industrielogistik

Masterarbeit

Requirements-Engineering für Intralogistik Software Projekte

eingereicht an der

Montanuniversität Leoben

betreut am

Lehrstuhl Industrielogistik

erstellt für die

KNAPP Systemintegration GmbH

Vorgelegt von:
Stefanie GROJER, BSc
0735264

Betreuer/Gutachter:
Ass. Prof. DI (FH) Dr. techn. Susanne Altendorfer-Kaiser
Univ.-Prof. Dr. Helmut Zsifkovits

Leoben, 10. März 2016

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Aus Gründen der Lesbarkeit wurde in dieser Arbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Es wird ausdrücklich festgehalten, dass die bei Personen verwendeten maskulinen Formen für beide Geschlechter zu verstehen sind.

Stefanie Grojer

Leoben, 10. März 2016

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meines Studiums und bei der Erstellung dieser Masterarbeit unterstützt haben.

Zunächst möchte ich mich bei Herrn Univ.-Prof. Dr. Helmut Zsifkovits für die Möglichkeit bedanken, dass ich diese Masterarbeit am Lehrstuhl für Industrielogistik verfassen durfte.

Ein besonders herzlicher Dank gilt Frau Ass. Prof. DI (FH) Dr. techn. Susanne Altendorfer-Kaiser, die meine Arbeit betreut hat. Ihre Unterstützung, die wertvollen Anregungen und die konstruktive Kritik waren für mich äußerst lehr- und hilfreich.

Im besonderen Maß bedanken möchte ich mich bei der KNAPP Systemintegration GmbH, mit meinem Vorgesetzten und Firmenbetreuer Herrn Ing. Johann Neuhauser, für die Unterstützung während meines Studiums und für die Möglichkeit, viel wertvolle Arbeitszeit in das Thema „Requirements-Engineering" investieren zu können. Weiters danke ich allen Kolleginnen und Kollegen, die zum Gelingen dieser Masterarbeit beigetragen haben. Ganz besonders gilt dies für die Mitglieder des Projektteams, Frau DI Karin Weiser, Herrn Ing. Johann Neuhauser, Herrn DI Markus Hübler, Herrn Dipl.-Wirtsch.-Inf. Stephan Wagner und Herrn DI (FH) MSc MA Walter-Rene Macku. Bedanken möchte ich mich für die zahlreichen interessanten Debatten und die Ideen, die maßgeblich dazu beigetragen haben, dass diese Masterarbeit in dieser Form vorliegt.

Herzlich danken möchte ich auf diesem Weg auch meiner Familie, die mich auf meinem Bildungsweg nicht nur finanziell sondern vor allem auch emotional unterstützt und mich in allen meinen Entscheidungen stets bestärkt hat.

Abschließend bedanke ich mich bei meinen Freundinnen und Freunden, die mir während meines Studiums und der Erstellung dieser Masterarbeit sowohl fachlich, als auch seelisch zur Seite gestanden sind. Dies gilt insbesondere für meinen Freund DI Clemens Seidler - danke für die Unterstützung, den Zuspruch und den Rückhalt.

Kurzfassung

Die vorliegende Masterarbeit befasst sich mit der Beschreibung und der Auswahl von klassischen Requirements-Engineering-Methoden für Intralogistik Software Projekte der KNAPP Systemintegration GmbH. Dabei ist es das Ziel der Arbeit, Methoden für das Ermitteln der Projektgrundlagen, die Anforderungserhebung, die Spezifikation und die Prüfung von Anforderungen zu Charakterisieren und im Rahmen eines Verbesserungsprojektes für die Anwendung bei der KNAPP Systemintegration GmbH auszuwählen. Gleichzeitig sollte das Verständnis für die Bedeutung und den Nutzen von professionellem Requirements-Engineering gefördert werden.

Um dieses Ziel zu erreichen, werden einleitend die Aufgabengebiete der KNAPP Systemintegration GmbH und der Abteilung „Software Engineering" sowie die Problemstellung und die Zielsetzung der Masterarbeit beschrieben. Im Anschluss an die Einleitung erfolgt im ersten Teil der Arbeit eine Definition der essentiellen Grundbegriffe des Themas. Zuerst werden die Hauptaufgaben des Requirements-Engineerings geklärt, danach wird auf den Begriff der Anforderung und der Spezifikation eingegangen. Anschließend werden die Dokumentation der Projektgrundlagen und die drei Hauptaufgaben des Requirements-Engineerings, die Anforderungsermittlung, die Anforderungsspezifikation und die Prüfung von Anforderungen näher beschrieben. Dabei werden die jeweilig durchzuführenden Teilaktivitäten erläutert und eine Reihe von Techniken vorgestellt, die den Requirements-Engineer in der jeweiligen Phase unterstützen können.

Im Anschluss an die theoretischen Grundlagen beschreibt der zweite Teil der Masterarbeit die Anwendung des erworbenen Wissens im Rahmen eines Verbesserungsprojektes, das in Zusammenarbeit mit der KNAPP Systemintegration GmbH durchgeführt wurde. Ziel des Projektes war es, gemeinsam mit einem Projektteam, Verbesserungspotentiale im Requirements-Engineering aufzudecken und konkrete Verbesserungsmaßnahmen zu definieren. Zunächst wird die IST-Situation des Requirements-Engineerings bei der KNAPP Systemintegration GmbH erläutert, bevor die Vorgehensweise und die Ergebnisse der Projektarbeit im Detail dargestellt werden.

Schlussendlich erfolgt im dritten Teil der Arbeit die Conclusio, in welcher die wichtigsten Erkenntnisse der vorliegenden Masterarbeit resümiert werden und das Ergebnis den Zielen gegenübergestellt wird.

Abstract

This master's thesis deals with the description and the selection of classic requirements-engineering techniques for the intralogistics software projects of the KNAPP Systemintegration GmbH. The purpose of this work is the characterization of approaches for the description of the project fundamentals as well as the requirements elicitation, the specification and the verification of requirements and the subsequent selection of suitable techniques for the future application in the course of an improvement project at the KNAPP Systemintegration GmbH. Additionally, this master's thesis aims to raise awareness of the importance of professional requirements-engineering for the successful completion of customer projects.

In order to achieve the defined goals, the preface of the thesis introduces the KNAPP Systemintegration GmbH and the department "Software Engineering" as well as the problem and the purpose statement. The subsequent first part of the thesis defines the necessary theoretical foundations of the topic. First, the main tasks of requirements-engineering as well as the concept of requirements and the requirements specification document are described. Then the thesis focuses on the documentation of the project fundamentals and the three main tasks of requirements-engineering, the requirements elicitation, the requirements specification and the verification of requirements. The necessary subtasks are highlighted and a set of techniques is characterized, that aims to facilitate the fulfillment of the mentioned tasks.

In a detailed second part, the thesis focuses on the practical application of the acquired knowledge in the context of an improvement project at the KNAPP Systemintegration. The main goal of the project was to identify improvement potential regarding the requirements-engineering practice and to determine concrete improvement measures in order to enhance the quality of the requirements and the requirements specification documents and to facilitate the integration of new employees in the projects. First, the thesis examines the current situation regarding the requirements-engineering practice at the KNAPP Systemintegration GmbH, before it depicts the proceedings and the deliverables of the improvement project.

