Please quote as: Janzen, A.; Hoffmann, A. & Hoffmann, H. (2013): Anforderungsmuster im Requirements Engineering. In: Working Paper Series, Nr. 2, Kassel, Germany.

Fachgebiet Wirtschaftsinformatik

UNIKASSEL VERSITÄT

Working Paper Series

Kassel University
Chair for Information Systems
Prof. Dr. Jan Marco Leimeister

Nr. 2

Andrej Janzen, Axel Hoffmann, Holger Hoffmann

Anforderungsmuster im Requirements Engineering

Kassel, Mai 2013

Series Editor:

Prof. Dr. Jan Marco Leimeister, Kassel University
Chair for Information Systems

Pfannkuchstr. 1, 34121 Kassel, Germany
Tel. +49 561 804-6068, Fax: +49 561 804-6067

leimeister@uni-kassel.de

http://www.wi-kassel.de

Anforderungsmuster im Requirements Engineering

Andrej Janzen, Axel Hoffmann und Holger Hoffmann

Kassel University, Information Science, Kassel, Germany
andrej.janzen@wingas.de
{axel.hoffmann, holger.hoffmann}@uni-kassel.de

Abstract. Das Arbeitspapier soll die Grundlagen von Anforderungsmuster im Requirements Engineering (RE) darstellen. Dazu werden zuerst Begriffe rund um den Umgang mit Anforderungen verständlich gemacht. Nachdem die Begriffe erläutert wurden, wird die Integration des REs in die Softwareentwicklung dargestellt. Weiterhin werden die Tätigkeiten im Rahmen des Requirements Engineering aufgezeigt und worauf bei den einzelnen Tätigkeiten zu achten ist. Dazu werden auch einige Grundlagen zu Anforderungen behandelt, wie die unterschiedlichen Arten von Anforderungen und die Qualitätskriterien von Anforderungen. Abgeschlossen wird dieses Arbeitspapier mit der Vorstellung des Ansatzes der Anwendung von Anforderungsmustern. Es werden ähnliche Methoden innerhalb des REs beschrieben, die mit der Verwendung von Anforderungsmustern implizit verwendet werden. Durch die Vorstellung anderer Arten von Mustern innerhalb des Software Engineerings wird eine Abgrenzung dieser zu den Anforderungsmustern vorgenommen.

Keywords: Anforderungsmuster, Requirements Engineering, Qualitätskriterien

1 Einleitung

In diesem Arbeitspapier sollen einige Grundlagen zu den Themen Requirements Engineering, Requirements Management und Anforderungsmuster ausführlich behandelt werden. Zunächst gilt es, einige zentrale Begriffe des Requirements Engineerings zu klären und zu präzisieren (Abschnitt 2). Bei der Definition von Begriffen im ersten Unterabschnitt dieses Arbeitspapiers stehen nicht nur zentrale RE-Begriffe in der Betrachtung, sondern auch einige Begriffe von angrenzenden Phasen des REs innerhalb des Entwicklungsprozesses einer Software. Dabei handelt es sich um Begriffe, die zwar Inhalt vor- oder nachgelagerter Schritte des REs sind, jedoch auch innerhalb der RE-Literatur oftmals genutzt werden. Es sind meist Objekte und Artefakte, die phasenübergreifende Anwendung finden. Dazu zählen Begriffe wie z.B. Pflichtenheft oder Stakeholder. Das durch die Präzisierung und Beschreibung relevanter Begriff aufgebaute oder erfrischte Basiswissen ist erforderlich für die Auseinandersetzung mit den weiteren Themen dieser Arbeit. Dazu zählen unter anderem die Einordnung des REs in den Entwicklungsprozess einer Software (Abschnitt 3) sowie eine detaillierte

Beschreibung der essenziellen Tätigkeiten des REs (Abschnitt 4). Erstes hat das Ziel, einen groben Überblick zu verschaffen, wo RE und somit auch diese Arbeit überhaupt angesiedelt ist. Ebenfalls aus diesem Grund folgt ein Unterabschnitt, indem eine häufig in der Praxis existierende Unterscheidung von Anforderungen beschrieben wird (Abschnitt 5). Hierbei kann die Qualität der dokumentierten Anforderungen eine relevante Größe darstellen. Um dies näher analysieren zu können, gilt es zunächst in diesem Grundlagenblock die Qualitätskriterien für Anforderungen zu nennen und zu erläutern (Abschitt 6). Diese Kriterien bestimmen die Qualität einer Anforderung. Dieses Arbeitspapier schließt mit den Grundlagen von Anforderungsmustern ab (Abschnitt 7). Mittlerweile sind viele Methoden zur Wiederverwendung und Standardisierung von Anforderungen von Wissenschaftlern erarbeitet worden. Es erfolgt eine Wiedergabe dieser Konzepte, um darauf aufbauend zu beschreiben, was in dieser Arbeit unter Anforderungsmustern zu verstehen und was nicht zu verstehen ist.

2 Begriffe rund um das Requirements Engineering

An dieser Stelle werden diese Begriffe aufgegriffen und präzisiert, um für den weiteren Verlauf dieser Arbeit ein eindeutiges und einheitliches Begriffsverständnis zu Grunde zu legen.

Anforderungen
Die Begriffserläuterungen werden begonnen mit dem für diese Arbeit wichtigsten Begriff, dem Begriff Anforderung. Für diesen Begriff existiert eine Vielzahl unterschiedlicher, allerdings ähnlicher Definitionen (Versteegen, Heßeler et al. 2004). An dieser Stelle wird auf die Wiedergabe, Beschreibung und dem Vergleich unterschiedlicher Definitionen verzichtet. Aus folgenden zwei Gründen soll stattdessen lediglich die Begriffsdefinition aus dem IEEE Standard Glossary of Software Engineering aufgegriffen und wiedergegeben werden. Zum einen ist diese Definition durch die Zusammenarbeit vieler führender RE-Forscher entstanden. Zum anderen findet diese Definition des Begriffes Anforderung eine weite Verbreitung in der RE-Literatur. Den Begriff Anforderung definiert IEEE (1990) wie folgt:

1. Eine Bedingung oder Fähigkeit, die ein Benutzer benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen.
2. Eine Bedingung oder Fähigkeit, die ein System oder Teil eines Systems erbringen oder besitzen muss, um einen Vertrag, einen Standard, eine Spezifikation oder ein verlangtes Dokument zu erfüllen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Fähigkeit wie in 1. oder 2. referenziert.„

Gemäß dieser Definition gibt eine Anforderung ein fachliches oder technisches Leistungsmerkmal an die zu entwickelnde Anwendung wieder. Dabei sind Anforderungen klar von Lösungen zu trennen. Jede Anforderung darf nur ein Bedürfnis oder einen

angestrebten Nutzen beinhalten, dass es zu erfüllen bzw. erreichen gilt. Die Realisierung dieser gewünschten Systemfähigkeit oder Eigenschaft gehört nicht in die Beschreibung einer Anforderung. Anwendungsgebiet von Anforderungen ist somit der Problemraum, nicht der Lösungsraum (Balzert 2009; Rupp and SOPHISTen 2009; Partsch 2010). Diese lösungsneutrale Problembeschreibung nimmt der Auftraggeber vor. Denn der Auftraggeber kennt den Anwendungsbereich, indem die Software ein Problem lösen soll, am besten. Gleichzeitig verfügt er i.d.R. kein Wissen über Techniken und Methoden der Softwareentwicklung. Er beschränkt sich somit auf das „Was“ (Was muss die Software können). Der Auftragnehmer hingegen konzentriert sich auf das „Wie“ (Wie muss die Software entwickelt werden um die Anforderungen des Auftragnehmers umzusetzen). Diese Softwareentwickler verfügen über das technische Know-How um eine Software zielführend zu entwickeln. Sie verstehen allerdings nicht das durch den Einsatz der Software zu lösende Problem. Daher benötigen sie zur Entwicklung der gewünschten Software Anforderungen, die den Entwicklern die Wünsche des Auftraggebers eindeutig beschreiben.

Stakeholder

Der Auftraggeber und der bzw. die Auftragnehmer stellen wichtige Systembeteiligte dar. Jedoch gibt es weitere Personen, Gruppen und Institutionen, die ebenfalls potenzielles Interesse an der zukünftigen Software haben. All diese Systembeteiligten werden in dem RE-Bereich umfasst mit dem Begriff Stakeholder. Eine geeignete Definition für den Stakeholder-Begriff liefert Pohl/Rupp (2009):

> „Ein Stakeholder eines Systems ist eine Person oder Organisation, die (direkt oder indirekt) Einfluss auf die Anforderungen des betrachteten Systems hat.“

Demnach gehören nicht nur Personen oder Organisationen, die an der Softwareentwicklung direkt beteiligt sind zu den Stakeholdern dieser Software, sondern auch Rollen, die von den Auswirkungen der neuen Software betroffen sind. Konkrete Beispiele für Stakeholder sind Softwarenutzer, Administratoren der Software, Schulungs- und Trainingspersonal, Entwickler, Auftraggeber, Tester, Gesetzgeber, Betriebsrat usw. Da häufig nicht alle betroffenen Personen an dem Entwicklungsprojekt mitwirken können, sollten für alle Stakeholderrollen mindestens ein Vertreter bestimmt werden. Hierbei ist zu beachten, dass die ausgewählten Repräsentanten über präzises und aktuelles Verständnis der Bedürfnisse für ihre Rolle verfügen. Stakeholder sind u.a. eine der wichtigsten Quellen für Anforderungen. Somit sollten keine wichtigen Stakeholder übersehen werden. Das würde dazu führen, dass ggf. wichtige Anforderungen keine Berücksichtigung finden (Pohl and Rupp 2009). Das könnte das ganze Vorhaben gefährden. Das Identifizieren relevanter Stakeholder ist somit eine wichtige Teilaufgabe des Requirements Engineering, welche jedoch vor der eigentlichen Ermittlung von Anforderungen durchgeführt werden sollte (Glinz and Wieringa 2007; Partsch 2010).

Requirements Engineering.
Grob beschrieben ist das Requirements Engineering eine Disziplin, in der es um Anforderungen geht. Für eine genauere Begriffsbestimmung lassen sich viele, teilweise widersprüchliche Definitionen in der Literatur finden. Dabei sind große Abweichungen zwischen den einzelnen Tätigkeiten des REs vorhanden. Die Definition von Pohl (2008) beinhaltet keine Festlegung einzelner RE-Tätigkeiten. Auch wurde diese Definition in vielen Büchern und Zeitschriften zitiert. Aus diesen Gründen wird hier ebenfalls diese Definition zur Klärung des Begriffes Requirements Engineering wiedergegeben: „Das Requirements Engineering ist ein kooperativer, iterativer, inkrementeller Prozess, dessen Ziel es ist zu gewährleisten, dass

- alle relevanten Anforderungen bekannt und in dem erforderlichen Detaillierungsgrad verstanden sind,
- die involvierten Stakeholder eine ausreichende Übereinstimmung über die bekannten Anforderungen erzielen,
- alle Anforderungen konform zu den Dokumentationsvorschriften dokumentiert bzw. konform zu den Spezifikationsvorschriften spezifiziert sind."

Das RE zielt darauf ab, eine gemeinsame Basis über die Anforderungen an eine Software zwischen den Stakeholdern zu erreichen. Hierfür gilt es, nicht nur die zu entwickelnde Software zu betrachten, sondern auch die Umgebung, in der die Software eingesetzt werden soll (Ebert 2008). Denn es gilt zu berücksichtigen, was die Stakeholder von der Software in einer bestimmten Umgebung erwarten. Daher stehen im RE die Ziele im Mittelpunkt, die die Software innerhalb des Anwendungsbereiches erreichen muss. Jedoch gibt es für diese Analyse keinen standardisierten RE-Prozess. Die Abgrenzung der Haupttätigkeiten des Requirements Engineerings variieren stark zwischen einzelnen RE-Autoren. Zum Zwecke der Wiederspruchsfreiheit innerhalb dieser Arbeit ist eine Festlegung auf eine sinnvolle Abgrenzung der Haupttätigkeiten notwendig. Dazu wird die Abgrenzung der Haupttätigkeiten von Rupp/SOPHISTen (2009) und Grande (2011) übernommen. Nach ihnen besteht das RE aus folgenden vier Haupttätigkeiten:

- Anforderungen ermitteln,
- Anforderungen dokumentieren,
- Anforderungen prüfen und
- Anforderungen verwalten.

Alle vier Haupttätigkeiten zusammen werden häufig unter den Oberbegriff Anforderungsdefinition – gelegentlich auch mit dem Ausdruck „Spezifizieren von Anforderungen" – zusammengefasst (Balzert 2009, 445). In dieser Arbeit wird der Ausdruck „Spezifizieren von Anforderungen" nicht in diesem Zusammenhang gebraucht, da der Begriff Anforderungsspezifikation vielfach als Synonym für den Begriff Anforderungsdokument verwendet wird. Stattdessen wird im weiteren Verlauf dieser Arbeit der Begriff Anforderungsdefinition für die Zusammenfassung der vier Haupttätigkeiten des REs gebraucht.

Requirements Management.
Um innerhalb des REs ein Chaos zu vermeiden, bedarf es einer gewissen Ordnung wie z.B. die Sortierung großer Mengen von Anforderungen. Strukturiertes RE wird durch das systematische Verwalten von Anforderungen erreicht, also dem Requirements Management (kurz RM). Der deutsche Begriff für diese Disziplin lautet Anforderungsmanagement. Wie bei dem Begriff Requirements Engineering ist auch das Requirements Management in der Literatur teilweise widersprüchlich definiert. Die Abgrenzung zwischen RE und RM variiert von Autor zu Autor. So sehen einige Autoren das Requirements Management als den systematischen Ansatz zur Erhebung, Organisation und Dokumentation von Anforderungen (Schienmann 2002; Leffingwell 2003; Hood 2005). Diese Autoren betrachten das RE als einen Teilbereich des RMs. Sie verwenden den Begriff Requirements Management umfassend für alle Aufgaben in Umgang mit Anforderungen. Für sie ist also das Ermitteln, Dokumentieren und Prüfen Teil des RMs. Das sehen einige Autoren anders (Wiegers 2005; Ebert 2008; Pohl 2008; Pohl and Rupp 2009; Rupp and SOPHISTen 2009; Partsch 2010). Diese betrachten hingegen das RM als ein Teil vom RE an. Nach ihnen befasst sich das RM mit der Pflege, Verwaltung und Weiterentwicklung von Anforderungen in ihrem gesamten Lebenszyklus. Im Folgenden wird eine Definition des Begriffs Requirements Management nach Pohl/Rupp (2009) zitiert:

> „Die Anforderungsverwaltung (Requirements Management) […] umfasst alle Maßnahmen, die notwendig sind, um Anforderungen zu strukturieren, für unterschiedliche Rollen aufzubereiten sowie konsistent zu ändern und umzusetzen."

Hierbei erfolgt das Requirements Management begleitend zu allen anderen Aktivitäten des REs. Diese Ansicht, in der das RM eine Teildisziplin vom RE ist, wird auch in dieser Arbeit für die Unterscheidung zwischen RE und RM vertreten. Somit wird der Begriff Requirements Engineering als Oberbegriff für alle Tätigkeiten, die bei der Entwicklung von Software mit Anforderungen anfallen, verwendet. Der Begriff Requirements Management wird in dieser Arbeit dagegen in den Fällen benutzen, in denen speziell auf die Pflege, Verwaltung und Weiterentwicklung eingegangen wird. Besonders bei der Entwicklung komplexer Produkte bzw. Systeme ist das RM von großer Bedeutung für die Ergebnisqualität. Der Requirements Engineer sollte gerade bei solchen Entwicklungsprojekten geeignete Techniken und Richtlinien des RMs anwenden, um die Anforderungsdefinition und die weitere Verwendung der Anforderungen bestmöglich zu unterstützen.

Requirements Engineer
Der Requirements Engineer ist das Bindeglied zwischen den Stakeholdern und steht häufig im Mittelpunkt des Geschehens. Er ist in der Regel der einzige Projektbeteiligte, der direkten Kontakt zu allen Stakeholdern hat. Als Vertreter des Auftragnehmers muss er die Bedürfnisse hinter den Aussagen der Stakeholder erkennen und so dokumentieren, dass fachfremde Entwickler sie richtig verstehen und umsetzen können. Dazu muss ein Requirements Engineer den Anwendungsbereich sowie die Termino-

logie ausreichend verstehen. Zugleich muss er über genug IT-Wissen verfügen, um Bedürfnisse der Stakeholder, deren Umsetzung äußerst aufwendig oder gar unmöglich ist, frühzeitig zu erkennen und zu entschärfen. Die Verantwortung für die Ermittlung, Spezifikation, Dokumentation, Prüfung und Verwaltung der Anforderungen liegt bei ihm (Ebert 2008). Um diesen Aufgaben gerecht zu werden, muss er mit den Methoden und Tools für das RE und RM vertraut sein. Es ist wichtig, dass ein Requirements Engineer über all diese Fähigkeiten verfügt, da er i.d.R. den größten Einfluss auf die Entwicklung hat (Rupp and SOPHISTen 2009). Deshalb muss er großes Fingerspitzengefühl für die Bedürfnisse der Stakeholder besitzen. Je nach Projektgröße kann diese Rolle eine oder mehrere Personen wahrnehmen. Aus wirtschaftlichen Gründen ist es bei kleineren Entwicklungsprojekten empfehlenswert, die Rolle Requirements Engineer einer Person zu geben, die im Projekt gleichzeitig eine andere Rolle einnimmt (Versteegen, Heßeler et al. 2004). Oft wird hierbei der Projektleiter gewählt. In der RE-Literatur werden häufig Synonyme verwendet für diese Rollenbezeichnung, wie z.B. Anforderungsanalytiker, Anforderungsmanager, Systemanalytiker, Requirements Analyst usw. Innerhalb dieser Arbeit wird jedoch nur der Begriff Requirements Engineer verwendet. Einen wichtigen Meilenstein für den Requirements Engineer stellt das Anforderungsdokument dar.

Anforderungsdokument

Anforderungen werden traditionell in einem Anforderungsdokument gesammelt und verwaltet. Ein Anforderungsdokument ist ein Oberbegriff für das Resultat der Tätigkeiten des REs. Es wird gelegentlich auch als Anforderungsspezifikation, Software Requirements Specification, Anforderungsdefinition und Requirement Katalog genannt. Ein Anforderungsdokument enthält neben allen dokumentierten Anforderungen an die zu erstellende Software auch weitere Informationen (Partsch 2010). Zu diesen weiteren Inhalten gehören u.a. relevante Standards, Referenzdokumente, Kontextinformationen, Abnahmekriterien, Glossare usw. Hierbei gilt jedoch nicht „je mehr desto besser“, sondern das Dokument muss das erwartete Verhalten des zu entwickelnden Softwaresystems nur so vollständig wie nötig beschreiben. Damit das Anforderungsdokument qualitativ hochwertig ist, müssen einige Qualitätskriterien eingehalten werden (Rupp and SOPHISTen 2009). So muss ein Anforderungsdokument z.B. vollständig, konsistent, klar strukturiert, verfolgbar usw. sein. Eine vollständige Auflistung aller Qualitätskriterien an ein Anforderungsdokument kann in IEEE (1998)(1998, 4ff.) oder in Rupp/SOPHISTen (2009) entnommen werden. Es existieren immer mehr unterschiedliche Werkzeuge zur Unterstützung der Dokumentation von Anforderungen. Ein erkennbarer Trend der Werkzeugunterstützung ist die datenbankbasierte Dokumentation. Anforderungsdokumente dienen im Laufe der Projektlaufzeit als Grundlage für verschiedene Aufgaben wie z.B. zur Durchführung einer Ausschreibung (Pohl and Rupp 2009). Der erfahrene RE-Praktiker Ebert (2008) weist darauf hin, dass es äußerst hilfreich ist, für die unterschiedlichen Aufgaben mit verschiedenen Dokumenten zu arbeiten. Daher existieren verschiedene Ausprägungsformen von Anforderungsdokumenten. Im deutschen Sprachraum haben sich drei

Typen von Anforderungsdokumenten etabliert. Dabei handelt es sich um die Dokumente Kundenanforderungen, Lastenheft und Pflichtenheft (Schienmann 2002).

Das Dokument Kundenforderungen umfasst alle vom Auftraggeber geäußerten Bedürfnisse. Diese Rohdaten nimmt der Requirements Engineer nach Durchführung von Gewinnungsaktivitäten auf. Dazu muss er die dabei gesammelten Informationen zunächst einmal klar beschreiben. Nach Aufnahme müssen diese vagen Anforderungen in enger Zusammenarbeit mit den Stakeholdern analysiert, gefiltert, spezifiziert, bewertet und priorisiert werden. Die von dem Auftraggeber final abgestimmten und spezifizierten Anforderungen werden in das Lastenheft aufgenommen. Nach Überführung der Rohdaten wird das Dokument Kundenanforderungen nur zu Zwecken der Verfolgbarkeit benötigt, es wird jedoch nicht weiterverwendet (Deifel 1998). Das Lastenheft beinhaltet somit nur endgültig konsentierte Anforderungen. Nach DIN-Norm 69905 (DIN 69905 1997) wird das Lastenheft folgendermaßen definiert:

> „Das Lastenheft beschreibt die vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines Auftrages."

Dieser Definition ist entnehmbar, dass die Inhaltsbestimmung des Lastenhefts dem Auftraggeber unterliegt. Somit sind die darin enthaltenen Anforderungen und Rahmenbedingungen aus der Anwendersicht formuliert. Die definierten Ziele der Software sind folglich relativ abstrakt beschrieben. Die Detaillierung der Anforderungsbeschreibung im Lastenheft ist für die Softwareentwickler im Allgemeinen nicht ausreichend. Für die Spezifizierung der Auftraggeber-Anforderungen verwendet der Auftragnehmer nach Erhalt des Lastenhefts das Pflichtenheft. Somit dient das Lastenheft als Grundlage für die Erstellung des Pflichtenhefts (Pohl 2008). In der DIN-Norm 69905 (DIN 69905 1997) ist der Inhalt des Pflichtenhefts wie folgt beschrieben:

> „Vom Auftragnehmer erarbeitete Realisierungsvorgaben aufgrund der Umsetzung des vom Auftraggeber vorgegebenen Lastenheftes."

Der Fokus des Pflichtenhefts liegt somit auf der präzisen Darstellung der Softwareeigenschaften, die für die erfolgreiche Entwicklung zu implementieren sind. Dazu werden die Softwareanforderungen und Rahmenbedingungen des Lastenheftes im Pflichtenheft unter dem Gesichtspunkt der Detaillierung und Umsetzung wieder aufgegriffen. Der Einfluss des Pflichtenhefts auf den Erfolg des Entwicklungsprozesses ist hoch, da es die Grundlage für zahlreiche Entwicklungsaktivitäten bildet und für den gesamten Entwicklungsprozess als Referenzdokument gilt (Pohl 2008). Das Pflichtenheft, sofern es auch als das juristisch relevante Anforderungsdokument bestimmt wurde, sollte für die Abnahme der Software klar definierte Abnahmekriterien beinhalten.

