aktivitäten durch, d.h. die Teammitglieder erheben, dokumentieren, prüfen und verwalten Anforderungen.

Je größer das Projekt, umso eher ergibt es Sinn, eine separate Rolle eines Requirements Managers zu definieren, der diesen Tätigkeitsbereich des Projektes überwacht. Aber auch bei kleinen Projekten kann man diese Rolle definieren, die dann eben keine Vollzeitaufgabe ist. Der Requirements Manager sollte die Person sein, die sich am besten mit RE und RM auskennt. Insbesondere muss diese Person sehr gute kommunikative Fähigkeiten haben und

auch mit allen Stakeholdern im ständigen Kontakt stehen. Zusätzlich zu methodischem RE- und RM-Wissen und fachlichen Kenntnissen, die der Requirements Engineer benötigt, braucht der Requirements Manager auch noch Management-Kenntnisse, um den RE-Prozess aufsetzen, verwalten und überwachen zu können.

Parameter des RE-Prozesses in der Beispiel-Bank

Die Ermittlung der Anforderungen erfolgt in unserer Fallstudie upfront, da es sich um ein Weiterentwicklungsprojekt an einer dem Team gut bekannten Software handelt. Es ist also gut möglich, die Anforderungen anfangs zu erheben und zu beschreiben. Die Entwicklung dagegen wird iterativ erfolgen.

Wie in Kapitel 2 definiert, werden die Anforderungen auf mehreren Detaillierungsebenen beschrieben. Dies unterstützt eine detaillierte Sicherheitsanalyse und eine komplette Dokumentation des Systems für die Zukunft.

Änderungsmanagement: Es handelt sich um ein Inhouse-Projekt mit fixem Budget. Da es jedoch keinen Werkvertrag gibt, ist man im Prinzip beim Inhalt – wenn schon nicht beim Projektumfang – flexibel.

Wenn sich neue wichtige Anforderungen ergeben, sollten diese ins Projekt mit aufgenommen werden und dafür andere, weniger wichtige zurück gestellt. Dies gilt insbesondere bei Gesetzesänderungen, die kurzfristig berücksichtigt werden müssen. Solche Anforderungsänderungen müssen jedoch ordentlich geprüft und kontrolliert durchgeführt werden.

Dazu ist ein dokumentierter Änderungsprozess nötig, bei dem während der Auswirkungsanalyse nicht nur die Umsetzungskosten geschätzt werden müssen, sondern auch das Sicherheitsteam eine Risikoeinschätzung bezüglich der IT-Sicherheit abgibt, die Business Analysten in Bezug auf die Geschäftsprozesse und die Usability-Expertin bezüglich der Barrierefreiheit. Schließlich entscheidet ein Change Control Board aufgrund dieser Beurteilungen darüber, ob die Anforderungsänderung angenommen, abgelehnt oder zurückgestellt wird.

Die Verantwortlichkeiten waren bereits in Kapitel 1 festgelegt worden: Peter Reber als der Requirements Manager plant und überwacht den RE-Prozess, während mehrere Experten RE betreiben, d.h. Anforderungen erheben, dokumentieren und abstimmen. Mehrere externe Business Analysten analysieren die Geschäftsprozesse, ein Team von IT-Sicherheits-Experten betreibt Risiko-Analysen, der Usability-Experte entwirft alternative Oberflächen-Designs und verbessert die Barrierefreiheit, ein Moderator trifft sich zu einem Ideenworkshop mit dem Kundenbeirat.

9.3 Den Requirements Engineering Prozess dokumentieren

Der RE-Prozess besteht aus zahlreichen Aktivitäten der oben genannten vier Typen, beispielsweise Ermittlungsworkshops, Dokumentenanalysen, Spezifikations-Reviews etc. Geplant werden viele dieser Aktivitäten in Form von Meetings bzw. Workshops, da an vielen Aktivitäten mehrere Personen beteiligt sind. Die Reihenfolge dieser Aktivitäten ergibt sich aus der Wahl der Prozessparameter, z.B. ob Anforderungen upfront ermittelt und spezifiziert werden oder wie Anforderungsänderungen gehandhabt werden sollen (siehe Kapitel 9.2).

Das Folgende gilt unabhängig davon, ob Sie einen generischen RE-Prozess definieren, der als Firmenvorgabe für alle Projekte gelten soll, oder ob Sie für ein konkretes Projekt dessen RE-Prozess planen.

Die Aktivitäten und deren Reihenfolge kann man als UML Aktivitätsdiagramm darstellen. Im Aktivitätsdiagramm kann man auch die Zuordnung der Aktivitäten zu Rollen darstellen. Sie kennen diese Notation bereits aus dem CPRE Foundation Level [PoRu2015].

Die Zuordnung der Verantwortlichkeiten für die Aktivitäten zu den Rollen kann auch differenzierter mit einer *RACI-Matrix* wie der folgenden dargestellt werden. RACI steht für:

- **R** = *responsible* = verantwortlich für die Durchführung
- **A** = accountable = rechenschaftspflichtig, d.h. man genehmigt beispielsweise die Aktivität und deren Budget
- **C** = *consulted* = (wird) konsultiert, insbesondere im Sinne von fachlicher, inhaltlicher Verantwortung
- **I** = informed = zu informieren, d.h. die Person wird über die Ergebnisse informiert

Die folgende Tabelle zeigt ein Beispiel eines Auszugs aus einer RACI-Matrix.

Aktivität	Requirements Manager	Business Analyst	IT-Sicherheits-Experten	Usability-Experte	Moderator	Kunden-beirat
Analyse der Geschäfts-prozesse	A, I	R	I	C, I	I	
Risiko-Analyse	A, I	C, I	R	C, I		
Oberflächen-Design	A	C	C	R		
Ideenworkshop	A, I	I	I	I	R	C
...						

Tabelle 9: Beispiel einer RACI-Matrix für das RE

Tabelle 9 zeigt einen Ausschnitt der RACI-Matrix für unsere Fallstudie. Die hier dargestellten Aktivitäten gehören alle zum Aktivitätentyp Anforderungserhebung. Zur Ermittlung gehören vermutlich noch weitere hier nicht näher spezifizierte Aktivitäten. Hinzu kommen außerdem noch Aktivitäten des Typs Anforderungs-Dokumentation, Anforderungen prüfen und abstimmen sowie Anforderungsverwaltung. Unser Ziel ist hier jedoch keine vollständige Planung des RE-Prozesses für das Gesamtprojekt, sondern v.a. die Illustration der entsprechenden Darstellungsmethoden.

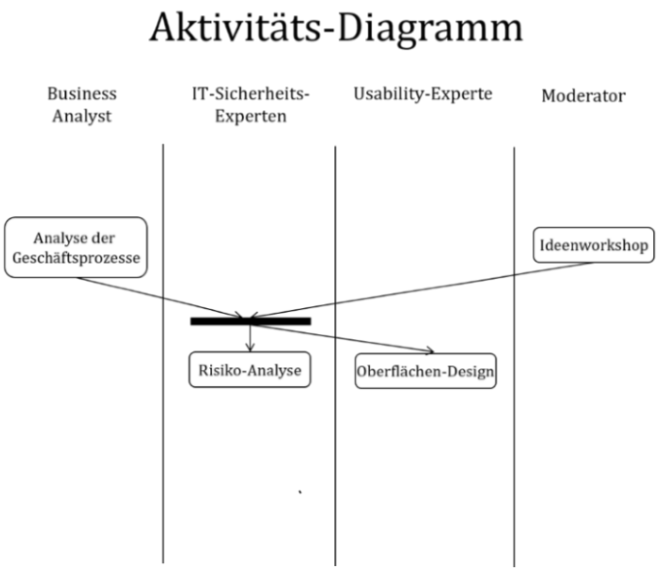


Abbildung 44: Aktivitätsdiagramm (Ausschnitt) des RE-Prozesses für die Fallstudie

Abbildung 44 stellt denselben Ausschnitt des RE-Prozesses als Aktivitätsdiagramm dar. Es wird offensichtlich, worin die beiden Notationen sich voneinander unterscheiden und einander ergänzen:

Die RACI-Matrix kann differenzierter die Verantwortung der verschiedenen Rollen für die Aktivitäten darstellen, während im Aktivitätsdiagramm eine Aktivität meist nur in einer einzigen Swimlane steht: in derjenigen, die zur verantwortlichen Rolle gehört. Wer sonst noch an der Aktivitität beteiligt ist, wird nicht dargestellt.

Das Aktivitätsdiagramm dokumentiert dagegen auch Abhängigkeiten der Aktivitäten voneinander, z.B. Reihenfolgen wie „Die Risiko-Analyse erfolgt nach der Geschäftsprozessanalyse (da sie auf deren Ergebnissen aufbaut)" oder „Risiko-Analyse und Oberflächen-Design können parallel erfolgen".

Noch differenzierter stellt das Gantt-Diagramm in Abbildung 45 den Zeitverlauf dar. Der Verantwortliche steht jeweils in der Spalte „Verant." (hier abgekürzt)und auch die anderen RACI-Verantwortlichkeiten könnten hier

dargestellt werden Der Zeitverlauf ist für jede Aktivität waagrecht als Zeile dargestellt und es wurde jede Kalenderwoche (KW) schwarz hinterlegt, in der an dieser Aktivität gearbeitet wird. Die Dauer und auch wiederholte Arbeiten können so noch detaillierter dargestellt werden als im Aktivitätsdiagramm. Hier ist dafür die Darstellung von Abhängigkeits-Beziehungen zwischen den Aktivitäten nicht so gut sichtbar, wenn sie auch durch Pfeile darstellbar sind.

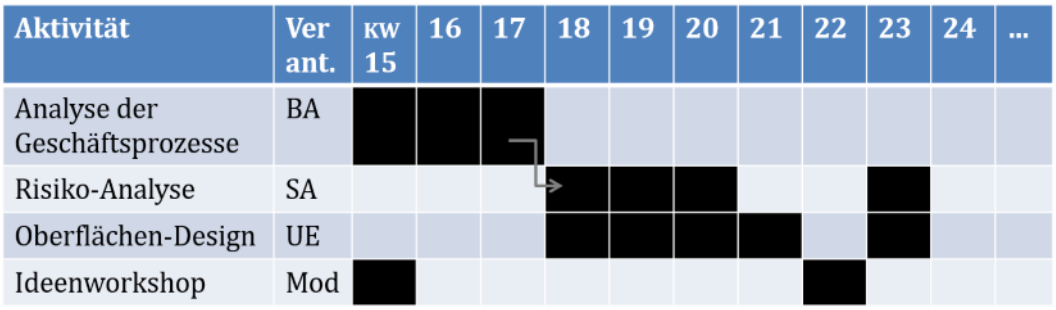


Abbildung 45: Gantt-Diagramm (Beispiel). BA steht für Business Analyst, SA für Sicherheitsanalyst (bzw. IT Sicherheits-Experte), UE für Usability-Experte, Mod für Moderator.

Die drei Darstellungsarten ergänzen einander also sehr gut. Sie können daher gemeinsam verwendet werden. Effizienter ist es jedoch, sich auf so wenige Darstellungen wie möglich zu konzentrieren. Denkbar ist auch, das Gantt-Diagramm für die Grobplanung der Arbeitspakete zu verwenden und je ein Aktivitätsdiagramm für die Feinplanung jedes Arbeitspakets.

Will man Termine und Budgets quantitativ verwalten, lässt sich der RE-Prozess auch als Projektplan darstellen. Weitere Dokumente, die den RE-Prozess darstellen und unterstützen können, sind: Checklisten, Vorlagen, Beispieldokumente und Richtlinien für die Durchführung einzelner Aktivitäten.

Sind viele Personen am RE-Prozess beteiligt, dann ist es ebenfalls sinnvoll, diesen Prozess mit einem Werkzeug zu unterstützen. Geeignet sind hierbei alle Workflow Management Systeme im weitesten Sinne.

9.4 Den Requirements Engineering Prozess überwachen und steuern

Den RE-Prozess zu überwachen, bedeutet sicherzustellen, dass alle Aktivitäten durchgeführt werden und die definierten Ergebnisse pünktlich erbracht werden sowie dass die Aktivitäten im Budget bleiben. Hilfreich hierzu sind Berichte, welche die Termine, das verbrauchte Budget, den Status und den Fertigstellungsgrad des RE-Prozesses und dessen einzelner Aktivitäten regelmäßig erfassen und die Ist-Werte mit den Soll-Werten aus der Planung vergleichen (siehe Kapitel 8).

Den RE-Prozess zu steuern bedeutet, ihn entsprechend dem Plan durchzuführen oder, wenn der Verlauf vom Plan abweicht, korrigierend einzugreifen. Zeichnet sich z.B. ab, dass ein

Termin oder Budget nicht eingehalten werden kann, so sind die Auswirkungen auf das Gesamtprojekt zu ermitteln und wenn sinnvoll Gegenmaßnahmen zu ergreifen. Hierbei gibt es zwei alternative Möglichkeiten der Korrektur: Man kann den Plan an den tatsächlichen Verlauf anpassen oder den Verlauf an den Plan. Ersteres ist einfacher, aber bei einem Projekt mit verbindlichem Endtermin und Budget oft nur schwer machbar. Um den Verlauf an den Plan anzupassen, müssen ggf. geplante Aktivitäten wegfallen, vorgezogen werden oder mit weniger Aufwand durchgeführt werden. Hierbei müssen kompetent Abstriche gemacht werden, wo sie am wenigsten Schaden anrichten, z.B. einzelne Stakeholder-Gruppen werden nicht befragt, einzelne offene Fragen nicht geklärt, Details nicht spezifiziert, unwichtige Änderungsanträge abgelehnt und so weiter. Die Voraussetzung für solch eine Schwerpunktsetzung ist natürlich eine Priorisierung der Anforderungen (vgl. Kapitel 4). Wichtig ist hierbei eine Risikobetrachtung: Überwiegt der Nutzen der Einsparung deren möglichen Schaden?

Tipp aus der Praxis: Wenn der definierte RE-Prozess nicht eingehalten wird, kann das verschiedene Ursachen haben. Beispielsweise kann es sein, dass die Mitarbeiter den Prozess nicht kennen oder nicht richtig verstanden haben. Es kann auch sein, dass der Prozess nicht die optimale Arbeitsweise beschreibt und darum nicht eingehalten wird. Aber auch Widerstände gegen neue Arbeitsweisen oder gegen bestimmte Arbeitsweisen können dahinter stecken. Da jede dieser Ursachen eine andere Maßnahme zur Behebung verlangt, muss unbedingt herausgefunden werden, warum der Prozess nicht gelebt wird.

9.5 Prozessverbesserung für den Requirements Engineering Prozess

Ein Prozess lässt sich immer noch weiter verbessern. Das Reifegradmodell CMMI{ XE "CMMI" } (*Capability Maturity Model Integration*) [CMMI] verlangt zwingend, dass ein reifer Entwicklungsprozess Aktivitäten für die kontinuierliche Verbesserung der Arbeitsprozesse vorsieht. Die Grundlage dafür ist eine Ist-Analyse bzw. Assessment oder Audit genannt, welche systematisch untersucht, wie gut der Prozess aktuell ist, wo er bereits gut ist und wo Verbesserungspotenzial besteht. Bei einem Audit wird meist der aktuelle Prozess mit einem Referenzprozess (z.B. vorgegeben durch einen Standard) verglichen und Prozesskennzahlen werden erhoben (vgl. Kapitel 8.2.2.1). Die Grundlage für eine Prozessanalyse sollten immer objektive, messbare Kriterien sein.

Bei der Durchführung eines Audits folgt man bspw. folgenden Schritten:

1. Erkennen eines Bedarfs für ein Audit,
2. Planen des Audits, z.B. des Zwecks und Ziels, des Umfangs (Prozess? Produkt? Welches?), des Teams, der Kriterien, der Ressourcen und Termine,
3. Durchführen des Audits und Dokumentieren der Ergebnisse,
4. Auswerten der Ergebnisse: Stärken und Schwächen, nötige Verbesserungen und die dringendsten Maßnahmen,
5. Umsetzen der Maßnahmen,
6. Messen der Verbesserungen.

Prozessverbesserung kann man abrupt als Prozessumstellung durchführen oder kontinuierlich. Eine Prozessumstellung verändert viele Aktivitäten und Parameter des Prozesses zugleich. Das hat den Vorteil, dass man so eine starke Effizienzsteigerung bewirken kann, die sich allerdings meist erst einstellt, nachdem sich alle Beteiligten an den neuen Prozess gewöhnt haben. Es besteht aber auch die Gefahr, dass der neue Prozess sich nicht bewährt und die Effizienz senkt. Eine Rückumstellung ist dann wieder mit großem Aufwand verbunden.

Die kontinuierliche Prozessverbesserung vermeidet dieses Risiko und führt mit wenig Aufwand kurzfristig zu (meist kleinen) Verbesserungen. Nach dem Prinzip der kontinuierlichen Prozessverbesserung{ XE "kontinuierliche Prozessverbesserung" } (KPV) optimiert man Prozesse allmählich, indem man die folgenden vier Aktivitäten (PDCA){ XE "PDCA" } des Deming-Zyklus{ XE "Deming-Zyklus" } [Demi1982] iterativ wiederholt:

- **Plan / Planen:** Man analysiert den Ist-Prozess und insbesondere vorhandenen Verbesserungsbedarf. Darauf aufbauend plant und dokumentiert man den Soll-Prozess.
- **Do / Umsetzen:** Man entwickelt und testet Verbesserungsmaßnahmen in einem Pilotprojekt und begleitet sie durch Messungen.
- **Check / Überprüfen:** Man prüft, ob die Maßnahmen die gewünschte Verbesserung bewirkt haben. Dafür vergleicht man die Ist-Werte mit den Plan-Werten.
- **Act / Handeln:** Aufgrund der Ergebnisse des Ist-Plan-Vergleichs werden Verbesserungsmaßnahmen durchgängig eingeführt oder falls nötig neue Maßnahmen geplant. Die Umsetzung der Maßnahmen wird überwacht und durch Messungen begleitet.

Den Ist- und Soll-Prozess charakterisiert man anhand von Messgrößen (vgl. Kapitel 8). Solche Messgrößen können sein:

- Der *Anteil des Projektbudgets*, der in das RE investiert wurde. Sowohl ein zu hoher Anteil als auch zu wenig kann bedenklich sein. Normal sind 10-30 % des Projektbudgets.
- Die (nach voraussichtlichem Aufwand gewichtete) *Anzahl der noch zu implementierenden Anforderungen* misst die bis Projektende noch zu erledigende Arbeit.
- *Anzahl der Anforderungen (bzw. Anzahl gewichtet nach Implementierungsaufwand), die pro Zeiteinheit implementiert werden.* Zusammen mit den Aufwandsschätzungen der noch nicht implementierten Anforderungen lassen sich so Prognosen über die Restdauer des Projektes berechnen.
- Änderungsrate der Anforderungen: Als normal gilt eine Rate von 1-5 % der Anforderungen pro Monat (gemessen in Aufwand) und schlimmstenfalls 30-50 % über eine mehrjährige Projektlaufzeit [Eber2012].
  Gibt es weniger Änderungen, kann das bedeuten, dass sich niemand wirklich für die Anforderungen interessiert und die Stakeholder nicht genügend eingebunden sind. Zu viele Änderungen sind ebenfalls ein Alarmzeichen: Die Anforderungen sind noch nicht stabil, eventuell sind die Stakeholder-Gruppen zu heterogen oder in Konflikt, und es ist noch zu früh, um die Anforderungen zu implementieren.

- *Durchlaufzeit von Änderungsanträgen* von der Beauftragung bis zur Umsetzung

Mit Hilfe von Benchmarking lässt sich herausfinden, welche Werte als Soll-Wert sinnvoll und erreichbar sind.

Verbesserungsmaßnahmen können sich entweder auf die in Kapitel 9.2 beschriebenen Prozessparameter beziehen oder darauf, wie die einzelnen Aktivitäten im Detail durchgeführt werden, z.B. mit Hilfe welcher Methoden.

Eine weitere Möglichkeit der Prozessverbesserung besteht darin, die Fehler zu analysieren, die im RE gemacht werden, z.B. die bei der Lastenheft-Inspektion gefundenen Fehler oder auch die in der Software ausgelieferten Fehler, die auf das RE zurückzuführen sind. Dann fragt man nach deren Ursachen und den Ursachen der Ursachen. So kommt man auf Ideen für Verbesserungsmaßnahmen.

Eine konkretere Hilfe zur Prozessverbesserung auch (aber nicht nur) im RE bieten Reifegradmodelle wie das auf der ISO 15504 basierende CMMI (Capability Maturity Model Integration) [Kneu2007], [CKS2011], [CMMI] oder auch ITIL für die Softwarewartung [Beim2012], [Ebel2014].

Nach [Eber2012] ist ein Reifegradmodell ein „Modell, das die Prozessfähigkeit in definierte Kategorien abbildet und damit eine verlässliche und wiederholbare Prozessbewertung erlaubt. Ein Reifegradmodell stellt Forderungen an Prozesse und schreibt selbst keine Prozesse vor. Es ist daher kein Vorgehensmodell. Wird zur Bewertung der Prozessreife und zur Prozessverbesserung sowohl für die eigenen Prozesse als auch für jene der Zulieferer eingesetzt."