Finally, the conclusion addresses the findings of the thesis and compares them with the initially stated purpose of the work.

Inhaltsverzeichnis

Tabellen- und Abbildungsverzeichnis

Einleitung

KNAPP Systemintegration GmbH

Die KNAPP Gruppe mit ihrer Konzernzentrale in Hart bei Graz ist ein international marktführendes Unternehmen im Bereich der Intralogistik und Lagerautomation mit weltweit 25 Standorten und 2700 Mitarbeitern.

Als Generalunternehmer konzipiert KNAPP weltweit kundenspezifische, manuelle bis vollautomatisierte Lagersysteme, die in weiterer Folge als Projekt geplant, installiert und in Betrieb genommen werden. Den weiteren Betrieb der Anlage übernimmt der Kunde, betreut durch ein umfassendes Kundenservice. Der Auftragseingang erreichte im Wirtschaftsjahr 2014/15 einen Betrag von 578 Millionen Euro bei einer Exportquote von 98 Prozent.[1]

Das Produktportfolio des gesamten Unternehmens umfasst dabei technische Anlagen wie Behälter-, Karton- und Palettenfördertechnik, automatische und manuelle Kommissioniersysteme, sowie Lager-Logistiksoftware, von der Steuerung der Gewerke durch die SPS[2] bis hin zum Warehouse Management System. Als Software-Umgebung wird hauptsächlich die, vom Unternehmen selbst entwickelte, KiSoft-Produktfamilie eingesetzt, um die logistischen Prozesse des Kunden abzubilden. Abhängig von der Projektkomplexität wird die Software individuell für den Kunden entwickelt, oder es können Standardkomponenten eingesetzt werden. Weltweit sind derzeit rund 1600 von KNAPP errichtete Anlagen in Betrieb.

Die KNAPP Systemintegration GmbH mit Sitz in Leoben ist, als Teil der KNAPP Gruppe, für die Entwicklung von logistischen Gesamtkonzepten und die Umsetzung von Kundenprojekten verantwortlich.

Die vorliegende Masterarbeit wurde in Zusammenarbeit mit der Abteilung „Software Engineering“ der KNAPP Systemintegration GmbH verfasst. Dieser Unternehmensbereich ist im Rahmen der technischen Projektleitung, sowohl intern als auch dem Kunden gegenüber, für die, auf den in der Projektierungsphase ermittelten logistischen Kundenanforderungen basierende, Erarbeitung und detaillierte Beschreibung der erforderlichen Prozesse des Lagers in den Spezifikationen[3] sowie deren Weitergabe

[1] vgl. (KNAPP Systemintegration GmbH (Hrsg.), 2015)
[2] Speicherprogrammierbare Steuerung
[3] Im deutschsprachigen Raum ist hierfür der Ausdruck „Pflichtenheft“ üblich. In der vorliegenden Arbeit wird jedoch der allgemein bekannte Begriff „Spezifikation“ verwendet.

an die Software-Entwicklung verantwortlich. In weiterer Folge sind die technischen Projektleiter dafür zuständig, sicherzustellen, dass die gelieferte Software die Prozesse der funktionalen Spezifikation abbildet. Der technische Projektleiter kontrolliert letztendlich, ob mit der Software die logistischen Anforderungen des Kunden erfüllt werden können. Als eine der essentiellsten Tätigkeit gilt die Erstellung der funktionalen Spezifikation, welche möglichst alle Kundenanforderungen hinsichtlich der Software in einem hohen Detaillierungsgrad beinhaltet.[4]

Problemstellung

Die Intralogistik befasst sich im Wesentlichen mit dem innerbetrieblichen Logistiknetzwerk eines Lagers des Kunden. Die Kernaufgabe der Intralogistik ist es, das Logistiksystem so zu gestalten, zu dimensionieren, zu realisieren und zu betreiben, dass die Leistungsanforderungen möglichst kostenoptimal erfüllt werden. Zudem gilt es, die einzelnen Logistiksysteme des Lagers so zu einem leistungsfähigen Logistiknetzwerk zu verknüpfen und die verfügbaren Ressourcen so zu disponieren, dass die Auftrags-, Leistungs- und Logistikprozesse des Lagers optimal ablaufen.[5] Ziel eines Intralogistikanbieters, wie der KNAPP Systemintegration ist es, ein Logistiksystem zu planen, dass es dem Anlagenbetreiber ermöglicht, seine Logistikprozesse effizient abzuwickeln und dadurch die stetig steigenden Erwartungen seiner Kunden an die Schnelligkeit, die Qualität, die Kosten und die Variantenvielfalt der Logistikleistung zu erfüllen. Der Planung der Systemstruktur der Intralogistikanlage kommt damit besondere Bedeutung zu, da nicht nur die aktuellen, sondern auch die mittel- bis langfristigen Leistungsanforderungen des Kunden an die Lagerprozesse berücksichtigt werden müssen. Insbesondere die Intralogistiksoftware steht dabei vor der Herausforderung, immer komplexere Prozesse abbilden und unterstützen zu müssen, dabei soll jedoch die Bedienung der Software durch den Lagermitarbeiter, immer einfacher und intuitiver, erfolgen können.[6] Die Anforderungen der Anlagenbetreiber an die Software können jedoch stark variieren und eine der Hauptaufgaben des Software-Engineers als Softwareplaner, ist es, sich in den Projektphasen der Erhebung und der Dokumentation der Anforderungen mit dem Kunden darauf zu einigen, was genau die Software, die geliefert werden soll, beinhalten wird. Die definierten Anforderungen werden in weiterer Folge geprüft und verwaltet. Ziel dieser Prozesse ist es, alle funktiona-

[4] Da sich die vorliegende Arbeit mit der Erstellung der funktionalen Spezifikation befasst, wird auf eine Beschreibung der sonstigen Tätigkeiten des technischen Projektleiters nicht eingegangen.
[5] vgl. (Gudehus, 2010, S. 550 f.)
[6] vgl. (Hompel & Schmidt, 2008, S. 8 ff.)

len und nicht-funktionalen Anforderungen des Kunden an die Intralogistikanlage, korrekt und vollständig in der Spezifikation abzubilden, um damit die Basis für alle weiteren Projektaktivitäten zu legen.[7] Es handelt sich dabei um besonders kritische Aufgaben, da der falsche Umgang mit Anforderungen einer der häufigsten Gründe für gescheiterte Projekte ist. Typischerweise werden 30 bis 50 Prozent des Entwicklungsaufwandes für Nacharbeiten, Fehlerbehebungen und nicht wertbildende Aktivitäten eingesetzt. In etwa die Hälfte dieser Projektschwierigkeiten resultieren aus dem falschen Umgang mit den Anforderungen des Kunden: aus unzureichendem Requirements-Engineering und -Management. Dies beinhaltet beispielsweise unklare oder sich unkontrolliert ändernde Anforderungen, Lücken zwischen Kundenerwartungen und Projektinhalten, zu enge Zeitpläne mit unerreichbaren Meilensteinen, oberflächliche oder ungenaue Aufwandseinschätzungen oder unkontrollierte Unteraufträge an Subunternehmer.[8] Betrachtet man das Requirements-Engineering, kommt es auch bei Projekten der KNAPP Systemintegration immer wieder vor, dass Anforderungen zum Beispiel aus zeitlichen Gründen in den Anforderungserhebungs-Workshops nicht mehr im Detail spezifiziert, missverstanden oder nicht als Anforderungen erkannt werden, Anforderungen unterschiedlich detailliert beschrieben und die Anforderungen nicht einheitlich verwaltet werden. Dies führt teils zu erheblichen ungeplanten Ressourcenaufwänden im weiteren Verlauf des Projektes.