Für die Entwicklung einer Software ist die Differenzierung zwischen Lasten- und Pflichtenheft im angelsächsischen Sprachraum dagegen unüblich (Pohl 2008). Stattdessen wird als Anforderungsdokument die Software Requirements Specification – kurz SRS – verwendet. Das SRS ist dabei grundsätzlich in drei Bereiche aufgeteilt

(IEEE 1998). Neben der Einleitung existiert jeweils ein Bereich für die allgemeine Beschreibung der zu erstellenden Software sowie für die spezifizierten Anforderungen. Die allgemeine Beschreibung wird i.d.R. von dem Auftraggeber vorgenommen. Somit sind hierbei die Anforderungen aus Sicht des Auftraggebers enthalten. Die spezifizierten Anforderungen stellen dagegen technische Aspekte in den Vordergrund. Die Aufnahme dieser Entwicklungsanforderungen unterliegt i.d.R. den Softwareentwicklern. Das SRS-Dokument umfasst somit den Zweck des Lasten- sowie des Pflichtenhefts in einem Anforderungsdokument. In dieser Arbeit wird dieser Anforderungsdokumenttyp nicht verwendet. Stattdessen erfolgt eine Anlehnung an die im deutschen Sprachraum etablierte Differenzierung der Anforderungsdokumente.

In diesem Abschnitt wurden einige Schlüsselbegriffe aus dem RE-Kontext aufgegriffen und erläutert. Zu diesen zählen die Begriffe Anforderung, Stakeholder, Requirements Engineering, Requirements Management, Requirements Engineer, und Anforderungsdokument. Diese Begriffsdefinition ist bei weitem nicht vollständig, da im RE viele weitere Fachbegriffe existieren. Das erläutern all dieser Begriffe wäre sehr umfangreich. Daher beschränkt sich dieses Arbeitspapier nur auf die Definition der wichtigsten Fachbegriffe. Sollten im weiteren Verlauf dieser Arbeit weitere Fachbegriffe verwendet werden, so werden diese in der Fußnote der gleichen Seite definiert.

3 Die Integration des RE in den Software-Entwicklungsprozess

Bevor weitere Grundlagen des Requirements Engineerings und Requirements Managements näher betrachtet werden, soll in diesem Abschnitt zunächst beschrieben werden, wie das RE in das Software Engineering eingebettet ist. An dieser Stelle soll aufgezeigt werden, welchen Einfluss eine bestimmte Vorgehensweise bei der Softwareentwicklung auf den Ablauf des REs hat. Dazu werden zunächst die unterschiedlichen Arten von Vorgehensmodellen zur Softwareentwicklung näher betrachtet. Anschließend werden die Konsequenzen der einzelnen Arten auf das RE-Vorgehen gezeigt.

Für die Abwicklung komplexer Entwicklungen ist ein koordiniertes Vorgehen unumgänglich. So ist es auch bei der Softwareentwicklung. Daher steht für die Softwareentwicklung mittlerweile eine Vielzahl von unterschiedlichen Vorgehensmodellen zur Verfügung. Nach Schienmann (2002) beschreibt ein Softwarevorgehensmodell „[…] den Zusammenhang aller Aktivitäten, Ergebnisse, Rollen und Techniken in der Anwendungsentwicklung.“ Solche Vorgehensmodelle sollen die Softwareentwicklung übersichtlicher gestalten, indem das Vorhaben in Phasen mit einzelnen kleinen Aktivitäten aufgespalten wird. Die Komplexität der Softwareentwicklung wird dadurch beherrschbarer. Neben des eigentlichen Entwicklungsprozesses beinhalten Vorgehensmodelle oft weitere Tätigkeitsbereiche oder Querschnittsprozesse. Die meisten Vorgehensmodelle beinhalten folgende Phasen: Planung, Analyse, Design, Implementierung, Test, Auslieferung sowie die Softwarewartung. Dennoch unterscheiden sich die Vorgehensmodelle teileweise sehr stark voneinander. Die ersten entwickelten Vorgehensmodelle schrieben einen streng sequentiellen Entwicklungs-

prozess vor, bei dem Rücksprünge zu vorherigen Phasen nur in seltenen Ausnahmen (Softwarelebenszyklus-Modell) oder nur zur vorhergehenden Phasen (Wasserfallmodell) möglich sind (Hansen and Neumann 2005). Diese stringenten Vorgehensmodelle fordern, dass eine neue Phase erst begonnen werden darf, wenn die vorhergehende Phase abgeschlossen ist und somit das Zwischenprodukt vollständig vorliegt. Die Weiterentwicklung dieser streng linearen Verfahren führte zu iterativen, inkrementellen Vorgehensmodellen wie z.B. dem RUP (Rational Unified Process) oder dem Spiralmodell. Bei diesen moderneren Vorgehensmodellen ist es nicht erforderlich, dass eine Phase komplett abzuschließen ist, bevor mit der nächsten begonnen wird. Stattdessen wird eine Tätigkeit in mehrere Durchläufen wiederholt. Ein sequenzieller Durchlauf der Entwicklungstätigkeiten wird als eine Iteration bezeichnet. Durch mehrere Iterationen der Tätigkeiten erfolgt eine inkrementelle Entwicklung, da in jeder Iteration die Software auf Grundlage der bereits vorhandenen Ergebnisse weiterentwickelt wird (Hansen and Neumann 2005). Nach dem Prinzip „Weniger ist mehr" sind die neusten, leichtgewichtigen Vorgehensmodelle entstanden. Zu diesen Vorgehensmodellen zählen u.a. das SCRUM wie auch XP (eXtreme Programming). Ein wesentliches Merkmal dieser agilen Vorgehensmodelle ist der ausgesprochene Minimalismus. Sie zielen darauf ab, den bürokratischen Aufwand sowie die Anzahl von Vorgehensregeln so gering wie möglich zu halten (Schienmann 2002). Die Anzahl der Ergebnistypen und -dokumente bleibt dabei sehr klein. Das Vorgehen ist stark inkrementell und iterativ, da kleine Teams in sehr kleinen Schritten arbeiten. Der Softwareentwicklungsprozess soll so flexibler und schlanker als bei den traditionellen Vorgehensmodellen werden. Agil steht hierbei für die Möglichkeit, die für die jeweilige Situation passende Methode aus einem schier unendlich großen Angebot zu wählen, da äußerst wenig Regeln vorab bestimmt werden. Schienmann (2002) prognostiziert, dass diese Form des Vorgehens in Zukunft zunehmen wird. Er begründet seine Prognose damit, dass diese weniger risikobehaftet sind. Diese verschiedenen Arten von Vorgehensmodellen weisen unterschiedliche Strategien auf, um Anforderungen zu erheben sowie mit Anforderungsänderungen umzugehen. Diese Unterschiede sollen im nachfolgenden kurz vorgestellt werden.

3.1 Lineare Vorgehensmodelle

Abbildung 1: RE im stringenten Entwicklungsprozess (in Anlehnung an Pohl 2008)

Um die Entwicklung einer Software, die den Auftraggeber nicht zufrieden stellt, zu vermeiden, wird oftmals ein unverhältnismäßig hoher Aufwand in der Analysephase

betrieben. Gemäß der Denkweise „To begin with the end in mind“ wird versucht, alle Anforderungen vollständig zu erheben, bevor die ersten Entwurfs- und Realisierungsentscheidungen getroffen werden. Somit integriert sich das RE bei dieser Vorgehensart wie in Abbildung 1 gezeigt.

Vertreter dieses stringenten Vorgehens plädieren für eine frühe, ausführliche Anforderungsermittlung, da spätere Änderungen äußerst aufwendig sind. Das Requirements Engineering ist hier eine abgeschlossene, zeitlich befristete frühe Phase der Softwareentwicklung. Änderungen im weiteren Verlauf sind nicht bzw. nur sehr eingeschränkt möglich. Sollten Änderungen in nachfolgenden Entwicklungsphasen doch nötig sein, so werden diese aufgrund zuvor abgeschlossenem RE in den Anforderungsdokumenten oft nicht nachgehalten Pohl (2008). Hierbei entsteht die Gefahr, dass bei längerfristigen Entwicklungen das Lasten- und Pflichtenheft nicht den aktuellen, sondern den Stand zu Projektbeginn abbilden. In solchen Fällen liegen veraltete Anforderungsdokumente des Systems vor. Das führt bei Folgeprojekten erneut zu einer Notwendigkeit einer zeit- und kostenintensiven Ist-Analyse für das RE. Um diese Problematik zu lösen, sollten Anforderungen, die sich nach Beendigung des REs ändern könnten, frühzeitig abgeschätzt werden (Ebert 2008).

3.2 Iterative/inkrementelle Vorgehensmodelle

Bei iterativen bzw. inkrementellen Vorgehensmodellen werden die RE-Tätigkeiten mehrfach durchlaufen. Das RE ist bei diesen Entwicklungsvorgehen keine anfängliche Projektphase, sondern eher ein begleitender Prozess der gesamten Entwicklung (Ebert 2008).

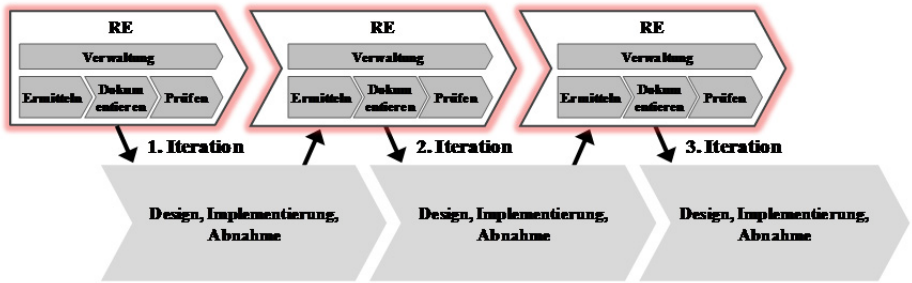

Abbildung 2: RE im iterativen/inkrementellen Entwicklungsprozess (in Anlehnung an Pohl 2008)

Das RE ist hier eine Querschnittstätigkeit. Anzahl der Iterationen sollten vor Projektstart definiert werden. Bei jeder Iteration werden weitere Anforderungen aufgenommen oder bestehende aktualisiert. Es herrscht eine kontinuierliche Wechselwirkung zwischen dem RE und dem restlichen Entwicklungsprozess. Die Intensität des REs ist jedoch nicht durchgehend gleich. Um eine qualitativ gute – nicht perfekte – Anforde-

rungsbasis zu erhalten, die es erlaubt, das erste Design mit einem akzeptablen Risiko zu beginnen, müssen in der ersten Iteration wesentlich mehr Ressourcen für das RE aufgewandt werden, als in den nachfolgenden Tätigkeitsdurchläufen (Schienmann 2002).

3.3 Agile Vorgehensmodelle

Auch bei dieser Vorgehensart ist das RE ein kontinuierlicher, phasenübergreifender Prozess. Jedoch existieren hier keine schwergewichtigen Regeln, die das Vorgehen einengen. Zu viele Einschränkungen beschränken die Kreativität und führen oft zu Bürokratie (Ebert 2008). Stattdessen soll der richtige Prozess für eine gegebene Projektumgebung sowie gegebene Produktziele geschaffen werden. Hierbei stehen das Projekt und der Anwender im Mittelpunkt, nicht die erschöpfende Dokumentation. Bei diesen leichtgewichtigen Vorgehensmodellen werden die benötigten Anforderungen erst kurz vor der Implementierung einer Funktion oder eines anderen Teils der Software ermittelt (Rupp and SOPHISTen 2009).

Abbildung 3: RE im agilen Entwicklungsprozess (in Anlehnung an Ebert 2008)

Der Aufwand für das RE ist über die gesamte Entwicklungszeit relativ gleichmäßig. Anforderungsdokumente wie z.B. Lasten- oder Pflichtenheft sind bei diesem Vorgehen fehl am Platz. Stattdessen werden viele Entscheidungen direkt im Code umgesetzt. Vertreter agiler Vorgehensmodelle begründen dies damit, dass eine sehr frühe Anforderungsdefinition einem „Hellsehen“ gleichkommt. Sie nehmen an, dass die Auftraggeber die Anforderungen zu Projektbeginn nicht komplett kennen (Ebert 2008). Einer der Nachteile dieses agilen REs ist die Gefahr, dass Inkonsistenzen in der Begriffsbildung und Realisierung entstehen können, da die Softwareentwicklung ohne ein Gesamtbild des Anwendungsbereiches beginnt. Das könnte zu Korrekturen führen, wodurch ein deutlicher Mehraufwand resultiert (Cao and Ramesh 2008).

Zusammengefasst kann festgehalten werden, dass die Intensität und der tatsächliche Ablauf des REs von dem gewählten Vorgehensmodell der Softwareentwicklung

abhängig ist. Dabei ist keines der Vorgehensmodelle pauschal das Beste für die Softwareentwicklung. Stattdessen muss abhängig der Problemstellung, also der spezifischen Projekt- und Produktcharakteristika wie z.B. Umfang und Komplexität der zu entwickelnden Software, die passende Vorgehensart ausgewählt werden (Rupp and SOPHISTen 2009). Eine vergleichende Bewertung der drei unterschiedlichen Arten von Vorgehensmodellen und deren Eignung in Abhängigkeit von kritischen Projekt- und Produktparametern ist in Ebert (2008) enthalten. Grundsätzlich sollte bei jedem Entwicklungsprojekt so wenig wie möglich aber so viel wie nötig dokumentiert werden.

4 Die RE-Tätigkeiten

Nachdem wichtige Begriffe rund um die Disziplin RE erläutert und die Integration des REs in den Prozess der Softwareentwicklung aufgezeigt wurden, ist in diesem Kapitel die Disziplin selbst Betrachtungsgegenstand. Die Disziplin RE ist ein Prozess, der die Aufgabe hat, alle relevanten Anforderungen an eine zu entwickelnde Software zu definieren. Im Folgenden werden die einzelnen RE-Aktivitäten näher betrachtet, die dazu nötig sind, um dieses Ziel zu erreichen (siehe Abbildung 4).

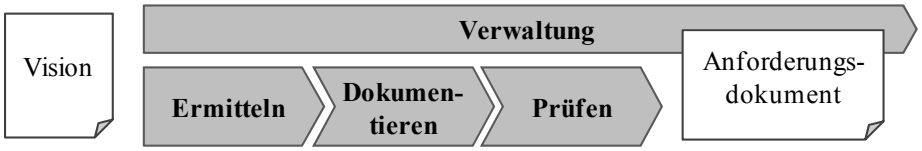

Abbildung 4: Die vier Haupttätigkeiten des Requirements Engineerings (in Anlehnung an Grande 2011)

Grundlage des REs ist eine zu Beginn vorgegebene Vision[1], die die angestrebte Veränderung beschreibt. Basierend auf der Vision werden für einen bestimmten Kontext Anforderungen ermittelt. Die zunächst informell vorliegenden Bedürfnisbeschreibungen und Anforderungen gilt es in einer weiteren Tätigkeit zu spezifizieren sowie zu dokumentieren. Doch bevor das Anforderungsdokument an die Entwickler weitergereicht wird, ist es nötig, die dokumentierten Anforderungen zu überprüfen. Dazu müssen sie nach der Verifikation und Validierung von den Auftraggebern abgenommen sowie von den Entwicklern akzeptiert werden. Die Pflege, Verwaltung und Weiter-

[1] Eine Vision definiert grob das Ziel eines Vorhabens, ohne dabei festzulegen, wie die gewünschte Veränderung erreicht werden kann. Es ist also eine Zielbeschreibung ohne Beschreibung der Lösung zur Erreichung des Ziels. In der Regel beinhaltet eine Vision eine Frist zur Erreichung des Ziels. Die Vision dient als Leitgedanke für alle Stakeholder der Entwicklung. Somit müssen alle an der Entwicklung beteiligten Personen ihr Handeln an die Realisierung der Vision ausrichten.

entwicklung von Anforderungen sollte nicht nur begleitend zu den zuvor genannten Tätigkeiten erfolgen, sondern über den gesamten Lebenszyklus der zu entwickelnden Software. Die an dieser Stelle nur grob beschriebenen RE-Tätigkeiten werden im Folgenden näher beleuchtet.

Doch davor sei hingewiesen, dass die optimale Vorgehensweise der Anforderungsdefinition von vielen Randbedingungen abhängt. Ist beispielsweise eine innovative Software zu entwickeln, so ist die Anforderungsdefinition ermittlungsintensiver als bei einer Weiterentwicklung einer schon in Anwendung befindlichen Software, für die ein ständig weitergepflegte Anforderungsdokument vorliegt. Solche Randbedingungen beeinflussen erheblich die Methodik, so dass keine allgemeingültige Vorgehensweise existiert (Hood 2005; Balzert 2009). Daher ist der Prozessablauf der oberen Abbildung nur als ein Vorgehensrahmen zu sehen, der an die jeweiligen Rahmenbedingungen anzupassen ist.

4.1 Ermittlung von Anforderungen

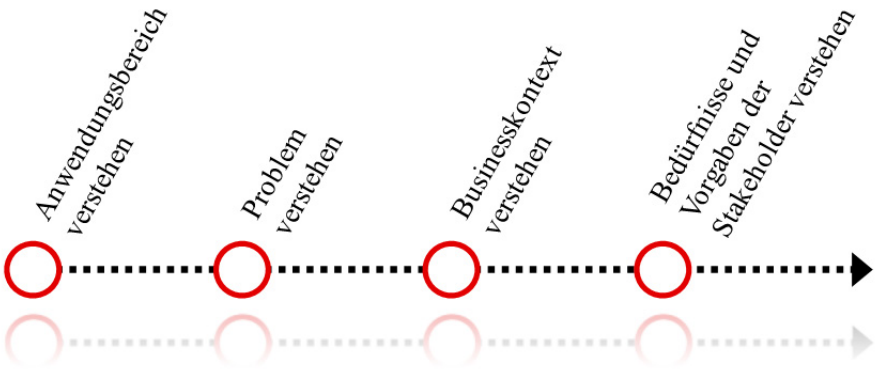

Abbildung 5: Phasen der Anforderungsermittlung (in Anlehnung an Kotonya and Sommerville 1998)

In dieser wohl schwierigsten Phase des REs geht es darum, alle Anforderungen an die zu erstellende Software zu identifizieren. Vor dem Beginn der Anforderungsermittlung sollte eine Vision vorliegen (Ebert 2008). Damit wird verhindert, dass Anforderungen ohne jegliche Orientierung an die endgültige Softwarelösung ermittelt werden. Denn in so einem Fall werden Anforderungen nicht im Kontext ermittelt und analysiert, da kein Ziel bzw. keine Vision zuvor vorgegeben wurde. Die Folge dieser falschen Anforderungsermittlung sind unnötige Fehler und weitere Aufwände zur Korrektur dieser Fehler. Daher sollten Anforderungen in dem vorgegebenen Kontext ermittelt und dokumentiert werden (Hood 2005). Ist der Anwendungsbereich der zu

entwickelnden Software nicht ausreichend bekannt, so ist es nötig, dass sich die Projektteilnehmer zunächst mit dem projektspezifischen Kontext auseinandersetzen. Die Abbildung 5 zeigt auf, womit sich die Stakeholder projektspezifisch auseinandersetzen müssen.

Anwendungsbereich verstehen
Der Bereich, in dem die Software angewandt werden soll, muss von dem Requirements Engineer und den Anforderungsstellern hinreichend verstanden werden. Anforderungen können nur brauchbar und vollständig ermittelt und beschrieben werden, wenn genügend Wissen über das Anwendungsgebiet vorhanden ist. Soll zum Beispiel ein Anwendungssystem für die Disposition von Sicherheitskräften auf einem Flughafen erstellt und eingeführt werden, so müssen alle Projektteilnehmer inkl. des Requirements Engineers über ausreichende Flughafenkenntnisse verfügen. Sie müssen wissen, welche unterschiedlichen Sicherheitsstufen auf einem Flughafen existieren und wie diese durch entsprechende Personalbesetzung zu erfüllen sind.

Problem verstehen
Eine erfolgreiche Softwareentwicklung bedarf auch der ausreichenden Kenntnis bezüglich des Grundes für die Entwicklung der Software. Details über das Problem, welches den Auftraggeber dazu veranlasst hat, die Entwicklung der Software zu beauftragen, müssen den Projektteilnehmern bekannt sein. Hiermit wird eine Zielvorgabe für das Projekt geschaffen. So ist es vorstellbar, dass in dem oben genannten Beispiel eine Gesetzesänderung die Sicherheitsstandards auf einem Flughafen erhöht, wodurch die Sicherheitskräfte in Zukunft in kürzerer Zeit an einen Ort gelangen müssen. Aus diesem Grund muss ein System zur Disposition der Sicherheitskräfte eingesetzt werden, welches besonders auf eine schnelle Disposition ausgelegt ist. Sind diese Informationen den RE-Beteiligten nicht bekannt, so kann daraus eine mangelhafte Fokussierung auf die Schnelligkeit der Disposition resultieren. In diesem Fall wäre möglicherweise nach Projektende zwar ein Dispositionssystem entwickelt und implementiert, jedoch kein zweckmäßiges. Das Vorhaben wäre somit gescheitert.

Businesskontext verstehen
Eine Software wird in der Regel für die Unterstützung oder sogar Ermöglichung von Geschäftstätigkeiten entwickelt und eingesetzt. Die an dem RE für die Entwicklung der Software beteiligten Personen müssen diese Geschäftstätigkeiten genügend durchschauen. Nur so kann eine Software aufgebaut werden, die bestmöglich ihren Einsatzzweck erfüllt. Bei dem oben erwähnten Beispiel sollten sich die RE-Beteiligten den Grund, die Durchführung, die Erfolgsfaktoren und ggf. weiteres Fachwissen zu der Disposition von Sicherheitskräften an einem Flughafen aneignen. Je größer dieses Fachwissen bei den Teilnehmern ist, desto besser können sie die erforderten Eigenschaften und Funktionen des Systems bestimmen.

Bedürfnisse und Vorgaben der Stakeholder verstehen

Die System-Stakeholder sind die Personen, die in direkter oder indirekter Beziehung mit der zu entwickelnden Software stehen (vgl. Abschnitt 2). System-Stakeholder können also die End-User, die Systemadministratoren, die Systemverantwortlichen usw. sein. Um die Stakeholder nach ihren Bedürfnissen zu befragen, gilt es zunächst die relevanten Stakeholder zu identifizieren. Sind die Stakeholder identifiziert, so gilt es ihre Wünsche hinsichtlich des Systems zu erfassen. Häufig beschreiben die Stakeholder jedoch nicht das was sie brauchen, sondern das was sie wollen (Ebert 2008). Sie schaffen es nicht, ihre Bedürfnisse aus der Benutzersicht zu beschreiben. Stattdessen äußern sie Lösungen. Dies kann dem Auftraggeber im Nachhinein schaden. Daher ist es für den Requirements Engineer äußerst wichtig, die Bedürfnisse und Vorgaben dieser Personengruppe im Detail intensiv zu analysieren und richtig zu verstehen. Auch reicht es nicht aus, die Stakeholder einfach nur zu fragen, was ihre Anforderungen sind. Das ist lange nicht ausreichend für eine gute Anforderungsermittlung. Denn Stakeholder beschreiben Bedürfnisse oft in den bestehenden Geschäftsprozessen und lehnen sich dabei stark an das existierende System an (Rupp and SOPHISTen 2009). Dabei ist das Ziel der meisten Softwareentwicklungsprojekte eine Lösung zu erarbeiten, die eine verbesserte Situation möglichst effektiv unterstützt. Die Autoren Hood/Wiebel (2005) beschreiben die Anforderungsermittlung als ein „[…] Versuch, Träume aus den Menschen heraus zu kitzeln, anstatt immer wieder die alten Lösungen zu sammeln."