Reifegradmodelle beschreiben Aktivitäten oder Praktiken, die zur Erreichung eines bestimmten Reifegrades durchgeführt werden müssen. Auch alle anderen Methoden der Prozessverbesserung sind für die Verbesserung des RE-Prozesses einsetzbar wie TQM (Total Quality Management) [HuMa2011] und Six Sigma [Tava2012], [BWJ2013].

TQM besteht insbesondere aus Prinzipien zur Erreichung von Qualität und wirtschaftlichem Handeln. Wichtig sind hier z.B. die Kundenorientierung, Prozessorientierung, Qualitätsorientierung, die gemeinsame Zuständigkeit aller Mitarbeiter, kontinuierliche Verbesserung und rationale Entscheidungen. Six Sigma dagegen ist ein Rahmenwerk zur Verbesserung, wobei Messungen und statistische Analysen durchgeführt werden, um durch einen fehlerfreien Prozess fehlerfreie Produkte zu erzeugen.

Speziell die Verbesserung des REs unterstützt die Sammlung von Best Practices von Sommerville und Sawyer [SoSa1997], die unterscheiden zwischen drei Kategorien von Best Practices: *basic*, *intermediate* und *advanced*. Zunächst gilt es bei der Verbesserung des RE-Prozesses, alle Basis-Praktiken umzusetzen, dann die Intermediate-Praktiken und zuletzt diejenigen, die als advanced klassifiziert sind.

Zu den Basis-Techniken gehört beispielsweise die Definition einer Standardvorlage für die Anforderungs-Spezifikation oder einer Checkliste für die Inspektion dieser Spezifikation.

Bei der konkreten Planung der Prozessverbesserung unterstützt die Vorlage für einen Action Plan von Karl Wiegers [Wieg2005]. Diese Vorlage enthält folgende Inhalte:

- Name des Verbesserungsprojektes
- Datum
- Ziele (der Verbesserung, ausgedrückt in geschäftlichen Zielen)
- Kenngrößen für den Erfolg (d.h. Erreichung der Ziele)
- Organisatorischer Einfluss der Veränderung
- Teilnehmer (Mitarbeiter, deren Rollen und zeitliche Budgets)
- Mess- und Berichtsprozess (wann wird wer wie den Fortschritt der Maßnahmen dieses Plans überwachen und berichten)
- Abhängigkeiten, Risiken und Randbedingungen
- Geschätztes Abschlussdatum aller Maßnahmen dieses Plans
- Maßnahmen (3–10 pro Plan) mit Identifikator, verantwortlicher Person, Zieldatum, Zweck, Beschreibung, Liefergegenständen und Ressourcenbedarf.

Zu beachten ist bei der Verbesserung des RE-Prozesses, dass er nicht für sich allein optimiert werden kann, sondern nur in Zusammenarbeit mit den anderen Tätigkeiten des Projektes wie Projektmanagement, Entwicklung und Testen. Veränderungen im RE-Prozess werden sich auch auf deren Arbeit auswirken.

Tipp aus der Praxis: Jeder Prozess sollte so einfach wie möglich und nur so komplex wie nötig sein. Je mehr verbindliche Vorgaben Sie machen, umso mehr schränken Sie Kreativität und Flexibilität ein. Da sich die Randbedingungen ändern, müssen auch Prozesse sich ständig anpassen. Ein Prozess, der sich nicht ändert, veraltet sonst.

Den RE-Prozess verbessern

Der RE-Prozess in unserer Fallstudie ist neu aufgesetzt und umfasst eine Vielzahl an Stakeholdern, die bisher nicht zusammen gearbeitet haben. Das sind Risikofaktoren, die die Beobachtung des RE-Prozesses besonders wichtig machen.

Welche Kennzahlen der Statusbericht enthalten soll und warum diese so wichtig sind, wurde bereits in Kapitel 8.2.2.1 diskutiert. Anhand dieser Daten kann wöchentlich eine aktualisierte Prognose für den Liefertermin abgegeben werden.

Treten beim Fortschritt des RE-Prozesses Verzüge oder andere Schwierigkeiten auf, dann werden die Ursachen gesucht und Maßnahmen ergriffen werden.

Doch es soll auch proaktiv vorgegangen werden und Risiken in Bezug auf die Termintreue identifiziert werden. Da im Gesamtprojekt die Termintreue sehr wichtig ist, gilt dies auch für den RE-Prozess. Hier gilt außerdem, dass verschiedene Aktivitäten aufeinander aufbauen und daher voneinander abhängen. Gemeinsam mit dem Projektleiter führt Peter Reber daher eine Kritische-Pfad-Analyse im Netzplan des RE-Prozesses durch, um herauszufinden, welche Aktivitäten für die Einhaltung des Endtermins besonders kritisch sind. Diese sollen dann auch besonders aufmerksam beobachtet werden. (Die Netzplantechnik beschreiben wir hier nicht näher, weil sie zu den Methoden

des Projektmanagements gehört. Sie können jedoch in der DIN 69900 [DIN69900] und in jedem Projektmanagementbuch mehr darüber erfahren.). Als besonders kritisch werden die Risiko-Analysen angesehen, die nach jeder Änderung der Anforderungen und für jede Anforderungsänderung durchgeführt werden sollen. Damit sich diese nicht unnötig verzögern, wird dafür gesorgt, dass mehrere IT-Sicherheits-Experten zur Verfügung stehen. Diese planen für jeden Montagnachmittag einen Workshop ein. Dieser kann bei Nichtbedarf ausfallen. So ist sichergestellt, dass regelmäßig Ressourcen für die Risiko-Analysen zur Verfügung stehen.

Auch die Beteiligten des RE-Prozesses werden nach ihrer Einschätzung gefragt. Die Usability-Expertin sieht es als ein Risiko für die Qualität des Ergebnisses, wenn die Oberflächen-Designs ohne Beteiligung der Anwender erstellt werden. Sie wünscht sich eine Möglichkeit, Rückmeldung von den Anwendern zu erhalten, die hier vom Kundenbeirat vertreten werden. Darum wird eine zusätzliche Aktivität „Anwendertests der Oberflächen" eingeplant. Diese führen die Usability-Expertin und der Moderator gemeinsam durch wobei sie den Kundenbeirat konsultieren. Tabelle 10 zeigt die ergänzte RACI-Matrix.

Aktivität	Requirements Manager	Business Analyst	IT-Sicherheits-Experten	Usability-Experte	Moderator	Kunden-beirat
Analyse der Geschäfts-prozesse	A, I	R	I	C, I	I	
Risiko-Analyse	A, I	C, I	R	C, I		
Oberflächen-Design	A	C	C	R		
Anwender-tests der Oberflächen	A			C	R	C
Ideenwork-shop	A, I	I	I	I	R	C
...						

Tabelle 10: Beispiel einer RACI-Matrix für das RE

9.6 Inhalte für den RMP

Der RMP dokumentiert den RE-Prozess in einer der oben beschriebenen Notationen. Es wird dabei auch festgelegt, ob die Anforderungen upfront oder iterativ ermittelt werden, wie Anforderungsänderungen eingegliedert werden sollen und wie die Verantwortlichkeiten für die RE-Aktivitäten verteilt sind. Die Detaillierungstiefe wird bereits durch das RIM festgelegt.

Es soll auch klar sein, wie der RE-Prozess überwacht wird (z.B. mit Hilfe von welchem Bericht). Auch Maßnahmen zur Prozessbewertung und -verbesserung sollten vorgesehen werden, z.B. *Lessons Learned* Analysen nach Projektende.

9.7 Vertiefende Literatur

[BWJ2013] Franz J. Brunner, Johann Wappis, Berndt Jung: Null-Fehler-Management: Umsetzung von Six Sigma, Carl Hanser Verlag GmbH & Co. KG; Auflage: 4., überarbeitete und erweiterte Auflage, 2013.

[DIN69900] DIN 69900 Projektmanagement – Netzplantechnik; Beschreibungen und Begriffe, 2009.

[HuMa2011] Thomas Hummel, Christian Malorny: Total Quality Management: Tipps für die Einführung, Carl Hanser Verlag GmbH & Co. KG; Auflage: 4., vollständig überarbeitete Auflage, 2011.

[Tava2012] Serkan Tavasli: Six Sigma Performance Measurement System: Prozesscontrolling als Instrumentarium der modernen Unternehmensführung, Deutscher Universitätsverlag, 2012.

10 Requirements Management in agilen Projekten

10.1 Vorwissen

10.1.1 Grundlagen agiler Entwicklung

Ein klassisches phasenweises, plangetriebenes Vorgehen in Reinform verliefe so: Zuerst wird das Gesamtprojekt geplant, dann werden die Anforderungen *upfront* vollständig spezifiziert und abgenommen, anschließend dann umgesetzt und getestet.

Dieses Vorgehen funktioniert allerdings nicht in allen Projekten und Anforderungs-Domänen. Insbesondere dann, wenn die Anforderungen mangels Wissens oder Erfahrung nicht gut genug bekannt sind (z.B. bei sehr innovativen Projekten) oder sich in einem volatilen Projektumfeld ständig ändern. Aus diesem Grund erfolgt gutes RE meist iterativ, z.B. mit Hilfe von Prototypen, und nicht nach einem reinen Wasserfall-Modell.

Die agilen Entwicklungsmethoden{ XE "agile Entwicklung" } empfehlen ebenfalls ein iteratives Vorgehen – allerdings ein sehr kurz getaktetes, leichtgewichtiges Vorgehen, das innerhalb einer Iteration nur diejenigen Dokumente erstellt, die absolut nötig sind. Darüber hinaus heißen agile Vorgehen neue Anforderungen und Anforderungsänderungen jederzeit willkommen, da diese in einer nächsten Iteration berücksichtigt werden können („Embrace change").

Natürlich gibt es nicht nur diese beiden extremen Vorgehen, sondern alle möglichen Abstufungen zwischen upfront und iterativem RE, zwischen schwergewichtigem und leichtgewichtigem RE.

Die gemeinsame Grundlage aller agilen Ansätze ist das Agile Manifesto{ XE "Das Manifest für agile Softwareentwicklung" } („agile Manifest"). Im agilen Manifest wird aus Sicht der Softwareentwickler gefordert *AgileManifesto*:

"Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

> *Individuen und Interaktionen mehr als Prozesse und Werkzeuge*
> *Funktionierende Software mehr als umfassende Dokumentation*
> *Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung*
> *Reagieren auf Veränderung mehr als das Befolgen eines Plans*

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein."

Beim agilen Vorgehen sind also die Zusammenarbeit, Produktivität und individuelle Stärken des Teams wichtiger als Verträge und Dokumente (einschließlich Anforderungs-Spezifikationen).

Hierin grenzen sich die agilen Methoden von plangetriebenen Ansätzen ab, die klare Vertragsgegenstände (z.B. Projekt-Scope, Anforderungs-Spezifikationen, Releasepläne oder einen definierten Änderungsprozess) fordern.

Weiter definiert das agile Manifest dreizehn Prinzipien AgileManifesto:

1. *„Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.*
2. *Heiße Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.*
3. *Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.*
4. *Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.*
5. *Errichte Projekte rund um motivierte Individuen.*
6. *Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.*
7. *Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.*
8. *Funktionierende Software ist das primäre Maß für den Fortschritt.*
9. *Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.*
10. *Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.*
11. *Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell.*
12. *Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.*
13. *In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an."*

Die wichtigsten agilen Methoden sind:

- Scrum [ScBe2001], [ScSu2013]
- Extreme Programming XP [Beck2000]
- Kanban [Ande2010]
- Lean Software Development [PoPo2003]
- Crystal Cock1997, [Cock2004], [Cock2006]
- Feature-Driven Development FDD [PaFe2002], [Nebu2014]

Tipp aus der Praxis: Tatsächlich lauten die Alternativen bei der Wahl eines Vorgehensmodells nicht nur „agil oder Wasserfall". Es gibt nicht nur zahlreiche agile und zahlreiche plangetriebene Vorgehensmodelle, sondern auch noch diverse Varianten jedes Vorgehensmodells sowie hybride Vorgehen mit Elementen aus beiden Welten. Sie können sich auch ein Vorgehensmodell ausgehend von einem existierenden maßschneidern [Herr2014]. Angesichts dieser umfangreichen Auswahlmöglichkeiten lohnt es sich darum, die Randbedingungen und Bedürfnisse Ihres Projektes genau zu analysieren und das Vorgehensmodell sorgfältig auszuwählen. Wie Sie dabei vorgehen, kann hier nicht ausführlich behandelt werden. Die Auswahl der passenden RM-Praktiken aus dem Repertoire der agilen Methoden werden wir jedoch diskutieren. RM-Praktiken aus agilen Methoden können auch in nichtagilen Projekten angewendet werden.

10.1.2 Scrum als Vertreter der agilen Methoden

Scrum{ XE "Scrum" } ist der derzeit am weitesten verbreitete agile Ansatz. Eine vollständige Beschreibung finden Sie im Scrum Guide 2013 [ScSu2013]. Scrum wird beschrieben durch seinen Ablauf (getrieben durch seine Events), seine Artefakte und seine Rollen. Diese sind ganz typisch für agile Frameworks und sind so ähnlich auch in anderen agilen Methoden zu finden.

10.1.2.1 Scrum-Ablauf

Scrum definiert den Arbeitsablauf wie folgt: Ein Sprint{ XE "Sprint" } (Iteration{ XE "Iteration" }) dauert bis zu vier Wochen. Am Ende jedes Sprints muss ein fertiges, nutzbares und potenziell auslieferbares Produkt (Komponente, Inkrement o.ä.) fertig gestellt sein.

Der **Sprint** enthält die folgenden Events oder Meetings:

- **Sprint-Planung:** Hier werden aus dem *Product Backlog* – der Liste aller aktuell ermittelten Anforderungen – diejenigen Einträge identifiziert, die im kommenden *Sprint* bearbeitet werden. Es wird ein *Sprint Backlog* erstellt, in dem die zu bearbeitenden Backlog Items abgelegt sind. Auf diesem *Backlog* werden Aufgaben oft durch *User Stories* dargestellt (siehe unten), d.h. es erfolgt eine anforderungsbasierte Arbeitsplanung.
- **Sprint:** definierter Zeitraum in dem das Team die Items des *Sprint Backlogs* bearbeitet.
- **Daily Scrum**: An jedem Arbeitstag findet ein Daily Scrum (oder auch: Stand-up Meeting) statt. D. h. die Besprechung des Teams zum Informationsaustausch über aktuelle Arbeiten und Schwierigkeiten sowie zur feingranularen Planung des Arbeitstages.
- **Sprint Review:** Hier werden am Sprint-Ende die Arbeitsergebnisse des vergangenen Sprints diskutiert. Der *Product Owner* nimmt das Sprintergebnis ab.
- **Sprint Retrospective:** Das *Scrum Team* (d.h. Product Owner, Scrum Master und Development Team) tauscht sich nach Ablauf eines *Sprints* über die Zusammenarbeit aus. Es soll herausgefunden werden, wie der Arbeitsprozess im Team verbessert werden kann.

10.1.2.2 Scrum-Artefakte

Das **Product Backlog**{XE "Produkt-Auftragsbestand (Produkt-Backlog)"}

- Enthält die *Backlog Items*, z.B. *User Stories* des zu entwickelnden Produkts sowie technische und administrative Aufgaben, in der Bearbeitungs-Reihenfolge. Die Backlog Items des Product Backlogs sollen vom Product Owner so sortiert werden, dass Ziele und die Missionen optimal erreicht werden.
- Muss nicht vollständig sein, sondern wird laufend fortgeführt.
- Beschreibt hoch priorisierte *User Stories* detaillierter als die niedrig priorisierten.

Das **Sprint Backlog**{XE "Sprint-Auftragsbestand, Sprint-Backlog"}

- Enthält alle *Backlog Items*, die in diesem *Sprint* zu realisieren sind, ergänzt um den Plan für die Lieferung des Produkt-Inkrements sowie zur Erfüllung des Sprint-Ziels.
- Macht die gesamte Arbeit sichtbar, die das Development Team für notwendig erachtet, um das Sprint-Ziel zu erreichen.
- Wird vom Development Team um weitere Arbeiten ergänzt, wenn diese zur Erreichung des Sprint-Ziels erforderlich sind.
- Als Good Practice werden die *Backlog Items* in Aufgaben (*Tasks*) von typischerweise einem Arbeitstag Dauer heruntergebrochen.

Ein in der Praxis weit verbreitetes „Werkzeug" ist das *Task Board*. Das *Task Board* ist eine Pinnwand zur Visualisierung des *Sprint Backlogs* und des Fertigstellungsgrads der *Backlog Items*. Hier wandern die *Tasks* des aktuellen *Sprints* von links nach rechts, entsprechend ihrem Bearbeitungsstatus. Dabei stellt je eine Spalte einen Status dar, z.B. „To Do" (zu tun), „*In Process*" (in Arbeit), „*To Verify*" (zu testen) und „*Done*" (fertig). Jeweils eine Zeile fasst die Tasks zusammen, die zu einem *Backlog Item* (z.B. *User Story)* gehören. So stellt das *Task Board* eine Sicht auf die Anforderungen (*User Stories*) und deren Status dar.

Ein weiteres in der Praxis häufig verwendetes Mittel zur Darstellung des Projektfortschritts ist das *Burndown Chart*. Das Burndown Chart ist eine graphische, pro Tag zu erfassende Darstellung des noch zu erbringenden Restaufwands pro Sprint. Im Idealfall fällt die Kurve kontinuierlich (daher *Burndown*) und der Restaufwand ist am Ende des *Sprints* Null. Der im *Task Board* dargestellte Status und Fertigstellungsrad des aktuellen *Sprints* wird hier quantitativ und grafisch visualisiert.

Das *Inkrement* ist das am Ende des *Sprints* fertiggestellte, lauffähige und potentielle lieferbare Produkt. Laut dem Agilen Manifest ist dieses das wichtigste Artefakt.

Das *Impediment Backlog* ist eine Liste aller Hindernisse des Projekts. Der *Scrum Master* ist dafür zuständig, diese gemeinsam mit dem Team auszuräumen.

Aus Sicht des RM ist die *User Story*{XE "User Story"} das zentrale Artefakt. Eine *User Story* beschreibt eine Anforderung auf einer Karteikarte mit einem definierten Satzbau. Eine User Story hat i.d.R. die folgende Form:

Als <ROLLE>,
möchte ich <FUNKTIONALITÄT>,
sodass <NUTZEN>

Beispielsweise:
Als Kunde,
möchte ich Geld von meinem Konto auf ein fremdes Konto überweisen,
um meine Rechnungen möglichst bald zu bezahlen.

Die Angabe des Nutzens ist hierbei optional, d.h. sie kann auch entfallen. Durch die Angabe des Nutzens kann der Mehrwert der Funktionalität für die Rolle explizit gemacht werden, was ihrerseits wieder das Verständnis der User Story verbessert.

Zusätzlich kann die *User Story* – z.B. an den Ecken – noch Informationen enthalten wie die Kostenschätzung (z.B. in *Story Points*), den Nutzen für die Benutzer (auf einer Punktskala) sowie das technische Risiko. Diese Angaben dienen der Priorisierung der *User Stories*.

Für jede *User Story* werden auch Akzeptanzkriterien und Testfälle genauer spezifiziert. Diese können knapp auf der Rückseite der Karteikarte dokumentiert werden in folgender Form:

Unter Voraussetzung, dass <Vorbedingung>,
wenn <Trigger>
dann <Ergebnis>.

Beispielsweise:
Unter der Voraussetzung, dass auf meinem Konto mehr als 100 € vorhanden sind,
wenn ich eine Überweisung über 100 € tätige,
dann werden anschließend auf meinem Konto 100 € weniger angezeigt und auf dem Zielkonto 100 € mehr als zuvor.

Durch das Notieren von Anforderungen (*User Story*) und Testfällen auf derselben Karte ist ohne viel Aufwand die Verfolgbarkeit zwischen beiden gegeben.

10.1.2.3 Scrum-Rollen

Scrum unterscheidet innerhalb des *Scrum-Teams* nur drei Rollen: *Product Owner, Scrum Master* und *Development Team*. Das *Development Team* organisiert sich selbst, und zwar nicht nur die Programmierung, sondern auch das RE, RM und das Projektmanagement. Bezüglich des RMs sind die Aufgaben folgendermaßen aufgeteilt:

- Der *Product Owner* trifft alle inhaltlichen Entscheidungen, d.h. welche *Backlog Items* (z.B. *User Stories)* es gibt, was sie umfassen, wie sie formuliert sind, wie sie getestet werden sollen und insbesondere welche Prioritäten sie haben.

- Der *Scrum Master* ist für das Verständnis und die Durchführung von Scrum verantwortlich. Er tut dies, indem er dafür sorgt, dass das Scrum Team die Theorie, Praktiken und Regeln von Scrum einhält, d.h. der Scrum Master coacht das Scrum Team.
- Das *Development Team* setzt die Anforderungen um. Die Teammitglieder informieren untereinander über den Bearbeitungsstatus im *Daily Scrum*.

10.2 Anforderungsmanagement im Rahmen agiler Produktentwicklung

In der agilen Entwicklung sind Anforderungen sehr wichtig. Besonders gerne werden Anforderungen in Form von *User Stories* verwendet: Die *User Stories* sind die Grundlage für die Iterations- und Arbeitsplanung sowie für die Fortschrittsüberwachung auf dem *Task Board* und im *Burndown Chart*. Allerdings ist es ganz typisch, dass die Anforderungen möglichst leichtgewichtig und „gerade gut genug" spezifiziert werden. Da üblicherweise die Teammitglieder bei den agilen Methoden eng zusammenarbeiten und täglich miteinander kommunizieren, stellt die User Story nur eine Protokollnotiz dessen dar, was besprochen wurde. Sie braucht darum nicht vollständig oder eindeutig für Dritte zu sein.