Im Zuge einer schriftlichen Befragung der technischen Projektleiter, Tester und Inbetriebnehmer sowie der Entwicklungsprojektleiter konnten folgende Probleme bei der Erfassung und Beschreibung von Anforderungen identifiziert werden:

- Die Qualität der Anforderungen schwankt stark je nach Verfasser.
- Die Anforderungen sind lückenhaft und geben zu viel Spielraum zur Interpretation.
- Die Anforderungen sind teilweise widersprüchlich.
- Nicht-funktionale Anforderungen werden nicht erfasst.
- Es wird nicht klar abgegrenzt, was zum Lieferumfang gehört und was nicht.

Auf die Frage nach den Gründen für die genannten Probleme wurden am häufigsten folgende Ursachen angeführt:

- Zeitdruck
- Kunde kennt seine Anforderungen nicht

[7] vgl. (Sharma & Pandey, 2013)
[8] vgl. (Ebert, 2005, S. 68 ff.)

- Sprachbarrieren/kulturelle Unterschiede
- Fehlende Stakeholder/falsche Personen in den Workshops auf Kundenseite
- Unerfahrenheit des Kunden mit automatisierten Lagersystemen
- Mangelhafte Beschreibungsart (Fließtext)
- Anforderungen werden vom technischen Projektleiter missverstanden
- Versteckte Anforderungen werden nicht erkannt

Die Ergebnisse der Befragung lassen den Schluss zu, dass es durchaus Optimierungspotential im Requirements-Engineering gibt. Besonders dann, wenn sich der Kunde seiner spezifischen Anforderungen noch nicht bewusst ist, was häufig mit der Unerfahrenheit mit automatisierten Lagersystemen zusammenhängt, passiert es, dass Anforderungen beispielsweise übersehen werden. Sehr oft kommt es auch vor, dass nichtfunktionalen Anforderungen keine oder zu wenig Aufmerksamkeit zu Teil wird. Später sind es jedoch genau diese Anforderungen, die große Konflikte auslösen.

Zusätzlich zur Befragung wurde im Vorfeld der Erstellung der vorliegenden Arbeit ein Workshop abgehalten, bei dem technische Projektleiter und Entwicklungs-Projektleiter gemeinsam erarbeiteten, welche Schwierigkeiten die Teilnehmer in Bezug auf das Requirements-Engineering und -Management bei der Knapp Systemintegration feststellen können. Dabei wurden folgende Probleme identifiziert:

- Die Anforderungen sind keine in sich geschlossenen Abläufe. Häufig sind Standardfälle und Ausnahmefälle vermischt und damit zu viele Funktionalitäten in einer Anforderung dokumentiert.
- Die Qualität und der Umfang einer Anforderung variieren sehr stark, abhängig vom Verfasser.
- Die Begutachtung durch den Entwicklungsprojektleiter erfolgt zu spät, in den meisten Fällen erst nach der Abnahme der Spezifikation durch den Kunden.
- Änderungen in den Anforderungen werden nicht durchgängig, über alle Systeme hinweg, dokumentiert.
- Teils werden Entwicklungsentscheidungen in der Spezifikation vorweggenommen, dies betrifft im Speziellen Entscheidungen über System- und Komponentengrenzen.

Zusätzlich zu den genannten firmeninternen Schwierigkeiten lässt sich feststellen, dass die Entwicklung softwareintensiver Systeme heute ganz allgemein vor einer Reihe

von Herausforderungen steht. Immer komplexere, innovativere, individuellere Systeme müssen immer schneller entwickelt und dennoch mit immer höherer Qualität, zu immer günstigeren Preisen, auf dem Markt gebracht werden. Software Engineering im Allgemeinen und Requirements-Engineering im Speziellen gewinnt damit seit Jahren an Bedeutung und ist heute essentiell für die erfolgreiche Systementwicklung.[9]

Ziele und Aufbau der Arbeit

Qualitativ hochwertige Anforderungen gelten als Voraussetzung dafür, ein System zu entwickeln, dass den Wünschen und Bedürfnissen des Kunden gerecht wird und damit den nachhaltigen Fortbestand des Intralogistik-Software-Unternehmens sichert. Die vorliegende Arbeit befasst sich daher mit der Beantwortung der folgenden Fragen:

- Welche Methoden des klassischen Requirements-Engineerings eignen sich im Hinblick auf die Intralogistikprojekte der KNAPP Systemintegration am besten, um alle Anforderungen der Kunden effizient und effektiv spezifizieren zu können?
- Wie könnten diese Methoden bei der KNAPP Systemintegration konkret umgesetzt werden?

Mit der Beantwortung dieser Fragen soll ein größeres Bewusstsein für die Bedeutung des Requirements-Engineerings und für die einsetzbaren Methoden entwickelt werden, um diese zielgerichtet verwenden zu können und damit einen wichtigen Beitrag zum erfolgreichen Projektverlauf zu leisten.

Um dieses Ziel zu erreichen, ergibt sich der folgende Aufbau der Arbeit: In der Einleitung wurden bereits die KNAPP AG und die KNAPP Systemintegration und die Tätigkeitsbereiche der Abteilung „Software Engineering“ kurz beschrieben. Darauf folgten die Problemstellung und die soeben dargestellte Zielsetzung der Arbeit.

Im darauffolgenden Hauptteil der Arbeit klärt Teil I die notwendigen theoretischen Grundlagen. Zuerst wird auf den Begriff der „Anforderungen“ eingegangen, danach erfolgt eine Definition des Begriffes der „funktionalen Spezifikation“. In den nachfolgenden Kapiteln werden die wichtigsten Phasen des Requirements-Engineerings beschrieben. Dazu wird in Kapitel 2 auf die Projektgrundlagen Stakeholderanalyse, Zieldefinition und Kontextanalyse eingegangen, danach beschreibt Kapitel 3 die Anfor-

[9] vgl. (Pohl, 2007, S. 7); (Sommerville I. , 2012, S. 29)

derungsermittlung, Kapitel 4 die Anforderungsdokumentation und abschließend, Kapitel 5, die Anforderungsvalidierung.

Der zweite Teil der vorliegenden Arbeit befasst sich mit der praktischen Anwendung des erlangten theoretischen Wissens im Rahmen eines Verbesserungsprojektes. Ziel dieses Projektes war es, Verbesserungspotentiale im Bereich des Requirements-Engineerings bei der KNAPP Systemintegration aufzudecken und konkrete Verbesserungsmaßnahmen abzuleiten. Zuerst erläutert Kapitel 6 die Vorgehensweise, die bei der Durchführung des Projektes gewählt wurde, anschließend wird in Kapitel 7 der IST-Prozess des Requirements-Engineerings bei der KNAPP Systemintegration im Detail beschrieben. In den nachfolgenden Kapiteln werden die Ergebnisse des Projektes dargestellt.

In Teil III der vorliegenden Masterarbeit erfolgt die Schlussbetrachtung in Form einer Conclusio und einem Ausblick.