Eine weitere bedeutsame Quelle für Anforderungen sind Dokumente. Oft lassen sich kontextbezogene Dokumente finden, die wichtige Informationen für das Entwicklungsprojekt enthalten. Solche branchen-, organisationsspezifische sowie allgemeingültige Dokumente wie Standards oder Gesetzestexte ermöglichen die Gewinnung erster Anforderungen. Die Erfahrungen von Hood/Wiebel (2005) zeigen, dass Anforderungsdokumente von Altsystemen oder vorhergehenden Versionen des gleichen Systems die bedeutsamste Anforderungsquelle ist. Denn viele der angestrebten Entwicklungen sind Weiterentwicklungen von bestehenden Systemen. Somit ist die Analyse von Dokumenten eine gängige und effektive Methode für die Ermittlung von Anforderungen.

Neben den Stakeholdern und Dokumenten existiert noch eine weitere wichtige Anforderungsquelle, die aktuell existierenden Systeme (Versteegen, Heßeler et al. 2004; Pohl 2008; Pohl and Rupp 2009). Durch das Analysieren existierender Systeme erhalten die Stakeholder einen Eindruck zu den Funktionalitäten und Eigenschaften der derzeitigen Lösung. Darauf aufbauend können sie Nachbildungen, Erweiterungen oder Änderungen fordern. Zugleich können im betrachteten System enthaltene Fehler identifiziert und somit in der zu entwickelnden Lösung vermieden werden. Relevante Systeme hierfür können neben den Alt- bzw. Vorgängersystemen auch Konkurrenzsysteme sein.

Ziel der Ermittlungsphase ist es, Anforderungen in einer hinreichend guten Qualität zu ermitteln (Ebert 2008). Sie müssen so detailliert spezifiziert sein, dass die Entwickler darauf aufbauen können. Das Ziel ist es also, Anforderungen „gut genug" zu ermitteln. So weist Ebert (2008) darauf hin, die Ermittlungsphase möglichst schnell –

jedoch nicht vorzeitig – abzuschließen. Dazu müssen alle wichtigen Stakeholder dem Ende zustimmen. Diesen möglichst schnellen Abschluss begründet Ebert damit, dass eine lange Ermittlungsphase zu einem unproduktiven Kreislauf von Spezifikationen, Nachbesserungen und Abstimmungsgesprächen führt. Es entsteht die Gefahr, dass eine übergewichtige Software resultiert, die auch noch möglicherweise aufgrund ihrer aufwendigen Entwicklung zu einer verspäteten Fertigstellung führt. Die Ermittlungsphase sollte nicht dadurch verkürzt werden, indem alle nötigen Schritte schneller oder verkürzt ausgeführt werden. Zur Verkürzung der Ermittlungsphase sollten stattdessen nicht ergebnismindernde Möglichkeiten genutzt werden. So schlägt Ebert (2008) vor, dass eine Verkürzung sinnvollerweise durch die Wiederverwendung von Anforderungen oder dem Aufbau von Druck in der Analysephase erreicht werden sollte. Weiter schreibt er, dass die Analysephase häufig zum „Aufwärmen“ dient und dadurch lange nicht den Druck hat, den andere Entwicklungsphasen haben. Mittels Druckaufbau sollen die Ergebnisse früher und dennoch in ausreichender Qualität kommen. Es existieren jedoch viele weitere Ansätze und Werkzeuge zur Ermittlung von Bedürfnissen und Anforderungen. Auf die Aufzählung und Beschreibung dieser einzelnen Ermittlungstechniken soll in dieser Arbeit verzichtet werden. Die Beschreibung der Techniken sowie der jeweiligen Vor- und Nachteile kann in den Werken von Rupp/SOPHISTen (2009) und Hood/Wiegel (2005) nachgelesen werden. Hickey/Davis (2003) warnen, dass es keine universelle Methode für die Anforderungsermittlung gibt sowie die Nutzung einer einzelnen Ermittlungstechnik oft nicht zielführend ist. Stattdessen empfehlen sie, Techniken situationsgerecht anzuwenden und ggf. zu kombinieren.

Der praxisorientierte RE-Autor Ebert (2008) weist zu Recht darauf hin, dass in der Ermittlungsphase des REs auf eine sorgfältige Strukturierung der gesammelten Bedürfnisse verzichtet werden sollte, da dies die Kreativität einschränkt. Es reicht also, die geäußerten Bedürfnisse zunächst nur vage zu dokumentieren. Wichtiger in dieser RE-Phase ist die Aufdeckung und Beseitigung von Konflikten zwischen Anforderungen. Wie bereits oben erwähnt werden Anforderungen aus verschiedenen Quellen gesammelt. In der Regel sind an der Anforderungsermittlung mehrere unterschiedliche Stakeholder beteiligt. Des Weiteren ist die Entwicklung erkennbar, dass die Anzahl der Stakeholder, die Anforderungen stellen, in den letzten Jahren aufgrund zunehmender Komplexität der Systeme gestiegen ist (Rupp and SOPHISTen 2009). Unterschiedliche Stakeholder haben oft verschiedene Standpunkte und Ziele, wodurch einander widersprechende Anforderungen leicht entstehen können (Boehm, Grunbacher et al. 2001; Ebert 2012). Pohl (Pohl 2008) warnt, dass solche Konflikte den Erfolg der Entwicklung gefährden. Daher gilt es im Rahmen der Anforderungsermittlung Konflikte aufzudecken und anschließend mit den Stakeholdern zu verhandeln (Deifel 1998). Hierfür müssen die verschiedenen Lösungsalternativen identifiziert und anschließend von den Stakeholdern gemeinsam eine der Alternativen ausgewählt werden. Dem Requirements Engineer stehen für die Verhandlungsphase verschiedene Lösungstechniken und -methoden zur Verfügung. Diese sind in Rupp/SOPHISTen (2009) sowie Pohl (2008) ausführlich beschrieben und bewertet. Die Priorisierung von Anforderungen sollte ebenfalls in der Anforderungsverhand-

lung erfolgen. Aufgrund von beschränkten Zeit-, Personal- und Kapitalressourcen ist es in vielen Entwicklungsprojekten nicht möglich, alle geäußerten Anforderungen umzusetzen (Pohl 2008). Während des REs liegt in der Regel eine gründliche Aufwandsschätzung für die Umsetzung einzelner Anforderungen noch nicht vor – diese erfolgen erst durch die Erstellung des Pflichtenhefts. Aus diesem Grund kann zu diesem Zeitpunkt nicht endgültig bestimmt werden, welche Anforderungen im Rahmen der gegebenen Ressourcen realisierbar sind. Ziel der Anforderungspriorisierung ist es, zu verdeutlichen, welche Anforderungen einen essenziellen Charakter und welche eher einen Goldrand-Charakter[2] besitzen (Schienmann 2002). Dazu werden die Anforderungen untereinander priorisiert. Hierfür muss eine Übereinstimmung im Hinblick auf die Bedeutung der jeweiligen Anforderungen unter den Stakeholdern verhandelt werden. Für die Priorisierung der Anforderungen ist es nützlich mehrere unterschiedliche Bewertungskriterien zu berücksichtigen. Neben Bewertungskriterien aus Sicht des Kunden wie beispielsweise der erwartete Nutzen müssen auch Bewertungskriterien aus Sicht der Entwickler wie etwas Kosten oder Risiken der Realisierung bei fließen (Schienmann 2002). Bei Zeitdruck oder drohenden Budgetüberschreitungen in der Entwicklung können dann Anforderungen mit einer geringeren Prioritätsstufe gestrichen werden.

4.2 Dokumentation von Anforderungen

Sind erste Informationen ermittelt worden, so gilt es sie anschließend geeignet zu dokumentieren. Die zunächst vagen Anforderungen werden mittels Ergänzung und Verfeinerung konkretisiert. Es wird die Brücke zwischen den vagen Beschreibungen der Ermittlungsphase hin zu korrekten, eindeutigen Produktanforderungen geschlagen (Kotonya and Sommerville 1998). Das zweckdienliche Niederschreiben der Anforderungen hat viele Gründe (Pohl and Rupp 2009). Einer der Hauptgründe ist die dauerhafte Erhaltung der im Projekt gewonnenen Informationen. Gleichzeitig entsteht eine gemeinsame Informationsbasis, auf die alle Stakeholder jederzeit zurückgreifen können. Durch das gemeinsame Dokumentieren wird eine kollektive Diskussion über das zu Dokumentierende gefördert. Solche schriftlich festgehaltene Anforderungen sind wesentlich stabiler als verbal geäußerte Informationen. Gleichzeitig wird die Abhängigkeit von einzelnen Wissensträgern vermindert. Dies kommt der Einarbeitung neuer sowie dem Ausscheiden bisheriger Mitarbeiter zugute. Besonders wichtig ist die Dokumentation der Anforderungen um die rechtliche Verbindlichkeit für den Auftragge-

[2] Der Begriff Gold Plating wird benutzt, wenn Funktionalitäten vom Requirements Engineer oder den Entwicklern hinzugefügt werden, die nicht von den Stakeholdern gewünscht waren. Durch Besitz tiefgreifender Kenntnisse dieser Akteure nehmen sie an, dass diese zusätzlichen Funktionalitäten die Software verbessern wird. Dieses Ausschmücken kann in vielen Fällen Projekte retten oder das Ergebnis wirklich verbessern, indem der Geschmack des Benutzers getroffen wurde. Jedoch ist diese Kreativität des Software Engineers oder Entwicklers schädlich, wenn sich die Zusatzfunktion für die Benutzer als nicht nützlich erweist. In diesem Fall waren die Spezifikation, die Entwicklung und die Implementierung unnötig.

ber und den Auftragnehmer zu fixieren. Sie erlaubt im Streitfall rechtliche Konflikte zügig zu klären.

Bei der Dokumentation von Anforderungen ist streng zu beachten, dass die dokumentierten Anforderungen von allen relevanten Beteiligten verstanden werden. Erreicht wird dies, indem die richtige Notation gewählt wird. Auf der einen Seite kann es zweckmäßig sein, unterschiedliche Notationen innerhalb eines Anforderungsdokumentes zu verwenden (Schienmann 2002). So sind bei den Entwicklern graphische Dokumentationsnotationen wie z.B. die UML favorisiert, wogegen für andere Stakeholder solche semi-formalen Sprachen unverständlich sind (Hood 2005). Ist die Beschreibung einer Anforderung für verschiedene Stakeholdergruppen bedeutsam, so sollte der abzubildende Inhalt parallel für jede Gruppe in der von ihnen am besten verstehende Notation dokumentiert werden. Auf der anderen Seite wird es mit zunehmender Anzahl unterschiedlich verwendeter Notationstechniken immer aufwendiger und komplexer, die Anforderungsbeschreibungen stets aktuell, miteinander verlinkt und widerspruchsfrei zu halten. Die Wahl der richtigen Notation hängt somit von vielen Faktoren ab. In der Praxis hat sich der Einsatz natürlichsprachlicher Dokumentationstechniken am besten bewährt (Pohl 2008; Rupp and SOPHISTen 2009). Auch wenn formale Notationen für einige Anforderungsbeschreibungen am geeignetsten sind, werden sie dennoch äußerst selten verwendet. Der Grund hierfür ist, dass die Anforderungen nach der Dokumentation überprüft und von dem Auftraggeber freigegeben werden müssen. Formale Notationen haben jedoch den Nachteil, dass sie von den Auftraggebern häufig nicht verstanden werden (Hood 2005). Eine grobe Auflistung der Vor- und Nachteile einzelner Notationsarten kann in Hood/Wiebel (2005) eingesehen werden. Rupp/SOPHISTen (2009) haben eine Empfehlung zur Beantwortung der Frage nach der geeigneten Notation geäußert. Nach ihren Erfahrungen sollte vorwiegend die natürliche Sprache zur Dokumentation von Anforderungen genutzt werden. Müssen für eine Anforderungsdokumentation komplexe Abläufe beschrieben werden, so sollten diese zusätzlich zu der natürlichsprachlichen Beschreibung mittels semi-formalen Diagrammen verfeinert werden.

Bei Nutzung der natürlichen Sprache zur Beschreibung von Anforderungen sind einige Hürden zu überwinden. So ist darauf zu achten, dass möglichst keine sogenannten Weak-Wörter[3] verwendet werden. Denn sie können leicht zu einer Mehrdeutigkeit der Beschreibung führen (Hood 2005). Ist die Verwendung eines solchen Wortes unumgänglich, so muss der Autor zuvor sicherstellen, dass eine Mehrdeutigkeit ausgeschlossen ist. Im anderen Fall würde bei der Prüfung einer mehrdeutigen Anforderungsbeschreibung eine intensive Diskussion entstehen. Daher gilt es, solche Gefahren bereits bei der Beschreibung der Anforderung zu umgehen. Ein konsistenter Still des Satzbaus der Anforderungsbeschreibungen hilft den Lesern, die Anforderungen besser zu verstehen (Hood 2005). Somit ist für die Verständlichkeit jegliche Nutzung von Formulierungsvorlagen sinnvoll. Problematisch wird es auch, wenn nicht

[3] Weak-Wörter oder oft auch als schwache Wörter bezeichnet stellen ungenaue Adjektive und Adverbien dar. Es handelt sich um Wörter, die keinen oder nur einen geringen Beitrag zur Satzinformation leisten (absolut, äußerst, schneller, circa usw.).

eindeutige Substantive verwendet werden. Damit alle Beteiligten eine „gemeinsame“ Sprache nutzen, ist es ratsam, die zu verwendenden Substantive zu begrenzen und eindeutig zu definieren (Pohl 2008). Dafür wird meist ein Glossar verwendet. In einem Glossar werden Glossarbegriffe mit anschließender Erklärung des jeweiligen Begriffes erfasst. Zu jedem Begriff gilt es auch die Synonyme zu sammeln. Für ein Beispiel eines solchen Glossars kann auf Rupp/SOPHISTen (2009) verwiesen werden. Das Niederschreiben einer verbindlichen Definition von Begriffen hat den weiteren Vorteil, dass dadurch alle Stakeholder die Bedeutung der Fachbegriffe nachlesen können (Partsch 2010). Denn sehr oft werden in dem Anwendungsbereich der zu erstellenden Software bestimmte Abkürzungen oder Fachbegriffe verwendet, die nicht allen Stakeholdern wie beispielweise dem Requirements Engineer oder den Entwicklern bekannt sind. Der Aufbau und die Verwendung eines Glossars ist daher vorteilhaft. Gleichzeitig erhöht sich jedoch dadurch der Aufwand für die Phase der Dokumentation von Anforderungen. Daher sollte bei jedem Projekt überlegt werden, ob es sinnvoll ist, ein Glossar zu verwenden.

Während des REs fallen bezüglich einer Anforderung viele zusätzliche Informationen an, die für spätere Phasen der Entwicklung von Relevanz sein können. Als zwei simple Beispiele sind hier die Priorität und die Art einer Anforderung zu nennen. Im Rahmen der Anforderungsspezifikation und -dokumentation müssen Maßnahmen getroffen werden, um diese wertvollen Informationen strukturiert festhalten zu können. Dazu offerieren Rupp/SOPHISTen (2009), den Inhalt der Anforderung von den zusätzlichen Informationen zunächst einmal zu trennen. Für die Dokumentation dieser zusätzlichen Informationen befürworten sie sogenannte Attribute[4] zu verwenden. Welche Anforderungsattribute zu dokumentieren sind, sollte in jedem Projekt individuell entschieden werden. Es ist jedoch zu beachten, dass nicht zu viele Attribute angelegt werden. Denn nicht nur die Planung und Erzeugung nimmt viel Zeit in Anspruch, sondern auch das anschließende Spezifizieren, Prüfen und Freigeben der einzelnen Attribute (Hood 2005). Wichtig ist auch, dass die Festlegung möglichst vor der Spezifikation der Anforderungen geschieht (Rupp and SOPHISTen 2009). Sind mehrere Iterationen für das RE vorgesehen, so sollte die Festlegung der zu dokumentierenden Attribute noch in der ersten Iteration erfolgen. Ein weiterer bedeutsamer Vorteil der Attributierung von Anforderungen ist die Möglichkeit einer Sortierung oder Filterung der Anforderungen. Dies ist für das Arbeiten bei großen Mengen von Anforderungen äußerst hilfreich. So können gezielt Teilmengen der Anforderungen herausgegriffen und bearbeitet werden.

Bei der Dokumentation der Anforderungen ist darauf zu achten, dass ein zweckausreichender Detaillierungsgrad erreicht wird. Generell ist der Auftraggeber für eine lösungsneutrale Problembeschreibung zuständig (Rupp and SOPHISTen 2009). Die Lösungsbeschreibung obliegt dem Auftragnehmer. Situationsbedingt kann es jedoch für den Auftraggeber sinnvoll sein, den Lösungsraum einzuschränken. Meist passiert

[4] „Ein Anforderungsattribut wird durch einen Attributnamen, eine zugehörige Attributsemantik sowie den Wertebereich des Attributs und der jeweiligen Wertesemantik definiert“ Pohl, K. (2008). Requirements Engineering. Heidelberg, dpunkt-Verlag..

eine derartige Einmischung in den Lösungsraum durch den Auftraggeber unbeabsichtigt. Der Anforderungssteller ist hierbei nicht in der Lage, die Anforderungen richtig zu beschreiben. Mögliche Folgen einer zu lösungsorientierten Anforderung ist, dass der Auftragnehmer z.B. durch den daraus bedingten Verzicht auf die Integration einer Standardlösung das tatsächliche Bedürfnis suboptimal löst.

Alle konkretisierten, um Attribute erweiterten Anforderungen[5] sowie ggf. weitere Informationen wie z.B. einem Glossar sind in einem Anforderungsdokument zusammenzutragen. Das Anforderungsdokument stellt somit das Ergebnis der Anforderungsdokumentation dar (Balzert 2009).

4.3 Prüfung von Anforderungen

Eine verbindliche Überprüfung der Qualität der einzelnen Anforderungsbeschreibungen sowie des gesamten Anforderungsdokuments ist im Verlauf des REs unentbehrlich. Eine solche Prüfung soll sicherstellen, dass die ermittelten und dokumentierten Anforderungen die Wünsche und Vorstellungen der Stakeholder adäquat wiedergeben. Dazu werden Mängel und Fehler in den einzelnen Anforderungsbeschreibungen sowie in den Anforderungsdokumenten aufgedeckt und gelöst (Pohl 2008). Diese Qualitätsüberprüfung ist zu unterziehen, bevor die Anforderungsdokumente für weitere Entwicklungsaktivitäten weitergegeben und verwendet werden. Die Anforderungsprüfung stellt somit den Abschluss der Anforderungsdefinition dar. Doch zu oft halten sich Stakeholder zurück, wenn es darum geht, eigene Arbeitsergebnisse zu prüfen (Hood 2005). Diese Einstellung ist fatal, denn die Nichtauflösung bestehender Fehler und Mängel in den Anforderungen gefährdet die gesamte Entwicklung. Fehler in den Anforderungsdokumenten beeinträchtigen alle weiteren Entwicklungsaktivitäten, da sie Referenzdokumente für alle weiteren Tätigkeiten sind. Die Anforderungsprüfung führt zu zusätzlichem Aufwand für die Anforderungsdefinition. Jedoch berichten Pohl/Rupp (2009), dass die durch eine Überprüfung resultierenden Vorteile wesentlich höher sind als die entstehenden Kosten. Die Prüfung der Anforderungen kann auf drei Teilaufgaben herunter gebrochen werden. Neben der Validierung und Verifikation gilt es in dieser Phase die Anforderungen für die nachfolgenden Entwicklungsaktivitäten freizugeben (Pohl 2008; Balzert 2009; Pohl and Rupp 2009; Grande 2011).

Den Unterschied zwischen den Aufgabeninhalten der Validierung und Verifikation hat Boehm (1984) wie folgt einprägsam dargelegt:

> „Verification. ‚Am i building the product right?'
>
> Validation. ‚Am i building the right product?'"

In Bezug auf das RE wird bei der Validierung also überprüft, ob die richtigen Anforderungen dokumentiert wurden. Die Verifikation hingegen geht der Frage nach, ob

[5] Für die ausführliche Erläuterung einer Anforderung sowie für die Attribute dieser Anforderung wird im weiteren Verlauf dieser Arbeit der Oberbegriff Anforderungsbeschreibung verwendet.

die ursprünglichen Bedürfnisse der Stakeholders treffend wiedergegeben werden. Balzert (2009) stuft diese Aufgabe als schwierig ein, da kein Dokument existiert, gegen das die Prüfung durchgeführt werden kann. Abhilfe kann hierfür mit der Verwendung von Prüftechniken erfolgen. Denn wie für die Ermittlung und die Dokumentation von Anforderungen stehen auch für die Überprüfung verschiedene Techniken zur Verfügung. Eine Auflistung, Erläuterung und Beurteilung der gängigsten Prüftechniken ist in dem Werk von Rupp/SOPHISTen (2009) nachlesbar. Die Techniken unterscheiden sich in dem Aufwand für die Anwendung sowie in der Anzahl der Personen, die daran teilnehmen. Aus der Fülle dieser Techniken gilt es die für die Projektgegebenheiten passenden einzusetzen (Rupp and SOPHISTen 2009). Eine Hilfe zur Auswahl der für ein Projekt geeigneten Techniken ist in Rupp/SOPHISTen (2009) enthalten.

Nach erfolgreicher Konsolidierung müssen die Anforderungen durch den Auftraggeber abgestimmt werden. Damit wird die Einigung bezüglich der Anforderungen unter den relevanten Stakeholdern erreicht. Ein weiterer positiver Aspekt der erforderlichen Freigabe der Anforderungen durch die relevanten Stakeholder ist, dass hiermit eine letzte Möglichkeit für Änderungen besteht, ohne dabei die nachfolgenden Entwicklungstätigkeiten stark zu beeinträchtigen (Pohl and Rupp 2009). Dies ist sinnvoll, da die Stakeholder im Verlauf des REs zusätzliches Wissen über die geplante Software erwerben, wodurch Meinungsänderungen entstehen können. Das Ergebnis der Anforderungsprüfung sind freigegebene und einem gewissen Qualitätsgrad genügende Anforderungen.

4.4 Verwaltung von Anforderungen

Das Anforderungsmanagement oder häufig auch als Anforderungsverwaltung bzw. Requirements Management bezeichnet widmet sich der Verwaltung von Anforderungen. Das Erfordernis dieser Tätigkeit ergibt sich aus den beiden Tatsachen, dass gesammelte Anforderungen im weiteren Verlauf der Entwicklung weiterverwendet werden sowie sich ändern können (Rupp and SOPHISTen 2009). Es ist daher wichtig, die dokumentierten Anforderungen dauerhafte zur Verfügung zu stellen und sinnvoll zu strukturieren. Auch der selektive Zugriff auf die Anforderungen muss in der gesamten Entwicklungszeit gewährleitstet sein. Das RM umfasst notwendige Maßnahmen um diese Aufgaben umzusetzen. Der Betrachtungsgegenstand der Anforderungsverwaltung sind einzelne Anforderungsbeschreibungen sowie ganze Anforderungsdokumente. Das RM erfolgt begleitend zu den anderen drei Haupttätigkeiten des REs und umfasst mehrere Teilaufgaben.