Aus Sicht des RM stellen sich die agilen Methoden folgendermaßen dar:

- Die agile Anforderungs-Landschaft ist in der Regel einfach: Besonders häufig werden *User Stories* verwendet sowie zusätzlich Akzeptanztests für die Spezifikation der Anforderungen. Falls nötig, können zusammengehörige *User Stories* durch *Epics* zusammengefasst werden (siehe unten).
- Jede Anforderung bzw. *User Story* hat nur wenige Attribute, z.B. Kosten- und Nutzen-Schätzung oder das Risiko, die auf der *User Story* Karte stehen. Den Nutzen bewertet der *Product Owner*, die technischen Risiken und Kosten das *Development Team*.
- Es gibt nur wenige Sichten auf die Anforderungen: Das *Product Backlog*, das *Sprint Backlog* und das *Task Board*.
- Die Aufwandsschätzung von Backlog Items geschieht häufig durch das so genannte *Planning Poker*, das wir bereits in Kapitel 4.5.5 beschrieben haben. Als Kriterien zur Priorisierung der Backlog Items dienen die wenigen vorhandenen Attribute, insbesondere Kosten und Nutzen. Das Kosten-Nutzen-Verhältnis bestimmt dann die Priorität. Oder umgekehrt: Es werden nur solche Attribute verwaltet, die für die Priorisierung nützlich sind.
- Die Implementierungs-Reihenfolge der *User Stories* ermittelt sich nach: dem Wert für den *Product Owner*, technischen Abhängigkeiten (Funktionen, welche die Grundlage für andere darstellen, müssen zuerst implementiert werden), nach technischem Risiko (*User Stories*, deren Implementierung mit technischen Risiken behaftet sind, werden möglichst früh implementiert, um frühzeitig die Risiken besser beurteilen zu können) sowie Sprint-Thema.
- Versionsverwaltung gibt es in diesem Sinne keine. Erledigte oder veraltete *User Stories* landen meist im Papierkorb.

- Der Änderungsprozess ist einfach: Neue Anforderungen oder Anforderungsänderungen werden ins *Product Backlog* geschrieben und bei der nächsten Sprint-Planung berücksichtigt. Soll eine noch nicht realisierte *User Story* ersetzt werden, landet diese im Papierkorb. Hierzu ist kein Genehmigungsprozess und kein Gremium nötig.
- Die volle Verantwortung hierfür trägt der *Product Owner*. Die Entscheidung über die tatsächliche Umsetzung neuer Ideen fällt bei der Sprint-Planung.
- Variantenmanagement ist in der agilen Entwicklung nicht vorgesehen. Auch Verfolgbarkeit gibt es höchstens implizit, z.B. durch die Zuordnung von *User Stories* zu Iterationen oder *User Stories* zu *Epics*. Die Verfolgbarkeit zwischen User Story und Code erfolgt in Form von Eincheck-Kommentaren im Versionsverwaltungssystem. Auch *User Stories* und Testfälle müssen verfolgbar miteinander verknüpft sein, z.B. indem sie physisch auf derselben *Story Card* stehen.
- Das Berichtswesen ist einfach und anforderungsbasiert: Das *Task Board* zeigt den Bearbeitungsstatus jeder Aufgabe und jeder *User Story* tagesaktuell an. Das *Burndown Chart* zeigt quantitativ an, was noch zu tun ist, und ob voraussichtlich am Sprintende alle geplanten *User Stories* realisiert sein werden. Darüber hinaus sind keine Berichte nötig, wenn das Team sich ohnehin täglich abstimmt und ständig miteinander kommuniziert.
- Einen definierten RE-Prozess gibt es in den agilen Methoden nicht ausdrücklich. Entweder kann der *Product Owner* alle Stakeholder des Systems vertreten und deren Anforderungen ausformulieren, oder er ist dafür verantwortlich, diese Anforderungen zu ermitteln. Dieser Anforderungserhebungsprozess ist nicht Teil von *Scrum* und daher hier nicht definiert.
- Zur Verbesserung des Arbeitsprozesses und somit auch des RE und RM dienen das *Impediment Backlog* und die Sprint Retrospektive.
- Die zu verwendenden Werkzeuge sind simpel. Ursprünglich wurde nur mit Karteikarten und einer Pinnwand gearbeitet. Je mehr jedoch agile Teams nicht am selben Ort arbeiten, umso mehr setzen sich einfache Software-Werkzeuge durch. Diese Werkzeuge setzen die Backlogs, Task Board und Burndown Chart elektronisch um. So können auch global verteilte Mitarbeitende jederzeit auf diese Artefakte zugreifen.

In der agilen Entwicklung fehlen also einige Elemente des RM, die wir empfehlen. Natürlich kann man diese nicht einfach risikofrei weglassen! Aus Sicht des RM setzt diese leichtgewichtige, iterative Agilität folgende Randbedingungen voraus:

- Der *Product Owner* kennt die Anforderungen oder er kann sie ermitteln. Ggf. kann der *Product Owner* auch mehrere weitere Personen konsultieren. Diese müssen jedoch ständig verfügbar sein und aktiv im Projekt mitarbeiten.
- Das *Development Team* hat genügend Domänenwissen, um die Anforderungen trotz leichtgewichtiger Beschreibung als *User Story* richtig zu verstehen. Ggf. können die leichtgewichtigen *User Stories* auch durch schwergewichtigere Darstellungsformen ergänzt und konkretisiert werden.
- Das Team arbeitet selbstorganisiert und eigenverantwortlich.

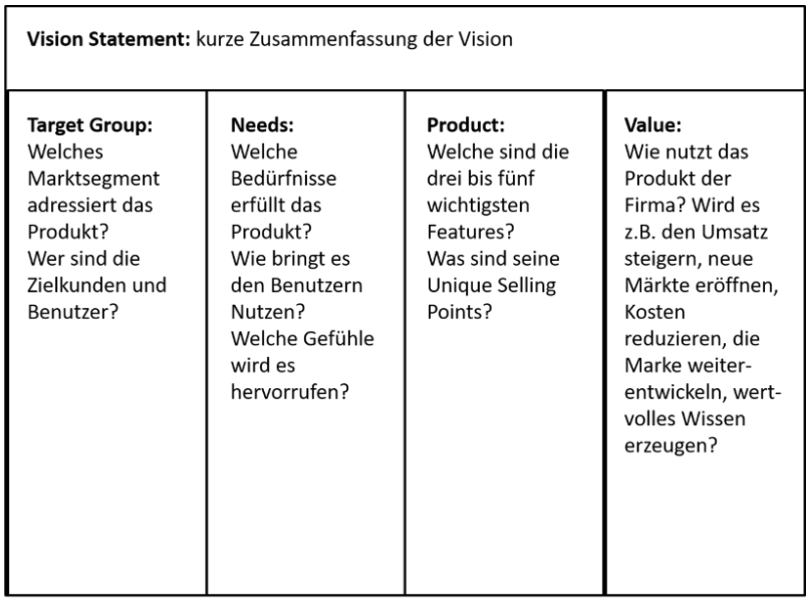
Tatsächlich stellt sich in der Praxis gelegentlich heraus, dass die leichtgewichtige Anforderungs-Spezifikation in Form von *User Stories* und Akzeptanztests (vgl. Kapitel 10.1.2.2) nicht genügt. Die agilen Prinzipien verbieten allerdings auch nicht, einzelne oder alle Anforderungen detaillierter, schwergewichtiger oder anders zu spezifizieren als durch *User Stories*.

Methoden des klassischen RE und klassischen RM können in agilen Projekten zusätzlich eingesetzt werden, wo das Team dies für sinnvoll erachtet oder Firmenrichtlinien dies vorschreiben. Der Anwendungsfall (Use Case), als Mittel auch in der klassischen Anforderungs-Spezifikation etabliert, wird ebenso als Artefakt im agilen Umfeld verwendet (z.B. [Cock2001] oder [JSB2011]).

Es sind auch Mischformen zwischen klassischem und agilem Projekt denkbar, z.B. dass man upfront eine Anforderungs-Spezifikation erstellt und anschließend iterativ und agil entwickelt und testet.

Die agilen Methoden haben sich bereits selbst um zusätzliche Formen der Anforderungs-Spezifikation erweitert, die aus dem klassischen RE stammen oder von dort in leichtgewichtiger Form adaptiert wurden. Im Folgenden sollen kurz das *Vision Board, Minimal Viable Product MVP* und *Minimal Marketable Product MMP*, *Epics* und *Story Maps* beschrieben werden.

Das Vision Board (auch Product Canvas genannt) [Pich2014] beschreibt die Vision und eine ganz knappe, leichtgewichtige Form eines Business Case für ein Produkt oder ein Projekt. Es umfasst nur fünf Felder wie in Abbildung 46 dargestellt.



Quelle: http://scaledagileframework.com/epics/

Abbildung 46: Vision Board nach Roman Pichler [Pich2014] (eigene Darstellung)

Minimal Viable Product MVP: Das ist das kleinste Produkt, das bereits verwendet werden kann, um Rückmeldungen von Stakeholdern zu erhalten. Es enthält gerade so viele Features, dass Benutzer seine Nützlichkeit beurteilen können [Ries2011].

Minimal Marketable Product MMP: Das ist das kleinste Produkt, das auf dem Markt verkauft werden kann. Es enthält gerade so viele Features, dass ein Benutzer es nützlich verwenden kann. Somit ist es das kleinste Produkt, das verkauft werden kann.

Epics sind Beschreibungen von Anforderungen auf einem höheren Detaillierungsgrad als User Stories. Sie fassen daher üblicherweise mehrere User Stories zusammen.

Ein *Epic* könnte beispielsweise heißen „Kontoverwaltung", und dazu gehören mehrere User Stories wie „Kontostand ansehen", aber auch „Konto eröffnen" und „Konto auflösen". *Epics* können als *Epic Value Statement* (vgl. Abbildung 47) diskutiert und dargestellt werden, um aufzuzeigen, für wen sie inwiefern nützlich sind. Diese Nutzendiskussion dient dann der Priorisierung der *Epics* und der dazu gehörigen *User Stories* und somit der Iterationsplanung.

Position Statement	
For	<customers>
Who	<do something>
The	<solution>
Is a	<something – the „how">
That	<provides this value>
Unlike	<competitor, current solution, or non-existing solution>
Our solution	<does something better the „why">
Scope	
Success Criteria	
In Scope	
Out of Scope	
NFRs	

Abbildung 47: Epic value statement template [Leff2011]

Story Maps{ XE "Story-Maps" } [Patt2008] sind eine Übersichtsdarstellung über den Zusammenhang von Anforderungen und Geschäftsprozessen. Story Maps dienen dazu, das „Walking Skeleton", also die minimale Umsetzung eines funktionierenden Geschäftsprozesses, zu ermitteln. Eine mögliche Darstellung ist, horizontal die Aktivitäten des Geschäftsprozesses darzustellen und unter den Aktivitäten die jeweils dazugehörigen Anforderungen (z.B. User Stories) zuzuordnen.

Die Skalierung von agilen Ansätzen auf große und verteilte Teams ist in den Anfängen, aktuell bilden sich einige Frameworks heraus. Ansätze hierzu finden sich z.B. in [Ecks2004], [Ecks2010], [Leff2011] sowie [KoBe2013].

10.3 Abbildung von RM-Tätigkeiten auf Scrum-Tätigkeiten

Scrum versteht sich selbst als „Rahmenwerk, innerhalb dessen Menschen komplexe adaptive Aufgabenstellungen angehen können, und durch das sie in die Lage versetzt werden, produktiv und kreativ Produkte mit höchstmöglichem Wert auszuliefern" [ScSu2013]. Scrum gibt jedoch nur allgemeine Arbeitsprozesse vor. In der folgenden Tabelle sind die RM-Tätigkeiten den Scrum-Tätigkeiten bzw. Artefakten zugeordnet. Weiterhin wird die ausführende Rolle in Scrum angegeben. Nicht zu allen RM-Tätigkeiten macht Scrum eine Aussage. (Das heißt: der Scrum-Guide. Neben dem Scrum-Guide gibt es eine

unübersichtliche Anzahl von Literatur, die mehr oder weniger erfolgreiche Ergänzungen zu Scrum machen. Hier beziehen wir uns ausschließlich auf den Scrum-Guide.) Ob und wie die entsprechende RM-Tätigkeit dann in einem Scrum-Projekt ausgeführt wird, bleibt dem Scrum-Team überlassen.

RM-Tätigkeit	Scrum-Tätigkeit bzw. Artefakt	Scrum-Rolle
Attributierung	User Stories im Backlog: Beschreibung, Reihenfolge, Schätzung, Status und Wert. Optional: Gruppierung	PO, DT
Bewertung und Priorisierung	Schätzung von Nutzen und Aufwand durch Planning Poker Anordnung der User Stories im Product Backlog Auswahl der User Stories für einen Sprint Priorisierung innerhalb eines Sprints	DT PO PO und DT DT
Verfolgbarkeit	Es besteht eine implizite Verfolgbarkeit von User Stories zu den zugehörigen Abnahmetestfällen sowie, bei geeigneter Attributierung, zurück zu den Quellen der User Stories. Darüber hinaus ist eine Verfolgbarkeit innerhalb des Product Backlog (Abhängigkeiten) und von User Stories zum Sourcecode möglich. Scrum sagt nichts über die Verbindung von User Stories innerhalb des Product Backlog aus. Denkbar wäre eine Verfolgbarkeit über Epics (gruppierte User Stories). Verfolgbarkeit wird nur dokumentiert, wenn deren Notwendigkeit gegeben ist.	Keine
Versionierung	Eine Versionierung von User Stories ist unnötig. Relevant ist immer die jeweils aktuelle Version einer User Story.	Keine oder PO
Änderungen	Änderungen können jederzeit vorgeschlagen werden. Neue Anforderungen führen zu neuen User Stories, Anforderungsänderungen dazu, dass eine User Story geändert oder durch eine neue ersetzt wird.	PO

RM-Tätigkeit	Scrum-Tätigkeit bzw. Artefakt	Scrum-Rolle
Varianten-management	Agile Methoden unterstützen Variantenmanagement nicht ausdrücklich. Der Einsatz üblicher Methoden des Variantenmanagements ist jedoch möglich.	PO
Berichtswesen	Berichte erfolgen vor allem mündlich. Die zur Verfolgung des Bearbeitungsstands verwendeten Artefakte können gleichzeitig auch als Berichte dienen: ▪ Daily Standup ▪ Sprint Review ▪ Sprint Retrospective ▪ Product-Backlog ▪ Sprint-Backlog ▪ Burndown Chart	DT
Prozess-management	Sprint Retrospective und Impediment Backlog	SM, DT

Tabelle 11: Abbildung der RM-Aktivitäten auf Scrum

10.4 Vertiefende Literatur

[AgileManifesto] Manifesto for Agile Software Development. Abrufbar unter http://agilemanifesto.org/. Zuletzt besucht im März 2024.

[Beck2000] Kent Beck: Extreme programming explained. Addison-Wesley, Upper Saddle River, 2000.

[JSB2011] I. Jacobson, I. Spence, K. Bittner: Use Cases 2.0. Ivar Jacobson International, 2011.

[KoBe2013] H.-P. Korn and J.P. Berchez (eds.): Agiles IT-Management in großen Unternehmen. Symposion, 2013.

[Leff2011] D. Leffingwell: Agile Software Requirements, Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional, 2011.

[PoPo2003] Mary Poppendieck, Tom Poppendieck: Lean Software Development. Addison Wesley, 2003.

[Ries2011] Eric Ries: The Lean Startup: How Constant Innovation Creates Radically Successful Businesses. Penguin, 2011.

[ScBe2001] Ken Schwaber, Mike Beedle: Agile Software Development with SCRUM. Prentice Hall, 2001.

[ScSu2013] Ken Schwaber, Jeff Sutherland: The Scrum Guide — The Definitive Guide to Scrum: The Rules of the Game, July 2013, http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf. Zuletzt besucht im März 2024.

11 Werkzeugeinsatz im Anforderungsmanagement

Auf dem Markt der RE- und RM-Werkzeuge{ XE "Werkzeug" } finden Sie mittlerweile eine Vielzahl unterschiedlicher Werkzeuganbieter mit verschiedenen Lizenzmodellen. Von Freeware bis Unternehmenslizenzen ist hier alles vertreten. Diese Werkzeuge unterscheiden sich vor allem in ihrem Kernfokus (Dokumentation, Zusammenarbeit, Verfolgbarkeit, Agilität).

Nicht alle auf dem Markt befindlichen Werkzeuge können als echte RM-Werkzeuge verstanden werden – auch wenn sie für das RM sicherlich hilfreich sein können (z.B. Modellierungs-Werkzeuge oder Versionskontroll-Systeme (VCS)).

Unter der folgenden URL finden Sie eine umfangreiche Liste von mehr als 100 RM-Werkzeugen: https://makingofsoftware.com/resources/list-of-rm-tools/ inklusive einer Klassifizierung ihres Kernfokus. In [HJD2011] finden Sie eine Beschreibung, wie das RM-Werkzeug DOORS® für die Verwaltung von Anforderungen genutzt werden kann.

Tipp aus der Praxis: In Unternehmen werden Sie nicht immer spezielle RM-Werkzeuge im Einsatz finden. Oftmals werden hier Standard-Büroanwendungen sowie webbasierte Plattformen zum Dokumentenaustausch und zur Zusammenarbeit gefunden. Selbst unter diesen Voraussetzungen lässt sich mit einigen organisatorischen Regeln und der nötigen Disziplin ein gutes RM umsetzen. Beachten Sie immer, dass die Auswahl irgendeines RM-Werkzeuges i.d.R. nicht zielführend ist, solange Sie nicht entschieden haben, wie Sie Ihren RE-Prozess umsetzen wollen (vgl. RMP). Eine kleine Unterstützungshilfe für die Werkzeugauswahl anhand der im Handbuch diskutierten RM-Aspekte finden Sie in Anhang B.

11.1 Rolle von Werkzeugen im Requirements Management

Der Einsatz von Werkzeugen soll dem Requirements Manager die Dokumentation und Verwaltung von Anforderungen vereinfachen. Werkzeuge für das Anforderungsmanagement ermöglichen durch ihre speziellen Funktionalitäten eine ganzheitliche Betrachtung von Anforderungen, indem u.a. Beziehungen zwischen unterschiedlichen Anforderungen (vgl. Kapitel 6 Verfolgbarkeit) sowie der Lebenszyklus einzelner Anforderungen (vgl. Kapitel 5 Versions- und Änderungsmanagement) abgebildet werden können.

Ein Werkzeug für das Anforderungsmanagement ist eine softwaretechnische Anwendung, deren Hauptziel die Unterstützung von Aktivitäten im RM ist.

In der Software- und Systementwicklung werden traditionell viele unterschiedliche Anwendungen eingesetzt. Viele davon beinhalten allerdings nur Teilaspekte des REs und/oder des RMs. Die Abgrenzung zu dedizierten Werkzeugen für das Anforderungsmanagement ist dabei nicht immer trennscharf.

Werkzeugen für das Anforderungsmanagement liegen spezifische Annahmen zugrunde, sodass sich diese Werkzeuge beispielsweise auf bestimmte Vorgehensmodelle, Arbeitsumgebungen oder Anwendungsdomänen konzentrieren:

- bestimmte Vorgehensmodelle, wie bspw. agile oder plangetriebene Entwicklung,
- bestimmte Arbeitsumgebungen, wie bspw. lokale oder verteilte Zusammenarbeit,
- bestimmte Anwendungsdomänen, wie bspw. Automobilindustrie oder Rüstungsindustrie

[SoSa1997] beschreiben die folgenden fünf Merkmale als Kernfunktionen eines RM-Werkzeugs:

- **Editor für Anforderungen** inkl. deren Attribute, um die Erfassung und Attributierung eindeutig identifizierbarer Anforderungs-Artefakte zu ermöglichen; hierbei kann es sich um einen rein textbasierten Editor handeln oder um einen Editor, der textuelle und modellbasierte Beschreibungen unterstützt
- **Import von Anforderungen** aus bestehenden Dokumenten in das Werkzeug (z.B. auf Basis des ReqIF-Formats, vgl. Kapitel 11.3) und **Export von verwalteten Anforderungen** in andere Formate (z.B. in dokumentenbasierte Spezifikationen)
- **Verfolgung von Anforderungen**, beginnend bei der Unterstützung zur Pflege von Verfolgbarkeits-Beziehungen bis hin zur Nutzung der gepflegten Verfolgbarkeits-Beziehungen, z.B. im Rahmen einer Auswirkungsanalyse
- **Versionierung** von Anforderungen sowie der Bildung von Anforderungs-Konfigurationen und Basislinien
- Bildung von **benutzerdefinierten Sichten** auf Anforderungen inkl. ihrer Attribute

Vergleichen wir diese Merkmale mit denen aus [PoRu2015] (Kap. 9.3), finden wir folgende zusätzlichen Merkmale, die bei der Auswahl eines RM-Werkzeugs eine wesentliche Rolle spielen:

- Verteilte Bearbeitung von Anforderungs-Artefakten inkl. Zugriffskontrolle
- Erstellung von rollenspezifischen Sichten für unterschiedliche Benutzergruppen
- Erzeugung von Berichten oder Auswertungen über die verwalteten Artefakte

Unabhängig von den Merkmalen, die ein Werkzeug bietet, ist für die Werkzeugauswahl und Werkzeugeinführung zu beachten, dass ein ausgewähltes RM-Werkzeug zu den im Unternehmen etablierten Vorgehensweisen und Prozessen passt.