Teil I Theoretische Grundlagen

Nachdem nun die Ausgangssituation und die Zielsetzung der vorliegenden Masterarbeit beschrieben wurden, erfolgt im ersten Teil der Arbeit die Erläuterung der notwendigen theoretischen Grundlagen des Requirements-Engineerings. Zuerst werden die wichtigsten Begriffe definiert, danach werden, geordnet nach den Hauptaktivitäten des Requirements-Engineerings, Techniken vorgestellt, die den Requirements-Engineer bei der Durchführung der jeweiligen Aktivität unterstützen können.

1 Requirements-Engineering

Im folgenden Kapitel soll eine Auswahl von grundlegenden Begriffen aus dem Requirements-Engineering erläutert werden. Nachdem kurz auf die wichtigsten Aufgaben im professionellen Requirements-Engineering eingegangen wird, erfolgt eine Betrachtung der Begriffe der „Anforderungen" und der „funktionalen Spezifikation".

1.1 Aufgaben im Requirements-Engineering

Die Anforderungen an eine weitgehend kundenspezifische Lagersoftware zu ermitteln, zu spezifizieren, zu validieren und zu verwalten gehört zu den anspruchsvollsten Aufgaben innerhalb der Systementwicklung. Lange Zeit wurde im deutschsprachigen Raum dafür hauptsächlich der Begriff „Systemanalyse" verwendet, heute wird auch im Deutschen fast ausschließlich der englische Begriff „Requirements-Engineering" verwendet. Ergänzend oder untergeordnet existiert oft auch der Begriff „Requirements-Management", der sich vorranging auf die Verwaltung von Anforderungen bezieht.[10,11]

Das Requirements-Engineering befasst sich, wie bereits eingangs erwähnt, mit Anforderungen. Eine genaue Definition und Beschreibung des Begriffes „Anforderung" erfolgt in Kapitel 1.2. Zum besseren Verständnis sei an dieser Stelle die folgende, zusammengefasste Definition angeführt:[12]

„Eine Anforderung ist eine Aussage über eine Eigenschaft oder Leistung eines Produkts, eines Prozesses oder der am Prozess beteiligten Personen."

Welche Aufgaben die Disziplin des Requirements-Engineerings konkret umfasst, zeigt die folgende Definition des IREB[13]:[14]

Das Requirements-Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen mit den folgenden Zielen:

- Die relevanten Anforderungen zu kennen, Konsens unter den Stakeholdern über die Anforderungen herzustellen, die Anforderungen konform zu vorge-

[10] vgl. (Balzert, 2009, S. 434)

[11] Auch in der vorliegenden Arbeit werden diese beiden Begriffe verwendet, wobei das Requirements-Management als untergeordnete Disziplin des Requirements-Engineerings verstanden wird. Im Rahmen der vorliegenden Arbeit wird jedoch ausschließlich das Requirements-Engineering behandelt. In Bezug auf den Begriff der „Anforderung", hält sich die Arbeit an den die gebräuchliche deutschsprachige Bezeichnung und spricht im Allgemeinen nicht von „Requirements".

[12] (Rupp & die SOPHISTen, 2014)

[13] International Requirements Engineering Board

[14] (Pohl & Rupp, 2011)

gebenen Standards zu dokumentieren und die Anforderungen systematisch zu managen,

- die Wünsche und Bedürfnisse der Stakeholder zu verstehen und zu dokumentieren,
- die Anforderungen zu spezifizieren und zu managen, um das Risiko zu minimieren, ein System auszuliefern, das nicht den Wünschen und Bedürfnissen der Stakeholder entspricht.

Analog zu dieser Definition beschreibt die Literatur weitgehend übereinstimmend die, in der folgenden Abbildung dargestellten, Haupttätigkeiten des Requirements-Engineerings: Die Ermittlung, die Dokumentation, die Validierung und die Verwaltung von Anforderungen.

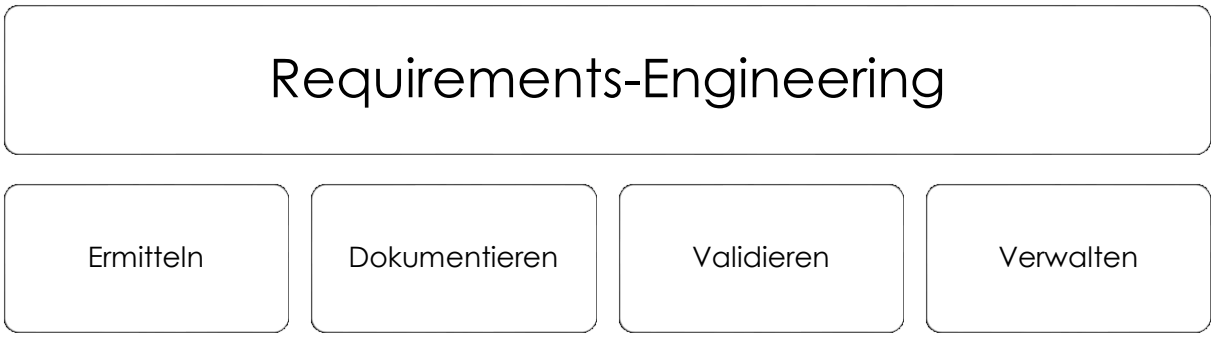


Abbildung 1: Haupttätigkeiten des Requirements-Engineerings[15]

Die **Anforderungsermittlung** befasst sich mit der Gewinnung von bereits existierenden Anforderungen von Stakeholdern und anderen Anforderungsquellen sowie der Entwicklung innovativer Anforderungen gemeinsam mit den Stakeholdern.[16]

Im Zuge der **Anforderungsdokumentation** werden die ermittelten Anforderungen spezifiziert, d. h. sie werden unter Berücksichtigung von festgelegten Methoden, Richtlinien, Konventionen, Checklisten und/oder Schablonen beschrieben.[17]

Die **Anforderungsvalidierung** hat zum Ziel, Fehler in den Anforderungen bezüglich des Inhalts, der Dokumentation und der Abgestimmtheit aufzudecken, um die hohe Qualität der Anforderungen sicherstellen zu können.[18]

[15] vgl. (Rupp & die SOPHISTen, 2014, S. 14), (Hull, Jackson, & Dick, 2011, S. 8), (Pohl, 2007, S. 44 ff.)
[16] vgl. (Pohl, 2007, S. 44)
[17] vgl. (Balzert, 2009, S. 444)

Die **Anforderungsverwaltung** bzw. das Requirements-Management befasst sich mit der sinnvollen und nachvollziehbaren Organisation der Anforderungen und Anforderungsdokumente, der Priorisierung und sonstigen Zustandszuweisung von Anforderungen und dem Management sämtlicher Änderungen in den Anforderungen.[19] Da die ausführliche Behandlung aller vier Haupttätigkeiten des Requirements-Engineerings den Umfang der vorliegenden Masterarbeit übersteigen würde, werden in weiterer Folge ausschließlich die Ermittlung, die Dokumentation und die Validierung von Anforderungen bearbeitet.

Auf die genannten Tätigkeiten wird in den Kapiteln 2 bis 5 des ersten Teils der vorliegenden Arbeit näher eingegangen. Zunächst erfolgt jedoch eine Definition und Klärung des Begriffes der Anforderung und der Spezifikation.