Die Nachvollziehbarkeit (Traceability) von Anforderungen steht im Mittelpunkt der Verwaltung von Anforderungen. Nach dem IEEE (1998) ist eine Anforderung nachvollziehbar, wenn sowohl ihre Herkunft als auch ihre Verwendung im Entwicklungsprozess verfolgbar sind. Einigen RE-Wissenschaftlern ist die alleinige Betrachtung der horizontalen Verfolgbarkeit für eine gute Nachvollziehbarkeit nicht ausreichend. So erweitern Gotel/Finkelstein (1994, 97) um Beziehungen zwischen den verschiedenen Verfeinerungs- und Spezifikationsschritten der Anforderungen – die evo-

lutionäre Verfolgbarkeit. Pinheiro (2003) sieht zusätzlich vertikale Verfolgbarkeit als unerlässlich an, die durch die Erfassung von Beziehungen zwischen Anforderungen entsteht. Eine hinreichende Nachvollziehbarkeit ist der Grundbaustein für viele weitere Hilfstechniken des REs und der Softwareentwicklung. So ist ein Änderungsmanagement ohne eine leistungsfähige Nachvollziehbarkeit von Anforderungen kaum zu bewältigen, da nicht feststellbar ist, welche Anforderungen von dieser Änderung betroffen sind (Edwards 1991). Ebenso wird die Wiederverwendung von Entwicklungslösungen erleichtert, indem die Verbindungen zwischen Anforderungen (Problembeschreibungen) und ihren Lösungen nachvollziehbar sind. Somit besitzt die Ausprägung der Nachvollziehbarkeit einen großen Einfluss auf die Ergebnisqualität der Softwareentwicklung (Pohl 2008). Die folgende Abbildung zeigt eine gängige Kategorisierung von Nachvollziehbarkeitsbeziehungen:

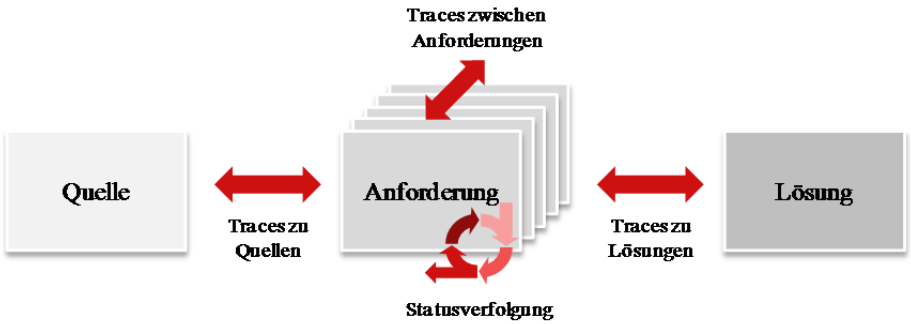

Abbildung 6: Die unterschiedlichen Kategorien von Nachvollziehbarkeitsbeziehungen (in Anlehnung an Brcina 2007)

Bei den Nachvollziehbarkeitsbeziehungen wird unterscheiden zwischen Traces in Richtung Umsetzung, Traces in Richtung Herkunft und Traces zwischen Anforderungen (Pohl 2008). Neben den Beziehungen der inhaltlichen Zusammenhänge sollten auch zeitliche Veränderungen der Anforderungen selbst erfasst werden, um die Aktivitäten im RE gezielt steuern und kontrollieren zu können. Dazu empfiehlt es sich zu jeder Anforderung ihren Status festzuhalten. Der Status einer Anforderung ist eine wichtige Information für das Projektmanagement. Er lässt erkennen, welche Aufgaben bisher durchgeführt wurden und welche Zustände noch erreicht werden müssen. Der Lebenszyklus einer Anforderung sowie die dabei annehmbaren Status werden in der Abbildung 7 dargestellt.

Wie bereits weiter oben erwähnt, sollte für eine hinreichende Verfolgbarkeit der Anforderungen der Fortschritt ihrer Implementierung hinreichend erfasst werden. Dazu empfiehlt Ebert (2008) den aktuell gültigen Zustand einer Anforderung der jeweiligen Anforderung als Statusattribut anzuhängen. Schienmann (2002) weist hin, dass für die Statusverfolgung Werkzeuge mit Workflow-Funktionalitäten genutzt werden sollten, da diese die Arbeit in weiten Teilen automatisieren können. Eine manuelle Statusverfolgung rät er ab, da diese durch die Vielzahl der möglichen Status

mit einem großen Aufwand verbunden ist. Auch sollten aus wirtschaftlichen Gründen und in Anbetracht der zeitlicher Beschränkungen auf eine vollständige Erfassung aller denkbaren Nachvollziehbarkeitsinformationen verzichtet werden (Pohl 2008). Eine Minimalstrategie ist hierbei angebrachter. Denn es ist viel sinnvoller, sich auf bedeutende Traces zu beschränken und diese umso professioneller zu pflegen, als Ressourcen für die Schaffung wertloser Informationen zu vergeuden (Schienmann 2002).

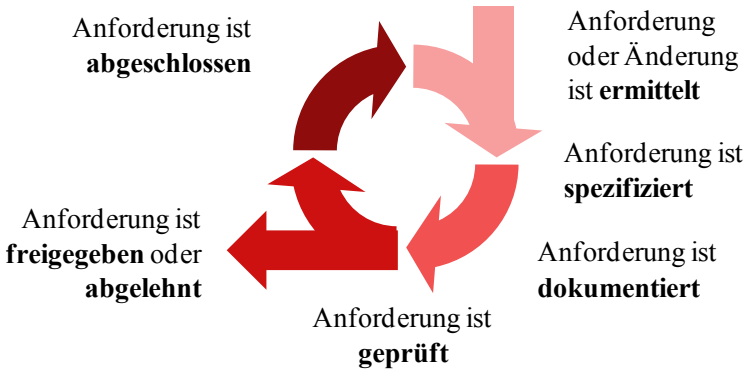

Abbildung 7: Der Lebenszyklus einer Anforderung (in Anlehnung an Ebert 2008)

Eine weitere Teilaufgabe des RMs ist das oben erwähnte Änderungsmanagement. Denn Anforderungen ändern sich im Laufe des Projektes. Es ist irrelevant, wie gewissenhaft und mühsam die Anforderungen zuvor erhoben, dokumentiert und geprüft wurden. Ebert (2008) berichtet, dass vor allem Anforderungen in Softwareentwicklungsprojekten sich im weiteren Entwicklungsverlauf stärker ändern als in anderen Entwicklungen in anderen Bereichen. Er begründet es damit, dass die Ermittlung und Bewertung der Anforderungen im Software Engineering im Vergleich zu anderen Gebieten äußerst undiszipliniert durchgeführt wird. Bei Entwicklungsvorhaben mit innovativen Produkten ergeben sich viele Anforderungsänderungen aufgrund der anfangs eingeschränkten Lösungsvorstellung auf die existierenden Erfahrungen (Ebert 2008). Bei dem nachfolgenden Spezifizieren dieser Anforderungen entwickelt sich ein Bild der neuen Situation wodurch neue Bedürfnisse auftreten. Ebert (2008) hat in den vergangen zehn Jahren mehr als 100 Projekte aus unterschiedlichen Branchen beobachtet und dabei festgestellt, dass die Änderungsrate der Anforderungen häufig mehr als 30% beträgt. Dies zeigt wie wichtig es ist, dass Änderungen beherrscht werden damit sie nicht die Entwicklung gefährden. Die zentrale Aufgabe des Änderungsmanagements ist es Änderungswünsche zu erfassen, bewerten, planen und umzusetzen (Schienmann 2002). Die Anforderungsbeschreibungen und -dokumente müssen dafür von Anfang an so aufgebaut sein, dass auch große Mengen von Änderungswünschen problemlos umsetzbar sind. Auch gilt es einen standfesten Prozess zum Änderungsmanagement zu definieren und etablieren (Pohl 2008). Natürlich soll-

te dieser in kleineren Projekten mit weniger Aufwand getrieben werden als in großen Projekten.

Mit der Nachvollziehbarkeit sowie dem Änderungsmanagement sind zwei der wichtigsten Teilaspekte des Requirements Managements behandelt worden. In der Literatur herrscht keine Einigkeit darüber, welches die einzelnen Maßnahmen des RMs sind. Da andere Maßnahmen des RMs für den Hauptteil der Arbeit von keiner Bedeutung sind, wird auf die Betrachtung dieser Maßnahmen verzichtet.

In diesem Abschnitt wurden die vier Haupttätigkeiten des REs aufgezeigt und erläutert. Dabei handelt es sich um die Ermittlung, die Dokumentationen, die Prüfung und die Verwaltung von Anforderungen. Die optimale Reihenfolge und Durchführungsintensität der Tätigkeiten hängt von vielen Randbedingungen ab und sollte daher projektspezifisch bestimmt werden.

5 Arten von Anforderungen

In der Literatur findet sich eine Vielzahl unterschiedlicher Herangehensweisen zur Klassifizierung von Anforderungen. Eine zweckmäßige Klassifikation ist hilfreich für die Verwaltung großer Mengen von Anforderungen. In diesem Abschnitt werden die häufigsten Klassifikationsmöglichkeiten vorgestellt.

Eine Möglichkeit ist die Unterteilung der Anforderungen nach dem Status, denn sie in ihrem Lebenszyklus einnehmen. Diese Klassifikation wurde bereits im Abschnitt 4.4 ausführlich aufgezeigt. Es existieren weitere Klassifizierungsmöglichkeiten von Anforderungen. So können sie nach der Priorität, der rechtlichen Verbindlichkeit oder mittels weiterer Attributwerte wie etwa dem Detaillierungsgrad klassifiziert werden (Rupp and SOPHISTen 2009). Die Einteilung der Anforderungen nach ihrer Priorität wird von dem Auftraggeber vorgenommen. Eine solche Anforderungspriorisierung ist nur bei der Anwendung inkrementeller Entwicklungsvorgehen sinnvoll (Ebert 2008). Die Software wird dabei gemäß der Prioritäten in Inkrementen entwickelt. Damit wird ein bestmögliches Ergebnis unter dem gegebenen Zeit- und Kostenrahmen erreicht. Die gleiche Absicht hat auch die Einteilung der einzelnen Anforderungen nach ihrer rechtlichen Verbindlichkeit. Hierbei beschreiben die Auftraggeber den Grad der Bedeutung, den sie für eine Anforderung ermessen. Dies ist eine äußerst wichtige Einteilung, die bei jedem Entwicklungsprojekt – unabhängig dem Vorgehen der Entwicklung – vorgenommen werden sollte (Rupp and SOPHISTen 2009). Die Einteilung der Anforderungen nach ihrer Priorität und nach ihrer rechtlichen Verbindlichkeit scheint auf den ersten Blick identisch zu sein. Der Unterschied ist, dass die Priorisierung bei komplexen Vorhaben, bei denen die Entwicklung inkrementell erfolgen soll, sinnvollerweise vorgenommen werden sollte. Die Einteilung nach rechtlichen Aspekten ist im Zusammenhang mit Verträgen unentbehrlich. Eine juristische Klassifizierung legt fest, welche Teile des Vertrages rechtlich einklagbar sind (Rupp and SOPHISTen 2009). Für diese Klassifizierung werden im deutschsprachigen Raum meist die Modalverben „muss“, „soll“ und „kann“ verwendet.

Eine andere, äußerst gebräuchliche Einteilung von Anforderungen ist die nach der Anforderungsart. Folgende Anforderungsarten haben sich dazu bewährt:

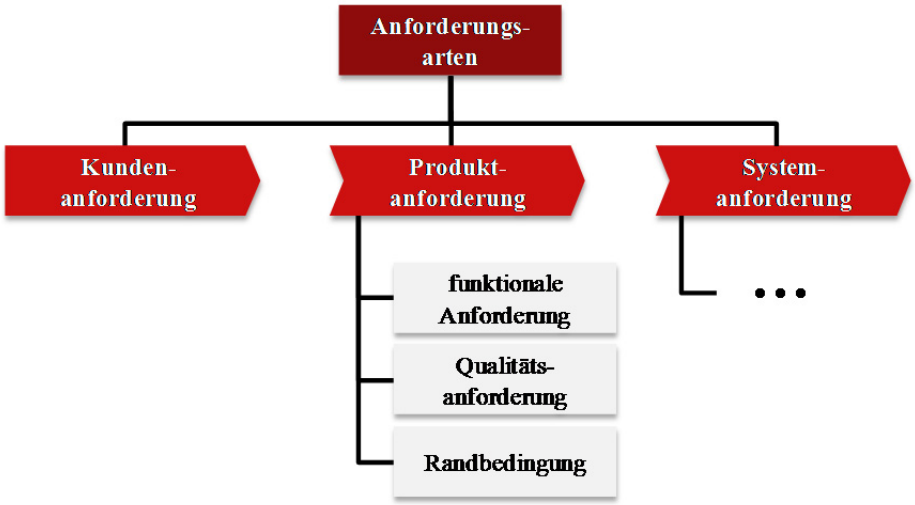

Abbildung 8: Klassifizierung von Anforderungen nach ihrer Art

Kundenanforderungen beschreiben die Bedürfnisse und Erwartungen des Auftraggebers (Chrissis 2003; Wiegers 2005). Zweck dieser Anforderungen ist es, den entstehenden Nutzen mit der zu entwickelnden Software zu erläutern. Diese Anforderungen beantworten das „Warum“ und „Weshalb“ diese Entwicklung überhaupt erfolgen soll (Ebert 2012). Sie werden auch als Markt-, Stakeholder- und Benutzeranforderungen sowie in dem angelsächsischen Raum als User-Requirements bezeichnet und sind in der Sprache des Kunden zu erfassen (Ebert 2012). Ein Beispiel für eine Kundenanforderung ist: „Wir möchten mit Hilfe der Software systematisch Flüge buchen und Sitzplatzreservierungen vornehmen können“. Viele Produktanforderungen werden von Kundenanforderungen abgeleitet (Schienmann 2002). Doch nicht jede Kundenanforderung wird in eine Produktanforderung übertragen. So ist es vorstellbar, dass im Rahmen der Anforderungsermittlung konfliktäre Bedürfnisse aus unterschiedlichen Quellen gesammelt wurden. Bei der folgenden Abstimmung müssen sich die Stakeholder auf eine dieser Kundenanforderungen festlegen.

Produktanforderungen sind die Anforderungen, die letztendlich in das Lastenheft aufgenommen werden. Sie werden in enger Zusammenarbeit zwischen dem Requirements Engineer und dem Softwarekunden definiert. Produktanforderungen beschreiben problemorientiert welche Geschäftsaktivitäten mit der Software wie unterstützt werden sollten. Somit beschreiben Produktanforderungen nicht, wie die Software technisch beschaffen sein muss. Daher sind sie noch sehr abstrakt, da sie keine konkreten Systemfunktionalitäten und -dienste beschreiben. Als Grundlage für einen

Entwicklungsvertrag sind sie somit ungeeignet. Formuliert werden sie in der Sprache der Anwender. Ist eine Ausschreibung der Entwicklung vorgesehen, so dienen die Produktanforderungen als Grundlage für die Ausschreibung. Produktanforderungen werden unterschieden in funktionale Anforderungen, Qualitätsanforderungen und Randbedingungen.

Funktionale Anforderungen beschreiben die geforderten Leistungen und das Verhalten der zu entwickelnden Software (Schienmann 2002). Sie lassen erkennen, wie eine Software auf bestimmte Eingaben oder Eingangsgrößen reagieren muss. Aktionen, die das System selbstständig ausführen soll, werden also ebenfalls mit funktionalen Anforderungen gefordert. Typischerweise nehmen sie den größten Teil der Anforderungsdokumente ein. Bei Verwendung der natürlichen Sprache für die Beschreibung der funktionalen Anforderungen ist darauf zu achten, möglichst in einfachen Sätzen zu schreiben. In vielen Fällen müssen funktionale Anforderungen mit Hilfe von Qualitätsanforderungen unterlegt werden.

Qualitätsanforderungen sind spezielle Produktanforderungen. Sie werden auch häufig als nicht-funktionale oder technische Anforderungen bezeichnet. Sie ergeben sich aus der Antwort auf die Frage, wie gut die Software die gestellten Aufgaben erfüllen soll (Balzert 2009). Als Beispiel kann hier das Antwortzeitlimit einer Funktion genannt werden. In der Regel beziehen sich Qualitätsanforderungen auf die Sicherheit, Zuverlässigkeit, Verfügbarkeit, Performance, Portabilität und Benutzbarkeit der Software (Schienmann 2002). Oft können sie funktionalen Anforderungen direkt zugeordnet werden. Manche Funktionen sind sogar ohne strenge Qualitätsanforderungen unbrauchbar. Nicht selten beziehen sie sich auf die ganze Software. Bei der Beschreibung der Qualitätsanforderungen ist darauf zu achten, dass sie gemessen und überprüft werden können. In der Praxis werden sie jedoch oft unterspezifiziert beschrieben (Ebert 2008; Partsch 2010). Sie werden als selbstverständlich erachtet oder nur unzureichend mittels Adjektiven dokumentiert. Vage Beschreibungen der Qualitätsanforderungen können zu Problemen bei der Realisierung und vor allem bei der Abnahme dieser Anforderungen führen. Aufgrund ihrer unpräzisen Formulierung werden sie häufig im Design nicht berücksichtigt oder können bei der Abnahme nicht auf Umsetzung überprüft werden. Für eine hohe Ergebnisqualität müssen daher Qualitätsanforderungen mit einem genauso hohen Maß an Präzision definiert werden wie funktionale Anforderungen. Die Ermittlung der Qualitätsanforderungen sollte parallel zur Definition von funktionalen Anforderungen erfolgen (Ebert 2008). Dass senkt das Risiko, dass wichtige Qualitätseigenschaften nicht in das Anforderungsdokument aufgenommen werden. Die Bestimmung ist jedoch nicht immer simpel, da Qualitätsanforderungen vielfach konfliktär sind (Balzert 2009). Ein Beispiel dafür ist die meist gegenläufige Beziehung zwischen Sicherheit und Benutzbarkeit.

„[…] organisatorische und/oder technische Restriktionen für das Softwaresystem und/oder den Entwicklungsprozess“ (Balzert 2009) werden mit Randbedingungen festgelegt. Alternativ werden sie als Rahmenbedingungen bezeichnet. Sie beziehen sich entweder direkt auf das Entwicklungsobjekt oder auf den Entwicklungsprozess des Objektes und sind daher ebenfalls produktspezifische Anforderungen. Beispiele

sind Kosten, Geschäftsprozesse, IT-Umfeld, Gesetze usw. Solche Restriktionen sind im Projekt nur schwer oder überhaupt nicht veränderbar (Pohl 2008).

Ist ein Entwickler bestimmt worden, so muss zunächst die zu entwickelnde Software von ihm designt werden. Um die Ziele und Absichten der Software zu berücksichtigen orientiert er sich an den Produktanforderungen. Als Ergebnis der Designphase resultieren die Systemanforderungen, die im Pflichtenheft festgehalten werden. Sie schaffen rechtliche Verbindlichkeit, da der Entwickler mit den Systemanforderungen detailliert beschreibt, was er entwickeln wird. Denn das Pflichtenheft ist die lösungsorientierte Antwort des Entwicklers auf die problemorientierte Anfrage des Auftraggebers. Doch vor ersten Entwicklungsaktivitäten müssen die Systemanforderungen vom Auftraggeber und Auftragnehmer beidseitig als Grundlage für die Realisierung und Abnahme des zukünftigen Systems akzeptiert werden. Die Formulierung der Systemanforderungen erfolgt in der Sprache der Techniker, da die Systemanforderungen von dem Softwareentwickler für die Entwicklung benötigt werden. Systemanforderungen lassen sich ebenfalls in unterschiedliche Klassen aufteilen. Da die Systemanforderungen jedoch kein Bestandteil des Requirements Engineerings darstellen, sollen diese hier nicht näher betrachtet werden.

Zusammengefasst kann festgehalten werden, dass Kundenanforderungen erste vage Bedürfnisäußerungen darstellen, die im Verlauf des REs genauer zu analysieren und spezifizieren sind. Für eine hinreichende Nachvollziehbarkeit müssen die Kundenanforderungen geordnet dokumentiert werden. Produktanforderungen umfassen alle entwicklungsrelevanten Anforderungen. Hierbei wird zwischen funktionalen Anforderungen, Qualitätsanforderungen und Randbedingungen unterschieden. Während die funktionalen Anforderungen festlegen, „was“ eine Software tun soll, beschreiben die Qualitätsanforderungen „wie“ die Software etwas durchführen soll (Balzert 2009). Sind Entwicklungseinschränkungen erforderlich, so werden diese mit der Vorgabe von Randbedingungen sichergestellt. Denn eine Entwicklung ist nur dann erfolgreich, wenn die entwickelte Software den vorgegebenen Randbedingungen genügt. Die Anforderungen mit dem höchsten Spezifikationslevel stellen die Systemanforderungen dar, da sie konkrete Lösungsansätze beschreiben. Sie geben detailliert Auskunft darüber, wie die Software entwickelt wird. Daher dienen die lösungsorientierten Systemanforderungen als Grundlage für den Entwicklungsvertrag.

6 Qualitätskriterien für Anforderungen

Um hochwertige Anforderungen zu erhalten, ist es wichtig zu wissen, was überhaupt eine qualitativ hochwertige Anforderung ausmacht. Daher sollen in diesem Abschnitt die qualitätsbestimmenden Eigenschaften der Anforderungen beschrieben werden.

Die Qualität von Anforderungen wird durch Qualitätskriterien bestimmt. Der Ingenieursverband IEEE hat dazu acht Qualitätskriterien für Softwareanforderungen definiert (IEEE 1998). Jedoch lag bei der Bestimmung dieser Qualitätskriterien der Fokus nicht auf eine einzelne Anforderung, sondern auf ein gesamtes Anforderungsdokument. Daher haben einige Wissenschaftler erkannt, dass die im IEEE-Standard 830

aufgelisteten Kriterien zur Qualitätssicherung einzelner Anforderungen nicht ausreichen, da hiermit nicht alle relevanten Aspekte abgedeckt sind. Es mangelt vor allem an Kriterien, die der Leserakzeptanz dienen. So fügt Ebert (2008) hinzu, dass Anforderungen auch verständlich, notwendig und realisierbar sein müssen. Den Grund für diese Erweiterung sieht er darin, dass diese drei Kriterien in der Praxis erforderlich sind. Ähnlich sehen es Rupp/SOPHISTen (2009) und erweitern die IEEE-Qualitätskriterien ebenfalls um diese drei sowie zusätzlich um die Kriterien abgestimmt, klassifizierbar und gültig & aktuell. Pohl (2008) sieht zusätzlich die Atomarität als eine relevante Eigenschaft an, die Anforderungen erfüllen müssen. Die folgende Abbildung gibt eine Übersicht der Qualitätskriterien nach dem IEEE sowie von drei bedeutsamen, deutschen RE-Forschern wieder:

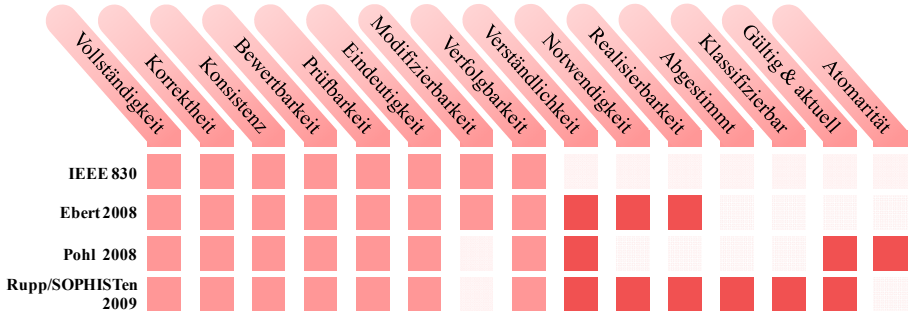

Abbildung 9: Auflistung der Qualitätskriterien an eine Anforderung unterschiedlicher Quellen (in Anlehnung an IEEE 1998; Ebert 2008; Pohl 2008; Rupp and SOPHISTen 2009)

Die in der Abbildung dargestellten Qualitätseigenschaften sind teilweise voneinander abhängig. So kann z.B. die Korrektheit einer Anforderung nur beurteilt werden, wenn sie auch der Verständlichkeit und Eindeutigkeit entspricht. Ebenfalls können Kriterien in einem Konflikt zueinander stehen. So kann beispielsweise das Reduzieren einer Anforderungsbeschreibung auf wenige Worte zwecks Verbesserung der Verständlichkeit gleichzeitig die Gefahr der Unvollständigkeit erhöhen.