11.2 Prinzipielle Vorgehensweise bei der Werkzeugauswahl

Das richtige Werkzeug auszuwählen ist nicht leicht. Es gibt viele Werkzeuge, und welches davon am besten zu Ihnen passt, hängt von Ihrem ganz persönlichen RE-Prozess ab. Denn der Prozess bestimmt die Anforderungen an das Werkzeug.

Die Auswahl eines RM-Werkzeugs betrifft meistens nicht nur ein einziges Projekt, sondern soll oft projektübergreifend stattfinden, z.B. für alle Projekte einer Abteilung oder eines Unternehmens. Das macht die Werkzeugauswahl in der Regel komplex, sodass die Einführung eines RM-Werkzeuges oftmals durch ein eigenständiges Projekt vorangetrieben wird, siehe auch [RuSo2009].

Es ist empfehlenswert, die Werkzeugevaluierung und Auswahl durch ein eigenständiges Projekt umzusetzen. [RuSo2009] setzen hier auf ein zweiphasiges Auswahlverfahren, bei dem zuerst eine Grobauswahl für eine erste potenzielle Werkzeugliste (Longlist) und in einem zweiten Schritt eine Feinauswahl zur Reduktion der Werkzeugliste auf die Favoriten (Shortlist) durchgeführt wird. Auf Basis der Shortlist wird dann eine Auswahlentscheidung getroffen, bei der das Werkzeug dann durch ein Projekt in das Unternehmen eingeführt und eventuell auf die unternehmensspezifischen Anforderungen maßgeschneidert wird (Customizing). Zur Steigerung der Akzeptanz wird darüber hinaus eine erste Nutzung durch ein Pilotprojekt empfohlen. Sollte das Pilotprojekt die Erkenntnis bringen, dass das ausgewählte Werkzeug doch nicht die gewünschte Unterstützung bringt, muss die Werkzeugauswahl wiederholt werden. Wenn kein Werkzeug genau den Anforderungen entspricht, kann statt einer Anpassung eines Werkzeuges auch eine Prozessanpassung erfolgen.

Werkzeugauswahl nach [RuSo2009]:

- **Projekt** zur Werkzeugauswahl aufsetzen
- Festlegung von **Kriterien für eine Grobauswahl**, indem grundlegende Anforderungen formuliert werden
- **Durchführung der Grobauswahl** (Longlist), um erste potenzielle Systeme zu identifizieren
- **Verfeinerung des Kriterienkatalogs** anhand neuer und verfeinerter Anforderungen an das Werkzeug
- **Durchführung einer Feinauswahl** (Shortlist), bis hin zu einem favorisierten Software-Kandidaten
- Optional: Sollte kein Werkzeug genau den Anforderungen entsprechen, ist eine **Werkzeuganpassung** der softwaretechnischen Anwendung notwendig (Customizing).
- Um die Akzeptanz im Unternehmen zu stärken und mögliche letzte Zweifel zu beseitigen, wird abschließend ein Pilotprojekt aufgesetzt.

Tipp: Im Anhang B finden Sie einige nützliche Kriterien zur Werkzeugauswahl auf Basis des RMP.

11.3 Datenaustausch zwischen RM-Werkzeugen

Der Import und Export von Anforderungen, Attributen, Metainformationen, Verknüpfungen und zugehöriger Sichten ist notwendig, um beispielsweise die Zusammenarbeit mit anderen Abteilungen, Partnern und Lieferanten zu unterstützen, die Werkzeuge anderer Anbieter verwenden. Auch bei anstehenden Migrationen von einem Werkzeug auf ein anderes, werden solche Funktionen benötigt.

In den meisten RM-Werkzeugen werden Anforderungen und deren Beziehungen untereinander in herstellerspezifischen (proprietären) Strukturen abgelegt. Hierdurch ist ein einfacher Austausch zwischen zwei RM-Werkzeugen von unterschiedlichen Herstellern (selbst bei identisch definiertem RIM) in der Regel nicht ohne weiteres möglich.

Die Object Management Group (vgl. [OMG2013]) hat den Industriestandard Requirements Interchange Format (ReqIF) definiert. Hiermit können Anforderungs-Artefakte und Metainformationen zwischen Werkzeugen unterschiedlicher Hersteller ausgetauscht werden. In erster Linie wird dies an der Schnittstelle zwischen Kunden und Lieferanten genutzt. Neben dem Austauschformat wird auch eine Vorgehensweise definiert.

Die Initiative stammt aus der Automobilbranche, in der eine enge Zusammenarbeit zwischen den Lieferanten und den Automobilherstellern besteht und genau definierte Versionen von Anforderungs-Artefakten ausgetauscht werden müssen.

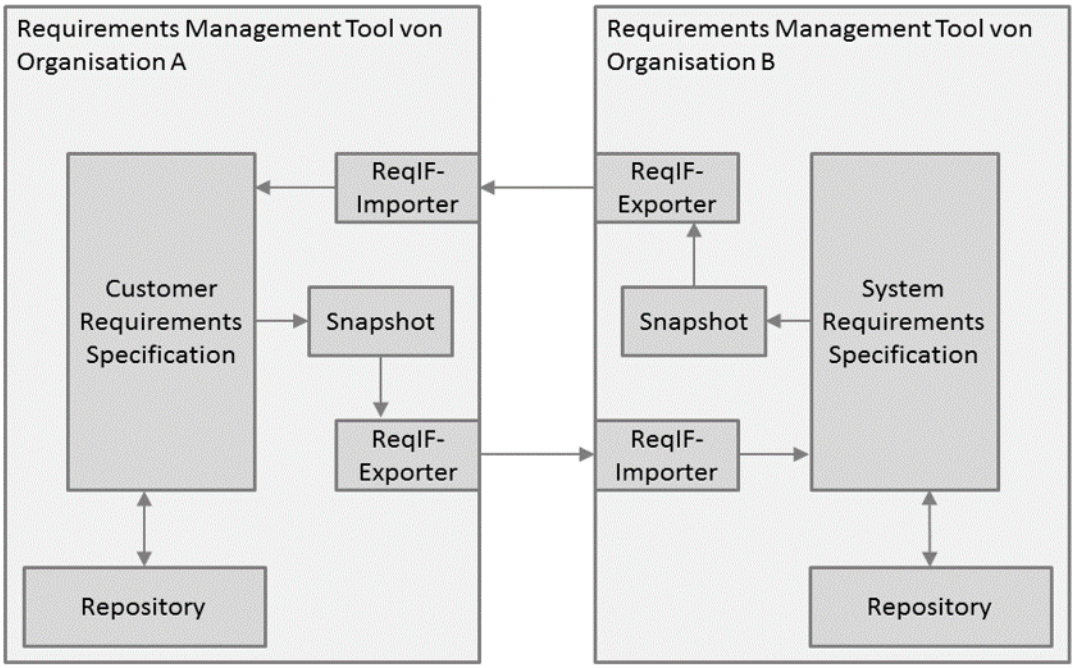


Abbildung 48: Beispiel eines Austauschs von Anforderungs-Artefakten zwischen zwei Organisationen (vereinfacht nach [OMG2013]).

ReqIF ist ein offenes, nicht-proprietäres Format. Es wird in XML-Dokumenten gespeichert. Mittels ReqIF wird also der Austausch von Anforderungen zwischen unterschiedlichen Werkzeugen und Partnern ermöglicht. Voraussetzung ist hier allerdings ein einheitlich abgestimmtes Datenmodell für den Austausch (z.B. ein RIM).

ReqIF bietet hier also folgende Vorteile im Datenaustausch:

- Die Partner müssen nicht mit dem gleichen Werkzeug arbeiten, sodass die Lieferanten nicht für jeden Kunden ein eigenes RE-Werkzeuge besitzen müssen.
- Mit ReqIF kann die Zusammenarbeit zwischen Unternehmen verbessert werden, indem Methoden des RM unternehmensübergreifend angewandt werden.
- Innerhalb einer Organisation können Anforderungen übertragen werden, auch über Werkzeuggrenzen hinweg.
- Mit ReqIF können Anforderungen mit allen Attributen und Metainformationen, im Gegensatz zu Dokumentenexporten in Word, PDF etc., verlustfrei ausgetauscht werden.

In Abbildung 48 ist der Ablauf eines Austausches von Anforderungs-Artefakten zwischen Organisationen dargestellt. Die Spezifikation der Anforderungen („Customer Requirements Specification" in Organisation A und „System Requirements Specification" in Organisation B) wird mit Hilfe eines Repositories versioniert.

Die Werkzeuge haben Schnittstellen zum Ex- und Import der Anforderungen. Mit Hilfe von Snapshots aus der Spezifikation werden Inhalte zwischen den Werkzeugen übertragen. Die Festlegung, wie der konkrete Datenaustausch im Rahmen des Projektes zu erfolgen hat, wird im RMP dokumentiert.

11.4 Inhalte für den RMP

Aus Werkzeugsicht halten Sie im RMP (siehe auch Anhang A) fest, wie Sie RM-Werkzeuge in Ihrem spezifischen Kontext einsetzen möchten. Orientieren Sie sich bei der Beschreibung evtl. an den zuvor beschriebenen Kapiteln. Dokumentieren Sie, was Sie, in welcher Form, durch welches Werkzeug unterstützen bzw. abbilden wollen, z.B.:

1. Kapitel 2 (Requirements-Information-Model) die Beschreibung aller textuellen Anforderungen des RIM sowie die dort festgelegten Detaillierungsebenen soll in DOORS® erfolgen. Die Beschreibung der modellbasierten Anforderungen (Klassendiagramme, BPMN Diagramme) erfolgt in Visual Paradigm.
2. Kapitel 3 (Attributierung und Sichten bei Anforderungen) die Anlage und Pflege der festgelegten Attribute für die in Kapitel 2 festgelegten textuellen Anforderungen erfolgt in DOORS®. Alle benutzerdefinierten Sichten werden in DOORS® definiert und mittels rollenbasierter Zugriffsrechte zugewiesen.

Der RMP beschreibt neben dem Requirements-Information-Model sowie der zu verwendenden Techniken zur Priorisierung, dem Versions- und Änderungsmanagement, der Umsetzung der Verfolgbarkeit, dem ausgewählten Verfahren für das Variantenmanagement, dem eigentlichen RE-Prozess sowie dem Berichtswesen, welche dieser Techniken bzw. Tätigkeiten durch ein bzw. durch welches Werkzeug unterstützt werden soll.

Zusätzlich sollte im RMP beschrieben werden, zwischen welchen Parteien Anforderungen ausgetauscht werden müssen und auf welchem Weg der Austausch mittels definierter Importe und Exporte zu erfolgen hat (vgl. Kapitel 11.3).

11.5 Vertiefende Literatur

[OMG2013] OMG: Requirements Interchange Format (ReqIF). Object Management Group, Version 1.1., 2013, https://www.omg.org/spec/ReqIF/1.1/PDF.

[Pohl2010] K. Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. Springer, 2010.

[RuSo2009] C. Rupp & die SOPHISTen: Requirements-Engineering und -Management. Hanser, 5. aktualisierte und erweiterte Auflage, 2009.

12 Abkürzungsverzeichnis

AC	actual cost
AHP	Analytical Hierarchy Process
BAC	Budget at Completion
CAB	Change Advisory Board
CCB	Change Control Board
CMMI	Capability Maturity Model Integration
CPRE	Certified Professional for Requirements Engineering
CR	Change Request
DIN	Deutsches Institut für Normung
EV	earned value
FDD	Feature-Driven Development
GQM	Goal-Question-Metric
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IREB	International Requirements Engineering Board
ISO	International Organization for Standardization
IT	Informationstechnologie
ITIL	IT Infrastructure Library
KPV	Kontinuierliche Prozessverbesserung
MMP	Minimal Marketable Product
MVP	Minimal Viable Product
PDCA	Plan, Do, Check, Act
PV	Planned Value
RE	Requirements Engineering
RAC	responsible, accountable, consulted, informed
RIM	Requirements-Information-Model
RM	Requirements Management
RMP	Requirements-Management-Plan
TBD	to be determined
TBR	to be resolved
UML	Unified Modelling Language
XP	Extreme Programming

13 Literaturverzeichnis

AgileManifesto-Manifesto for Agile Software Development. Abrufbar unter http://agilemanifesto.org/. Zuletzt besucht im März 2024.

[Ande2010] David J. Anderson: Kanban. Successful Evolutionary Change for Your Technology Business. Blue Hole Press, Sequim, Washington 2010.

[BaWe1984] Victor R. Basili, David M. Weiss: A methodology for collecting valid software engineering data. Software Engineering, IEEE Transactions on 6 (1984): 728-738.

[Basi1992] V.R. Basili: Software Modeling and Measurement: The Goal Question Metric Paradigm. Computer Science Technical Report Series, CS-TR-2956 (UMIACS-TR-92-96), University of Maryland, College Park, MD, September 1992.

[BBHK2014] Braun, P.; Broy, M.; Houdek, F.; Kirchmayr, M.; Müller, M.; Penzenstadler, B.; Pohl, K.; Weyer, T.: Guiding Requirements Engineering for software-intensive embedded systems in the automotive industry. Computer Science - R&D 29(1): (2014).

[BCR] Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach: The Goal Question Metric Approach. Tutorial, University of Maryland, http://www.cs.umd.edu/~mvz/handouts/gqm.pdf. Zuletzt besucht im März 2024.

[Beck2000] Kent Beck: Extreme programming explained. Addison-Wesley, Upper Saddle River, 2000.

[Beim2012] Martin Beims: IT-Service Management mit ITIL®: ITIL® Edition 2011, ISO 20000:2011 und PRINCE2® in der Praxis, Carl Hanser Verlag GmbH & Co. KG, 3rd updated edition, 2012 (available in German only).

[BSB2008] Christoph Bommer, Markus Spindler, Volkert Barr: Softwarewartung – Grundlagen, Management und Wartungstechniken. Dpunkt.verlag, 2008

[Bout2011] E. Boutkova: Experience with Variability Management in Requirement Specifications. In: D.E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid (eds): Software Product Lines – 15th International Conference (SPLC), Munich, 2013.

[BoHo2011] E. Boutkova, F. Houdek: Semi-automatic identification of features in requirement specifications. In: Proceedings of the 19th International Requirements Engineering Conference, Trento, Italy, September 2011.

[BWJ2013] Franz J. Brunner, Johann Wappis, Berndt Jung: Null-Fehler-Management: Umsetzung von Six Sigma, Carl Hanser Verlag GmbH & Co. KG; edition: 4, revised and extended edition, 2013 (available in German only).

[BLP2004] S. Bühne, K. Lauenroth, K. Pohl: Why is it not Sufficient to Model Requirements Variability with Feature Models. In: Aoyama, M.; Houdek, F.; Shigematsu, T.

(eds) Proceedings of Workshop: Automotive Requirements Engineering (AURE04). IEEE Computer Society Press, Los Alamitos 2004.

[CMMI] https://insights.sei.cmu.edu/library/cmmi-for-development-version-13/. Zuletzt besucht im März 2024.

[CNAT2011] J.M. Carillo de Gea, J. Nicolás, J.L.F. Alemán, A. Toval, C. Ebert, A. Vizcaíno: Requirements Engineering Tools In: IEEE Software, July/ August 2011.

[CNAT2012] J.M. Carillo de Gea, J. Nicolás, J.L.F. Alemán, A. Toval, C. Ebert, A. Vizcaíno: Requirements Engineering Tools: Capabilities, survey, and assessment. In: Information and Software Technology, Volume 54, Issue 10, October 2012.

[CKS2011] Mary Beth Chrissis, Mike Konrad, Sandy Shrum: CMMI 1.3 für die Entwicklung: Richtlinien für Prozessintegration und Produktverbesserung, Addison-Wesley Verlag, 2011.

[Cock1997] A. A. Cockburn: Surviving Object-Oriented Projects. Addison-Wesley, 1997.

[Cock2001] A. Cockburn: Use Cases effektiv erstellen. Addison-Wesley, 2001,

[Cock2004] A. Cockburn: Crystal Clear, A Human-Powered Methodology for Small Teams. Addison-Wesley, 2004.

[Cock2006] A. Cockburn: Agile Software Development. Addison-Wesley, 2006.

[ClNo2007] P. Clements, L. Northrop: Software Product Lines: Practices and Patterns. Addison Wesley, Boston, 6. Auflage, 2007.

[CHW1998] J. Coplien, D. Hoffmann, D. Weiss: Commonality and Variability in Software Engineering. In: IEEE Software, Volume 15, Issue 6, 1998.

[CzEi2000] K. Czarnecki, U.W. Eisenecker: Generative Programming: Methods, Tools, and Applications. Addison Wesley, 2000.

[CHQW2022] Thorsten Cziharz, Peter Hruschka, Stefan Queins, Thorsten Weyer: Handbook Requirements Modeling, Education and Training for IREB Certified Professional for Requirements Engineering, Advanced Level Requirements Modeling, IREB, Version 2.0.0, July 1, 2022.

[Davi2003] A. Davis: The Art of Requirements Triage. IEEE Computer, Volume 36, Issue 3, 2003.

[Davi2005] Alan M. Davis: Just Enough Requirements Management – Where Software Development Meets Marketing. Dorset House Publishing, 2005.

[DeMa1982] Tom DeMarco: Controlling Software Projects: Management, Measurement, and Estimation. Prentice Hall/Yourdon Press, 1982

[DeMa2009] Tom DeMarco: Software Engineering: An Idea Whose Time Has Come and Gone? IEEE Software, July/August 2009.

[Demi1982] W.E. Deming: Out of the Crisis. Massachusetts Institute of Technology, Cambridge 1982.

[DIN61508] IEC DIN EN 61508-2 Funktionale Sicherheit sicherheitsbezogener elektrischer /elektronischer/ programmierbarer elektronischer Systeme. VDE Verlag, 2002.

[DIN69900] DIN 69900 Projektmanagement – Netzplantechnik; Beschreibungen und Begriffe, 2009.

[Ebel2014] N. Ebel: ITIL®(R) 2011 Edition: Grundlagen und Know-how für das IT Service Management und die ITIL®(R)-Foundation-Prüfung, dpunkt.verlag GmbH, 1. Auflage, 2014.

[Eber2012] C. Ebert: Systematisches Requirements Engineering. Dpunkt, 4. Auflage, 2012.

[Ecks2004] J. Eckstein: Agile Software Development in the Large. Dorset House Publishing, 2004.

[Ecks2010] J. Eckstein: Agile Software Development with Distributed Teams. Dorset House Publishing, 2010.

[Gabl2014a] Springer Gabler Verlag (Herausgeber), Gabler Wirtschaftslexikon, Stichwort: Produktfamilie, online im Internet: (Stand: 11.

[Gabl2014b] Springer Gabler Verlag (Herausgeber), Gabler Wirtschaftslexikon, Stichwort: Produktlinie, online im Internet: https://wirtschaftslexikon.gabler.de/definition/produktlinie-43488/version-176677. Zuletzt besucht im März 2024.

[GoFi1994] O.C.Z. Gotel, A.C.W Finkelstein: An Analysis of the Requirements Traceability Problem. Proceedings of IEEE International Conference on Requirements Engineering, 1994.

[Glin2014] Martin Glinz: A Glossary of Requirements Engineering Terminology. Version 1.6 May 2014.

[Herr2014] A. [Herr2014] A. Herrmann: Leichte Dellen – Wenn agil nicht geht: Feature Driven Development. iX 9/2014, S. 110-113.

[HJD2011] E. Hull, K. Jackson, J. Dick: Requirements Engineering. Springer, 3rd Ed, 2011.

[HuMa2011] Thomas Hummel, Christian Malorny: Total Quality Management: Tipps für die Einführung. Carl Hanser Verlag GmbH & Co. KG; Auflage: 4., vollständig überarbeitete Auflage, 2011.

[IEEE830] IEEE: IEEE 830-1998 Recommended Practice for Software Requirements Specifications, 1998.

[IEEE1233] IEEE: IEEE Standard 1233 Guide for Developing of System Requirements Specifications, 1998.

[IREB2015] IREB: Syllabus IREB Certified Professional for Requirements Engineering – Foundation Level, Version 2.2, 2015.

[ISO9000] ISO: ISO 9000-1 Quality systems – Model for Quality Assurance in Design, Development, Production, Installation and Servicing. International Organization for Standardization (ISO), 1994.

[ISO9241] ISO: DIN EN ISO 9241 Ergonomie der Mensch-System-Interaktion.

[ISO12207] ISO: ISO/IEC 12207: 1995, Information Technology – Software life cycle processes. International Organization for Standardization (ISO), 1995.

[ISO29148] ISO: ISO/IEC/IEEE 29148:2018: Systems and software engineering – Life cycle processes – Requirements engineering, 2018.

[ISO14102] ISO/ IEC 14102:1995 Information Technology – Evaluation and Selection of CASE Tools, 1995.

[ISO15288] ISO/IEC 15288:2008 Systems and software engineering — System life cycle processes, 2008.

[ISO24766] ISO: ISO/IEC TR 24766:2009: Information technology – Systems and software engineering – Guide for requirements engineering tool capabilities. International Organization for Standardization (ISO), 2009.