1.2 Anforderungen

1.2.1 Definition

In der Literatur lassen sich zahlreiche Definitionen für den Begriff der Anforderung finden, am häufigsten wird die folgende Charakterisierung des „Institute of Electrical and Electronics Engineers“ zitiert:[20]

1. Eine Bedingung oder Eigenschaft, die **ein Benutzer** benötigt um ein Problem zu lösen oder ein Ziel zu erreichen.
2. Eine Bedingung oder Eigenschaft, die **ein System oder eine Systemkomponente** aufweisen muss, um einem Vertrag, einem Standard, einer Spezifikation oder einem sonstigen formell auferlegten Dokument zu genügen.
3. Eine **dokumentierte Repräsentation** einer Bedingung oder Eigenschaft wie unter Punkt 1 oder 2 definiert.

Insofern existieren verschiedene Ebenen von Anforderungen, je nach Perspektive. Während die Definition von Punkt 1 eine benutzernahe Sicht zum Ausdruck bringt (Benutzeranforderungen), spiegelt Punkt 2 die Sicht des Systemlieferanten oder Entwicklers wider (Systemanforderungen).[21] Zusätzlich zu den verschiedenen Ebenen von Anforderungen, auf die im nächsten Kapitel eingegangen wird, lassen sich verschiedene Arten von Anforderungen unterscheiden (siehe Kapitel 1.2.3).

[18] vgl. (Pohl & Rupp, 2011)
[19] vgl. (Rupp & die SOPHISTen, 2014, S. 368 f.)
[20] vgl. (Institute of Electric and Electronic Engineers, 1990)
[21] vgl. (Goeken, 2006, S. 108)

Wie in der folgenden Abbildung ersichtlich, kann die Entwicklung von Anforderungen als Zyklus beschrieben werden. Am Beginn werden Anforderungen erhoben, dann analysiert, priorisiert und schließlich wird das Erarbeitete spezifiziert. Mit größter Wahrscheinlichkeit werden im Laufe dieser Tätigkeiten Lücken oder Unklarheiten identifiziert, die weiterer Erhebungsarbeit nach sich ziehen und der Prozess beginnt von vorne. Ziel des gesamten Prozesses sind Anforderungen, die, den im Unterkapitel 1.2.4 beschriebenen Qualitätskriterien, bestmöglich genügen.[22]

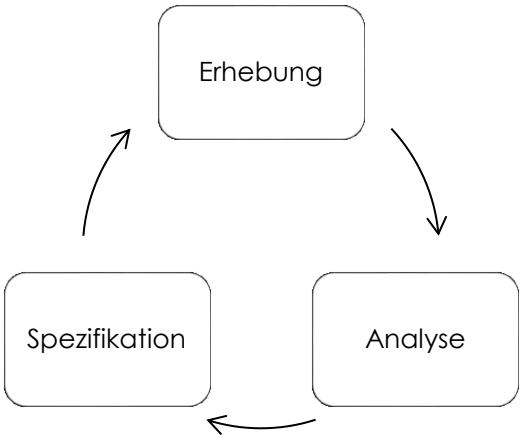


Abbildung 2: Zyklus der Anforderungserhebung, -analyse und -spezifikation[23]

1.2.2 Ebenen von Anforderungen

Anforderungen in der Ebene der „Benutzeranforderungen" beschreiben problemorientiert „Was" das zu erstellende System, aus Sicht der direkt damit arbeitenden Benutzer, leisten soll.[24] Oftmals sind diese Anforderungen anfangs vage, inkonsistent oder unvollständig, da der zukünftige Nutzer noch nicht weiß, wie er sein Problem formulieren soll. Im Zuge des Requirements-Engineerings soll daher iterativ eine Lösung modelliert werden, die den Problembereich bestmöglich adressiert.[25]

In der nächsten Ebene, den „Systemanforderungen", werden bereits spezifikationsorientiert Lösungen beschrieben. Neben der Beschreibung „Was" das zukünftige System leisten soll, wird auch das „Wie" konkretisiert. Allerdings ist diese Beschreibung der Merkmale und Eigenschaften des zu erstellenden Systems von der technischen Lösung im Rahmen der Implementierung zu unterscheiden.[26]

[22] vgl. (Wiegers & Beatty, 2013, S. 120)
[23] (Wiegers & Beatty, 2013, S. 120)
[24] vgl. (Goeken, 2006, S. 107)
[25] vgl. (Ebert, 2005, S. 40)
[26] vgl. (Savolainen, 2002), (Ebert, 2005, S. 41)

1.2.3 Arten von Anforderungen

Neben den genannten Ebenen werden Anforderungen häufig nach bestimmten Kriterien unterteilt. Meist erfolgt eine Teilung in funktionale und nicht-funktionale Anforderungen:[27]

- **Funktionale Anforderungen** definieren Funktionen oder Dienste, die das System seinen Nutzern (Personen oder anderen Systemen) zur Verfügung stellen soll, wie das System auf bestimmte Eingaben reagieren und sich in bestimmten Situationen verhalten soll. In manchen Fällen drücken funktionale Anforderungen auch explizit aus, was das System nicht tun soll.
- **Nicht-funktionale Anforderungen** definieren Beschränkungen der durch das System angebotenen Dienste oder Funktionen. Dies beinhaltet Zeitbeschränkungen, Beschränkungen des Entwicklungsprozesses und Einschränkungen durch Standards. Nicht-funktionale Anforderungen beziehen sich eher auf das Gesamtsystem als auf einzelne Systeme oder Dienste.

Für die Klassifizierung nicht-funktionaler Anforderungen gibt es keine einheitliche Definition. Sommerville beispielsweise kategorisiert die verschiedenen Arten der nicht-funktionalen Anforderungen folgendermaßen:[28]

- **Produktanforderungen** legen das Verhalten der Software fest oder schränken dieses ein. Dies betrifft beispielsweise Anforderungen an die Effizienz oder Zuverlässigkeitsanforderungen, die die akzeptable Fehlerquote festlegen, Informationssicherheitsanforderungen und Anforderungen bezüglich der Benutzbarkeit.
- **Unternehmensanforderungen** sind umfassende Systemanforderungen, die sich aus der Politik oder Arbeitsweise des Unternehmens des Kunden und des Entwicklers ergeben. Beispiele dafür sind Entwicklungsprozessanforderungen, die die Programmiersprache festlegen und bestimmen, welche Entwicklungsumgebung oder Prozessstandards eingesetzt werden.
- **Externe Anforderungen** beziehen sich auf alle Anforderungen, die sich aus Faktoren außerhalb des Systems und aus seinem Entwicklungsprozess ergeben. Dies beinhaltet beispielsweise gesetzliche oder ethische Anforderungen,

[27] vgl. (Sommerville I. , 2012, S. 116 ff.), (Ebert, 2005, S. 13)
[28] vgl. (Sommerville I. , 2012, S. 119)

die sicherstellen, dass das System von seinen Benutzern und der Öffentlichkeit akzeptiert wird.