Wie in dieser Übersicht zu erkennen ist, sehen Pohl (2008) und Rupp/SOPHISTen (2009) das IEEE-Qualitätskriterium **Modifizierbarkeit** nicht als notwendig für eine gute Anforderung an. Ebert (Ebert 2008), der jedoch diese Eigenschaft als notwendig betrachtet, beschreibt dieses Kriterium wie folgt: „Die Anforderung ist so beschrieben, dass erkennbar ist, was sich ändern könnte, wo Unsicherheiten vorhanden sind. Sie ist über Verknüpfungen so mit anderen Dokumenten verwoben, dass bei Änderungen der Einfluss erkennbar ist.“ Pohl (2008) und Rupp/SOPHISTen (2009) haben die Entbehrlichkeit dieses Kriteriums nicht begründet. Im nachfolgenden werden die restlichen Qualitätskriterien aus der obigen Abbildung grob charakterisiert.

Vollständig ist eine Anforderung, wenn die geforderte Funktion oder Eigenschaft komplett beschrieben ist. Unvollständige Anforderungen sollten durch einen entsprechenden Status oder einer Textanmerkung auch als solche gekennzeichnet werden.

Damit wird ein systematisches Suchen bzw. Hinzufügen fehlender Informationen ermöglicht.

Gibt eine Anforderung vollständig und ohne Wiedersprüche die Vorstellung des Stakeholders wieder, der sie geäußert hat, so entspricht diese Anforderung dem Qualitätskriterium der **Korrektheit**. Empfehlenswert ist es, wenn hierfür die Stakeholder die Anforderungen nach dem Formulieren lesen um zu erkennen, ob ihr Wunsch von dem Requirements Engineer richtig interpretiert und dokumentiert wurde. Eine Anforderung ist nicht korrekt, wenn sie unnötigerweise den Funktionsumfang vergrößert (sogenanntes Gold Plating).

Damit eine Anforderung dem Qualitätskriterium **Konsistenz** entspricht, muss sie hinsichtlich des Inhaltes und der Beschreibung gegenüber allen anderen Anforderungen frei von Wiedersprüchen und unkonfliktär sein. Aber auch in sich selbst müssen die Anforderungen konsistent sein.

Bewertbarkeit: Dieses Kriterium ist erfüllt, wenn die Wichtigkeit der Anforderung ermittelt und dokumentiert wurde. Diese Gewichtung der Anforderung sollte anhand von Regeln, Markterfordernissen und der Produktstrategie erfolgen. Rupp/SOPHISTen (2009) fügen hinzu, dass eine Gewichtung der Anforderungen umso relevanter ist, je höher die Komplexität oder Größenordnung der zu entwickelnden Software. Denn in solchen Situationen können meist nicht alle Anforderungen sofort im ersten Release umgesetzt werden. Eine zuvor von den Stakeholdern aufgestellte Rangfolge ermöglicht die Beschränkung auf die wichtigsten Anforderungen. Das erleichtert später die Auswahl der weiteren Anforderungen die umgesetzt werden sollen, falls der Umfang der Funktionalitäten im Nachhinein erweitert wird.

Eine Anforderung ist **prüfbar**, wenn sie so beschrieben ist, dass sie testbar ist. Hierzu muss sie eindeutig beschrieben sein und sich durch eine begrenzte Anzahl von Testfällen prüfen lassen. Dafür werden oft Abnahmekriterien definiert. Unterspezifizierte Anforderungen können nicht objektiv auf Erfüllung an der entwickelten Software überprüft werden.

Wenn eine Anforderung nur auf eine Art und Weise verstanden werden kann, dann ist diese **eindeutig**. Es ist wichtig, dass eine Anforderung nicht unterschiedlich interpretierbar ist, da sonst die Gefahr entsteht, dass diese Anforderung nicht wie gewünscht realisiert wird. Daher ist es ratsam, die Anforderungen von mehreren bzw. allen Stakeholdern lesen zu lassen. Alle Leser sollten dabei den Inhalt einer Anforderung gleich interpretieren. Ist ein Teil oder die ganze Anforderung für einen Leser nicht eindeutig interpretierbar, so sollte diese als offener Punkt markiert werden. Die Beschreibung der Anforderungen in der natürlichen Sprache hat den signifikanten Nachteil, dass die natürliche Sprache inhärent mehrdeutig ist (Pohl 2008). Berry/Kamsties/Krieger (2003) schreiben, dass hierfür vier Ursachen existieren. Zum einen kann eine Mehrdeutigkeit lexikalische Gründe haben. In der natürlichen Sprache sind Synonyme und Homonyme vorhanden, die unterschiedliche Interpretationen zulassen können. Als Beispiel kann das Wort Ball genannt werden, welches sowohl für ein kugelförmiges Sportgerät wie auch für ein feierliches Tanzvergnügen verwendet wird. Zum anderen kann eine Mehrdeutigkeit entstehen, wenn einer Wortfolge mehr als eine grammatikalische Struktur zugeordnet werden kann. So ist z.B. die

banale Anforderung „Das System muss mehrsprachige Hilfe und Systemeigenschaften anzeigen können“ unterschiedlich interpretierbar. Es kann so verstanden werden, dass das System sowohl die Hilfe als auch die Systeminformationen in mehreren Sprachen anzeigen muss. Andererseits kann die Anforderung so verstanden werden, dass sich das Eigenschaftswort mehrsprachig nur auf die Hilfe bezieht. In diesem Fall liegt eine strukturelle Mehrdeutigkeit vor. Auch kann eine Mehrdeutigkeit aufgrund von semantischen Fehlern entstehen. Diese liegen vor, wenn in einem gegebenen Kontext ein Satz unterschiedlich interpretierbar ist. Ein Negativbeispiel hierfür wäre die Anforderung „Alle Systemfunktionen müssen einen Auslöser haben“. Einerseits ist die Anforderung so zu verstehen, dass jede Systemfunktion einen beliebigen Auslöser haben muss. Liegt jedoch bei der Betrachtung der Anforderung der Fokus auf den Auslöser, so kann die Anforderung meinen, dass alle Systemfunktionen den gleichen Auslöser haben müssen. Letzte Fehlerquelle für Fehlinterpretationen ist die referentielle Mehrdeutigkeit. Diese liegt vor, wenn für ein Element in einem Satz, welches einen Verweis auf einen sprachlichen Ausdruck referenziert, der referenzierte sprachliche Ausdruck nicht eindeutig bestimmt werden kann. So ist es z.B. bei dieser Anforderung: „Das System muss eine Druck- und Exportfunktion enthalten. Diese muss vom Administrator konfigurierbar sein“. Hier ist nicht eindeutig, welche Funktion konfigurierbar sein muss. Die Druckfunktion? Oder doch etwa die Exportfunktion?

Für die Vermeidung von Mehrdeutigkeiten in natürlichsprachlichen Anforderungen müssen all diese beschriebenen Gefahrenquellen dem Requirements Engineer bekannt sein. Die Eindeutigkeit von natürlichsprachlichen Anforderungen kann auch durch Einsatz geeigneter Techniken gewährleistet werden. Pohl (2008) zählt zu diesen Techniken die Verwendung von Glossaren, Einsatz von syntaktischen[6] Vorlagen und die Einschränkung der natürlichen Sprache zu einer Normsprache auf. Ein Glossar ist eine Sammlung der Definitionen von Fachbegriffen für einen bestimmten Anwendungsbereich. Durch das Niederschreiben der Terminologie wird die Gefahr der lexikalischen Mehrdeutigkeit reduziert, indem der irrtümliche Gebrauch von Synonymen und Homonymen unter der Projektbeteiligten vermieden wird. Mithilfe der Nutzung von syntaktischen Schablonen werden typische Fehler bei der Formulierung von natürlichsprachlichen Anforderungen vermieden (Rupp and SOPHISTen 2009). Fokus solcher natürlichsprachlicher Satzschablonen ist die Syntax der natürlichsprachlichen Anforderungen, nicht ihr Inhalt. Eine Normsprache hingegen legt die syntaktischen Strukturen sowie teilweise den Inhalt in Bezug auf die zugrunde liegende Domäne fest (Schienmann 2002). So wird z.B. für Synonyme ein Begriff vereinbart. Nur diese festgelegten Strukturen und Begriffe dürfen für die Formulierung von Anforderungen genutzt werden. Der Einsatz einer solchen Normsprache vermeidet durch die Standardisierung der natürlichen Sprache Mehrdeutigkeiten und Missverständnisse zwischen den Stakeholdern (Pohl 2008). Anforderungen können allerdings nicht nur natür-

[6] Syntax – hier speziell die Satzsyntax gemeint – definiert die Lehre des Satzbaus. Die Syntax behandelt Regeln und Muster, nach denen Wörter zu Sätzen oder Satzteilen zusammengestellt werden. Die Syntax ist somit Teil der Grammatik.

lichsprachlich sondern auch in Diagrammen formuliert werden. Rupp/SOPHISTen (2009) werten, dass nach festen Regeln definierte Anforderungen in Form einer grafischen Darstellung weniger Interpretationsspielraum zulassen als wenn sie in der natürlichen Sprache aufgenommen werden. Jedoch kann der Aufwand für die grafische Anforderungsbeschreibung schnell unverhältnismäßig groß werden. In diesem Fall ist es ratsam, auf diese Beschreibungsart zu verzichten und stattdessen die Anforderungen natürlichsprachlich zu dokumentieren.

Das Kriterium **Verfolgbarkeit** verlangt, dass der Ursprung der Anforderung, deren Umsetzung und die Beziehung zu anderen Dokumenten nachvollziehbar sind (IEEE 1998). Die Anforderungen sollten ebenso mit abhängigen Anforderungen gleicher und verschiedener Anforderungsarten verknüpft werden. Sind z.B. aus einer funktionalen Anforderung weitere abhängige funktionale und nicht-funktionale Anforderungen abgeleitet, so müssen auch diese bei der Annullierung der ursprünglichen Anforderung aufgehoben werden. Für die Verfolgbarkeit von Anforderungen wird meist eine eindeutige Anforderungsnummer verwendet. Es ist wichtig, dass die Anforderungsnummer über den gesamten Lebenszyklus der Anforderung unverändert bleibt. Nur so kann ein funktionierendes Tracing gewährleistet werden, was jedoch Voraussetzung für eine nachhaltige Systementwicklung unter sich ändernden Anforderungen ist.

Die Anforderungen müssen für alle Beteiligten und Betroffenen **verständlich** beschrieben sein. Eine Anforderung ist dann verständlich, wenn ihr Inhalt möglichst einfach für alle verschiedenen Interessenvertreter interpretierbar ist. Durch die Verwendung kurzer Sätze in Aktivform für die Anforderungsbeschreibung kann eine gute Verständlichkeit erreicht werden. Pro Satz sollte auch nur maximal eine Anforderung ausgedrückt werden. Die Qualitätsmerkmale Eindeutigkeit und Verständlichkeit sind unabhängige Kriterien. Somit kann eine Anforderung verständlich aber auch gleichzeitig mehrdeutig oder eindeutig und zugleich nicht verständlich sein. Die Verständlichkeit ist auch stark von der gewählten Dokumentationsform abhängig. Daher kann es sinnvoll sein, die Dokumentationstechnik entsprechend der Zielgruppe anzupassen. Rupp/SOPHISTen (Rupp and SOPHISTen 2009) merken an, dass es für die Verständlichkeit vorteilhaft ist, in der Analysephase eine gemeinsame Sprache für alle zu schaffen, da hierbei das breiteste Spektrum der Stakeholder teilnimmt.

Ein weiteres Qualitätskriterium stellt die **Notwendigkeit** dar. Eine Anforderung sollte nur dann aufgenommen werden, wenn ihr Inhalt für die Zielgruppe einen konkreten Nutzen darstellt. Unnötig Eigenschaften und Leistungen, die nicht der Erfüllung eines Systemziels dienen, dürfen demnach nicht aufgenommen werden. Somit müssen sich alle Anforderungen auf eine konkrete Zielvorgabe beziehen. Um die Notwendigkeit einer Anforderung prüfen zu können, muss diese Anforderung zurück zu den Zielen verfolgbar sein. Somit setzt dieses Qualitätskriterium die Erfüllung der Verfolgbarkeit einer Anforderung voraus.

Ist eine Anforderung bzw. ihre Funktionalität oder Eigenschaft nicht **realisierbar**, so darf diese auch nicht aufgenommen und in nächsten Entwicklungsphasen berücksichtigt werden. Eine Anforderung ist eine realisierbare Anforderung, wenn sie innerhalb der gegebenen organisatorischen, rechtlichen, technischen, zeitlichen und finan-

ziellen Rahmenbedingungen umsetzbar ist. Um die technologischen Grenzen der Umsetzung zu berücksichtigen, empfiehlt es sich, einen Mitarbeiter aus dem Entwicklungsteam an der Anforderungsbestimmung zu beteiligen (Schienmann 2002). Für die Beurteilung der Realisierung einzelner Anforderungen müssen auch die dafür geschätzten Umsetzungskosten einbezogen werden. Viele Funktionalitäten werden von den Stakeholdern verworfen oder zurückgezogen, nachdem ihnen die für die Anforderung anfallenden Umsetzungskosten verdeutlicht werden (Rupp and SOPHISTen 2009). Im Verlauf des Entwicklungsprojektes sollte auch möglichst früh geprüft werden, was technisch innerhalb des vorhandenen Budgets machbar ist. Dies würde helfen, sich frühzeitig auf die nötigsten Funktionalitäten zu beschränken und daher weitere Ressourcen für die Erhebung und Beschreibung nicht umsetzbarer Anforderungen zu ersparen.

Eine Anforderung erfüllt das Qualitätskriterium **abgestimmt**, sofern alle Stakeholder die Anforderung als einwandfrei und gültig akzeptieren. Wichtig ist, dass die Akzeptierung sowohl von Auftraggebern als auch Seitens der Auftragnehmer erfolgt. Die Auftraggeber müssen die Anforderungen akzeptieren, da sie für die Umsetzungskosten aufkommen müssen. Die Geltung der Auftragnehmer ist deshalb erforderlich, weil sie für die Umsetzung der Anforderungen unter den geltenden Rahmenbedingungen verantwortlich sind. Daher ist es zwingend erforderlich, dass alle Beteiligten einen konfliktfreien Konsens über die Softwareanforderungen erreichen. Dies wird jedoch immer schwieriger, da Systeme immer komplexer werden und immer häufiger mehrere Unternehmensbereiche vernetzen, wodurch die Anzahl der System-Stakeholder zunimmt (Rupp and SOPHISTen 2009). Um trotz dieser Veränderungen weiterhin möglichst einfach und schnell einen gemeinsamen Konsens zu erreichen, wurden diverse Konsolidierungstechniken geschaffen. Ebenfalls kann es eine große Hilfe sein, wenn vor der Abstimmung die Wichtigkeit der einzelnen Anforderungen ermittelt und dokumentiert wurde. Daher steht dieses Qualitätskriterium in enger Beziehung mit dem Kriterium Bewertbarkeit.

Rupp/SOPHISTen (2009) postulieren, dass Anforderungen bezüglich der juristischen Verbindlichkeit **klassifiziert** werden müssen. Für jede einzelne Softwareanforderung ist die rechtliche Relevanz festzulegen. Dies soll zum einen die Bedeutung für die Auftraggeber aufzeigen. Zum anderen wird hiermit die Einklagbarkeit als rechtlich verbindlichen Vertragsbestandteil sichergestellt.

Ein weiteres Qualitätskriterium ist, dass Anforderungen **gültig und aktuell** sein müssen. Damit ist gemeint, dass eine dokumentierte Anforderung die aktuellen Gegebenheiten der zu entwickelnden Software und des zugehörigen Kontexts wiedergibt. Unter aktuelle Gegebenheiten können beispielsweise die gegenwärtigen Stakeholderwünsche, aktuell gesetzliche Verordnungen oder programmübergreifende Schnittstellen fallen. Bei Änderung einer der bestimmenden Größen müssen alle dadurch betroffenen Anforderungen angepasst werden.

Anforderungen sollten nach Pohl (Pohl 2008) auch **atomar** sein. Hierzu müssen sie einen Sachverhalt isoliert beschreiben. Gibt jedoch eine Anforderung mehr als nur eine Softwarefunktionalität oder -eigenschaft wieder, so ist diese Anforderung nicht atomar. Ein Beispiel für eine solche Anforderung könnte lauten: „Ein Systembenutzer

muss sich zur Benutzung der Anwendung anmelden können. Er muss dann die Möglichkeit haben Nachrichten zu lesen, zu verfassen, zu verschicken und auszudrucken. Der Ausdruck muss im A4- und A3-Format möglich sein.“ Solche nicht atomaren Anforderungen müssen in mehrere Anforderungen aufgeteilt werden. Anforderungen, die nicht in der natürlichen Sprache sondern durch Nutzung anderer Darstellungsformen beschrieben sind, müssen ebenfalls atomar sein.

In diesem Abschnitt wurden die Qualitätskriterien von Anforderungen vorgestellt. Um zeitraubende Nachbesserungen zu vermeiden, sollten diese Kriterien nicht nur zur nachträglichen Überprüfung dienen, sondern bereits bei der Anforderungsdokumentation genutzt werden. Sie sind weniger als eine Messlatte zu verstehen, sondern vielmehr als ein Wegweiser. Darum müssen diese Eigenschaften dem Requirements Engineer bereits vor der Anforderungsdokumentation ausreichend bekannt sein.

7 Grundlagen zu Anforderungsmustern

Nachdem die Grundlagen des REs hinreichend beschrieben wurden, leitet dieser letzte Abschnitt den Ansatz der Anwendung von Mustern im RE ein. Es wird ausführlich erläutert, was unter Anforderungsmuster zu verstehen ist. Ebenso wird der Sammelort der Anforderungsmuster – der sogenannte Anforderungsmusterkatalog – vorgestellt. Doch zuvor werden einige RE-Ansätze vorgestellt, die bereits eine große Akzeptanz in der Praxis erlangt haben. Es sind Methoden und Techniken, die in einem Anforderungsmuster ebenfalls enthalten sind. Konkret handelt es sich dabei um die Wiederverwendung von Anforderungen, der Nutzung von Satzmustern, der Verwendung einer Normsprache und dem Einsatz von Anforderungsschablonen. Es existieren neben den Anforderungsmustern weitere Musterarten, die innerhalb des Software Engineerings angewandt werden können. Zu den bekanntesten zählen die Design Patterns und die Analysemuster. Die Abgrenzung von Anforderungsmustern zu diesen Musterarten stellt den Abschluss dieses Abschnitts dar. Dazu erfolgt die Vorstellung der anderen Muster.

7.1 Grundlegende Ansätze

Anforderungsmuster beinhalten mehrere bereits in der Praxis etablierte Hilfsmittel für das RE. Sie ermöglichen es, Anforderungen auf eine einfache und systematische Art wiederzuverwenden. Für die Formulierung der Anforderungsbeschreibung können Satzmuster in den Anforderungsmustern eingesetzt werden. Sie unterstützen auch die Methode der Normsprache, indem Attributwerte vordefiniert sowie Begriffe über alle Muster vereinheitlicht werden. Durch die vordefinierte Struktur der Anforderungsmuster decken sie auch die Methode der Anforderungsschablonen ab. Diese Methoden, die in dem Ansatz der Anwendung von Anforderungsmustern vereint werden, werden im Folgenden genauer beschrieben.

Wiederverwendung von Anforderungen

In den letzten Jahren hat die projektübergreifende Wiederverwendung von Anforderungsformulierungen immer mehr an Bedeutung zugenommen (Partsch 2010). In der Studie von Neill/Laplante (2003) hat sich herausgestellt, dass in etwa 20% der Entwicklungsprojekte Anforderungsdokumente von Altsystemen für die Ermittlung von Anforderungen genutzt werden. Andere Autoren werten diese Anforderungsquelle sogar als die bedeutsamste (Hood 2005). Der hohe Anteil von Anforderungen mit dem Ursprung aus dieser Quelle ist wohl auch damit begründet, dass die Zugänglichkeit an diese Informationen leicht ist. Ein anderer, mindestens genauso wichtiger Grund ist, dass diese aufgespürten Anforderungen bereits umfassend und ausreichend formuliert sind (Hood 2005). Oft reichen leichte Anpassungen der Anforderungsbeschreibung aus, um diese wiederzuverwenden. So schreiben Rupp/SOPHISTen (Rupp and SOPHISTen 2009), dass durch die Wiederverwendung von Anforderungen der Aufwand für die Anforderungsdefinition reduziert werden kann. Jedoch nur, wenn die Wiederverwendung selbst mit weniger Aufwand verbunden ist als die Neuschaffung der Anforderung. Für eine aufwandsarme Wiederverwendung müssen die bereits vorhandenen Anforderungen gut wiederverwendbar sein. Doch wann ist eine Anforderung gut wiederverwendbar? Die Wiederverwendbarkeit von Anforderungen lässt sich leicht überprüfen. So existieren einige Faktoren, die die Wiederverwendbarkeit von Anforderungen beeinflussen. Je stärker diese Einflussfaktoren ausgeprägt sind, umso wiederverwendbarer ist eine Anforderung. Nach Rupp/SOPHISTen (Rupp and SOPHISTen 2009) sind es folgende Faktoren:

- Eine Anforderung ist umso besser wiederverwendbar, **je abstrakter und technologieunabhängiger die Anforderungsformulierung** ist. Sind in einer Anforderungsformulierung z.B. „Bankverbindung“ statt „Kontoinhaber, Kontonummer und BLZ“ oder „Tabellenkalkulationsprogramm“ statt „Microsoft Excel 2010“ vorzufinden, so ist in diesem Fall die Anforderungen einfacher wiederzuverwenden. Die Beinhaltung einer konkreten Technologieentscheidung ist ebenfalls eine zeitliche Begrenzung der Wiederverwendung einer Anforderung. Denn gerade in heutiger Zeit ändert sich die Technologie in einem rasanten Tempo.
- - Eine Anforderung ist umso besser wiederverwendbar, **je homogener die Produktpalette des Unternehmens ist**. Eine schmales Leistungssortiment hat zur Folge, dass viele vorhandene Anforderungen 1:1 oder leicht verändert in späteren Softwareentwicklungsprojekten wiederverwendet werden können. So ist die Wiederverwendung von Anforderungen z.B. für einen Dispositionssystemanbieter förderlicher als für einen Flughafenbetreiber, welcher ein breites Dienstleistungssortiment anbietet.
- Eine Anforderung ist umso besser wiederverwendbar, **je unabhängiger diese in einen Ablauf eingebunden ist**. Anforderungen ohne festgelegen Ablauf (z.B.: System muss die Möglichkeit bieten, den Vornamen und Nachnamen einzugeben) sind wiederverwendbarer als mit festgelegten Ablauf (z.B.: System muss die Möglichkeit bieten, erst den Vornamen und danach den Nachnamen einzugeben). Die

Einbindung von chronologischen Abläufen senkt somit die Wahrscheinlichkeit der Wiederverwendung.