[ISO25010] ISO: ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, 2011.

[ISO29110] ISO: ISO 29110 Lifecycle process standard for Very Small and Medium Entities (VSME), 2011.

[Oran2013] Foundations of IT Service Management with ITIL®2011. 2nd Edition ITILyaBrady, 2013 (Kindle Edition).

[JSB2011] I. Jacobson, I. Spence, K. Bittner: Use Cases 2.0. Ivar Jacobson International, 2011.

[KCHN1990] C. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson: Feature-Oriented Domain Analysis (FODA) – Feasibility Study. Software Engineering Institute, 1990.

[KKLK1998] K. Kang, S. Kim, J. Lee, K. Kim, E. Shin, M. Huh: FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering, Nr. 5, 1998.

[KLD2002] K. Kang, J. Lee, P. Donohoe: Feature-Oriented Product Line Engineering. IEEE Software 19(4): 2002.

[Kano1984] N. Kano et al: Attraktive Qualität und Must-be-Quote. Journal of the Japanese Society for Quality Control, H. 4, 1984.

[KaRy1997] J. Karlsson, K. Ryan: A Cost-Value Approach for Prioritizing Requirements. IEEE Software 14, Nr. 5, 1997.

[Kerz2003] H. Kerzner: Project Management. Ein systemorientierter Ansatz zur Planung und Steuerung. Mitp-Verlag, Bonn, 2003.

[Kneu2007] Ralf Kneuper: CMMI: Verbesserung von Software- und Systementwicklungsprozessen mit Capability Maturity Model Integration. Dpunkt.verlag, 3. Auflage, 2007.

[KoBe2013] H.-P. Korn and J.P. Berchez (eds.): Agiles IT-Management in großen Unternehmen. Symposion Publishing, 2013.

[Küpp2005] H.-U. Küpper: Controlling: Konzeption, Aufgaben, Instrumente. Schäffer-Poeschel, 4th edition, 2005.

[KuSt2001] K. Kurbel, E. Stickel: Informationsmanagement. Oldenbourg Wissenschaftsverlag, 2001.

[Leff2011] D. Leffingwell: Agile Software Requirements, Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional, 2011.

[LeWi2000] D. Leffingwell, D. Widrig: Managing Software Requirements – A Unified Approach. Reading, Addison-Wesley, 2000.

[LoKa1995] P. Loucopoulos, V. Karakostas: System Requirements Engineering. McGraw-Hill, 1995.

[MGP2009] P. Mäder, O. Gotel, I. Philippow: Getting Back to Basics: Promoting the Use of a Traceability Information Model in Practice. Proceedings of 5th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE2009), Vancouver, Canada, May 2009.

[MJZC2013] P. Mäder, P.L. Jones, Y. Zhang, J. Cleland-Huang: Strategic Traceability for Safety-Critical Projects. IEEE Software, Volume 30, Issue 3, May / June 2013.

[Mois2002] F. Moisiadis: The fundamentals of prioritizing requirements. Systems Engineering, Test & Evaluation Conference, Sydney, October 2002.

[Nebu2014] Nebulon Pty. Ltd: Feature Driven Development. http://www.featuredrivendevelopment.com/. Zuletzt besucht im März 2024.

[Nuse2001] B. Nuseibeh: Weaving the Software Development Process between Requirements and Architecture. Proceedings of ICSE2001 Workshop STRAW-01, Toronto, May 2001.

[OMG2013] OMG: Requirements Interchange Format (ReqIF). Object Management Group, Version 1.1., 2013, http://www.omg.org/spec/ReqIF/1.1/. Zuletzt besucht im März 2024.

[PaFe2002] Stephen R. Palmer, John M. Felsing: A Practical Guide to the Feature-Driven Development. Prentice Hall International, 2002.

[Patt2008] Jeff Patton: The new user story backlog is a map, 10/08/2008 http://www.agileproductdesign.com/blog/the_new_backlog.html. Zuletzt besucht im März 2024.

[PHAB2012] Pohl, K., Hönninger, H., Achatz, R., Broy, M. (Eds.): Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology, Springer 2012.

[Pich2014] Roman Pichler: The Product Vision Board, https://www.romanpichler.com/tools/product-vision-board/. Zuletzt besucht im März 2024.

[PMI2013] PMI: Project Management Book of Knowledge (PMBOK). Project Management Institute, 5th Ed., 2013.

[Pohl1996] K. Pohl: Process-Centered Requirements Engineering. John Wiley Research Science Press, 1996.

[Pohl2010] K. Pohl: Requirements Engineering: Grundlagen, Prinzipien, Techniken. Springer, 2010.

[PBL2005] K. Pohl, G. Böckle, F. van der Linden: Software Product Line Engineering – Foundations, Principles, and Techniques. Springer, 2005.

[PoRu2015] K. Pohl and Chris Rupp: Requirements Engineering Fundamentals – A Study Guide for the Certified Professional for Requirements Engineering Exam – Foundation Level – IREB compliant. dpunkt.verlag GmbH, 2015.

[PoPo2003] Mary Poppendieck, Tom Poppendieck: Lean Software Development. Addison Wesley, 2003.

[PriEsT] PriEsT, http://sourceforge.net/projects/priority/. Zuletzt besucht im März 2024.

[RBSP2002] M. Riebisch, K. Böllert, D. Streitferdt, and I. Philippow: Extending Feature Diagrams with UML Multiplicities. in Proc. World Conf. Integrated Design and Process Technology (IDPT), 2002.

[Ries2011] Eric Ries: The Lean Startup: How Constant Innovation Creates Radically Successful Businesses. Penguin, 2011.

[RoRo2014] S. Robertson, J. Robertson: Mastering the Requirements Process – Getting Requirements Right. Addison-Wesley, 3rd Edition, 2014.

[RuSo2009] C. Rupp & die SOPHISTen: Requirements-Engineering und -Management. Hanser, 5. aktualisierte und erweiterte Auflage, 2009.

[Saat1990] Thomas L. Saaty: Multicriteria decision making – the analytic hierarchy process. Planning, priority setting, resource allocation. 2. Auflage, RWS Publishing, Pittsburgh 1990.

[Schi2001] B. Schienmann: Kontinuierliches Anforderungsmanagement. Prozesse – Techniken – Werkzeuge. Addison-Wesley, 2001,

[SHT2006] P.-Y. Schobbens, P. Heymans, J.C. Trigaux: Feature Diagrams: A Survey and a Formal Semantics. Proceedings of the 14th International Requirements Engineering Conference (RE'06), September 2006.

[ScBe2001] Ken Schwaber, Mike Beedle: Agile Software Development with SCRUM. Prentice Hall, 2001.

[ScSu2013] Ken Schwaber, Jeff Sutherland: The Scrum Guide, July 2013 http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf. Zuletzt besucht im März 2024.

[SEI1999] Carnegie Mellon SEI: The Capability Maturity Model, Guidelines for Improving the Software Process. Addison Wesley, 1999.

[SEI2010] Carnegie Mellon SEI: CMMI for Services, Version 1.3, Improving processes for providing better services. 2010.

[SEI2011] SEI: CMMI® for Development, Version 1.3 CMU/SEI-2010-TR-033. Abrufbar unter https://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15287.pdf. Zuletzt besucht im März 2024.

[SMK2013] S. Siraj, L. Mikhailov, J.A. Keane: PriEsT: an interactive decision support tool to estimate priorities from pairwise comparison judgments. International Transactions in Operational Research, 2013.

[SoSa1997] I. Sommerville, P. Sawyer: Requirements Engineering: A Good Practice Guide. John Wiley & Sons, 1997.

[SpLi2007] A. Spillner, T. Linz: Basiswissen Softwaretest – Aus- und Weiterbildung zum Certified Tester. Dpunkt.verlag, 3. Auflage, 2007.

[SuSc2013] J. Sutherland, K. Schwaber: Scrum Guide, July 2013, available at www.scrum.org.

[Syra2014] Ingo Geppert, Torsten Lodderstedt: Projektanforderungsmanagement – Eine pragmatische Lösung für effiziente Toolunterstützung. Zuletzt besucht im März 2024.

[Tava2012] Serkan Tavasli: Six Sigma Performance Measurement System: Prozesscontrolling als Instrumentarium der modernen Unternehmensführung. Deutscher Universitätsverlag, 2012.

[USCo2002] US Congress: Sarbanes-Oxley Act. Washington, USA, 107th Congress of the United States of America, 23.01.2002.

[VanL2009] A. van Lamsweerde: Requirements Engineering – from System Goals to UML Models to Software Specifications. John Wiley and Sons, 2009.

[VDI2001] VDI: VDI guideline 2519 sheet 1 – The procedure for the creation of tender and performance specifications, 2001.

[Wann2013a] Roland Wanner: Earned Value Management: The Most Important Methods and Tools for an Effective Project Control. CreateSpace Independent Publishing Platform, 2013.

[Wann2013b] Roland Wanner: Earned Value Management: So machen Sie Ihr Projektcontrolling noch effektiver Taschenbuch. CreateSpace Independent Publishing Platform, 3. Auflage, 2013

[Wieg2005] Karl Wiegers: Software Requirements. Microsoft Press Deutschland, 1. Auflage, 2005.

[WiBe2013] K. Wiegers, J. Beatty: Software Requirements. 3rd Edition. Microsoft Press, 2013.

[Youn2014] R. Young: The Requirements Engineering Handbook, Artech House, Boston, 2004.

[Zieg1998] K. Ziegbein: Controlling. Kiehl Friedrich Verlag, 6. Auflage, 1998.

Index

Anhang A: Vorlage RMP

Wie in der Einleitung unseres Handbuchs IREB Certified Professional for Requirements Engineering Modul "Requirements Modeling" erwähnt, muss der Requirements Manager bereits zu Beginn den RE-Prozess planen. Das Ergebnis dieser Festlegungen dokumentiert er in einem **Requirements-Management-Plan (RMP)**. Im Verlauf des Buchs haben wir die zu treffenden Entscheidungen diskutiert und anhand einer Fallstudie Auszüge eines beispielhaften RMP erstellt. Dieser Anhang präsentiert nun zusammenhängend eine Vorlage für einen RMP, dessen Kapitelüberschriften und eine kurze Kapitelbeschreibung. Sie können als Requirements Manager Ihren RMP nach diesem Schema aufbauen.

Certified Professional for Requirements Engineering

Requirements-Management-Plan für <Ihr Projekt>

<Ihr Name>

Basierend auf: IREB CPRE Requirements Management Practitioner | Specialist

2.1.0 | 5. April 2024

Inhaltsverzeichnis

1 RE- und RM-Prozess

Definieren Sie in diesem Kapitel den Prozess, nach dem Sie Anforderungen ermitteln, dokumentieren, prüfen, abstimmen und verwalten wollen. Legen Sie dabei folgende Parameter fest:

- Zeitverlauf der Ermittlung, d.h. upfront versus iterativ
- Detaillierungstiefe der Dokumentation, d.h. der höchste Detailgrad, auf dem spezifiziert wird (mit den beiden Extremen schwergewichtige versus leichtgewichtige Spezifikation)
- Eingliederung von Änderungen, konkret: Änderungsantrag versus Product Backlog
- Verteilung der Verantwortlichkeit

Dokumentieren Sie den Prozess z.B. als Aktivitätsdiagramm, als RACI-Matrix oder Gantt-Diagramm.

Dokumentieren Sie darüber hinaus, wie der RE-Prozess überwacht wird (z.B. mit Hilfe von welchem Bericht und welchen Kennzahlen). Auch Maßnahmen zur Prozessverbesserung können hier vorgesehen werden, z.B. Lessons Learned Analysen nach Projektende.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 9.

1.1 RE- und RM-Werkzeuge

Legen Sie fest, welches Werkzeug bzw. welche Werkzeuge in Ihrem Projekt verwendet werden sollen, um den RE-Prozess zu unterstützen. Die Werkzeuge können alternativ auch im Prozessmodell dokumentiert sein.

Weitere Details zu Werkzeugen und speziell zur Werkzeugauswahl finden Sie im Handbuch unter Kapitel 11.

1.2 Requirements Information Model

Legen Sie in diesem Kapitel Ihre Anforderungs-Landschaft fest und beschreiben Sie, wie Sie Ihre Anforderungen dokumentieren wollen. Hierzu gehört beispielsweise:

- Welche Anforderungsarten wollen Sie betrachten?
- Wie wollen Sie diese Anforderungen dokumentieren?
- Wie detailliert werden Sie die Anforderungen beschreiben und welche Detaillierungsebenen sollen betrachtet werden?
- Welches Formalitätsniveau müssen Ihre Anforderungen erreichen?

Dies kann beispielsweise in Form eines Requirements-Information-Model RIM dokumentiert werden.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 2.

1.3 Attributierungsschema

Dokumentieren Sie in diesem Kapitel z.B. als Tabelle oder Informationsmodell, welche Attribute Ihre Anforderungen haben sollen:

- Name des Attributs
- Bedeutung
- Verantwortliche Person
- Erlaubte Werte
- Default-Wert
- Pflichtfeld sowie
- Abhängigkeiten zwischen Attributen

Die Attribute können in Abhängigkeit der Anforderungsart oder Detaillierungsebene unterschiedlich sein. Zum Attributierungsschema gehören auch Attribute, die zur Priorisierung der Anforderungen dienen sollen.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 3.

1.4 Priorisierung

Dokumentieren Sie hier bereits zu Beginn, nach welchen Kriterien Sie Priorisierungen vornehmen wollen. In der Regel sind Priorisierungen notwendig, um auf einen bestimmten Zeitplan oder ein bestimmtes Budget hinzuarbeiten – im Zweifel sogar beides. Legen Sie in diesem Kapitel fest, nach welchen Kriterien Ihre Anforderungen priorisiert werden sollen, auch wann, von wem und mit welcher Priorisierungsmethode.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 4.

1.5 Verfolgbarkeit

In diesem Kapitel dokumentieren Sie die Verfolgbarkeits-Strategie: Verfolgbarkeitsziel, Nutzungsstrategie, Aufzeichnungs-Strategie und das projektspezifische Verfolgbarkeits-Modell.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 6.

1.6 Sichten und Berichte

Auf Basis der in Kapitel 4 beschriebenen Attribute für Anforderungen lassen sich in diesem Kapitel die notwendigen Sichten und deren Inhalte definieren. Beschreiben Sie hier auch, wer (welche Stakeholder) die jeweilige Sicht aus welchem Grund (Ziel) wann benötigt. Der Inhalt einer Sicht entsteht z.B. durch die Filterung und Sortierung der Anforderungen nach Attributen.

Legen Sie hier ebenfalls fest, welche (anforderungsbezogenen) Berichte wann erstellt werden. Für jeden Bericht sind der Berichtsempfänger und das Ziel des Berichts dokumentiert, z.B. tabellarisch. Die Herleitung von Berichtsinhalten aus Zielen kann als Goal-Question-Metric-Baum grafisch dargestellt werden. Es ist auch festgelegt, wie diese Inhalte aus welchen Attributen ermittelt oder berechnet werden und wie sie dargestellt werden (z.B. mit welcher grafischen Darstellung). Die Festlegung kann auch in Form einer Berichtsvorlage oder Sicht dokumentiert sein.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 3 und 8.

1.7 Versionierung

Dokumentieren Sie hier, wie Sie in Ihrem Projekt Anforderungen und Dokumente versionieren wollen. Legen Sie fest, welche Zustände eine Anforderung einnehmen darf, wie die Zustandsübergänge zu erfolgen haben und wer den Zustand von Anforderungs-Artefakten ändern darf.

Legen Sie darüber hinaus fest, auf welcher Basis eine Anforderungs-Basislinie erstellt wird und was es für den folgenden Anforderungsmanagement-Prozess bedeutet, sobald eine Anforderungs-Basislinie erstellt wurde, z.B. nach einer Anforderungs-Basislinie werden Änderungen nur noch über einen Änderungsmanagement-Prozess akzeptiert. Legen Sie grundsätzlich im RMP fest, wie Sie mit Änderungen im Projekt umgehen wollen, wie diese zu dokumentieren sind, ob es ein Änderungsgremium gibt, wie dieses besetzt wird usw.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 5.

1.8 Änderungsprozess

In diesem Kapitel beschreiben Sie Ihren Änderungsmanagement-Prozess und die dazugehörigen Dokumente. Beschreiben Sie hier, wer Anforderungsänderungen beantragen kann, wie diese beantragt, bewertet und entschieden werden, d.h. durch wen, wann und nach welchen Kriterien. Zum Änderungsprozess kann auch eine Vorlage für eine Anforderungsänderung (Change Request) gehören, die festlegt, welche Informationen für eine Anforderungsänderung zu ermitteln und dokumentieren sind.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 5.

1.9 Variantenmanagement

Definieren Sie in diesem Kapitel, ob und wie Sie Variabilität – also Variations-Punkte, Varianten und deren Abhängigkeiten – in Ihren Anforderungen dokumentieren wollen. Dies kann beispielsweise eine Dokumentation im Text, als orthogonales Modell oder Merkmals-Modell sein.

Weitere Details zum Inhalt finden Sie im Handbuch unter Kapitel 7.

Anhang B (Werkzeugauswahl)

In diesem Anhang beschreiben wir Ihnen einige Kriterien zur Werkzeugauswahl. Diese Kriterien entstammen zum einen der Literatur und zum anderen dem in diesem Handbuch Schritt für Schritt erstellten RMP und der Frage, welche der im RMP beschriebenen Tätigkeiten, Prozesse bzw. Techniken durch ein Management Werkzeug unterstützt werden sollten / könnten. Hierzu haben wir einen Kriterienkatalog erstellt und diesen anhand drei gänzlich unterschiedlicher Werkzeuge angewendet. Diese stehen jeweils stellvertretend für eine Werkzeug-Kategorie. Die Bewertungen entsprechen dem Stand von 2015 und veralten natürlich, sobald die Werkzeuge weiterentwickelt werden. Microsoft Word ® und Microsoft Excel ® empfehlen wir natürlich nicht für das RM, weil sie viele der RM-Aktivitäten nicht gut unterstützen.

Wir wünschen viel Spaß beim Studieren und Ausprobieren.

Inhaltsverzeichnis

1 Herausforderungen bei der Einführung und Nutzung von Werkzeugen

Die Einführung eines (neuen) Requirements Management Werkzeugs ist wie jede Applikations-Einführung ein Thema, bei dem neben den fachlichen und methodischen Aspekten vor allem auch menschliche Aspekte berücksichtigt werden müssen – denn Menschen sind diejenigen, die später mit dem Werkzeug arbeiten müssen. Nur ein akzeptiertes Werkzeug, das dem Nutzer einen direkten oder indirekten Vorteil bringt, wird auf Dauer akzeptiert und korrekt verwendet werden. Mit der Einführung eines Requirements Management Werkzeugs sind soziale und kognitive, organisatorische und unternehmerische Aspekte zu beachten.

Zur Illustration wollen wir einige Herausforderungen der Werkzeugeinführung beispielhaft beschreiben. Diese Beispiele haben wir den in [PoRu2011a] eingeführten Sichten zugeordnet.

Anbietersicht

- **Marktposition:** Die Auswahl des Werkzeuges bedeutet für ein Unternehmen in der Regel eine längerfristige Festlegung, da mit der Werkzeugeinführung oftmals auch organisatorische Umstellungen notwendig werden. Mit der Festlegung auf ein Werkzeug erfolgt auch eine Bindung an dessen Anbieter. Existiert dieser Anbieter oder das Werkzeug bald nicht mehr, wird guter Rat buchstäblich teuer.
- **Trends und Updates:** Es kann erwünscht sein, dass der Anbieter zukünftige Trends unterstützt und in seinem Werkzeug anbietet. Das Folgen von Trends kann aber auch unerwünscht sein, falls der Anbieter seine Strategie vollständig umstellt und bspw. nur noch auf agile Vorgehensweisen setzt.

Benutzersicht

- **Nutzerakzeptanz**: Das Werkzeug muss für viele Nutzergruppen zur Verfügung stehen, um den Entwicklungszyklus abbilden zu können (Marketing, Entwicklung, Finanzabteilung etc.). Aus diesem Grunde ist die Akzeptanz der Nutzer von äußerster Wichtigkeit für die korrekte spätere Nutzung. Dabei geht es nicht nur darum, dass ihre Arbeitsprozesse unterstützt werden, sondern dies auf eine effiziente und ergonomische Weise geschieht. Das Werkzeug muss also insbesondere auch benutzerfreundlich zu verwenden sein.

Betriebswirtschaftliche Sicht

- **Das Lizenzmodell** der Software muss zum Nutzungsprofil und zur Kostenstruktur des Unternehmens passen.
- **Die Betreibbarkeit** der Software muss zum einen mit der Kostenstruktur und zum anderen mit der bestehenden IT-Infrastruktur des Unternehmens vereinbar sein.