Ein weiteres Beispiel für ein Kategorisierungsschema von nicht-funktionalen Anforderungen ist im Volere-Schema von James und Suzanne Robertson bzw. der Atlantic Systems Guild enthalten. Dieses Schema wurde vor vielen Jahren hauptsächlich für organisatorische Systeme und Software entwickelt und seine Kategorien seither ständig an den aktuellen Stand der Technik und der Projektrealität angepasst. Die nachfolgende Abbildung zeigt jenen Teil des Volere-Schemas (Kapitel 10 bis 17), in dem die acht Haupt- und die dazugehörigen Unterkategorien von nicht-funktionalen Anforderungen, aufgelistet werden.[29]

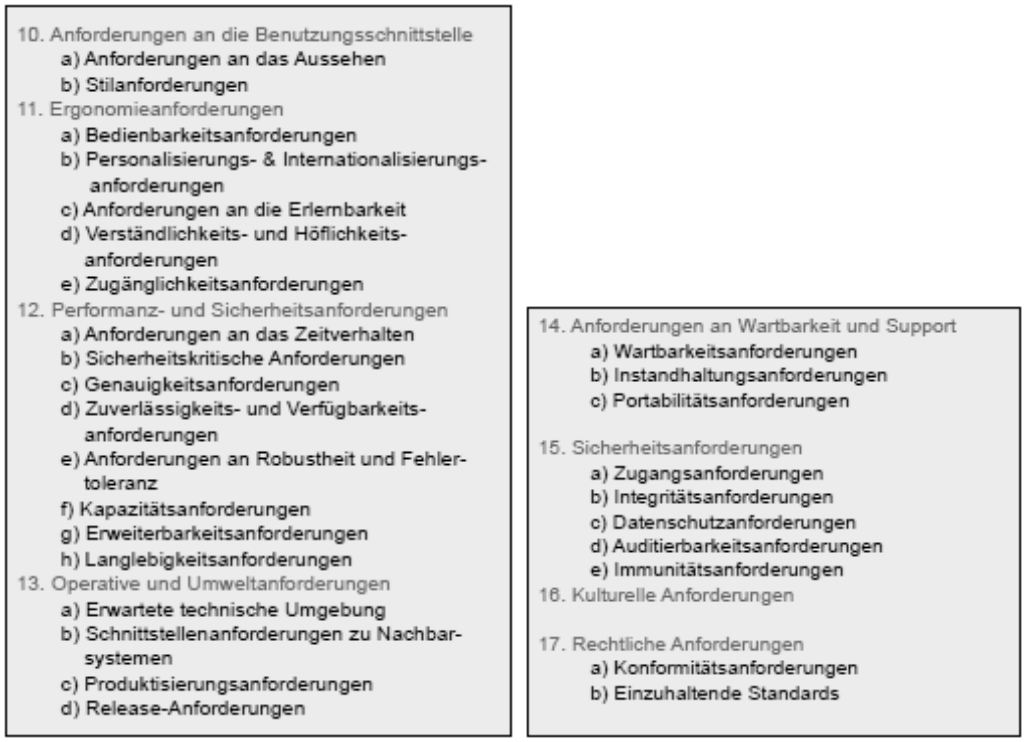


Abbildung 3: Kategorien nicht-funktionaler Anforderungen nach Volere[30]

Als drittes Beispiel für die Einteilung von nicht-funktionalen Anforderungen dient der ISO/IEC-25000-Standard „Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE)“ (Nachfolger der ISO/IEC-9126[31]). Diese Norm befasst sich ausschließlich mit Qualitätsanforderungen, also Anforderungen in Bezug auf die Güte des Produkts, des Prozesses oder der am Prozess beteiligten Personen und betrachtet damit einen wesentlich kleineren Bereich nicht-funktionaler Anforderungen

[29] vgl. (Robertson & Robertson, 2012)
[30] (Hruschka, 2014, S. 208)
[31] siehe (ISO/IEC 9126, 2001)

als das angeführte Volere-Schema. Der Standard unterscheidet die folgenden Haupt- und Unterkategorien:[32]

- **Änderbarkeit**
  Vorgaben in Bezug auf den Aufwand und die Auswirkungen, die mit spezifischen Änderungen am System oder dem Prozess verbunden sind. Dies betrifft die folgenden Unterkategorien: Analysierbarkeit, Modifizierbarkeit, Stabilität und Testbarkeit.

- **Benutzbarkeit**
  Anforderungen bezüglich der Benutzung des Produkts oder an die Durchführung von Tätigkeiten. Dies beschreibt einerseits den Aufwand, um das Produkt/den Prozess je nach Ausbildungsstufe zu erlernen, andererseits wird beschrieben inwiefern die Anforderungen konform zu Standards (z. B. gesetzte Rollendefinitionen oder Benutzungsoberflächenstandards im Unternehmen) sein müssen. Dies beinhaltet die folgenden Unterkategorien: Verständlichkeit, Erlernbarkeit, Bedienbarkeit, Attraktivität und Konformität.

- **Effizienz**
  Diese Anforderungen spezifizieren das Leistungsniveau, das Zeitverhalten und das Verbrauchsverhalten des Produkts oder des Prozesses. Submerkmale sind: Zeitverhalten, Verbrauchsverhalten und die Konformität.

- **Funktionalität**
  Diese Kategorie betrifft die Eigenschaften von Funktionalitäten in den folgenden Unterkategorien: Angemessenheit, Korrektheit, Interoperabilität, Sicherheit und Ordnungsmäßigkeit.

- **Übertragbarkeit**
  Unter diesem Abschnitt findet man Anforderungen an die Anpassungsfähigkeit des Produktes/des Prozesses an verschiedene neue Umgebungen (z. B. technische Plattformen, betriebliche Abläufe). Submerkmale sind: Anpassbarkeit, Installierbarkeit, Koexistenz, Austauschbarkeit und Konformität.

- **Zuverlässigkeit**
  Diese Kategorie befasst sich mit Eigenschaften eines Produkts oder Prozesses, das geforderte Leistungsniveau über einen bestimmten Zeitraum zu gewähr-

[32] vgl. (ISO/IEC 25000 Norm, 2005)

leisten. Dies betrifft: Reife, Fehlertoleranz, Robustheit, Wiederherstellbarkeit und Konformität.

Unabhängig vom gewählten Schema weisen viele Autoren[33] übereinstimmend darauf hin, dass die Beachtung nichtfunktionaler Anforderungen erfolgsentscheidend sein kann, dennoch werden diese, im Gegensatz zu den funktionalen Anforderungen, in der Praxis häufig vernachlässigt.[34] Aus diesem Grund verzichtet Pohl auf den Begriff der „nicht-funktionalen Anforderungen" mit der Begründung, dass es sich häufig um unzureichend verstandene und unzureichend spezifizierte funktionale Anforderungen handelt. Stattdessen wird in funktionale Anforderungen, qualitätsbezogene Anforderungen und Rahmenbedingungen gegliedert:[35]

- **Funktionale Anforderungen:** Pohl greift hier weitgehend auf die zuvor beschriebene Definition von Sommerville zurück.
- **Qualitätsanforderungen:** Diese definieren gewünschte Qualitätsmerkmale des geplanten Systems, z. B. die Performanz des Systems, die Zuverlässigkeit oder die gewünschte Ausfallsicherheit.
- **Rahmenbedingungen:** Eine Rahmenbedingung ist eine organisatorische oder technologische Anforderung, die die Art und Weise einschränkt, wie ein Produkt entwickelt wird. Rahmenbedingungen sind für die Projektbeteiligten schwer oder gar nicht veränderbar.