- Eine Anforderung ist umso besser wiederverwendbar, **je konstanter der von der Anforderung betroffene Anwendungsbereich** ist. Die Konstanz des betrachteten Bereichs wirkt sich ebenfalls auf die zeitliche Begrenzung der Wiederverwendung aus. Anforderungen, die dynamische Anwendungsbereiche betreffen, können bereits nach kurzer Zeit durch Veränderungen der Betriebsabläufe für die Wiederverwendung unbrauchbar sein.

Eine durch die Wiederverwendung erreichte Aufwandsreduzierung hat den positiven Nebeneffekt, dass die gesamte Entwicklungszeit verringert wird. Neben den aus der Aufwandsreduktion entstehenden monetären Vorteilen bewirkt die Wiederverwendung von Anforderungen eine Steigerung der Qualität der Anforderungserhebung (Ebert 2008; Pohl and Rupp 2009). Denn der Wiederverwendungsansatz zielt darauf ab, bereits in der Vergangenheit als erfolgreich entpuppte Lösungen – in diesem Fall als Anforderungsbeschreibungen – wiederzuverwenden. Es handelt sich also hierbei um Anforderungsbeschreibungen, die fehlerfrei waren und somit zu dem gewünschten Ergebnis führten. Sie waren somit prüfbar, eindeutig, verständlich usw. Mit anderen Worten wird durch die Wiederverwendung von Anforderungen versucht, Erfahrungen und Wissen aus der Vergangenheit zu nutzen. Durch Gebrauch bewährter Anforderungen wird das Risiko von Fehlern in der Formulierung von Anforderungen stark vermindert.

Für die Wiederverwendung von Anforderungen ist eine sorgfältige Nachvollziehbarkeit sehr hilfreich (Pohl 2008; Grande 2011). Sind zusammengehörende Anforderungen, Testfälle und Lösungen miteinander verbunden, so können diese „Pakete“ in späteren Entwicklungsprojekten leicht wiederverwendet werden. Eine systematische Verwaltung von Altanforderungen mit Hilfe eines RE-Tools erlaubt eine effizientere Suche nach geeigneten Anforderungen (Hood 2005).

Die Wiederverwendung von Anforderungen erfolgt jedoch häufig intuitiv und ohne feste Regeln (Rupp and SOPHISTen 2009). Das Wiederfinden einer Anforderung in den bereits existierenden Anforderungsdokumenten der Altsysteme und aktuellen Systeme ist dabei sehr schwierig (Knethen 2002). Das Wissen, welche Anforderung in welchem Anforderungsdokument bereits vorliegt ist nur implizit bei den betroffenen Personen vorhanden (Pohl 2008). Des Weiteren besteht bei einer Absicht der späteren Wiederverwendungsmöglichkeit die Gefahr, dass Anforderungen mehrdeutig und unverständlich z.B. durch eine hohe Abstraktion formuliert werden (Rupp and SOPHISTen 2009). Abhilfe für diese Probleme sollen regelgeleitete Wiederverwendungskonzepte wie das IVENA XT schaffen. Dieses wurde von der SOPHIST GmbH zur systematischen Wiederverwendung von Anforderungen entwickelt. IVENA ist die Abkürzung für „Integriertes Vorgehen zur Ermittlung nicht-funktionaler Anforderungen“. Das Vorgehen wurde weiterentwickelt und eignet sich seitdem auch für alle anderen Arten von Anforderungen. Aus diesem Grund wurde das XT (extended) der Bezeichnung angehängt.

Eines der Merkmale dieses Vorgehens ist die Generalisierung und Spezialisierung von Anforderungen. Die innovative Besonderheit stellt die Verwendung einer Anforderungsbibliothek dar. Darin werden Anforderungen aus vorangegangenen Projekten gesammelt und so jedem Requirements Engineer zur Verfügung gestellt. Die Anforderungsbibliothek dient als Ideengeber und Nachschlagewerk für die Anforderungsermittlung und als Formulierungshilfe für die Dokumentation von Anforderungen. Die folgende Abbildung stellt das Wiederverwendungsvorgehen mit IVENA XT dar:

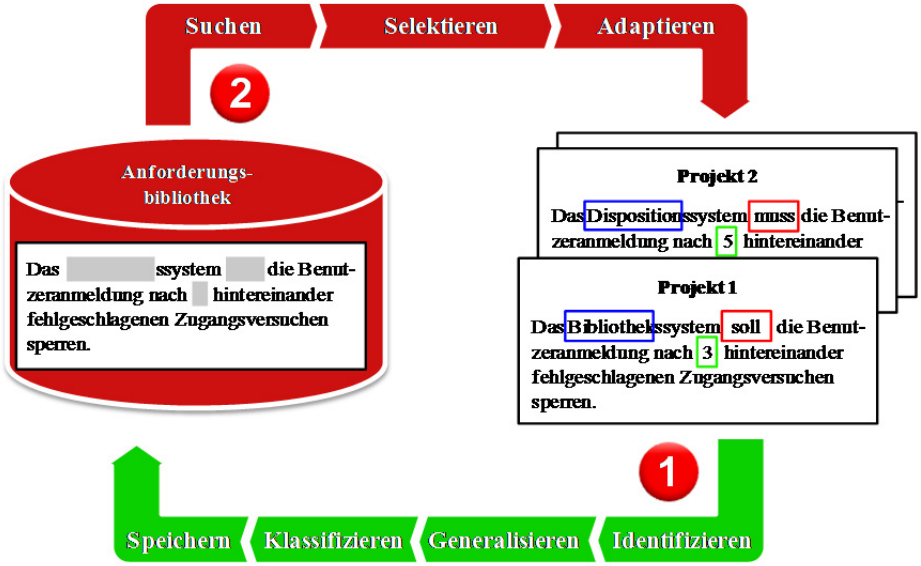

Abbildung 10: Ablauf der Wiederverwendung von Anforderungen mit dem Wiederverwendungskonzept IVENA XT (in Anlehnung an Rupp and SOPHISTen 2009)

Anforderungen sind meist sehr speziell auf die projektspezifische Problemstellung zugeschnitten. Die für eine Wiederverwendung geeigneten Anforderungen werden daher vor der Einbringung in die Anforderungsbibliothek generalisiert. Dazu werden alle projektspezifischen Werte durch neutrale Platzhalter ersetzt. Durch die Generalisierung wird eine allgemeinere, den Projektgegebenheiten unabhängige Formulierung erreicht. Dies ermöglicht eine vielfache Wiederverwendung dieser Anforderungen. Anschließend werden diese neutralisierten Anforderungen in der Anforderungsbibliothek gespeichert. Die Ablegung der Anforderungen in der Anforderungsbibliothek sollte in einer sinnvollen Struktur erfolgen. Das erleichtert das Auffinden einer Anforderung. Bei jedem neuen Projekt wird geprüft, welche der generalisierten Anforderungen aus der Anforderungsbibliothek für das Projekt relevant sind. Bei der Verwendung solcher generalisierter Anforderungen besteht wiederum die Notwendigkeit der Spezialisierung. Dabei müssen den selektierten Anforderungen projektspezifische Werte hinzugefügt werden. Während die Neutralisierung bei jeder Anforderung nur

einmalig bei dem Aufnehmen in die Anforderungsbibliothek erfolgt, ist die Spezialisierung bei jeder Verwendung erforderlich.

Die Anforderungsbibliothek sollte ständig erweitert und verbessert werden. Nach dem Abschluss eines Entwicklungsprojektes müssen die darin erarbeiteten Anforderungsdokumente nach Anforderungen durchsucht werden, die Wiederverwendungspotenzial aufweisen und in der Anforderungsbibliothek bisher nicht enthalten sind. Diese werden dann ebenfalls generalisiert und in die Anforderungsbibliothek eingebunden. Auf diese Weise wird die Anforderungsbibliothek immer vollständiger, wie in der folgenden Abbildung graphisch dargestellt ist:

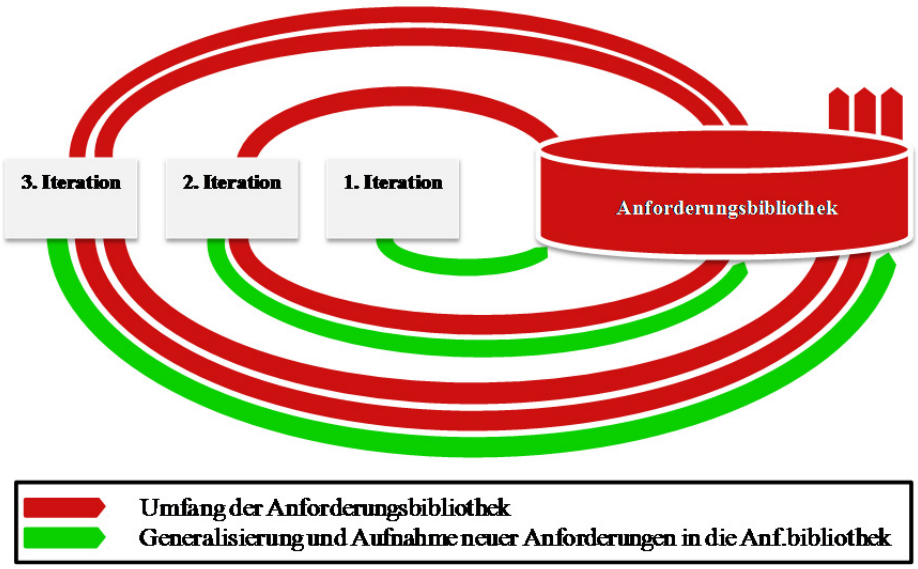

Abbildung 11: Inkrementelle Erweiterung der Anforderungsbibliothek (in Anlehnung an (in Anlehnung an Rupp and SOPHISTen 2009)

Unbestritten führt die Einführung eines solchen Wiederverwendungskonzeptes zunächst einmal zu einem zusätzlichen Aufwand. Vor allem der Aufbau und die Pflege der Anforderungsbibliothek sind sehr zeitaufwendig (Götz 1997). Auf der anderen Seite reduziert sich dadurch der Aufwand zur Anforderungsdefinition bei allen nachfolgenden Entwicklungsprojekten. Zu Beginn wird sich der Nutzen jedoch nicht deutlich zeigen. Aber bereits nach einigen Entwicklungsprojekten wird dieser zusätzliche Aufwand wieder kompensiert (Rupp and SOPHISTen 2009). Und je länger dieses Wiederverwendungskonzept angewandt wird, umso vollständiger wird die Anforderungsbibliothek. Hiermit wird einen stetig zunehmenden Gebrauchswert sichergestellt.

Neben der Wiederverwendung von Anforderungsbeschreibungen gibt es alternative Instrumente für die Verbesserung der Qualität einzelner Anforderungen. Im Folgen-

den wird aufgezeigt, inwieweit die natürliche Sprache teilformalisiert werden kann, um dadurch die Qualität der Anforderungsformulierung zu verbessern.

Satzschablone

Für einen Erfolg des Entwicklungsprojektes müssen Anforderungen eindeutig und verständlich formuliert werden. Eine klare Strukturierung des Anforderungstextes erhöht selbst bei sehr heterogenen Anforderungen die Verständlichkeit (Ebert 2012). Aus diesem Grund schlagen diverse RE-Forscher die Verwendung einfacher Vorlagen für den Satzbau des Anforderungstextes einer Anforderungsbeschreibung vor (Hood 2005; Pohl 2008; Balzert 2009; Pohl and Rupp 2009; Rupp and SOPHISTen 2009; Partsch 2010; Grande 2011; Ebert 2012). In der RE-Literatur werden diese Formulierungsbaupläne unterschiedlich bezeichnet – Satzmuster, Satzbaumuster, Satzschablone, Textschablone, syntaktisches Anforderungsmuster. Sie basieren auf bisherigen Erfahrungen. Bei Anwendung dieses Werkzeugs müssen die Anforderungen zwar bei jedem Projekt immer wieder neu definiert werden, jedoch existieren hierfür vordefinierte Satzbauelemente. Diese führen den Anforderungsautor bei der Formulierung. Abweichungen sind nicht zulässig, da die Satzschablonen so konstruiert sind, dass die vorbestimmte Verknüpfung der Satzbauelemente einen korrekten Satz ergibt (Abbildung 12). Satzschablonen zielen weniger darauf ab, den Inhalt von Anforderungen zu revidieren. Vielmehr steht bei diesem RE-Werkzeug die syntaktische Struktur einer einzelnen Anforderungsformulierung im Fokus. Allerdings ist es manchmal förderlich, auch Begriffe vorzudefinieren (Hood 2005). So sollten z.B. die für die Bestimmung der rechtlichen Verbindlichkeit der Anforderung verfügbaren Modalverben vordefiniert werden, um diese für alle Anforderungen innerhalb eines Projektes sowie auch projektübergreifend zu vereinheitlichen.

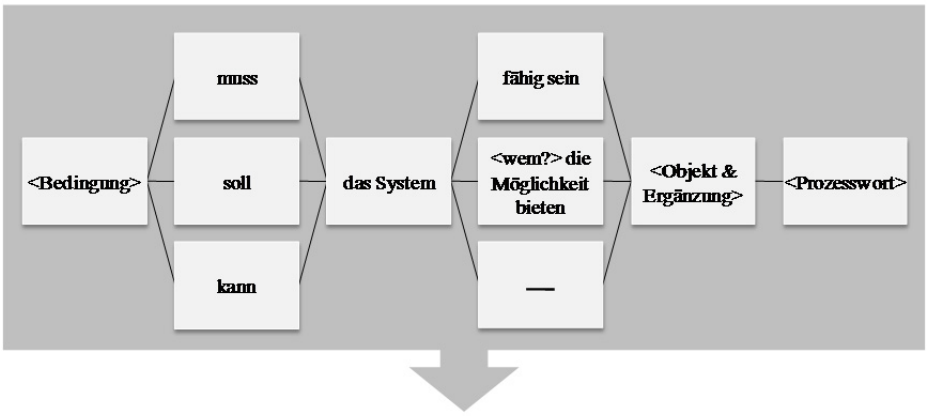

Abbildung 12: Beispielhafte Satzschablone für die Formulierung von Anforderungen (in Anlehnung an Pohl and Rupp 2009)

Durch die Verwendung solcher Satzschablonen resultieren Anforderungstexte, die hinsichtlich des Satzbaus standardisiert sind. Dies wiederrum bewirkt eine Erhöhung der Lesbarkeit und der Verständlichkeit bei gleichzeitiger Reduzierung der Komplexität der Anforderungsformulierungen. Die Gefahr der syntaktischen Mehrdeutigkeit wird vermieden. Empfehlenswert ist es, mehrere den unterschiedlichen Anforderungsarten (funktionale, nicht-funktionale usw.) angepasste Satzmuster einzusetzen (Ebert 2008). Solche Satzschablonen haben auch das Ziel, den Prozess der Formulierung von Anforderungstexten zu vereinfachen. Besonders den unerfahrenen Requirements Engineers kommt das zugute. Ein weiterer Vorteil dieser Methode ist die einfache Anwendung. Es ist weder ein zeitintensives erlernen als auch eine aufwendige Vorbereitung notwendig (Pohl and Rupp 2009).

Die Nutzung von Satzschablonen ist folglich eine gute Methode, um Anforderungen zu beschreiben, bei denen die korrekte Syntax weitestgehend automatisch sichergestellt ist. Eine korrekte Syntax ist aber noch lange keine Gewähr für fehlerfreie Anforderungen. Denn auch Fehler in der Semantik können zu einer Mehrdeutigkeit führen.

Normsprache

Durch die Nutzung einer Normsprache kann die syntaktische und semantische Mehrdeutigkeit (siehe Qualitätskriterium Eindeutigkeit im Abschnitt 6) bei natürlichsprachlichen Anforderungen vermieden werden. Die Verwendung einer Normsprache verhilft zu einer effektiveren Gestaltung des REs (Schienmann 2002). Durch eine Normsprache – häufig auch unter der Bezeichnung Metasprache anzutreffen – wird ebenfalls eine Einschränkung in der Formulierung einer natürlichsprachlichen Anforderung vorgenommen. Jedoch wird bei dieser Methode neben der zu verwendenden Grammatik zusätzlich die Semantik festgelegt (Lehmann 1998; Schienmann 2002). Daher können Normsprachen als Erweiterung von Satzschablonen verstanden werden (Pohl 2008). Für die Festlegung der Semantik wird die Bedeutung zulässiger Wörter innerhalb einer Domäne (z.B. Verkehrswesen, Medizintechnik) a priori definiert und festgelegt. Somit ergibt sich die Bedeutung von Begriffen nicht erst durch den Kontextbezug der Aussage. Pohl (2008) definiert eine Normsprache wie folgt:

> „Eine Normsprache besitzt eine in Bezug auf eine spezifische Domäne eingeschränkte natürlichsprachliche Grammatik (Syntax) und definiert eine Menge von Begriffen (Semantik), die unter Verwendung der vorgegebenen Grammatik zur Konstruktion von Aussagen über die Domäne verwendet werden können."

Demzufolge wird mit Hilfe einer Normsprache eine definierte Fachsprache zur Formulierung von Anforderungen über eine zugrunde liegende Domäne bereitgestellt. Die Verwendung einer solchen Sprache für die Formulierung von Anforderungen bietet einige Vorteile. Bei dieser Methode werden die Anforderungen trotz starker Formalisierung in der natürlichen Sprache beschrieben. Das hat den Vorteil, dass mit Hilfe einer Normsprache erstellen Anforderungen für die Anwender weiterhin einfach

zu verstehen sind (Lehmann 1998). Denn wie bereits in Abschnitt 4.2 erwähnt präferieren Anwender die natürliche Sprache für die Beschreibung von Anforderungen. Die Einschränkungen empfinden sie nicht als lästig, da lediglich der Gebrauch gewohnter Sprachkonstrukte reglementiert wird (Pohl 2008). Die Verwendung einer Normsprache führt zu einer höheren Eindeutigkeit von Anforderungen, da sie auf Basis einer durchdachten Vorgabe der Syntax und eines festgelegten Wortschatzes formuliert werden (Hood 2005). Durch die normierte Terminologie wird auch die Kommunikation zwischen Projektbeteiligten verbessert. Denn es werden Mehrdeutigkeiten und Missverständnisse bei der Benennung von Situationen und Gegenständen vermieden. Die Entwicklung von rechnergestützten Informationssystemen ist eine potenzielles Einsatzgebiet einer normierten Sprache, da hierbei eine intensive Kommunikation zwischen der Anwender- und Entwicklerseite erforderlich ist. Jedoch unterscheiden sich stark die bevorzugt eingesetzten Sprachen beider Seiten (Lehmann 1998). Die Lücke zwischen der gewachsenen Sprache einzelner Beteiligter und einer gemeinsamen Sprache wird mit der Normsprache geschlossen. Eine Normsprache hat jedoch den bedeutenden Nachteil, dass sie sich immer auf einen bestimmten Anwendungsbereich fixiert. Denn Begriffe unterscheiden sich häufig von Domäne zu Domäne. Daher ist es nicht möglich, eine domänenübergreifende Normsprache zu erstellen. Ein weiterer Nachteil ist, dass die Anwendung einer Normsprache eine umfassende Schulung voraussetzt (Pohl 2008). Denn die Entwicklung und die akribische Anwendung einer Normsprache sind nicht ganz einfach. Für die Entwicklung schlägt Schienmann (2002) folgendes 4-stufiges Vorgehen vor:

Abbildung 13: 4-stufiges Vorgehen zur Erstellung einer Normsprache (in Anlehnung an Schienmann 2002)

Zur Entwicklung einer Normsprache müssen zuerst sachbezogene, umgangssprachliche Aussagen gesammelt werden. Aus diesen Aussagen werden anschließend Fachbegriffe extrahiert, gemeinsam definiert und in einer für alle Betreffenden zugänglichen Sammelstelle festgehalten. In diesem Schritt werden auch Regeln geschaffen, die den Gebrauch der Fachbegriffe klarstellen. Nachdem eindeutig geklärt ist, welche Gegenstände durch einen Fachbegriff in der betrachteten Domäne bezeichnet werden können und wie die Fachbegriffe zu gebrauchen sind, geht es um die Erzeugung von

standardisierten Aussagen mit Hilfe der Normsprache. Hierbei können zusätzlich Satzschablonen eingesetzt werden, um die Syntax vorzugeben. Die standardisierten Aussagen werden anschließend bei der Formulierung von Anforderungen eingesetzt. Die ersten drei Stufen bilden den Kern der Normsprache (Pohl 2008). Bei Bedarf können in einer vierten Stufe die Aussagen klassifiziert werden. Hiermit wird der Übergang in eine Modellierungssprache vorbereitet. Die Klassifikation sollte jedoch nur vorgenommen werden, wenn eine objektorientierte Beschreibung und Dokumentation der Anforderungen beabsichtigt ist. Die Entwicklung einer Normsprache muss als ein inkrementeller Prozess verstanden werden (Schienmann 2002). Die Fachbegriffe und Aussagen werden dabei nach und nach ermittelt, präzisiert und stabilisiert.

Für weitere Details bezüglich der Normsprache kann auf Pohl (2008) und auf Schienmann (2002) verwiesen werden. Sowohl die Satzschablone als auch die Normsprache sind Hilfsmethoden für die Formulierung einer natürlichsprachlichen Anforderung. Es existieren jedoch weitere Methoden, die sich dem strukturellen Aufbau einer gesamten Anforderungsbeschreibung widmen. Eine davon ist die Verwendung von Anforderungsschablonen.

Anforderungsschablone

Für die Qualitätsverbesserung der Beschreibung und Dokumentation einzelner Anforderungen ist es hilfreich, eine Anforderungsschablone zu benutzen (Schienmann 2002; Hood 2005; Robertson and Robertson 2006; Pohl 2008; Partsch 2010). Eine Anforderungsschablone – häufig auch als Attributierungsschema, Anforderungs-Template oder Snow Card bezeichnet – definiert eine vorgegebene Struktur einer vollständigen Anforderungsbeschreibung. Implizit legt eine solche Schablone dadurch fest, welche Eigenschaften zu jeder Anforderung dokumentiert werden müssen. Eine beispielhafte Anforderungsschablone, die an die etablierte und von Robertson/Robertson (2006) eingeführte Snow Card angelehnt ist, zeigt die Abbildung 14.

Die Schablonen dienen als Strukturvorgabe für die Beschreibung einzelner Anforderungen. Somit sind sie ein einfaches Hilfsinstrument für die Anforderungsspezifizierung und -dokumentation (Pohl 2008). Doch vor dem Einsatz solcher Schablonen gilt es zunächst die Struktur und somit auch die benötigten Informationsarten zielabhängig festzulegen. Hierfür sollte ausreichend Zeit und Energie aufgebracht werden, damit diese bereits vor dem breiten Einsatz optimal sind. Es ist ratsam, Anforderungsschablonen bei Bedarf an die projektspezifischen bzw. unternehmensspezifische Gegebenheiten temporär anzupassen (Schienmann 2002; Ebert 2012). Die Grundstruktur sollte aber projektübergreifend gleich bleiben. Weiterhin empfiehlt Pohl (2008) die Eigenschaftswerte so weit wie möglich vorzugeben. Als Beispiel kann hier die Verbindlichkeit einer Anforderung genannt werden, bei der nur die im deutschen Sprachraum etablierten Verben „Soll“, „Muss“ und „Kann“ zur Verfügung stehen sollten. Sind die Schablonen vor den ersten Ermittlungsbemühungen erstellt, so sollten sie im weiteren Verlauf des Entwicklungsprojektes nicht mehr umgestaltet werden. Wurde im Projektverlauf eine neue Anforderung ermittelt, so ist es nicht zwingend erforder-

lich, in der Anforderungsschablone sofort alle Felder auszufüllen. Empfehlenswert ist es, die Anforderungsschablone schrittweise auszufüllen (Robertson and Robertson 2006). Eine Differenzierung der Attribute in Basisattribute und Nicht-Basisattribute ist dafür zweckmäßig (Pohl 2008).