Produktsicht

- **Die Funktionsmerkmale** der Software müssen den an das Werkzeug gestellten Anforderungen genügen. Das bedeutet zum einen, dass die für Sie relevanten Kriterien des RMP (Attributierung, Sichtenbildung, Verfolgbarkeit, Berichterstellung etc.) erfüllt werden müssen. Zum anderen bedeutet dies aber auch, dass das Werkzeug in allen Bereichen die notwendige Benutzerunterstützung leisten muss. Es ist in der Regel ein Irrglaube, dass eine Funktionalität nur unterstützt werden muss, egal wie, und der Nutzer sich schon damit abfindet. Nehmen wir hier gern das Beispiel der Verfolgbarkeit: Falls das Werkzeug zwar grundsätzlich das Erzeugen von Verfolgbarkeits-Beziehungen zwischen Artefakten unterstützt, die Erstellung aber nicht benutzerorientiert und für den Benutzer zeitsparend ist, werden diese Beziehungen wahrscheinlich nicht oder nur unzureichend gesetzt werden. Bei der Vielzahl an (möglichen) funktionalen und nichtfunktionalen Anforderungen an das RM-Werkzeug fällt es schwer bis unmöglich, das perfekte Werkzeug zu finden. Sie müssen darum Ihre Auswahlkriterien geschickt priorisieren und dürfen nur auf die wirklich unwichtigen verzichten.

Projektsicht

- **Anpassbarkeit:** Kein Projekt ist wie das andere. Genauso wie der RE-Prozess und die Anforderungs-Dokumente an die Größe und andere Eigenschaften des Projektes angepasst werden müssen, so muss auch das Werkzeug diese Anpassung mittragen. Tut es das nicht, ist es nur in manchen Projekten nützlich, in anderen nicht.

Prozesssicht

- **Werkzeug folgt dem Vorgehen**: Vor der Werkzeugauswahl muss das Vorgehen im Unternehmen klar und etabliert sein, z.B. auf Basis eines vorhandenen RMP für das Unternehmen. Das Werkzeug hilft nur, ein vorhandenes Vorgehen zu unterstützen. Gibt es kein definiertes Vorgehen, fällt die Werkzeugauswahl schwer und schränkt die möglichen RM-Vorgehensweisen für später ein.
- Methodisches Wissen: Die Verwendung eines Werkzeugs allein stellt nicht sicher, dass nur korrekte Daten bzw. Anforderungen erfasst werden. Es unterstützt jedoch die Erfassung und Pflege von Daten. Aus dem Grunde ist es auch bei der Verwendung von Werkzeugen zwingend notwendig, dass alle am Entwicklungsprozess Beteiligten ausreichend geschult sind, um Anforderungen korrekt zu dokumentieren.
- **Projektspezifische Datenmodelle (RIMs)** müssen durch das Werkzeug abgebildet werden können. Zur Austauschbarkeit ist hier eventuell eine vorherige Vereinheitlichung der Datenmodelle erforderlich.

Technische Sicht

- **Austauschbarkeit von Daten:** Bei der Zusammenarbeit mit anderen Abteilungen, Partnern, Lieferanten stellt die Austauschbarkeit von Daten eine besondere Herausforderung dar, da der Austausch zwischen heterogenen RIMs und unterschiedlichen Werkzeugen sichergestellt werden muss. Auch der Export und Import bei der Migration auf ein neues Werkzeug oder Werkzeug-Release muss möglich sein.

2 Kriterien für die Auswahl eines Requirements-Management-Werkzeugs

Bei der Auswahl eines RM-Werkzeugs stellt sich wie bei jeder Softwareeinführung in einem Unternehmen die Frage nach den Anforderungen. Was soll also durch das Werkzeug unterstützt werden? Wie soll die Integration stattfinden? Wer soll die Software betreiben? Gibt es bestehende Systeme, die abgelöst werden müssen? Ist eine Migration notwendig? Und so weiter.

Die Literatur bietet hier bereits eine Reihe hilfreicher Anhaltspunkte, Checklisten und Fragen, die Sie bei der Einführung eines RM-Werkzeuges unterstützen können [PoRu2011a], [CNAT2011], [CNAT2012], [ISO24766], [Eber2012].

In [PoRu2011a] wird ein sichten-basiertes Vorgehen vorgeschlagen, um so die Anforderungen an das Werkzeug von allen relevanten Stakeholdern (analog einer Software-Einführung) berücksichtigen zu können.

- **Anbietersicht:** Die Anbietersicht betrachtet unter anderem die Marktposition und die Dienstleistungsmöglichkeiten (im Sinne von Schulungen, Benutzersupport, unternehmensspezifische Anpassungen etc.) des Werkzeuganbieters. Diese Sicht ist notwendig, da die Einführung einer Werkzeugunterstützung i.d.R. eine längerfristige Bindung bedeutet.
- **Benutzersicht:** Die Benutzersicht betrachtet die Anforderungen, die sich aus der Sicht der unterschiedlichen Systemnutzer ergeben. Hierunter fallen beispielsweise Anforderungen an Rollenkonzepte, Mehrbenutzerfähigkeit etc.
- **Betriebswirtschaftliche Sicht:** Die betriebswirtschaftliche Sicht betrachtet die gesamten Kosten als Vollkostenrechnung, die für die Einführung, den Betrieb und die laufenden Kosten für Lizenzen und Support etc. benötigt werden.
- **Produktsicht:** Die Produktsicht betrachtet die Funktionalitäten, die das einzuführende Werkzeug benötigt, um das RM zu unterstützen. Hierunter fallen beispielsweise Anforderungen an die Attributierung, Sichtenbildung, Verfolgbarkeit.
- **Projektsicht:** Die Projektsicht betrachtet, inwieweit das Werkzeug zukünftige Projekte z.B. im Hinblick auf die Planung, Berichterstellung etc. unterstützen kann
- **Prozesssicht:** Die Prozesssicht betrachtet Anforderungen an die methodische Unterstützung, die an das Werkzeug gestellt werden, z.B. durch geeignete Workflows. Hier sollte allerdings darauf geachtet werden, dass nicht das Werkzeug die Methodik vorgibt.
- **Technische Sicht:** Die technische Sicht betrachtet Anforderungen an die Betreibbarkeit, Portierbarkeit, Skalierbarkeit, die Integration in eine bestehende Werkzeuglandschaft von beispielsweise Testwerkzeugen sowie den Datenaustausch oder die Migration von Daten.

Diese Sichten helfen dabei, die Anforderungen an Ihr Werkzeug zu definieren. In der Literatur gibt es darüber hinaus zahlreiche Checklisten zur Werkzeugauswahl, vgl. [CNAT2011], [CNAT2012], [ISO24766], [Eber2012].

Wir möchten in diesem Abschnitt gezielt auf die Aspekte des RM eingehen, die für Ihr Unternehmen bzw. Projekt unterstützt werden müssen. Bei der Auswahl konzentrieren wir uns aus diesem Grunde hauptsächlich auf die Punkte, die in Ihrem RMP berücksichtigt werden sollen.

In den vorigen Kapiteln dieses Buchs haben Sie gelernt, wie Sie einen RMP erstellen.

Hierbei haben Sie entweder einen konkreten Plan für ein Projekt oder auch einen abstrakten Plan für das RM im Unternehmen definiert. Damit Ihr RMP bestmöglich unterstützt wird, sollten Sie sich rückblickend auf die vorherigen Kapitel und auf Ihren RMP überlegen, welche Kriterien zur Werkzeugauswahl folglich für Sie besonders wichtig sind und bspw. folgende Fragen in Ihrer Auswertung berücksichtigen:

- Unterstützt das Werkzeug die Umsetzung Ihres Requirements-Information-Models?
  - Werden die unterschiedlichen Anforderungsarten unterstützt?
  - Werden unterschiedliche Anforderungs-Artefakte unterstützt?
  - Werden unterschiedliche Darstellungsformen unterstützt?
  - Werden unterschiedliche Detaillierungsebenen unterstützt?
  - Können die im Werkzeug dokumentierten Anforderungen in strukturierter und lesbarer Form (z.B. als Anforderungs-Spezifikation) exportiert werden?
- Unterstützt das Werkzeug die Erstellung der notwendigen Attribute und Sichten?
  - Werden unterschiedliche Attribute je Anforderungsart unterstützt?
  - Wird die Definition von Wertebereichen für Attribute unterstützt?
  - Sind Mehrfachauswahlen bei Attributen möglich?
  - Können Werteübergänge der Attribute definiert werden?
  - Wird der Nutzer bei der Befüllung durch automatische Werte (z.B. Erstellungsdatum, Ersteller) unterstützt?
  - Können Default-Werte für Attribute definiert werden?
  - Wird zwischen optionalen und obligatorischen Attributen unterschieden?
  - Werden Abhängigkeiten zwischen Attributen unterstützt?
  - Können ad-hoc Sichten erzeugt werden?
  - Können erzeugte Sichten gespeichert werden?
  - Können Sichten über Rollenkonzepte eingeschränkt werden?
- Unterstützt das Werkzeug die Priorisierung von Anforderungs-Artefakten?
  - Werden ad-hoc Priorisierungsmethoden unterstützt?
  - Werden analytische Priorisierungsmethoden unterstützt?
  - Können Priorisierungsentscheidungen historisiert werden?
- Unterstützt das Werkzeug die Versionskontrolle von Anforderungen?
  - Werden neue Versionen von Artefakten automatisch erzeugt?
  - Können unterschiedliche Versionen miteinander verglichen werden?
  - Kann der Änderungsgrund dokumentiert und nachvollzogen werden?
  - Führt die Änderung von Attributen zu neuen Versionen des Artefakts?
  - Können einzelne Attribute von der Versionierung ausgenommen werden?
  - Ist das Zurückrollen zu alten Anforderungs-Versionen möglich?
  - Können Anforderungs-Konfigurationen erstellt werden?

    - Ist das Zurückrollen zu alten Anforderungs-Konfigurationen möglich?
    - Ist der Vergleich zwischen Anforderungs-Konfigurationen möglich?
    - Können Anforderungs-Basislinien erstellt werden?
    - Ist das Zurückrollen zu alten Anforderungs-Basislinien möglich?
    - Ist der Vergleich zwischen Anforderungs-Basislinien möglich?
- Unterstützt das Werkzeug das Änderungsmanagement?
    - Kann ein Änderungsmanagement-Prozess hinterlegt werden?
    - Werden Vorlagen für Änderungsanträge angeboten oder unterstützt?
    - Ist eine rollenbasierte Erstellung und Bearbeitung von Änderungsanträgen möglich?
    - Wird die Bearbeitung und Bewertung von Änderungsanträgen unterstützt?
    - Können Änderungsaufträge später durch Verlinkung zu den zu ändernden Anforderungen in Beziehung gesetzt werden?
- Unterstützt das Werkzeug die Verfolgbarkeits-Strategie des RMP?
    - Wird Verfolgbarkeit zwischen Artefakten unterstützt?
    - Können unterschiedliche Beziehungs-Typen angelegt werden?
    - Können Beziehungs-Typen auf Artefakte eingeschränkt werden, um zu verhindern, dass alle Beziehungs-Typen unkontrolliert verwendet werden?
    - Ist eine Verlinkung zu vor- und nachgelagerten Artefakten (Ziele bzw. Testfälle) möglich (Stichwort Werkzeugintegration)?
    - Wird eine rollenbasierte Pflege von Verfolgbarkeits-Beziehungen unterstützt oder darf jeder Benutzer alle Beziehungen erstellen, ändern, entfernen?
    - Wird die Verfolgbarkeit zwischen textuellen und modellbasierten Artefakten unterstützt (ggf. auch werkzeugübergreifend)
    - Wie können Verfolgbarkeits-Beziehungen dargestellt werden (Matrix, Tabelle, Graph, etc.)?
    - Sind Auswirkungsanalysen für Änderungen möglich, die dem Benutzer die vor- und nachgelagerten Artefakte darstellen?
    - Über wie viele Ebenen ist eine Auswirkungsanalyse möglich?
    - Können Auswertungen über Verfolgbarkeits-Beziehungen erstellt werden (z.B. Anzahl der Beziehungen zwischen Testfällen und Anforderungen zur Anzahl der Testfälle und Anforderungen)?
- Unterstützt das Werkzeug die Dokumentation von Variabilität?
    - Wird die explizite Dokumentation von Variabilität unterstützt?
    - Wird die implizite Dokumentation von Variabilität unterstützt?
    - Werden Beziehungen zwischen Variations-Punkten und Varianten unterstützt?
    - Wird die Merkmals-Modellierung unterstützt?
    - Werden orthogonale Verfolgbarkeits-Modelle unterstützt?
    - Wird die Ableitung von konkreten Produkten aus der definierten Variabilität unterstützt?
    - Kann nach Varianten und Variations-Punkten gesucht werden?
- Unterstützt das Werkzeug das Berichtswesen im Rahmen des RMs?
    - Gibt es Vorlagen für die Definition von Berichten?

        - Können eigene Berichte erstellt werden?
        - Wird eine automatisierte Erstellung von Berichten (z.B. zu bestimmten Zeitpunkten) unterstützt?
        - Ist ein Export von Berichten z.B. als PDF-Datei möglich?
        - Können Berichte automatisiert versendet werden?
        - Ist der Ausdruck von Berichten möglich?
    - Unterstützt das Werkzeug die Definition von RE-Prozessen?
        - Können Workflows für die definierten RE-Aktivitäten (z.B. Dokumentation, Überprüfung, Abnahme) definiert werden?
        - Wird die Definition von Rollen, Verantwortlichkeiten und (Benutzer-)Rechten unterstützt?
        - Können unternehmensweite Vorgehensmodelle abgebildet werden, die in einzelnen Projekten angepasst werden?
        - Wird die parallele und rollenbasierte Arbeit unterstützt?
        - Werden Offene-Punkte-Listen (bzw. Aufgaben / Tasks) unterstützt, um unklare Punkte und Aufgaben zu dokumentieren und bestimmten Personen zuzuweisen?
        - Können Entscheidungen festgehalten werden (z.B. Beschlussprotokolle)?
        - Können RE-Prozesse überprüft werden (Soll / Ist Vergleich zur Prozesskonformität)?
    - Unterstützt das Werkzeug agile Methoden?
        - Werden Storyboards und Kanban-Boards unterstützt?
        - Werden Burndown-Charts unterstützt?
        - Werden Product Backlogs und Sprint Backlogs unterstützt?
        - Werden Retrospektiven unterstützt?

Überlegen Sie sich, welche dieser Punkte für Sie relevant sind, und gewichten Sie diese Punkte für ihr Projekt zur Werkzeugeinführung. Auf Basis des RMPs können Sie eine strukturierte Fragenliste für Ihre Werkzeugauswahl erstellen.

Mit Hilfe dieser Liste bewerten Sie dann die ausgewählten Werkzeuge in Bezug auf zu unterstützende RM-Funktionen. Denn wesentlich ist, dass das Werkzeug Ihren RE-Prozessen gerecht wird.

3 Analyse ausgewählter Werkzeuge mittels der RMP-Bewertungskriterien

In diesem Kapitel werden wir auf Basis der im vorherigen Kapitel eingeführten Kriterien eine exemplarische Werkzeugevaluierung vornehmen. Hierzu haben wir uns bewusst drei sehr unterschiedlichen Klassen von Werkzeugen gewidmet.

- Standard-Büroanwendungen
- Systems-Engineering-Werkzeuge
- RM-Werkzeuge

Hierbei wollen wir gar nicht so sehr auf das eigentliche Ergebnis schauen und vor allem keine unabhängige Bewertung von RM-Werkzeugen erstellen. Vielmehr wollen wir einen Einblick geben, wie diese Kriterien angewendet und für die Werkzeugauswahl genutzt werden können.

In der Klasse der Standard-Büroanwendungen werden wir uns dem am weitesten verbreiteten Werkzeug widmen, womit weltweit wahrscheinlich mehr als die Hälfte der Anforderungen erfasst und später verwaltet werden. Auch wenn diese Anwendungen bei weitem nicht die eingangs erwähnten notwendigen Eigenschaften von RM-Werkzeugen [PoRu2011a] (Kapitel 9.3) unterstützen. Dennoch ist diese Art der Dokumentation und Verwaltung mit einigen methodischen und organisatorischen Richtlinien zur Attributierung, Versionierung und Verfolgbarkeit besser als keine Dokumentation.

Der riesige Vorteil bei diesen Applikationen liegt in der weiten Verbreitung – also dem initialen Vorhandensein, der bereits bestehenden Nutzerakzeptanz sowie dem Vorteil, dass fast jeder mit diesen Applikationen umzugehen weiß, und solche Dateien leicht per E-Mail zwischen Beteiligten ausgetauscht werden können. Word bietet beispielsweise den Vorteil, dass Spezifikationen genau so strukturiert werden können, wie es erforderlich ist. Mit Excel lassen sich darüber hinaus mit wenigen Vorkenntnissen Attribute anlegen und Sichten erzeugen. Excel hat hier gegenüber von Word den entscheidenden Vorteil, dass man Filterungen, Sortierungen, Auswertungen usw. auf Basis der definierten Attribute vornehmen kann.

Zur Erstellung einer Versionierung kann man sich dazu über kleinere Makros das Leben erleichtern, um „auf Knopfdruck" neue Versionen eines Anforderungs-Artefakts zu erzeugen. In Abbildung 49 ist eine beispielhafte Darstellung einer Excel-basierten Anforderungsliste mit verschiedenen Attributen. Die grau hinterlegte Zeile spiegelt hier eine alte Version einer Anforderung wider, die unterhalb als aktuelle und überarbeitete Version dargestellt wird.

	RID	Version	Artefakt Typ	Requirement Typ	Source	Requirement/Description	Kommentar	Maturity (Requirement)	Prio (Requirement)	Status (Requirement)
Add new last requirement / Add new inline requirement										
Neue Version erzeugen	R-00010	V-001	requirement	functions/process	Meeting	Das neue Onlineportal muss die Bearbeitung von Kundenstammdaten erlauben.	Der Kunde muss in der Lage sein, seine Email-Adresse oder seine Telefunnummer zu ändern.	safe	Must	ready for review
Neue Version erzeugen	R-00020	V-001	requirement	functions/process	Meeting	Das neue Onlineportal muss die Erstellung neuer Unterkonten erlauben.	Der Kunde muss in der Lage sein, ein neues Unterkonto zu seinem Sparkonto zu eröffnen	change likely	Should	draft
Neue Version erzeugen	R-00020	V-002	requirement	functions/process	Meeting	Das neue Onlineportal muss die eErstellung neuer Unterkonten für ein Suprtkonto erlauben	Der Kunde muss in der Lage sein, ein neues Unterkonto zu seinem Sparkonto zu eröffnen	safe	Should	draft
Neue Version erzeugen	R-00030	V-001	requirement	functions/process	Meeting					

Abbildung 49: Screenshot einer excelbasierten Anforderungsliste

Die Erstellung von Strukturen und Hierarchien ist in tabellenorientierten Applikationen wie Excel allerdings schwieriger als in Word. In Tabelle 12: Analyse ausgewählter Werkzeuge: Ergebnistabelle: Analyse ausgewählter Werkzeuge: haben wir Fähigkeiten von Word und Excel für das RM etwas mehr unter die Lupe genommen.

Auch die zweite Klasse der hier betrachteten Werkzeuge ist kein klassisches RM-Werkzeug, sondern eine Anwendung, die ursprünglich dem Issue-Tracking dient. Mittels dieser Klasse wollen wir vor allem darauf hinweisen, dass Sie vor der Einführung eines RM-Werkzeuges in Ihrem Unternehmen nach links und rechts schauen und prüfen, welche anderen Werkzeuge bereits im Einsatz sind und eventuell sogar für das RM eingesetzt werden können. Das Issue-Tracking System Jira von der Firma Atlassian (www.atlassian.com) bietet durch seine Anpassbarkeit deutlich mehr Möglichkeiten als nur das Issue-Tracking. In Jira können Sie unterschiedliche Anforderungs-Artefakte mit eigenen Attributen erzeugen, Workflows für z.B. Statusübergänge definieren, Anforderungs-Artefakte miteinander verlinken und so weiter. Neben seinem Standard-Funktionsumfang bietet Jira einige Plug-Ins, um beispielsweise agile Projektmethoden wie Scrum zu unterstützen. Confluence bietet darüber hinaus eine webbasierte Anwendung zur Organisation von Dokumenten, Besprechungsnotizen, Entscheidungen oder zur Erstellung von Berichten und zur Beschreibung des gesamten Projektkontextes. Confluence lässt sich dabei nahtlos in Jira integrieren. Die Nutzung von Jira und Confluence für das RM wurde beispielsweise in [Syra2014] beschrieben. In Abbildung 50 werden zwei Screenshots auf Jira dargestellt – links eine Vorlage zur Erstellung von Anforderungen mit den definierten Attributen und rechts eine exemplarische Sicht auf Anforderungen. Abbildung 51 zeigt ein Beispiel, wie Verlinkungen in Jira dargestellt werden (Issue Links). Im Beispiel ist die Anforderung mit zwei weiteren Anforderungen über den Beziehungs-Typ „blocks" und mit weiteren zwei Anforderungen über den Beziehungs-Typ „relates to" verbunden. Über diese Hyperlinks lässt sich bidirektional zwischen den Artefakten navigieren.