Nichtsdestotrotz hält sich die vorliegende Arbeit an die Unterteilung in funktionale und nicht-funktionale Anforderungen, da dies den, bei der KNAPP Systemintegration festgelegten Bezeichnungen, entspricht.

1.2.4 Qualitätskriterien für Anforderungen

Unabhängig von der Art der Anforderung gibt es eine ganze Reihe von Eigenschaften, die eine gut beschrieben Anforderung aufweisen muss. Vor allem am Beginn erfüllen Anforderungen jedoch meist die wenigsten Qualitätskriterien, die Entwicklung von qualitativ hochwertigen Anforderungen ist ein Prozess, der häufige Reflexion und Überarbeitung verlangt.[36] Die „perfekte" Anforderung per se existiert zwar nicht,

[33] vgl. zb. (Pohl, 2007, S. 14 ff.), (Ebert, 2005, S. 27), (Rupp & die SOPHISTen, 2014, S. 269), (Lawrence, Wiegers, & Ebert, 2001)
[34] vgl. (Ebert, 2005, S. 27)
[35] vgl. (Pohl, 2007, S. 14 ff.)
[36] vgl. (Alexander & Stevens, 2002, S. 67)

die folgende Liste von Charakteristiken bietet jedoch einen Anhaltspunkt zur Bewertung und lässt sich in ähnlicher Art und Weise in mehreren Quellen finden:[37]

- **Vollständig:** Damit eine Anforderung als vollständig beschrieben gilt, muss sie mehrere Kriterien erfüllen. Dazu zählt die Vollständigkeit hinsichtlich der minimalen Inhalte der Anforderung (Titel, Inhalt, Quelle, Erfüllungskriterium, Referenznummer etc.), der Beschreibung von Annahmen und Voraussetzungen, der Systemgrenzen, der Datenelemente und der zulässigen Wertebereiche.[38] Die Anforderung muss messbar sein und genug Informationen enthalten, um die Forderung der Stakeholder adäquat wiederzugeben. Zusätzlich müssen jene Bereiche, die bekanntermaßen unvollständig sind, als solche gekennzeichnet werden.[39]
- **Verständlich:** Dieses Kriterium kann durch mehrere Eigenschaften erfüllt werden. Einfache, direkte und aktiv formulierte Sätze sind in der Regel einfacher zu verstehen. Des Weiteren gilt es, ein einfaches Vokabular zu verwenden. Akronyme, Abkürzungen oder branchenspezifischer Jargon müssen in den Anforderungen geklärt, oder in einem Abkürzungsverzeichnis bzw. Glossar angeführt sein.[40]
- **Konsistent:** Die Aussagen der Anforderung dürfen sich nicht widersprechen oder konfliktär zueinander sein. Zusätzlich zur Konsistenz innerhalb einer Anforderung darf auch die gesamte Menge der Anforderungen nicht inkonsistent sein.[41]
- **Eindeutig:** Mehrdeutigkeit ist eines der häufigsten und schwierigsten Probleme im Zuge der Anforderungsbeschreibung. Das Ziel sind klare, eindeutige Anforderungen, ohne die Beschreibung unlesbar zu machen. Ein möglicher Hinweis auf eine mehrdeutige Anforderung ist die Verwendung des Wortes „oder".[42]
- **Korrekt bzw. notwendig:** Eine Anforderung ist korrekt, wenn sie den Forderungen der relevanten Stakeholder entspricht und die Anforderung vollständig im zukünftigen System umgesetzt sein muss. Erhöht eine Anforderung unnötiger-

[37] vgl. (Institute of Electric and Electronic Engineers, 2011), (Bijan, Yu, Stracener, & Woods, 2013), (Ebert, 2005, S. 127 ff.), (Pohl, 2007, S. 222 ff.), (Balzert, 2009, S. 475 f.), (Rupp & die SOPHISTen, 2014, S. 26 ff.)
[38] vgl. (Ebert, 2005, S. 128 f.)
[39] vgl. (Rupp & die SOPHISTen, 2014, S. 26)
[40] vgl. (Alexander & Stevens, 2002, S. 68)
[41] vgl. (Pohl, 2007, S. 223)
[42] vgl. (Alexander & Stevens, 2002, S. 69)

weise den Funktionsumfang, ist diese nicht notwendig und damit nicht korrekt.[43]

- **Verifizierbar:** Anforderungen müssen prüfbar beschrieben werden. Zulässige Eingaben, Ausgaben, Abnahmekriterien inkl. gültiger Wertebereiche müssen definiert sein. Wörter wie „schnell", „einfach", „ausreichend" oder „benutzerfreundlich" weisen auf nicht verifizierbare Anforderungen hin.[44]
- **Nachvollziehbar bzw. verfolgbar:** Eine Anforderung ist nachvollziehbar, wenn ihre Quelle, die Evolution der Anforderung und ihr Nutzen für das zukünftige System nachverfolgbar sind.[45] Vereinfacht wird dies durch eine eindeutige Anforderungsnummer, die während des gesamten Lebenszyklus der Anforderung unverändert bleibt.[46]
- **Unteilbar bzw. atomar:** Es sollten niemals mehrere Anforderungen in einer Anforderung enthalten sein. Häufig entsteht dies durch die Verschachtelung von Sätzen mit Konjunktionen wie beispielsweise „und", „oder" und „außerdem". Probleme können auftreten, wenn es an dem jeweiligen Leser liegt herauszufinden, welcher Teil gerade zutrifft.[47]
- **Technisch lösungsneutral:** Eine Anforderung beschreibt, was gefordert wird und nicht wie die Anforderung umgesetzt werden soll. Ziel der technischen Lösungsneutralität von Anforderungen ist es, keine technische Lösung von vornherein auszuschließen, solange keine Randbedingung existiert, die eine bestimmte Lösung vorschreibt.[48]

[43] vgl. (Balzert, 2009, S. 475)
[44] vgl. (Ebert, 2005, S. 125)
[45] vgl. (Pohl, 2007, S. 222)
[46] vgl. (Rupp & die SOPHISTen, 2014, S. 27)
[47] vgl. (Alexander & Stevens, 2002, S. 70)
[48] vgl. (Rupp & die SOPHISTen, 2014, S. 27)

1.3 Die Spezifikation

Nachdem im vorangegangenen Kapitel näher auf den Begriff der Anforderungen eingegangen wurde, soll nun die funktionale Spezifikation beschrieben werden, da sie als Gesamtdokument die Summe aller Anforderungen in entsprechender Qualität enthält und damit das Zieldokument des Requirements-Engineering-Prozesses darstellt.

1.3.1 Definition

Die funktionale Spezifikation detailliert sämtliche logistische[49] Anforderungen an die Anlage und Rahmenbedingungen im Hinblick auf die angestrebte technische Lösung. Die Spezifikation umfasst in vielen Fällen einen ersten grobgranularen Entwurf der Systemarchitektur und wichtiger Schnittstellen des Systems sowie eine Zuordnung der Anforderungen zu Architekturkomponenten.[50] Hinzu kommt die Beschreibung der Behandlung aller in den Prozessen möglicherweise auftretenden Störungen und Fehlerfälle.