Anforderung #	*Identifiziert die Anforderung eindeutig*
Anforderungstyp	*Anforderungstyp aus vordefinierten Kategorien (funktionale Anforderung, nicht-funktionale Anforderung usw.)*
Event/use case #	*Liste von Anwendungsfällen, die die Anforderung benötigen - Ermöglicht Gruppierung zusammengehörender Anforderungen*
Beschreibung	*Beschreibt die Anforderung in einem Satz*
Auslöser	*Begründung für die Anforderung*
Quelle	*Stakeholder, der die Anforderung eingebracht hat*
Abnahmekriterium	*Messbare Vorgabe, um Erfüllung der Anforderung entscheiden zu können*
Kundenzufriedenheit	*Subjektive Zufriedenheit des Stakeholders bei Realisierung der Anforderung*
Kundenunzufriedenheit	*Subjektive Unzufriedenheit des Stakeholders bei Realisierung der Anforderung*
Abhängigkeiten	*Abhängigkeiten zu anderen Anforderungen*
Konflikte	*Nicht umsetzbare Anforderungen bei Umsetzung dieser Anforderung*
Priorität	*Priorität dieser Anforderung*
Weitere Referenzen	*Verweise auf Dokumente, die diese Anforderung näher beschreiben*
Historie	*Status der Anforderung und Datum der letzten Statusänderung*

Abbildung 14: Schablone für Aufbau einer Anforderung (in Anlehnung an Robertson and Robertson 2006)

Pohl (2008)(2008, 267) und Schienmann (2002) zählen zu den Vorteilen dieser Methode auf, dass damit eine strukturierte Dokumentation von Anforderungen erheblich erleichtert wird. Des Weiteren bringen Anforderungsschablonen den Vorteil, dass dokumentationsrelevante aber bisher nicht dokumentierte Eigenschaften einer Anforderung sehr leicht zu erkennen sind (Pohl 2008; Partsch 2010; Ebert 2012). In diesem Fall ist die entsprechende Zeile der Anforderungsbeschreibung leer. Durch dieses einfache Erkennen von Lücken kann die Qualität der Anforderungsbeschreibungen stark gesteigert werden. Gleichzeitig wird dadurch die Durchführung der für das Projekt als essentiell erachteten RE-Tätigkeiten sichergestellt (Hood 2005). Müssen z.B. die Abhängigkeiten einer Anforderung zu anderen Anforderungen dokumentiert werden, so ist es dafür nötig, sich mit den Abhängigkeiten der Anforderungen zu befassen. Ebenfalls gestaltet sich der Vergleich von Anforderungen und der Zugriff auf dokumentierte Anforderungseigenschaften leichter, da gleichartige Informationen immer an der gleichen Stelle hinterlegt sind (Pohl 2008). Anforderungsschablonen sind somit auch für viele Kontrolltätigkeiten innerhalb des REs eine Hilfe.

In diesem Abschnitt wurde expliziert, dass mit Hilfe der Wiederverwendung Anforderungen effektiver und effizienter erhoben werden. Jedoch eignet sich dieser Ansatz für ein Unternehmen besser als für ein anderes Unternehmen. Entscheidend dafür ist, wie wiederverwendbar die Anforderungen eines Unternehmens sind. Ist das RE

eines Unternehmens dafür prädestiniert, so sollten Anforderungen wiederverwendet werden. Denn dadurch resultiert eine Aufwandsreduzierung und Qualitätssteigerung der Anforderungsdefinition. Folglich werden das Floprisiko sowie die gesamte Entwicklungszeit gesenkt. Die Wiederverwendung sollte aber nicht intuitiv erfolgen, da dabei zahlreiche Nachteile sowie Gefahren bestehen. Stattdessen ist die Anwendung eines Wiederverwendungskonzeptes wie dem INVENA XT ratsam. Die Anforderungsbibliothek stellt dabei Anforderungen in adäquater Weise zur Verfügung, die das Potenzial der Wiederverwendung aufweisen. Bei der Verwendung werden diese Anforderungen wiederum projektspezifisch spezialisiert. Die Anforderungsbibliothek wird inkrementell nach jedem Entwicklungsprojekt vervollständigt. Des Weiteren wurden in diesem Abschnitt Methoden aufgezeigt, die die Formulierung von Anforderungen unterstützen. Diese Methoden geben konkret vor, wie die Beschreibungstexte von Anforderungen aufzubauen sind und welche Begriffe darin zu verwenden sind. Durch die daraus resultierende Standardisierung von Anforderungsbeschreibungen wird das Dokumentieren von Anforderungen erleichtert und die Verständlichkeit von sowie der Umgang mit Anforderungen verbessert (Schienmann 2002). Satzschablone und Normsprache sind Techniken, die in erster Linie darauf abzielen, Mehrdeutigkeiten, Vagheiten sowie Unvollständigkeiten bei der Formulierung von natürlichsprachlichen Anforderungen zu verhindern. Satzschablonen stellen dafür die syntaktische Eindeutigkeit bei Anforderungstexten sicher. Hierfür wird der Anforderungsautor bereits bei der Formulierung des Anforderungstextes unterstützt, indem er dabei eine Schablone benutzt, die eine konkrete syntaktische Struktur vorgibt. Eine Weiterentwicklung solcher Satzschablonen stellt die Normsprache dar. Denn hierbei wird neben der Syntax auch die Semantik in Bezug auf eine spezifische Domäne festgelegt. Grund dafür ist, dass die Bedeutung und der Gebrauch von Begriffen oft nicht für alle Beteiligten unmittelbar verständlich sind. Mit Hilfe beider Methode können Anforderungen in äußerst hoher Qualität geschrieben werden. Auf der anderen Seite kann es jedoch manchmal schwierig sein, eine Anforderungen unter der Beachtung solcher Formulierungsbeschränkungen verständlich zu beschreiben. Denn in manchen Fällen können solche Formulierungsvorgaben dazu führen, dass eine Anforderungsbeschreibung schwerer zu lesen ist als wenn diese ohne solche Einschränkungen formuliert worden wäre (Balzert 2009). Die letzte in diesem Abschnitt vorgestellte Methode sind die Anforderungsschablonen. Mit der Anwendung von Anforderungsschablonen wird eine schablonenbasierte Dokumentation der Anforderungen ermöglicht. Anforderungsschablonen geben die Struktur einer Anforderungsbeschreibung vor. Dazu sind zu beschreibende Attribute sowie fakultativ auch verfügbare Attributwerte definiert. Sie machen somit die strukturierte Dokumentation von Anforderungen einfacher und effektiver. Weitere Vorteile entstehen dadurch, dass gleiche Informationen sich bei jeder Anforderungsbeschreibung an gleicher Stelle befinden. Der Vergleich von Anforderungen wie auch der selektive Zugriff auf dokumentierte Attribute wird erheblich erleichtert.

7.2 Anforderungsmuster im Detail

Im Folgenden werden die in dieser Arbeit im Mittelpunkt stehenden Anforderungsmuster vorgestellt. Es wird aufgezeigt, was in dieser Arbeit unter einem Anforderungsmuster sowie unter einem Anforderungsmusterkatalog zu verstehen ist. Zunächst folgen einige Sätze zu der Entwicklung dieser Musterart sowie zu der Grundidee dieser Muster.

Die Softwareentwicklung ist eine Disziplin, bei der die Phasen des Entwicklungsprozesses sich projektübergreifend in ähnlicher Form ständig wiederholen (Wahono 2002). Bei jeder Entwicklung gilt es, das Entwicklungsprojekt zu planen, die relevanten Stakeholder zu identifizieren, die Anforderungen zu definieren, das System zu entwickeln und zu implementieren sowie es zum Schluss abzunehmen. Es sind immer wieder dieselben Probleme zu lösen. Lediglich das Umfeld variiert dabei. Diese regelmäßigen Wiederholungen des Requirements Engineerings macht sich das Konzept der Anforderungsmuster zu Nutze (Wahono 2002).

Ein Muster (engl. Pattern) ist eine wiederverwendbare Grundstruktur einer Lösung eines speziellen Problems. Ein Muster ist somit keine vorgefertigte Lösung, die eins zu eins eingesetzt werden kann, sondern nur ein generisches Lösungsschema für einer Gruppe ähnlicher Probleme (Buschmann 1998). Der Einsatz von Mustern zum Entwerfen komplexer Systeme erfreut sich in anderen Disziplinen bereits großer Beliebtheit (Konrad and Cheng 2002). Inspiriert durch das erfolgreiche Buch „Design Patterns – Elements of Reusable Object-Oriented Software“ im Jahre 1994 von der Gang of Four hat sich auch in der Softwareentwicklung die Anwendung der sogenannten Entwurfsmustern für die objektorientierte Softwareentwicklung stark etabliert. Die Verwendung von Mustern zur Anforderungsdefinition ist hingegen eine relativ neue Entwicklung. Dieser Ansatz der Musternutzung ist somit noch nicht weit verbreitet (Pantoquilho 2003; Ketabchi, Karimi Sani et al. 2011).

Anforderungsmuster

Ein Anforderungsmuster[7] ist ein wiederverwendbares Schema für die Definition ähnlicher Anforderungen. Rupp/Sophist Group (2002) charakterisieren ein Anforderungsmuster wie folgt:

> „Ein Anforderungsmuster ist ein Lösungskonzept zu einer speziellen Problemstellung der Anforderungsanalyse. […] Die Lösung bildet eine unter synergetischen Gesichtspunkten abgewogene Kombination aus verschiedenen Komponenten (z.B. natürlichsprachliche Anforderungen, Analyse, Abnahmekriterien).“

[7] Im englischsprachigen Raum wird der Begriff „Requirements Pattern“ für Anforderungsmuster verwendet. Der Begriff „Software Requirements Pattern“ bezieht sich demnach auf Anforderungsmuster in der Domäne des Software Engineerings und wird als SRP abgekürzt. Diese Arbeit beschränkt sich auf den deutschen Ausdruck. Dabei wird auf die Differenzierung zwischen Anforderungsmuster und Softwareanforderungsmuster verzichtet.

Grob beschrieben sind Anforderungsmuster eine Sammlung von Wissen und Erfahrungen, die in aktuellen Projekten durch Adaption genutzt werden können (Wahono 2002). Ihre Anwendung ist dem Software-Entwicklungsprozess der Phase zu zuordnen, in der das Anforderungsdokument erstellt wird (Rupp and Sophist Group 2002). Soweit existiert in der Wissenschaft Konsens bezüglich Anforderungsmuster. Geht man aber ins Detail, so wird ersichtlich, dass in der Fachliteratur unter Anforderungsmustern bzw. Requirements Patterns nicht einheitlich dasselbe verstanden wird. Stattdessen haben sich abweichende Herangehensweisen von Anforderungsmustern parallel entwickelt (Franch, Palomares et al. 2010). Zum einen unterscheiden sie sich hinsichtlich des Einsatzgebiets. So existieren Ansätze, in denen Anforderungsmuster sich auf eingebettete Systeme, Businessanwendungen, Sicherheitsanforderungen oder auch auf das RE allgemein fokussieren. Zum anderen herrscht auch in der Notation und in der beabsichtigten Anwendung der Muster Uneinigkeit. Ein detaillierter Überblick der existierenden Ansätze ist in Franch et al. (2010) ersichtlich. Diese haben eine Literaturrecherche durchgeführt, um die unterschiedlichen Ansätze von Anforderungsmustern zu vergleichen.

Zugang sperren (Muster 12.1)

Details	Autor: Andrej Janzen Gruppe: Zugangskontrolle Art: Funktionale Anforderung
Anwendbarkeit	Nutze das Anforderungsmuster wenn das System den Zugang nach einer bestimmten Anzahl fehlgeschlagener Zugangsversuche deaktivieren soll.
Beziehungen	Muster 12.0 → Log-In & Log-Out

ID	FA–
Name	Zugang sperren
Beschreibung	Der Zugang ▼ nach hintereinander fehlgeschlagenen Zugangsversuchen gesperrt werden.
Autor/Quelle	
Nutzen	▼
Abhängigkeit	FA– Log-In & Log-Out

■ ■ ■

Abbildung 15: Beispielhaftes Anforderungsmuster

In dieser Arbeit wird der Ansatz von Withall (2008) angewandt. Lediglich bei dem Template wird von diesem Ansatz abgewichen. Für die Verwendung eines Anforderungsmusters hat Withall (2008) Templates den einzelnen Mustern beigefügt. Hierfür hat er vordefinierte Beschreibungssätze erstellt, welche er mit Platzhaltern versehen hat. Solche Templates eignen sich nur für die Inhaltsbeschreibung von Anforderungen. Werden diese Templates ohne Kombination mit anderen Dokumentationstechniken verwendet, so entstehen Anforderungen, die nur mangelhaft dokumentiert sind. Es fehlen weitere Kontextinformationen zu einer Anforderungen wie zum Beispiel die Quelle bzw. der Anforderungssteller, der Anforderungstyp, die Priorität, die Abhän-

gigkeiten zu anderen Anforderungen usw. Es sind Informationen, die das Steuern und Verwalten der Anforderungen ermöglichen (Pohl 2008; Rupp and SOPHISTen 2009). Aus diesem Grund werden die in den Anforderungsmustern beinhalteten Templates in dieser Arbeit nach dem Ansatz von Durán Toro et al. (1999) definiert, um damit auch die Dokumentation dieser relevanten Informationen zu ermöglichen. Die Anforderungsbeschreibung wird dabei strukturiert und übersichtlich in einer tabellarischen Form dokumentiert. Abbildung 15 zeigt anhand eines Beispiels, wie der Aufbau und der Inhalt eines Anforderungsmusters in dieser Arbeit definiert werden.

Ein Anforderungsmuster besteht aus mehreren Grundelementen. Withall (2008) hat für seine Anforderungsmuster zehn solcher Grundelemente definiert. In dieser Arbeit werden nicht alle zehn Grundelemente verwendete. Stattdessen erfolgt eine Beschränkung auf die folgenden Elemente:

- Name: Jedes Anforderungsmuster wird durch einen eindeutigen Namen benannt. Dieser ist so zu wählen, dass er das Anforderungsmuster prägnant bezeichnet.
- Details: Hier können Metainformationen[8] zu dem Anforderungsmuster erwähnt werden wie der Autor des Musters, die Art der beinhalteten Anforderung, das letzte Änderungsdatum usw.
- Anwendbarkeit: Die Anwendbarkeit beschreibt die Situation und die Voraussetzungen für den Einsatz des Anforderungsmusters.
- Beziehungen: Durch das Explizieren von Beziehungen zwischen den Anforderungsmustern entsteht ein Mustersystem[9]. Damit wird eine Navigation durch das Mustersystem ermöglicht.
- Template: Das Template ist der Teil eines Anforderungsmusters, welcher bei der Anwendung des Musters genutzt wird. Diese generische Anforderungsbeschreibung wird für die Beschreibung und Dokumentation der für das aktuelle Projekt benötigten Anforderung herangezogen. Ein Template stellt somit ein Startpunkt für das Schreiben einer Anforderung dieses Typs dar.

Diese Grundelemente können in zwei Gruppen differenziert werden. Der Name, die Details, die Anwendbarkeit und die Beziehungen beschreiben das Anforderungsmuster an sich. Sie geben somit Informationen über das jeweilige Anforderungsmuster. Das Template hingegen ist der Teil des Anforderungsmusters, welcher bei Anwendung in das Anforderungsdokument übernommen wird. Jedes Anforderungsmuster bezieht sich immer auf genau einen Sachverhalt (Rupp 2004; Withall 2008). Alle auf

[8] Metainformationen bzw. Metadaten unterscheiden sich zu gewöhnlichen Daten darin, dass sie für den Verwender bzw. Betrachter nur von mittelbarer Relevanz sind. Nicht Metadaten sind somit unmittelbare relevant für den Betrachter. Welche Daten Metadaten und welche gewöhnliche Daten sind ist abhängig von dem Standpunkt. In diesem Fall wurde der Anwender der Anforderungsmuster als Betrachter gewählt.

[9] Ein Mustersystem fasst die einzelnen Muster systematisch zusammen. Dabei werden die Zusammenhänge zwischen den einzelnen Mustern ausgearbeitet. Im Mustersystem wird also u.a. beschrieben, welche Muster voneinander abhängig sind und welche sich evtl. gegenseitig ergänzen.

einen Sachverhalt zutreffenden Alternativen oder zusätzlichen Anhänge sind in einem Anforderungsmuster zu erfassen. Somit werden alle möglichen Anforderungen zu einem Sachverhalt in einem Anforderungsmuster gruppiert. Sind zum Beispiel in einem Unternehmen sowohl eine Einzel- als auch Gruppeanmeldung gemäß den Sicherheitsrichtlinien zugelassen, so sind in dem Anforderungsmuster zu dem Sachverhalt „Benutzeranmeldung und -abmeldung“ beiden Optionen zu berücksichtigen. Das Anforderungsmuster muss in diesem Fall zwei Templates beinhalten, da für jede der beiden Optionen eine gesonderte Anforderung zu schreiben wäre. Nun kann projektspezifisch ausgewählt werden, ob die zu entwickelnde Software keine An- und Abmeldung, nur eine Einzelanmeldung und -abmeldung, nur eine Gruppenanmeldung und -abmeldung oder beide Optionen ermöglichen muss. Im ersten Fall wird das Anforderungsmuster gar nicht verwendet.

Die Struktur des in der Abbildung 15 dargestellten Templates ist nur ein Beispiel. Die dort aufgelisteten Templatefelder sind intuitiv festgelegt worden. Bevor jedoch die ersten Anforderungsmuster erstellt werden, muss die Struktur für das Template verbindlich festgelegt werden. In den Aufbau des Templates sollten die Attribute mit aufgenommen werden, die zu jeder Anforderung dokumentiert werden sollen. Unumgängliche Informationen sind dabei eine eindeutig Anforderungsnummer, der Name der Anforderung und die Beschreibung der Anforderung. Somit sollte die Struktur der festgelegten Templates immer mindestens diese drei Felder beinhalten. Ist zusätzlich beabsichtigt die Abhängigkeiten zwischen Anforderungen zu dokumentieren, so sollte dafür ebenfalls ein Feld in dem Template vorgesehen werden. Ist ein Mustersystem vorhanden, so ist es einfach, die Abhängigkeiten zwischen Anforderungen in den einzelnen Templates vorzudefinieren. Aber auch alle anderen Felder sollten in den Templates soweit wie möglich vordefiniert werden. Bei dem Vordefinieren ist jedoch unbedingt die Absicht der Wiederverwendung zu berücksichtigen, d.h. sie müssen ausreichend Variabilität aufweisen. Hierfür dürfen keine konkreten Werte in den Templates vordefiniert werden. Stattdessen müssen konkrete Werte generalisiert werden, indem Platzhalter an den Stellen konkreter Werte eingesetzt werden. Wo möglich, sollten statt Freitextfeldern Auswahlfelder als Platzhalter verwendet werden. Wird nun ein Anforderungsmuster angewandt, so wird das jeweilige Template in das Anforderungsdokument kopiert und für das konkrete Projekt verfeinert (Wahono 2002). Für diese Adaption müssen die Platzhalter projektspezifisch ausgefüllt werden.

Die Anforderungsmuster nach dem eben vorgestellten Verständnis schließen die Verwendung anderer RE-Methoden mit ein. So werden implizit Anforderungen wiederverwendet als auch Satzschablonen, eine Normsprache und eine Anforderungsschablone eingesetzt. Abbildung 16 zeigt wie die zuvor im Abschnitt 7.1 vorgestellten RE-Methoden durch das Template eines Anforderungsmusters implizit verwendet werden.

Durch die Nutzung der Templates von Anforderungsmustern werden die Anforderungen genau wie bei der Verwendung einer **Anforderungsschablone** in einer einheitlichen, tabellarischen Form dokumentiert. In beiden Fällen werden zuvor die Attribute projektübergreifend einheitlich festlegt, die beim Spezifizieren der Anforderung zu beschreiben bzw. zu vervollständigen sind. Somit sind die Grundgedanken

einer Anforderungsschablone ebenfalls in dem Template eines Anforderungsmusters enthalten. Der einzige fundamentale Unterschied ist, dass bei der Methode Anforderungsschablone nur eine Vorlage vorhanden ist, welche die Grundlage aller erstellten Anforderungen ist. Bei der Methode Anforderungsmuster existiert jedoch für jede typische Anforderung ein individuelles Template und somit eine angepasste Grundlage. Somit sind Templates gegenüber einer Anforderungsschablone spezifizierter. Sie weisen mehr Inhalt auf. Bei den Attributwerten wie z.B. Name, Beschreibung der Anforderung usw. sollten die Templates nur Inhalte vordefiniert sein, die bereits mit Erfolg praktiziert wurden. Die Verwendung von fehlerhaften Inhalten sollte verhindert werden. Denn ist in einem Template ein inhaltlicher Fehler enthalten, so sind dann auch alle damit erstellten Anforderungen fehlerhaft.

Abbildung 16: Template eines Anforderungsmusters vereint Verwendung mehrerer RE-Techniken

Auf der anderen Seite bewirken inhaltliche Vordefinierungen, welche fehlerfrei sind, dass die damit erstellten Anforderungen ebenfalls frei von Fehlern sind – jedenfalls innerhalb dem Teil der Anforderung, welcher aus der Grundlage übernommen wurde. Und genau hierbei ist das Prinzip der **Wiederverwendung von Anforderungen** erkennbar. Es werden bereits erstellte und erfolgreich verwendete Anforderungen – also Lösungen – wiederverwendet. Die Beschreibung der typischen Anforderung sollte in den Templates möglichst verständlich, simpel und eindeutig sein. Um das zu erreichen können für die Formulierung der jeweiligen Beschreibung **Satzschablonen** genutzt werden. In diesem Fall ist die Methode der Verwendung von Satzschablonen, die eine korrekte Syntax sicherstellen sollen, ebenfalls indirekt in einem Template enthalten. Des Weiteren ermöglichen Templates ebenfalls sinnvolle Eingriffe in die Semantik der damit erstellten Anforderungen. So kann z.B. ein Begriff für einen be-

stimmten Bedeutungsinhalt über alle Templates hindurch verwendet werden. Das würde dazu führen, dass in den Anforderungen eines Anforderungsdokumentes, die mit Hilfe der Anforderungsmuster erstellt wurden, keine Synonyme vorkommen. Jedenfalls nicht in dem Teil der Anforderungen, welche in den Templates bereits vordefiniert waren. Die Begriffsbestimmung ist dadurch innerhalb eines Projektes aber auch projektübergreifend einheitlich. Auch Begriffsbeziehungen können mit Hilfe der Templates der Anforderungsmuster vorbestimmt werden. Dies zeigt, dass auch die Verwendung einer **Normsprache** indirekt mit dem Gebrauch von Anforderungsmustern für die Spezifizierung und Dokumentation von Anforderungen erfolgt. Indirekt bedeutet hierbei, dass bei der Anforderungsdefinition zwar keine Normsprache direkt zum Einsatz kommt, sondern Templates von Anforderungsmustern, bei welchen eine Normsprache bei der Erstellung verwendet wurde.