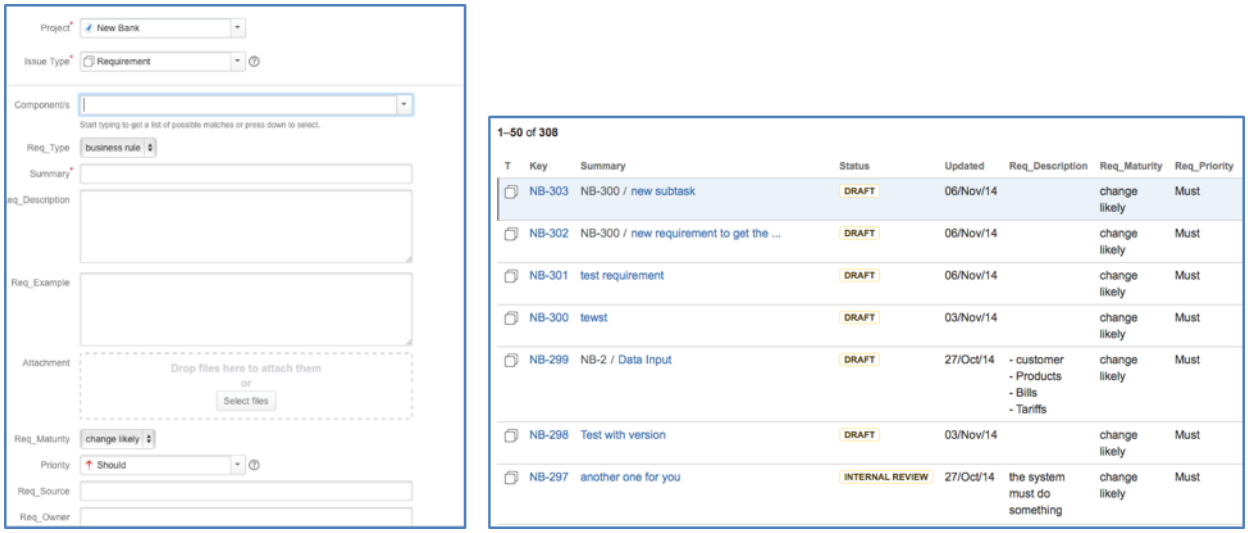


Abbildung 50: Beispiel-Vorlage zur Erfassung von Anforderungen (links) und Statussicht auf Anforderungen (rechts)

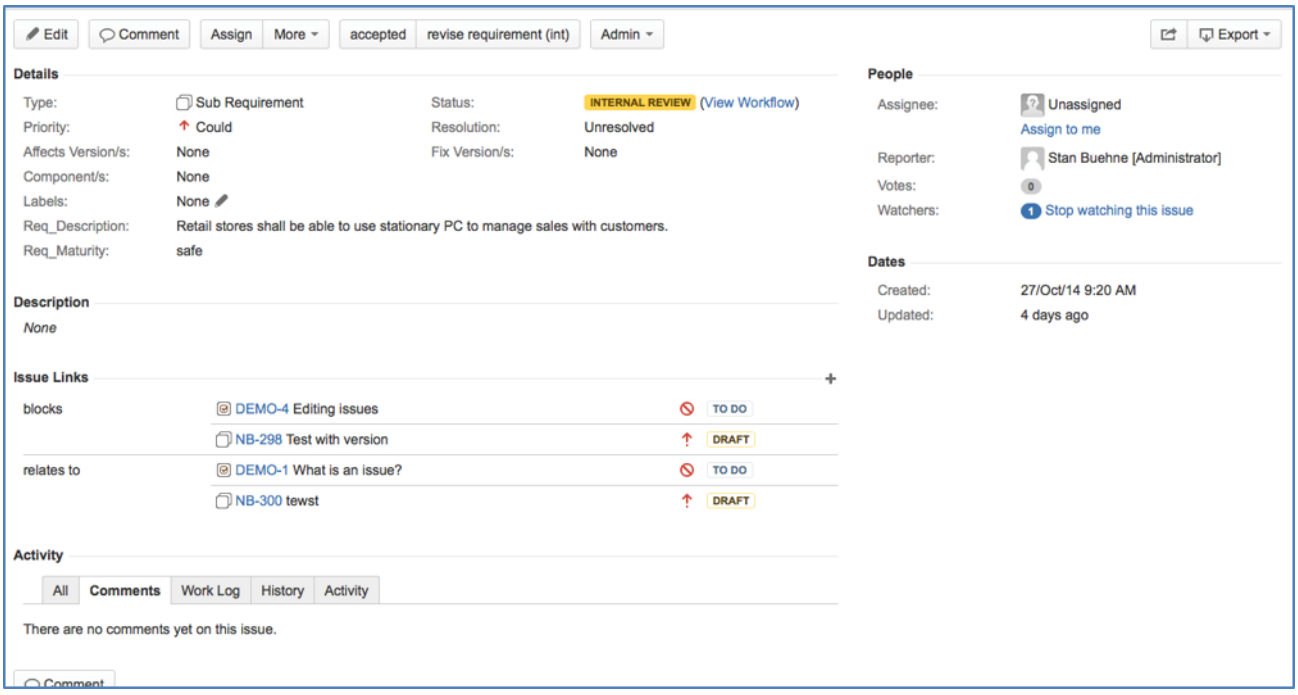


Abbildung 51: Beispiel Verfolgbarkeits-Beziehungen einer Anforderung in Jira

In der dritten Klasse der RM-Werkzeuge widmen wir uns ProR/ RMF, einem Open Source Werkzeug für das RM mit Eclipse.

Sie finden die Software zum kostenlosen Herunterladen sowie Dokumentation zum Werkzeug im Internet unter: http://eclipse.org/rmf, www.pror.org bzw. die Download-Seite hier:

oder auf der Download-Seite: http://eclipse.org/rmf/download.php

ProR setzt normalerweise die Installation von Eclipse voraus. Es gab eine Standalone-Version, die ohne Eclipse auskommt. Dies wird jedoch nicht mehr vom RMF-Projekt unterstützt (RMF steht für Requirements Modeling Framework). Eine Standalone-ProR-

Variante, die es jedoch noch kostenlos zum Herunterladen gibt, ist formalmindStudio der Formal Mind GmbH: http://www.formalmind.com/studio.

formalmindStudio enthält Erweiterungen, die sich ProR Essentials nennen und das Arbeiten effizienter machen. Dokumentationen zu diesem Werkzeug finden Sie hier: http://wiki.eclipse.org/RMF

und den RMF-Guide hier: http://download.eclipse.org/rmf/documentation/rmf-latex/main.html.

Das Werkzeug wird ständig weiter entwickelt. Wenn Sie Ihre eigenen Ideen umsetzen möchten, können Sie hier nachlesen, wie: https://wiki.eclipse.org/RMF/Contributor_Guide/Presentations

Abbildung 52 zeigt einen Screenshot einer hierarchischen Anforderungsliste in formalmindStudio. Die Attribute können selbst definiert werden, ebenso wie die Linktypen, die Verfolgbarkeit zwischen Anforderungen herstellen. Wie Sie in Tabelle 12: Analyse ausgewählter Werkzeuge: Ergebnistabelle sehen, unterstützt formalmind Studio aktuell vor allem Grundfunktionalitäten des RM. Jedoch existiert eine Vielzahl von Eclipse-Plugins, mit denen das Werkzeug integriert und funktional erweitert werden kann.

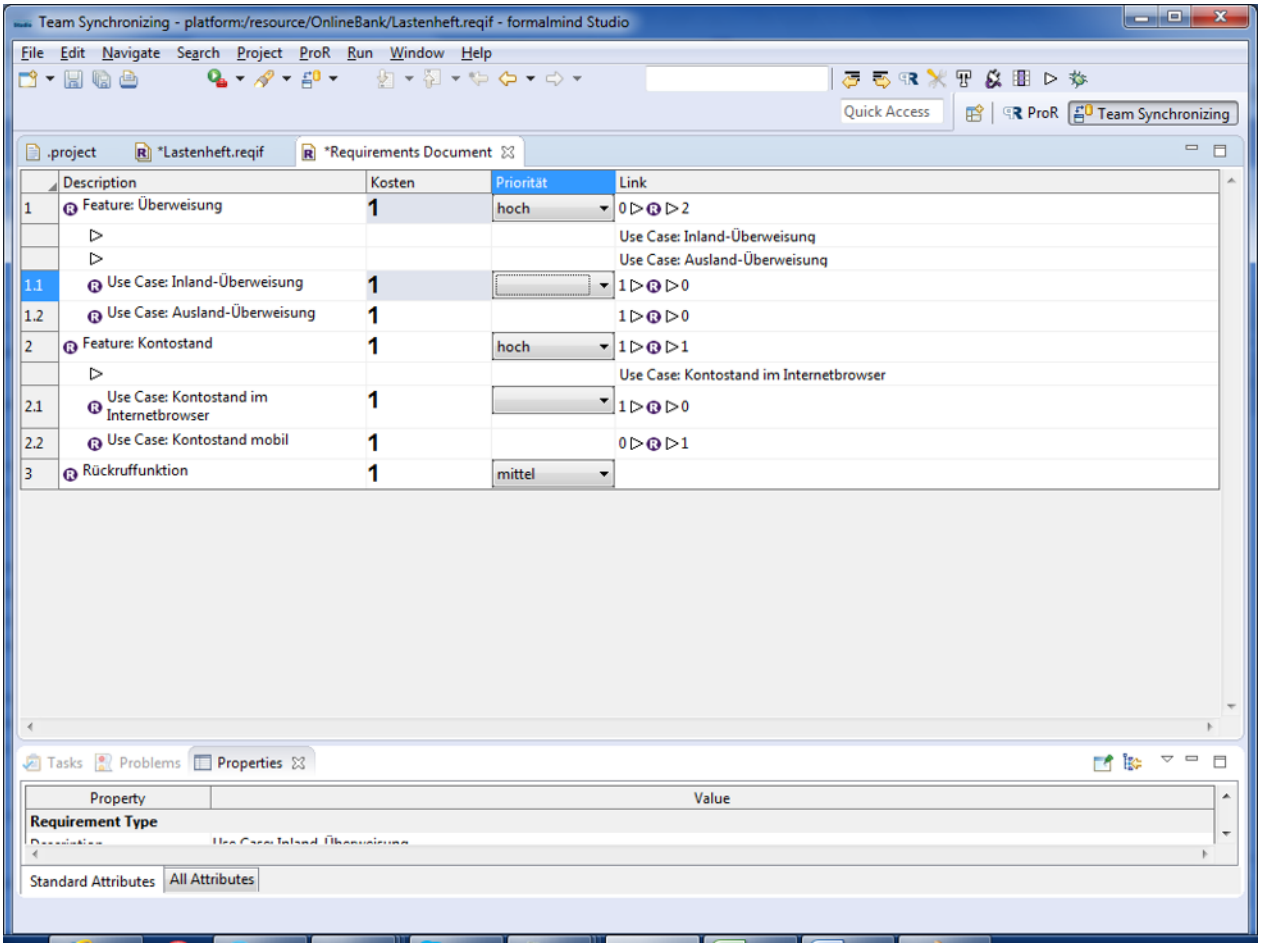


Abbildung 52: Screenshot von formalmindStudio: Anforderungshierarchie mit zwei Attributen (Kosten und Priorität) sowie Links zwischen den Features und Use Cases.

Tabelle 12: Analyse ausgewählter Werkzeuge: Ergebnistabelle gibt hier einen Überblick über die Auswertung. Wir haben an dieser Stelle ausschließlich drei Werte für die Bewertung verwendet – „Ja" bedeutet an dieser Stelle, das Werkzeug unterstützt das Kriterium vollständig, „teilweise" bedeutet, das Werkzeug unterstützt das Kriterium mit Abstrichen und „Nein" bedeutet, das Werkzeug unterstützt dieses Kriterium nicht. Die Bewertung soll an dieser Stelle weniger eine Werkzeugempfehlung darstellen, sondern vielmehr eine Kriterienvorlage zur Werkzeugbewertung, die beispielhaft für drei nicht typische RM-Werkzeuge angewendet wurde.

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Unterstützt das Werkzeug die Umsetzung Ihres Requirements-Information-Models?						
Werden die unterschiedlichen Anforderungsarten unterstützt?	Ja	Keine Einschränkung – solange textuell darstellbar	Ja	Über Attribute	Ja	Keine Einschränkung – solange textuell darstellbar
Werden unterschiedliche Anforderungs-Artefakte unterstützt?	Ja	Keine Einschränkung – solange textuell darstellbar	Ja	Über neue Issue-Typen und eigene Attribute und Sichten	Ja	
Werden unterschiedliche Darstellungsformen unterstützt?	Teilweise	Text und Templates	Teilweise	Text und Templates	Teilweise	Nur Text, aber Elemente von Diagrammen können referenziert werden, z.B. bei Integration mit Rodin
Werden unterschiedliche Detaillierungsebenen unterstützt?	Teilweise	Über Dokumentenstrukturen oder Attribute	Teilweise	Über Sub-Requirements und Verlinkung	Ja	Durch hierarchische Verlinkung – Diese kann flexibel durch Drag & Drop geändert werden.
Können die im Werkzeug dokumentierten Anforderungen in strukturierter und lesbarer Form (z.B. als Anforderungs-Spezifikation) exportiert werden?	Ja	Bereits als Dokument verfasst	Teilweise	Standard-Export ist rein Tabellenorientiert	Teilweise	Als HTML-Datei und als reqIF-Datei
Unterstützt das Werkzeug die Erstellung der notwendigen Attribute und Sichten?						

	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
Kriterium	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**
Werden unterschiedliche Attribute je Anforderungsart unterstützt?	Ja		Teilweise	Abhängig von der Realisierung	Ja	Man kann beliebig viele Anforderungsarten definieren, die jeweils unterschiedliche Attribute haben. Diese können innerhalb eines Artefakts gemischt werden.
Wird die Definition von Wertebereichen für Attribute unterstützt?	Ja	Wertebereiche nur in Excel sinnvoll abbildbar	Ja		Ja	Wertebereiche für Zahlen sowie Wertelisten
Sind Mehrfachauswahlen bei Attributen möglich?	Nein	Maximal als Werteliste	Ja	In unterschiedlicher Art und Weise (Checkbox, Label, Listenauswahl)	Ja	Bei Attributen, bei denen Wertelisten hinterlegt sind
Können Werteübergänge der Attribute definiert werden?	Nein	Könnten in Excel maximal durch ein Makro definiert werden	Teilweise	Für Attribute wie Status können explizite Status und Transitionen definiert werden.	Nein	

	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
Kriterium	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**
Wird der Nutzer bei der Befüllung durch automatische Werte (z.B. Erstellungsdatum, Ersteller) unterstützt?	Teilweise	In Word und Excel könnten bereits Einträge vorbeschriftet werden.	Ja		Teilweise	Es werden Default-Werte unterstützt.
Können Default-Werte für Attribute definiert werden?	Teilweise	In Word und Excel könnten bereits Einträge vorbeschriftet werden.	Ja		Ja	
Wird zwischen optionalen und obligatorischen Attributen unterschieden?	Teilweise	Nur durch spezielle Markierung möglich	Ja		Nein	
Werden Abhängigkeiten zwischen Attributen unterstützt?	Nein		Nein		Nein	Mit Hilfe des Eclipse Validation Frameworks möglich
Können ad-hoc Sichten erzeugt werden?	Nein		Ja		Teilweise	Ja, aber immer nur eine – Aus- und Einblenden von Attributen geht, Filtern nach Attributen ist möglich, Sortieren geht nicht.
Können erzeugte Sichten gespeichert werden?	Nein		Ja		Ja	Aber immer nur eine

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Können Sichten über Rollenkonzepte eingeschränkt werden?	Nein		Ja		Nein	
Unterstützt das Werkzeug die Priorisierung von Anforderungs-Artefakten?						
Werden ad-hoc Priorisierungsmethoden unterstützt?	Nein		Nein		Nein	
Werden analytische Priorisierungsmethoden unterstützt?	Nein		Nein		Nein	
Können Priorisierungsentscheidungen historisiert werden?	Nein		Nein		Nein	
Unterstützt das Werkzeug die Versionskontrolle von Anforderungen?						
Werden neue Versionen von Artefakten automatisch erzeugt?	Nein	Einzige Möglichkeit ist eine Änderungsverfolgung.	Ja		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle
Können unterschiedliche Versionen miteinander verglichen werden?	Teilweise	Eingeschränkt über die Änderungsverfolgung in Word -Excel bietet keine Möglichkeit.	Ja		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle

	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
Kriterium	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**
Kann der Änderungsgrund dokumentiert und nachvollzogen werden?	Teilweise	Rein textuell kann auf Artefakt und/oder Dokumentenebene ein Grund erfasst werden.	Ja	Änderungen können bspw. über Kommentare oder über eigene Issue-Typen dokumentiert werden.	Teilweise	Das wäre in einem eigenen Attribut möglich.
Führt die Änderung von Attributen zu neuen Versionen des Artefakt?	Nein	Versionierung kann nur manuell erfolgen	Ja		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle
Können einzelne Attribute von der Versionierung ausgenommen werden?	Nein	Versionierung kann nur manuell erfolgen	Nein		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle
Ist das Zurückrollen zu alten Anforderungs-Versionen möglich?	Teilweise	Eingeschränkt über die Änderungsverfolgung in Word -Excel bietet keine Möglichkeit.	Nein	Vorherige Versionen können nur dargestellt, aber nicht zurückgerollt werden.	Nein	Möglich durch Eclipse-Plugin für Versionskontrolle
Können Anforderungs-Konfigurationen erstellt werden?	Nein	Eine Anforderungs-Konfiguration kann maximal als Dokumentenversion erstellt werden.	Nein		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Ist das Zurückrollen zu alten Anforderungs-Konfigurationen möglich?	Nein		Nein		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle
Ist der Vergleich zwischen Anforderungs-Konfigurationen möglich?	Nein		Nein		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle und EMF Compare
Können Anforderungs-Basislinien erstellt werden?	Nein	Eine Baseline kann maximal als Dokumentenversion erstellt werden.	Nein		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle
Ist das Zurückrollen zu alten Anforderungs-Basislinien möglich?	Nein		Nein		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle
Ist der Vergleich zwischen Anforderungs-Basislinien möglich?	Nein		Nein		Nein	Möglich durch Eclipse-Plugin für Versionskontrolle und EMF Compare
Unterstützt das Werkzeug das Änderungsmanagement?						

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Kann ein Änderungsmanagement-Prozess hinterlegt werden?	Nein		Teilweise	Falls Änderungen als eigener Issue-Type angelegt werden, könnte hierfür ein Workflow definiert werden.	Nein	
Werden Vorlagen für Änderungsanträge angeboten oder unterstützt?	Nein		Teilweise	Falls Änderungen als eigener Issue-Type angelegt werden, sind eigene Attribute für einen Änderungsantrag möglich.	Nein	
Ist eine rollenbasierte Erstellung und Bearbeitung von Änderungsanträgen möglich?	Nein		Ja		Nein	
Wird die Bearbeitung und Bewertung von Änderungsanträgen unterstützt?	Nein		Teilweise	Die Bearbeitung kann über einen Workflow abgebildet werden – die Bewertung an sich muss manuell erfolgen.	Teilweise	Änderungsanforderungen können als Anforderungen dokumentiert werden.

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Können Änderungsaufträge später durch Verlinkung zu den zu ändernden Anforderungen in Beziehung gesetzt werden?	Nein		Ja		Teilweise	Ja, wenn die Änderungsanforderungen als Anforderung verwaltet werden.
Unterstützt das Werkzeug die Verfolgbarkeits-Strategie des RMP?						
Wird Verfolgbarkeit zwischen Artefakten unterstützt?	Teilweise	Nur über die manuelle Pflege von textuellen Referenzen, Hyperlinks, oder Matrizen	Ja	Über Verknüpfung von Issue-Types	Nein	
Können unterschiedliche Beziehungs-Typen angelegt werden?	Teilweise	Rein textuell ist es möglich.	Ja		Ja	
Können Beziehungs-Typen auf Artefakte eingeschränkt werden, um zu verhindern, dass alle Beziehungs-Typen unkontrolliert verwendet werden?	Nein		Nein		Nein	
Ist eine Verlinkung zu vor- und nachgelagerten Artefakten (Ziele bzw. Testfälle) möglich (Stichwort Werkzeugintegration)?	Teilweise	Nur über manuelle textuelle Referenzen	Ja	Falls alle Artefakte in Jira beschrieben werden	Ja	Falls alle Artefakte in formalmindStudio abgelegt sind oder durch Integration mit Rodin

	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
Kriterium	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**	**Bewer-tung**	**Begründung**
Wird eine rollenbasierte Pflege von Verfolgbarkeits-Beziehungen unterstützt oder darf jeder Benutzer alle Beziehungen erstellen, ändern, entfernen?	Nein				Nein	Alle Benutzer dürfen dasselbe.
Wird die Verfolgbarkeit zwischen textuellen und modellbasierten Artefakten unterstützt (ggf. auch werkzeugübergreifend)	Teilweise	Über textuelle Referenzen, URLs und eingebettete Objekte	Teilweise	Über textuelle Referenzen, URLs und Anhänge	Ja	Es gibt eine Integration mit dem Modellierungs-Werkzeug Rodin.
Wie können Verfolgbarkeits-Beziehungen dargestellt werden (Matrix, Tabelle, Graph, etc.)?		Über manuellen Aufwand in jeder Form		Durch Hyperlinks		Durch Hyperlinks
Sind Auswirkungsanalysen für Änderungen möglich, die dem Benutzer die vor- und nachgelagerten Artefakte darstellen?	Nein		Teilweise	Durch Hyperlinks	Teilweise	Durch Hyperlinks
Über wie viele Ebenen ist eine Auswirkungsanalyse möglich?		Es ist immer nur das direkt verknüpfte Artefakt sichtbar.		Es ist immer nur das direkt verknüpfte Artefakt sichtbar.		Es ist immer nur das direkt verknüpfte Artefakt sichtbar.