Anforderungsmusterkatalog

Analog zu anderen Musterarten der Softwareentwicklung werden auch Anforderungsmuster in einem Musterkatalog (nachfolgend Anforderungsmusterkatalog (AMK)) gesammelt (Mendez-Bonilla, Franch et al. 2008). In der Fachliteratur wird diese Sammelstelle auch als Requirement Pattern Pool und Requirement Pattern Catalogue bezeichnet. Der AMK ist nicht mit einem Mustersystem gleichzusetzen. Während der AMK nur eine Sammelstelle für Anforderungsmuster darstellt, versucht ein Mustersystem Muster sinnvoll miteinander zu verbinden. In dem Mustersystem, welches eher eine abstrakte Ebene ist, werden die Beziehungen zwischen Anforderungsmustern wie z.B. Abhängigkeiten oder gegenseitige Ausschließungen erfasst. Sind Anforderungsmuster in ein AMK aufgenommen worden, so können die Anforderungsschreiber anschließend geeignete Anforderungsmuster aus dem AMK für das jeweilige Projekt auswählen (Rupp 2004). Jedoch sollte die Sammlung von Anforderungsmustern in dem AMK sinnvoll gegliedert werden, um das Suchen und Auswählen der Muster zu vereinfachen (Renault, Mendez-Bonilla et al. 2009; Franch, Palomares et al. 2010). Dafür ist eine Struktur empfehlenswert, bei der die Anforderungsmuster klassifiziert werden. Renault et al. (2009) empfehlen zur Gliederung des AMKs entweder das Klassifikationsschema nach ISO 9126-1 (2001) oder eins, welches sich an den Volere-Ansatz (Robertson and Robertson 2006).

In diesem Abschnitt wurde der Ansatz der Verwendung von Anforderungsmustern innerhalb des REs vorgestellt. Die Auswirkungen auf ein Softwareentwicklungsergebnis sind umso größer, je früher eine Wiederverwendung stattfindet (Pantoquilho 2003). Wenn diese These akzeptiert wird, dann sollte bei Softwareentwicklungsprojekten die Wiederverwendung bereits beim RE beginnen. Dazu sollten hier Anforderungsmuster eingesetzt werden, die die Erfahrungen und das Lösungswissen zu bestimmten Problemstellung aus vorhergehenden Projekten zur Wiederverwendung bereitstellen. Dieses Erfahrungswissen wird in einzelnen Anforderungsmustern niedergeschrieben und in einem Anforderungsmusterkatalog benutzungsfreundlich hinterlegt. Die Anforderungsmuster können in der Anforderungsphase zukünftiger Projekte aus dem Anforderungsmusterkatalog selektiert und an die projektspezifischen

Erfordernisse adaptiert werden. In diesem Abschnitt wurde der Aufbau eines Anforderungsmusters aufgezeigt sowie die Sammelstelle der Muster, der Anforderungsmusterkatalog, erläutert. Ein Anforderungsmuster besteht aus zwei grundlegenden Teilen. Der erste Teil sind Informationen bezüglich des jeweiligen Musters. Es sind Informationen, die der Verwaltung dienen und den richtigen Gebrauch der Muster sicherstellen sollen. Der zweite Teil wesentliche Teil ist das Template. Jedes Anforderungsmuster beinhaltet mindestens ein Template. Wird ein Anforderungsmuster innerhalb eines Projektes eingesetzt, so werden die jeweiligen Templates für die Beschreibung der Anforderung verwendet. Die Templates sind unter Berücksichtigung der Wiederverwendungsabsicht soweit wie möglich vordefiniert. Somit müssen bei der Verwendung nur die Platzhalter projektspezifisch ausgefüllt werden. Die Anforderungsmuster werden in einem Anforderungsmusterkatalog gesammelt. Dieser sollte sinnvoll gegliedert werden um die Arbeit mit dem Katalog zu erleichtern.

7.3 Abgrenzung zu anderen Mustern des Software Engineerings

Neben den noch nicht weit verbreiteten Anforderungsmustern existieren weitere Musterarten, die die Softwareentwicklung effektiver und effizienter machen sollen. Die Design Patterns und die Analysemuster gehören zu den Bedeutsamsten dieser Muster. Nachfolgend werden diese beiden Musterarten kurz vorgestellt. Ebenfalls werden die Unterschiede zwischen ihnen und den Anforderungsmuster verdeutlicht.

Anforderungsmuster versus Design Patterns

In der objektorientierten Softwareentwicklung hat sich die Nutzung von Mustern recht früh etabliert. Einer der treibende Faktoren dafür war das im Jahr 1995 bestverkaufte Buch der Informatik mit dem Titel „Design Patterns: Elements of Reusable Object-Oriented Software“. In diesem Buch stellen die vier Autoren Gamma, Vlissides, Helm und Johnson einige Design Patterns vor, mit denen immer wiederkehrende, gleichartige Aufgabestellungen simpel und elegant gelöst werden können. Jedes Design Pattern „benennt, erläutert und bewertet systematisch einen wichtigen und wiederkehrenden Entwurf in objektorientierten Sytemen.“ (Gamma 2011). Dabei handelt es sich um keine neuartigen Entwürfe oder gar erstaunliche Programmiertricks, sondern lediglich um Entwurfserfahrungen, die Entwicklern verhelfen sollen, eine Lösung schneller „richtig“ zu erstellen. Es sind etablierte Problemlösungen, die im Laufe der Zeit durch viele Entwurfsrevisionen, die die Entwickler durch das Bestreben nach größerer Wiederverwendung und Flexibilität vorgenommen hatten, gefunden wurden. Da jedoch die Problemlösungen noch nie zuvor auf eine solche leicht zugängliche Weise in einem einheitlichen Format niedergeschrieben wurden, werden sie von vielen Anwendern als neu erfasst. Diese Design Patterns der Entwurfsebene der Softwareentwicklung werden im deutschen Sprachraum auch als Entwurfsmuster bezeichnet. Die Entwurfsmuster sind auf einer sehr abstrakten Ebene beschrieben. Dadurch wird eine Kommunikation von bewährten Lösungen ohne Bezug auf bestimmte Programmiersprachen ermöglicht. Die Umsetzung der Muster kann in allen

handelsüblichen Programmiersprachen erfolgen. Die an einem Entwurfsmuster orientierte Lösung eines Entwurfsproblems ist häufig mit einem geringfügig höheren Aufwand als die am nächsten liegende Lösung verbunden. Dieser zusätzliche Aufwand wird jedoch durch eine höhere Flexibilität und Wiederverwendbarkeit kompensiert. Denn die Verwendung von Design Patterns ermöglicht die damit erstellten Entwürfe in zukünftigen Entwicklungsprojekten aufwandsarm wiederzuverwenden (Gamma 2011).

Jedes Entwurfsmuster besteht aus vier grundlegenden Elementen (Gamma 2011). Der Musternamen benennt das Entwurfsproblem und seine Lösungen mit wenigen Worten. Wann ein Muster anzuwenden ist wird in dem Problemabschnitt beschrieben. Der Lösungsabschnitt enthält eine Lösungsschablone für das adressierte Problem, die in vielen unterschiedlichen Situationen angewandt werden kann. Konkrete Entwürfe und Implementierungen sind in einem Entwurfsmuster vergebens zu finden. Das letzte Element eines Entwurfsmusters sind die Konsequenzen. Hier werden die Vor- sowie auch Nachteile der Musteranwendung aufgelistet.

Gamma et al. (2011) klassifizieren Entwurfsmuster hinsichtlich ihrer Aufgaben in Erzeugungsmuster, Strukturmuster und Verhaltensmuster. Fokus der Erzeugungsmuster ist der Prozess der Objekterzeugung. Strukturmuster hingegen beschäftigen sich mit der Zusammensetzung von Klassen und Objekten. Die Art und Weise in der Klassen und Objekte zusammenarbeiten und Zuständigkeiten aufteilen wird mit den Verhaltensmustern charakterisiert.

Heutzutage spielen Entwurfsmuster eine wichtige Rolle in der objektorientierten Softwareentwicklung. Sie stellen ein probates Mittel dar, um sich das Wissen und die Erfahrungen anderer Entwickler nutzbar zu machen. Es ist kaum ein objektorientiertes System zu finden, dass nicht wenigstens einige solcher Entwurfsmuster verwendet. Viele der Entwurfsmuster sind soweit bekannt, dass ihr Name als feststehender Begriffe im Wortschatz vieler Softwareentwickler verankert ist. Sie unterscheiden sich von Anforderungsmustern grundlegend darin, dass sie an die Design-Phase einer objektorientierten Softwareentwicklung adressiert sind. Anforderungsmuster sind ein Teil der Analysephase und befassen sich somit mit dem Problembereich der Softwareentwicklung. Design Patterns hingegen beschreiben Lösungskonzepte. Des Weiteren sind sie ein Hilfsmittel nur für die objektorientierte Softwareentwicklung. Der Erfolg der Entwurfsmuster löste eine Übertragung des Musterkonzeptes auch auf andere Phasen der Softwareentwicklung aus. Ein Beispiel hierfür sind die von Fowler (1997) vorgestellten Analysemuster, die die Analysephase der Softwareentwicklung unterstützen.

Anforderungsmuster versus Analysemuster.
Mit den Requirements Patterns verwandt sind die Analysemuster (häufig auch als OOA-Muster bezeichnet). Sie werden ebenfalls wie die Anforderungsmuster in der Analysephase einer Softwareentwicklung eingesetzt. Ihre Anwendung ist jedoch nur auf objektorientierte Systemanalyse zu beschränken. Denn die darin enthaltenen Lösungsvorschläge für wiederholt anzutreffende Sachverhalte der Realwelt sind modell-

haft abgebildet. Ebenso wie bei dem Systementwurf und dem Requirements Engineering gibt es bei der objektorientierten Systemanalyse häufig ähnliche Probleme (Balzert 1999; Balzert 2009). Analysemuster erlauben den Rückgriff auf erprobte, wiederholende Analysemodelle für objektorientierte Systeme auf einer hohen Abstraktionsstufe. In dem englischen Sprachraum sind sie unter dem Begriff Analysis Patterns bekannt. Bekanntheit erlangten die Analysemuster im Jahr 1997 durch das erfolgreiche Buch von Martin Fowler mit dem Titel „Analysis Patterns: Reusable Object Modells". Hierin beschreibt er objektorientierte Muster zu konzeptionellen Modellierung von typischen Analyseproblemen. Diese Modelle sind laut Fowler (1997) für das Business Engineering ebenso relevant wie für das Software Engineering. Balzert (2009) unterscheidet zwischen allgemeinen und anwendungsspezifischen Analysemustern. Zu der zweiten Art gehören Muster, die beispielsweise Problemlösungen ausdrücklich für Planungssysteme beinhalten. Selbstverständlich resultieren auch aus der Verwendung von Analysemustern Vorteile. Neben der Qualitätssteigerung der Analyseergebnisse besteht der Hauptnutzen in der Verkürzung des kritischen Zeitfaktors „Time-to-market"[10] (Wulff 2002). Denn durch den Zugriff auf Beschreibungen und Lösungen häufig auftretender Probleme muss nicht jedes Analyseproblem vollständig neu gelöst werden. Somit wird die Analysephase schneller beendet, was auch zu einer Verkürzung der gesamten Entwicklungszeit beiträgt (Geyer-Schulz 2001).

Wie auch die Design Patterns enthalten die Analysemuster neben einem Lösungsansatz zu einem wiederkehrenden Problem auch einen eindeutigen Namen, eine Beschreibung des mit diesen Muster betrachten Problems sowie die Auflistung der Konsequenzen durch die Verwendung des Musters. Analysemuster befassen sich nicht mit der eigentlichen Entwicklung der Software. Sie unterstützen stattdessen das Verstehen eines Problems. Dabei fokussieren sie sich auf die organisationalen, sozialen und wirtschaftlichen Aspekte eines Systems (Geyer-Schulz 2001). Auch stellen sie nur einen Ausgangspunkt dar und nicht schon das konkrete Ziel. Somit sind sie analog zu den Anforderungsmustern dem Problembereich zuzuordnen. Im Gegensatz zu Anforderungsmustern sind die Analysemuster aber abstrakter (Withall 2008). Ein weiterer wesentlicher Unterschied besteht darin, dass die Analysemuster an den IT-Architekten adressiert sind (Pantoquilho 2003). Die Anforderungsmuster hingegen sind an die Domäne der Person gerichtet, die die Anforderungsdefinition übernimmt. Analysemuster werden auch wie die Design Patterns und die Anforderungsmuster in einem Katalog abgelegt.

In dem Abschnitt 7.3 wurden zwei weitere Musterarten vorgestellt, die in der Softwareentwicklung zum Einsatz kommen können. Die Design Patterns nach Gamma et al. (2011) und die Analysemuster nach Fowler (1997). Beide Musterarten sind der objektorientierten Softwareentwicklung zuzuordnen. Design Patterns sind Lösungsvorlagen für häufig anzutreffende Sachverhalte bei IT-Systemen. Sie erlauben somit Rückgriff auf bereits erprobte Lösungen in der Entwurfsphase. Die Analy-

[10] Unter dem Begriff time-to-market wird die Zeitdauer von Beginn der Entwicklung eines Produktes bis zu dessen Markteinführung verstanden.

semuster beschreiben hingegen typische wiederkehrende Fälle der Anforderungsanalyse. Sie beinhalten in der Praxis bereits bewährte Modellierungen für wiederholt anzutreffende Sachverhalte der Realwelt. Somit unterscheiden sie sich zu den Anforderungsmustern darin, dass sie ein Hilfsmittel der objektorientierte Softwareentwicklung sind und dass sie nicht für die Definition von Anforderungen konzipiert sind, sondern für die Analysearbeit. Jedoch weisen die Analyse- und Anforderungsmuster eine Gemeinsamkeit auf. Beide sind in der Analysephase der Softwareentwicklung einsetzbare Muster. Bei den Unterschieden zwischen Anforderungsmustern und Design Patterns ist es zum einen die Beschränkung auf die objektorientierte Softwareentwicklung der Design Patterns. Zum anderen unterstützen sie die Entwurfsphase und stellen somit Wissen und Erfahrungen des Lösungsbereiches dar. Sie operieren somit auf der Implementierungsebene.

In dem Abschnitt 7 wurden die Grundlagen zu Anforderungsmustern vermittelt. Dazu wurden zuerst für das RE bereits etablierte Methoden vorgestellt, die ebenfalls durch die Verwendung von Anforderungsmustern, wie sie hier in der Arbeit definiert werden, zum Einsatz kommen. Im Anschluss daran wurde die Methode der Anwendung von Anforderungsmustern ausführlich erläutert. Es wurde aufgezeigt, was innerhalb dieser Arbeit unter einem Anforderungsmuster und Anforderungsmusterkatalog verstanden wird. Da für die Softwareentwicklung weitere Musterarten existieren, wurden in dem letzten Teil des Abschnitts eine Abgrenzung der Anforderungsmuster zu den Design-Patterns und Analysemustern vorgenommen.

8 Zusammenfassung

Die in dem Abschnitt 7 wiedergegebenen Grundlagen zu Anforderungsmustern stellen den letzten Teil des Arbeitspapiers dar. Zunächst wurden in dem Arbeitspapier die Begriffe rund um das RE (Abschnitt 2) definiert. Nachdem aufgezeigt wurde, wie das RE sich in die Softwareentwicklung fügt (Abschnitt 3), wurden die RE-Tätigkeiten (Abschnitt 4), die unterschiedlichen Arten (Abschnitt 5) und die Qualitätskriterien (Abschnitt 6) von Anforderungen thematisiert. Der restliche Teil der Arbeit widmet sich der Beantwortung der Forschungsfragen. Dafür wird zunächst das Forschungsdesign beschrieben. Schließlich werden die Ergebnisse detailliert dargelegt.

Literatur

Balzert, H. (1999). Lehrbuch der Objektmodellierung – Analyse und Entwurf. Heidelberg, Spektrum Akademischer Verlag.

Balzert, H. (2009). Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. Heidelberg, Spektrum Akademischer Verlag.

Berry, D. M. K., E.; Krieger, M.M. (2003). From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity – A Handbook, http://se.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf.

Boehm, B., P. Grunbacher, et al. (2001). "Developing groupware for requirements negotiation: lessons learned." Software, IEEE **18**(3): 46-55.
Boehm, B. W. (1984). " Verifying and Validating Software Requirements and Design Specifications." Software, IEEE **1**(1): 75 - 88.
Brcina, C. (2007). "Arbeiten zur Verfolgbarkeit und Aspekte des Verfolgbarkeitsprozesses." Softwaretechnik-Trends **27**(1): 3-8.
Buschmann, F. M., R.; Rohnert, H.; Sommerlad, P.; Stal, M. (1998). Pattern-orientierte Software-Architektur – Ein Pattern-System. New York, Addison-Wesley.
Cao, L. and B. Ramesh (2008). "Agile Requirements Engineering Practices: An Empirical Study." Software, IEEE **25**(1): 60-67.
Chrissis, M. B. K., M.; Shrum, S. (2003). CMMI: Guidelines for Process Integration and Product Improvement. Boston, Addison-Wesley Professional.
Deifel, B. (1998). Theoretische und praktische Ansätze im Requirements Engineering für Standardsoftware und Anlagenbau, Technische Universität München.
DIN 69905 (1997). DIN 69905: Projektabwicklung – Begriffe, Deutsches Institut für Normung.
Durán Toro, A., B. Bernárdez Jiménez, et al. (1999). A Requirements Elicitation Approach Based in Templates and Patterns. Workshop em Engenharia de Requisitos.
Ebert, C. (2008). Systematisches Requirements Engineering und Management – Anforderungen ermitteln, spezifizieren, analysieren und verwalten. Heidelberg, dpunkt.verlag.
Ebert, C. (2012). Systematisches Requirements Engineering – Anforderungen ermitteln, spezifizieren, analysieren und verwalten. Heidelberg, dpunkt.verlag.
Edwards, M. H., S.L. (1991). A Methology for Systems Requirements Specification and Tracebility for large Real-Time Complex Systemy. Dahlgren, Virginia, Naval Surface Warfare Center.
Fowler, M. (1997). Analysis Patterns: Reusable Object Models. Amsterdam, Addison-Wesley Longman.
Franch, X., C. Palomares, et al. (2010). A Metamodel for Software Requirement Patterns. 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), Essen, Germany, Springer.
Gamma, E. H., R.; Johnson, R.; Vlissides, J. (2011). Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software. München, Addison-Wesley.
Geyer-Schulz, A. H., M. (2001). Software Engineering with Analysis Patterns. Technical Report. Wien, Institut für Informationsverarbeitung und -wirtschaft.
Glinz, M. and R. J. Wieringa (2007). "Stakeholders in Requirements Engineering." Software, IEEE **24**(2): 18-20.
Götz, R. S., H. (1997). IVENA: Integriertes Vorgehen zur Erhebung nichtfunktionaler Anforderungen.
Grande, M. (2011). 100 Minuten für Anforderungsmanagement – Kompaktes Wissen nicht nur für Projektleiter und Entwickler. Wiesbaden, Vieweg + Teubner Verlag.
Hansen, H. R. and G. Neumann (2005). Wirtschaftsinformatik 1 – Grundlagen und Anwendungen. Stuttgart, Lucius & Lucius.
Hickey, A. and A. Davis (2003). Elicitation technique selection: how do experts do it? Proceedings of the 11th IEEE International Requirements Engineering Conference (RE´03), Monterey Bay.
Hood, C. W., R. (2005). Optimieren von Requirements Management & Engineering. Berlin, Springer-Verlag.
IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990. New York.

IEEE (1998). IEEE recommended practice for software requirements specifications. IEEE Std 830-1998. New York.

ISO 9126-1 (2001). ISO/IEC Standard 9126-1: Software Engineering – Product Quality – Part 1: Quality Model.

Ketabchi, S., N. Karimi Sani, et al. (2011). A Norm-Based Approach towards Requirements Patterns. 35th IEEE Annual Computer Software and Applications Conference (COMPSAC), München.

Knethen, A. v. P., B.; Kiedaisch, F.; Houdek, F. (2002). Systematic Requirements Recycling through Abstraction and Traceability. Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02). Essen**:** 273-281.

Konrad, S. and B. H. C. Cheng (2002). Requirements patterns for embedded systems. Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02), Essen.

Kotonya, G. and I. Sommerville (1998). Requirements engineering: Processes and techniques. Chichester, Wiley.

Leffingwell, D. W., D. (2003). Managing Software Requirements – A Use Case Approach. Boston, Addison-Wesley.

Lehmann, F. R. (1998). "Normsprache." Retrieved 08.08.2012, from http://www.gi.de/service/informatiklexikon/detailansicht/article/normsprache.html.

Mendez-Bonilla, O., X. Franch, et al. (2008). Requirements Patterns for COTS Systems. Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS)

Neill, C. J. and P. A. Laplante (2003). "Requirements engineering: the state of the practice." IEEE Software **20**(6): 40-45.

Pantoquilho, M. R., R.; Araujo; J. (2003). Analysis Patterns Specifications: Filling the Gaps. 2nd Viking Conference on Pattern Languages of Programs (PLoP), Monticello.

Partsch, H. (2010). Requirements-Engineering systematisch – Modellbildung für softwaregestützte Systeme. Heidelberg, Springer-Verlag.

Pinheiro, F. A. C. (2003). Requirements Traceability. Perspectives on Software Requirements. J. C. S. P. Leite and J. H. Doorn. Massachusetts, Kluwer Academic Publishers**:** 91-113.

Pohl, K. (2008). Requirements Engineering. Heidelberg, dpunkt-Verlag.

Pohl, K. and C. Rupp (2009). Basiswissen Requirements Engineering. Heidelberg, dpunkt.verlag.

Renault, S., O. Mendez-Bonilla, et al. (2009). PABRE: Pattern-based Requirements Elicitation. Proceedings of the Third International Conference on Research Challenges in Information Science (RCIS)

Robertson, S. and J. Robertson (2006). Mastering the requirements process. Boston, Addison-Wesley Professional.

Rupp, C. (2004). Requirements Templates – The Blueprint of your Requirement. Nürnberg, Sophist Group.

Rupp, C. and Sophist Group (2002). Requirements-Engineering und -Management – Professionelle, iterative Anforderungsanalyse für die Praxis. München, Carl Hanser Verlag.

Rupp, C. and SOPHISTen (2009). Requirements-Engineering und -Management – Professionelle, iterative Anforderungsanalyse für die Praxis. München, Carl Hanser Verlag.

Schienmann, B. (2002). Kontinuierliches Anforderungsmanagement – Prozesse, Techniken, Werkzeuge. München, Addison-Wesley Verlag.

Versteegen, G., A. Heßeler, et al. (2004). Anforderungsmanagement. Berlin, Springer-Verlag.
Wahono, C. (2002). On the Requirements Pattern of Software Engineering. Proceedings of the Temu Ilmiah XI, Ilmiah.
Wiegers, K. E. (2005). Software Requirements. Unterschleißheim, Microsoft Press Deutschland.
Withall, S. (2008). Software Requirement Patterns. Redmont, Washington, Microsoft Press.
Wulff, A. (2002). Ein Werkzeug zur Nutzung von Analysemustern. Proceedings of the Modellierung 2002, Tutzing.