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Können Auswertungen über Verfolgbarkeits-Beziehungen erstellt werden (z.B. Anzahl der Beziehungen zwischen Testfällen und Anforderungen)?	Nein		Nein		Nein	
Unterstützt das Werkzeug die Dokumentation von Variabilität?						
Wird die explizite Dokumentation von Variabilität unterstützt?	Nein		Nein		Nein	
Wird die implizite Dokumentation von Variabilität unterstützt?	Teilweise	Über Vorlagen kann die implizite Dokumentation von Variabilität unterstützt werden.	Teilweise	Über Attribute kann die implizite Dokumentation von Variabilität unterstützt werden,	Nein	
Werden Beziehungen zwischen Variations-Punkten und Varianten unterstützt?	Nein		Nein		Nein	
Wird die Merkmals-Modellierung unterstützt?	Nein		Nein		Nein	
Werden orthogonale Verfolgbarkeits-Modelle unterstützt?	Nein		Nein		Nein	

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Wird die Ableitung von konkreten Produkten aus der definierten Variabilität unterstützt?	Nein		Nein		Nein	
Kann nach Varianten und Variations-Punkten gesucht werden?	Nein		Teilweise	Falls durch eigene Attribute abgebildet	Nein	
Unterstützt das Werkzeug das Berichtswesen im Rahmen des RMs?						
Gibt es Vorlagen für die Definition von Berichten?	Nein	Maximal eigene Word-Vorlagen	Nein		Nein	
Können eigene Berichte erstellt werden?	Ja	Eigene Berichte können in Word und Excel erstellt werden.	Nein		Nein	Nur Sichten
Wird eine automatisierte Erstellung von Berichten (z.B. zu bestimmten Zeitpunkten) unterstützt?	Nein		Nein		Nein	
Ist ein Export von Berichten z.B. als PDF möglich?	Ja	Ab Office 2010 kann ein Dokument als pdf gesichert werden.	Nein		Ja	Nur in HTML

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Können Berichte automatisiert versendet werden?	Nein		Nein		Nein	
Ist der Ausdruck von Berichten möglich?	Ja		Nein		Ja	In HTML-Format über den Browser
Unterstützt das Werkzeug die Definition von RE-Prozessen?						
Können Workflows für die definierten RE-Aktivitäten (z.B. Dokumentation, Überprüfung, Abnahme) definiert werden?	Nein		Ja		Nein	
Wird die Definition von Rollen, Verantwortlichkeiten und (Benutzer-)Rechten unterstützt?	Nein		Ja		Nein	
Können unternehmensweite Vorgehensmodelle abgebildet werden, die in einzelnen Projekten angepasst werden?	Teilweise	Über Dokumentenvorlagen können Spezifikationsvorlagen erstellt werden.	Nein		Nein	
Wird die parallele und rollenbasierte Arbeit unterstützt?	Nein		Ja		Nein	

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Werden Offene-Punkte-Listen (bzw. Aufgaben / Tasks) unterstützt, um unklare Punkte und Aufgaben zu dokumentieren und bestimmten Personen zuzuweisen?	Nein	Nicht mit direkter Zuordnung – grundsätzlich kann man natürlich offene Punkte Listen pflegen.	Teilweise	Über die Erstellung von Tasks, die zu Anforderungen in Beziehung gesetzt werden	Nein	
Können Entscheidungen festgehalten werden (z.B. Beschlussprotokolle)?	Nein		Teilweise	Durch Anhänge zu einer Anforderung oder Verlinkung zu Confluence	Nein	
Können RE-Prozesse überprüft werden (Soll / Ist Vergleich zur Prozesskonformität)?	Nein		Nein		Nein	
Unterstützt das Werkzeug agile Methoden?						
Werden Storyboards und Kanban-Boards unterstützt?	Nein		Ja	Über das Jira „Agile" Plugin sowohl Kanban- als auch Storyboards	Nein	
Werden Burndown-Charts unterstützt?	Nein		Ja	Über das Jira „Agile" Plugin	Nein	

Kriterium	Excel/ Word		JIRA/Confluence		ProR/formalmind Studio	
	Bewer-tung	Begründung	Bewer-tung	Begründung	Bewer-tung	Begründung
Werden Product Backlogs und Sprint Backlogs unterstützt?	Nein		Ja	Über das Jira „Agile" Plugin	Nein	
Werden Retrospektiven unterstützt?	Nein		Ja	Über das Jira „Agile" Plugin	Nein	

Tabelle 12: Analyse ausgewählter Werkzeuge: Ergebnistabelle

An dieser Stelle sei noch einmal erwähnt, dass wir hier keine umfassende Bewertung von Werkzeugen bzw. dieser ausgewählten Werkzeuge vornehmen wollten, sondern dass wir diese Werkzeuge nur genutzt haben, um unsere auf das RM bezogenen Auswahlkriterien in der Anwendung zu zeigen. Aus diesem Grund sei uns verziehen, falls wir bestimmte Funktionen eines der Werkzeuge nicht korrekt eingeschätzt haben.

Anhang C (Earned-Value-Analyse)

Die Earned-Value-Analyse{ XE "Earned-Value-Analyse" } (auch: Nutzwertanalyse{ XE "Nutzwertanalyse" }) [Kerz2003], [Wann2013a], [Wann2013b] verfolgt den Status eines Projektes anhand seines Ergebnisfortschritts und des Kostenverbrauchs. Dabei wird der Fortschritt (bzw. Fertigstellungsgrad bzw. *Earned Value*) verglichen mit dem zu diesem Zeitpunkt geplanten Fortschritt (*Planned Value*) und außerdem verglichen mit dem bisher verbrauchten Budget (*Actual Cost*).

Mit dieser Methode können Sie frühzeitig Abweichungen des tatsächlichen Projektverlaufs vom Zeitplan oder vom Budget feststellen. Ist das Projekt nicht im Plan, müssen entweder Maßnahmen ergriffen werden, um doch noch plangemäß fertig zu werden, oder es können mit Hilfe der Earned-Value-Analyse der Lieferverzug und die Kostenüberschreitung vorausberechnet werden.

Die Earned-Value-Analyse benötigt die folgenden vier Kennzahlen des Projektes:

- *Budget (Budget at Completion BAC):* Das für das gesamte Projekt verfügbare Budget. In der Earned-Value-Analyse geht man davon aus, dass dieses Budget den Gesamtkosten und dem *Planned Value* (Nutzwert) des Projektes zum Projektende entspricht. Ist dies nicht der Fall, verwenden Sie hier für Ihre Rechnung die geplanten Gesamtkosten. Zu diesem Gesamtvolumen des Projektes kommen noch drei Größen hinzu, für die Werte jeweils zu jedem Berichtszeitpunkt ermittelt werden müssen.
- *Geplanter Fertigstellungsgrad (Planned Value, PV):* Hier wird in % angegeben, zu welchem Anteil das Projekt zum aktuellen Zeitpunkt fertig gestellt sein sollte. Er wird berechnet als der Quotient aus dem geplanten Arbeitsvolumen und dem Gesamtvolumen des Projektes.
- *Fertigstellungsgrad{*XE "Fertigstellungsgrad" *} (Earned Value, EV):* Hier wird in % angegeben, zu welchem Anteil das Projekt zum aktuellen Zeitpunkt tatsächlich fertig gestellt wurde.
- *Bisherige Kosten (Actual Cost AC):* Hier wird angegeben, welche Kosten bisher entstanden sind. Der Kostenindex ist der Quotient aus den bisherigen Kosten und dem Gesamtbudget.

Mit Hilfe dieser Zahlen wissen Sie, welcher Teil des Ergebnisses bisher fertig gestellt wurde und welcher Anteil des Budgets dafür verbraucht wurde. Daraus lässt sich berechnen, ob das Projekt im Zeitplan liegt und ob es effizient arbeitet, d.h. das erzeugte Ergebnis zum verbrauchten Budget passt. Entspricht der Kostenindex dem Fertigstellungsgrad, dann wurden mit x % der Ressourcen auch x % des Ergebnisses fertiggestellt, und das Projekt liegt im Budget. Entspricht der Fertigstellungsgrad dem geplanten Fertigstellungsgrad, dann liegt das Projekt im Zeitplan. Aufgrund der Größe eventueller Abweichungen vom Plan lassen sich Prognosen erstellen, wie unpünktlich oder überteuert das Projekt voraussichtlich abgeschlossen werden wird.

Um die Termintreue zu prüfen, rechnet man folgendermaßen:

Man vergleicht für denselben Zeitpunkt den *Earned Value* mit dem *Planned Value*, d.h. den tatsächlichen Fertigstellungsgrad mit dem geplanten. Sind beide gleich, liegt man genau im

Plan. Ist der tatsächliche Fertigstellungsgrad höher als der geplante, ist das auch günstig, denn man wird voraussichtlich früher fertig werden als geplant. Am häufigsten ist jedoch der ungünstige Fall, dass der Fertigstellungsgrad geringer ist als der geplante. Das Projekt ist also in Verzug.

Aus dem Verzug folgt ein verspäteter Liefertermin für die geplanten Ergebnisse. Bleibt der Liefertermin fix, dann wird bei Verzug der Projektumfang beschränkt und ein Teil des Lieferumfangs gegebenenfalls nachgeliefert.

Um die Kostentreue zu prüfen, rechnet man folgendermaßen:

Zuerst berechnet man den Kostenindex. Dies ist der Quotient aus den bisherigen Kosten und dem Gesamtbudget des Projektes in %. Man vergleicht dann den tatsächlichen Fertigstellungsgrad mit dem Kostenindex. Die Erstellung von 26 % des Projektergebnisses sollte auch maximal 26 % des Gesamtbudgets verbrauchen. Ist der Fertigstellungsgrad jedoch kleiner als der Kostenindex, dann wird das Budget voraussichtlich überzogen werden.

Um die *Zeit- und Kostenpläne an die Realität anzupassen,* also Verzug und Budgetüberschreitung vorherzusehen, gibt es verschiedene Ansätze:

1. **Ursprungsplan beibehalten***:* Man geht davon aus, dass die bereits eingetretene Abweichung keine Auswirkung hat und die Verspätung oder Budgetüberzug wieder eingeholt werden kann. Endtermin und Budget bleiben also gleich. Diese optimistische Hoffnung erfüllt sich jedoch selbst bei ausgefeiltesten Begründungen selten.
2. **Aufaddieren des Verzugs bzw. der Teuerung***:* Man addiert den bisher entstandenen Verzug oder die zu viel entstandenen Kosten auf die Planwerte auf, wenn man davon ausgeht, dass der Rest des Projektes nach Plan verlaufen wird. Auch diese Annahme sollte begründet werden. Hierbei rechnet man so:

    - **Terminprognose**: Ist der aktuelle Fertigstellungsgrad (z.B. 26 %) erst sieben Tage später erreicht worden als geplant, dann beträgt der Verzug also sieben Tage. Das Gesamtprojekt wird um sieben Tage später fertig. Zum Endtermin werden sieben Tage dazu addiert.
    - **Kostenprognose**: Sind zum Erreichen des aktuellen Fertigstellungsgrads (z.B. 26 %) beispielsweise 30 % des Budgets (=Kostenindex) verbraucht worden, so wird das Projekt am Ende voraussichtlich 104 % des geplanten Gesamtbudgets verbrauchen. Man errechnet die voraussichtlichen Gesamtkosten, indem man das Gesamtbudget (BAC) mit 1,04 multipliziert.
3. **Lineare Prognose***:* Diese pessimistischste Annahme ist meist die realistischste. Man geht davon aus, wenn der erste Teil des Projektes bereits zu einem bestimmten Prozentsatz teurer wurde als geschätzt, dann liegt ein systematischer Schätzfehler vor und auch die restlichen Arbeiten werden entsprechend teurer.

    - Terminprognose: Ist der aktuelle Fertigstellungsgrad (z.B. 26 %) erst sieben Tage später erreicht worden als geplant, so ist das beispielsweise nach 33 statt 26 Tagen. Man macht also eine Dreisatzrechnung: Wenn 26 % Fertigstellung 33 Tagen Dauer entsprechen, wie lange dauern dann 100 %?

Rechnen Sie: 33 Tage x 100 % / 26 % = 127 Tage. Geplant war ursprünglich eine Dauer von 100 Tagen. Der Verzug beträgt dann also 27 Tage.
Auf dasselbe Ergebnis kommt man leichter, indem man so rechnet: 26 % Fertigstellung entsprechen 7 Tagen Verzug, 100 % Fertigstellung also 7 Tage x 100 % / 26 % = 27 Tage Verzug. Diese 27 Tage werden auf den ursprünglichen Liefertermin aufsummiert, um den voraussichtlichen Liefertermin zu berechnen. Das macht hier fast einen ganzen Monat.

- **Kostenprognose**: Sind zum Erreichen des aktuellen Fertigstellungsgrads (z.B. 26 %) beispielsweise 30 % des Budgets (=Kostenindex) verbraucht worden, dann berechnet man ebenfalls nach einem Dreisatz: 26 % Fertigstellung à 30 % Budget macht 100 % Fertigstellung für 30 % Budget x 100 % / 26 % = 115,4 %. Das heißt, voraussichtlich werden am Ende die Projektkosten 115,4 % des bisher geplanten Gesamtbudgets betragen. Die bisher klein erscheinende Kostenüberschreitung wird sich zum Projektende also schlimmstenfalls sehr aufsummieren.

Eine Earned-Value-Analyse kann man mit Hilfe des RM-Werkzeugs am besten für die RE- und RM-Tätigkeiten durchführen, weil das RM-Werkzeug vor allem die Informationen bezüglich der Anforderungen und deren Status enthält. Verwaltet das RM-Werkzeug jedoch den gesamten Entwicklungsprozess (im Sinne von anforderungsbasiertem Projektmanagement), dann kann auch eine Earned-Value-Analyse für das Gesamtprojekt mit den Daten aus dem RM-Werkzeug durchgeführt werden. Dazu muss z.B. das Status-Attribut der Anforderungen den gesamten Lebenszyklus abbilden.

Um die Earned-Value-Analyse mit Hilfe eines RM-Werkzeugs zu unterstützen, muss das Werkzeug die Verwaltung folgender Inhalte unterstützen:

- Für jede Anforderung wird in einem Status-Attribut ihr gesamter Lebenszyklus abgebildet und verwaltet, also von der Ermittlung über die Vereinbarung für ein bestimmtes Release, über das Implementieren, das Testen und die Auslieferung. Jedem Status-Wert wird auch ein Fertigstellungsgrad zugewiesen, z.B. wie in Tabelle 13 dargestellt.
- Für jede Anforderung ist in einem Attribut „Aufwand" ihr geplanter Umsetzungsaufwand hinterlegt. Diese Aufwände werden als Gewichtungsfaktoren gebraucht für die Ermittlung des tatsächlichen Fertigstellungsgrades des Gesamtprojektes. Die Anzahl der fertiggestellten Anforderungen allein ergibt noch nicht den Fertigstellungsgrad, da jede Anforderung verschieden aufwendig umzusetzen ist.
- Für einen Earned-Value-Bericht wird jeweils für jede Anforderung ihr prozentualer Fertigstellungsgrad ermittelt, beispielsweise wie in Tabelle 13 dargestellt. Dann ermittelt sich der Fertigstellungsgrad als gewichtetes Mittel über die Fertigstellungsgrade aller Anforderungen, wobei die Anforderungen nach ihrem geplanten Budget gemittelt werden. Aufwändige Anforderungen wiegen so also schwerer. Ist das nicht möglich, können alternativ nur die abgeschlossenen Anforderungen als fertiggestellt (100 %) gerechnet werden und alle anderen als unfertig (0 %).

Die Berechnung sowie alle anderen Analysen bleiben gleich. Die Formel für den Fertigstellungsgrad bleibt die gleiche und auch alle Analysen. Es muss allerdings bei der Festlegung des Planwertes des Fertigstellungsgrades bereits berücksichtigt werden, wie später der tatsächliche Fertigstellungsgrad ermittelt wird.
Wertet man nur die fertigen Anforderungen als „fertig", dann wird es zu Projektbeginn erst mal keinen messbaren Fortschritt geben. Dafür werden zum Projektende hin innerhalb kurzer Zeit viele Anforderungen fertig Ein solcher Planverlauf erlaubt jedoch keine ganz so frühzeitige Entdeckung von Planabweichungen. Darum empfehlen wir die anteilige Berücksichtigung von in Arbeit befindlichen Anforderungen.

- Um das tatsächlich verbrauchte Budget zu einem Zeitpunkt zu ermitteln, ist es nötig, zu jeder Anforderung in einem Attribut die bisher aufgelaufenen Kosten zu verwalten. Eine solche anforderungsbasierte Kostenerfassung im RM-Werkzeug könnte aber je nach Firma auch eine Doppelerfassung bedeuten und den Entwicklern zu kleinteilig vorkommen. Darum kann das verbrauchte Budget auch zu jedem Zeitpunkt aus einem anderen Werkzeug (Zeiterfassung, Projektmanagementwerkzeug, Controlling) ermittelt werden.

Zusätzliche Informationen, die nicht anforderungsbezogen sind, aber für den Earned-Value-Bericht gebraucht werden, sind der geplante Fertigstellungsgrad des Projektes zu einem Zeitpunkt und das Gesamtbudget des Projektes (*Budget at Completion*), die ebenfalls im Werkzeug verwaltet werden müssen, wenn hier der Earned-Value-Bericht berechnet wird.

Die folgende Tabelle (Tabelle 13) zeigt die Fertigstellungsgrade von Anforderungen, abhängig von ihrem Status, nach verschiedenen Autoren. [RuSo2009] beziehen sich offensichtlich nur auf den Erhebungs-, Dokumentations- und Vereinbarungsprozess der Anforderung. Darum entspricht ein Fertigstellungsgrad von 100 % dem Status „genehmigt", während bei [Eber2012] dieser Status 0 % entspricht, weil hier der Projektverlauf nach der Anforderungs-Genehmigung betrachtet wird. Die Prozentsätze in der linken Spalte eignen sich also für eine Earned-Value-Analyse des RE, die Status in der mittleren Spalte für die Earned-Value-Analyse der Implementierung. Da wir den gesamten Projekt-Zyklus umfassen wollen, schlagen wir ganz rechts eine weitere Skala vor.

Zur Unterstützung der Earned-Value-Analyse müssten für jede Anforderung die Attribute wie in Tabelle 13 dargestellt im Attributierungsschema enthalten und gepflegt sein. Einmalig eingetragen wird „Plan-Kosten", d.h. die für jede Anforderung geschätzten Kosten. Für jeden zu betrachtenden Zeitpunkt muss man die zeitabhängigen Werte „geplanter Wert", „tatsächliche Kosten" und „Status" einpflegen. Der Fertigstellungsgrad der Anforderung wird automatisch berechnet aus dem Status, dem jeweils ein Fertigstellungsgrad zugeordnet ist (vgl. Tabelle 14). Der erreichte Wert einer Anforderung wird berechnet aus dem Fertigstellungsgrad mal dem Aufwand. Die Werte für das Gesamtprojekt werden so berechnet:

- **Budget at Completion** = Summe der geplanten Kosten aller Anforderungen
- **Tatsächliche Kosten** = Summe der tatsächlichen Kosten aller Anforderungen
- **Status**: wird manuell bestimmt, als Ergebnis der Earned-Value-Analyse und deren Prognosen, aber auch der Einschätzung des Projektleiters darüber, ob eingetretene Verzüge wieder eingeholt werden können

- **Fertigstellung**: Der Fertigstellungsgrad des Projektes ist der gewichtete Mittelwert der Fertigstellungsgrade der Anforderungen, gewichtet nach Aufwand. In diesem Beispiel: (4000 € x 50 % + 2000 € x 10 % + 2000 € x 100 % + …)/ 30.000 €. Das Ergebnis ist eine Prozentzahl.
- **Erreichter Wert** = Summe der erreichten Werte aller Anforderungen

Anforderungs-Status	[Rupp & Sophist 2004]	[Eber2012]	[RuSo2009]	Unser Vorschlag
	Fertigstellungsgrad in Bezug auf Erhebungs-, Dokumentations- und Genehmigungs-prozess	Fertigstellungsgrad in Bezug auf Implementierung	Fertigstellungsgrad in Bezug auf das Gesamtprojekt	Fertigstellungsgrad in Bezug auf das Gesamtprojekt
Erzeugt	0 %		20 %	10 %
Abgezeichnet	30 %		30 %	
Abnahmekriterien vollständig	60 %			
Konsistenz mit dem Objektmodell erfüllt	75 %			
Konsistenz mit dem Prototyp erfüllt	90 %			
Verifiziert / geprüft	95 %		40 %	20 %
Genehmigt / vereinbart / freigegeben	100 %	0 %	50 %	25 %
Entworfen			60 %	
In Implementierung		10 %		50 %
Implementiert		50 %	80 %	70 %
Getestet/ abgeschlossen		100 %	100 %	100 %

Tabelle 13: Fertigstellungsgrade von Anforderungen, abhängig von ihrem Status

Datum: heute	Aufwand / geplante Kosten	Tatsächliche Kosten	Status	Fertigstellung	Erreichter Wert
Anforderung 1	4.000 €	1.800 €	in Implementie-rung	50 %	2.000 €
Anforderung 2	2.000 €	150 €	erzeugt	10 %	200 €
Anforderung 3	2.000 €	2.100 €	abgeschlossen	100 %	2.000 €
...					
Gesamtprojekt	**30.000 €**	**9.500 €**	**gelb**	**30 %**	**9.000 €**

Tabelle 14: Attributierung der Anforderungen, damit die Earned-Value-Analyse unterstützt wird.