Die Spezifikation bildet somit die Basis für zahlreiche Entwicklungsaktivitäten und dient über den gesamten Entwicklungsprozess hinweg als Referenzdokument.[51] Beispielweise ist es die Grundlage für die Anforderungen an die Entwicklung sowie der Erstellung der prozessbezogenen Testfälle. Es ist häufig wichtiger Vertragsbestandteil und Quelldokument für Abnahmekriterien, neben seiner Rolle im Änderungsmanagement, wo es oft verwendet wird, um Auswirkungen von Anpassungen zu analysieren.[52]

1.3.2 Qualitätskriterien für Spezifikationen

Es lässt sich klar erkennen, dass die Qualität der Spezifikation von zentraler Bedeutung für den Projekterfolg ist. Diese Qualität basiert auf den Qualitätseigenschaften der einzelnen Anforderungen sowie der übergreifenden Qualität des Gesamtdokuments. Somit gelten für die Spezifikation im Grunde genommen die gleichen Qualitätskriterien wie für jede einzelne Anforderung. Darüber hinaus lassen sich für das Gesamtdokument drei wesentliche, übergreifende Qualitätseigenschaften erkennen:[53]

[49] Anforderungen an Mechanik, Elektrik etc. sind in separaten Dokumenten spezifiziert
[50] vgl. (Pohl, 2007, S. 235)
[51] vgl. (Pohl, 2007, S. 235)
[52] vgl. (Hompel & Schmidt, 2008, S. 315)
[53] vgl. (Pohl, 2007, S. 237 ff.)

Vollständigkeit

Eine Spezifikation gilt als vollständig, wenn sie die folgenden Bestandteile enthält:[54]

- Alle Anforderungen zur Funktionalität, Performanz, Entwurfseinschränkungen und externen Schnittstellen sowie externe Anforderungen, die durch die Spezifikation beeinflusst werden.
- Vorgegebene Reaktionen des Systems auf alle denkbaren Arten von Eingabedaten und in allen denkbaren Situationen. Die Reaktion muss dabei sowohl für gültige, als auch für ungültige Daten spezifiziert sein. Die Daten können dabei sowohl von einem Benutzer als auch beispielsweise von einem anderen System stammen.
- Beschriftungen und Referenzen aller Abbildungen, Tabellen und Diagramme der Spezifikation sowie Definitionen aller Termini und Messgrößen.

Bekannte Lücken sollten mit dem Akronym „TBD" („To Be Defined") gekennzeichnet werden. Es muss auf jeden Fall die Ursache der Lücke angegeben werden, wann sie geschlossen sein muss und, falls bekannt, wie das TBD aufgelöst werden soll und wer dafür verantwortlich ist.

Konsistenz

Die Spezifikation ist konsistent, wenn jede einzelne Anforderung konsistent ist und wenn keine Konflikte zwischen den Anforderungen existieren. Prinzipiell lassen sich drei Arten von Inkonsistenzen unterscheiden:[55]

- Inkonsistente Beschreibung eines realen Objekts: Zumindest zwei Anforderungen beschreiben das gleiche Objekt mit unterschiedlicher Terminologie (z. B. „Eingabe Artikelnummer", „Eingabe SKU-Nummer").
- Inkonsistente Eigenschaften eines realen Objekts: Die, in zumindest zwei Anforderungen, spezifizierten Eigenschaften eines Objektes stehen zueinander in Konflikt (z. B. eine Anforderung beschreibt die Eingabe der Artikelnummer grundsätzlich über die Tastatur, eine andere Anforderung fordert, dass die Eingabe immer über einen Barcode-Scan erfolgt).

[54] vgl. (Institute of Electric and Electronic Engineers, 1998)
[55] vgl. (Institute of Electric and Electronic Engineers, 1998)

- Logische oder zeitliche Konflikte: Mindestens zwei Anforderungen stehen in einem logischen/zeitlichen Konflikt (z. B. eine Anforderung fordert, dass Funktion B immer Funktion A folgt, während eine andere Anforderung definiert, dass B vor A ausgeführt wird).

Änderungsfreundlichkeit und Lesbarkeit

Das Layout und die Struktur der Spezifikation gewährleisten Änderungsfreundlichkeit, wenn eine einfache, vollständige und konsistente Modifikation der Anforderungen möglich ist.[56] Die Anforderungen sollten so beschrieben sein, dass sie möglichst einfach vom Leser, ohne detailliertes technisches Wissen, erfasst werden können.[57]

Änderungsfreundlichkeit und Lesbarkeit können durch eine kohärente Struktur des Dokuments, eine eindeutige Identifizierbarkeit von Anforderungen, durch Vermeidung von unnötigen Redundanzen und einer atomaren Beschreibung von Anforderungen gewährleistet werden.[58]

[56] vgl. (Institute of Electric and Electronic Engineers, 1998)
[57] vgl. (Sommerville I. , 2012, S. 126)
[58] vgl. (Pohl, 2007, S. 238)

2 Dokumentation der Projektgrundlagen

Bevor die eigentlichen Anforderungen an das zukünftige System ermittelt werden können, müssen eine Reihe von Grundlagen des Requirements-Engineering-Prozesses geklärt und dokumentiert werden. Am Beginn der Feinplanung müssen demnach die Stakeholder und der Systemkontext analysiert und die grundlegenden Ziele des Projektes festgelegt werden. Diese ersten Schritte sind so offensichtlich, dass eine gründliche Ausführung dieser Tätigkeiten in vielen Projekten vernachlässigt wird.[59] Der Umfang dieser Anfangstätigkeiten umfasst in der Regel nicht mehr als1 bis 5 Prozent des Gesamtprojektes. Die Bearbeitung der drei genannten Themen ist kein großer Aufwand, dennoch ist sie so wichtig, dass kein Projekt ohne sie begonnen werden sollte, denn vergessene Stakeholder, nicht definierte Ziele und unklarer Systemkontext haben durchaus das Potential das Projekt scheitern zu lassen.[60] Deshalb soll in den nachfolgenden Kapiteln näher auf diese Themen eingegangen werden.

2.1 Stakeholderanalyse

Sinngemäß sind die Stakeholder eines Projektes alle Personen, die benötigte Ressourcen zur Verfügung stellen oder ein Risiko eingehen, indem sie finanzielle Mittel, Zeit oder Arbeit in das Projekt investieren, um die strategischen Ziele des Unternehmens zu verfolgen. Noch allgemeiner formuliert ist jeder ein Stakeholder, der Interesse am Projekt hat oder von ihm in irgendeiner Weise betroffen ist. Es kann sich dabei um natürliche, juristische oder abstrakte Personen, die eine Gruppe repräsentieren, handeln. Stakeholder haben untereinander häufig Interessenskonflikte, diese gilt es zu koordinieren, um ein für alle Seiten zufriedenstellendes Ergebnis zu erzielen.[61] Eine Unterstützung, um neben den offensichtlichen Stakeholdern auch die Anforderungen der weniger offensichtlichen Stakeholder früh genug berücksichtigen zu können, ist die Einteilung der Stakeholder in Kategorien. Als Beispiel dient die folgende Unterteilung:[62]

- **System-Stakeholder:** Diese Gruppe umfasst jeden, der direkt mit dem System oder den Ergebnissen des Systems arbeitet oder durch den Einsatz des Systems betroffen ist. Stakeholder dieser Gruppe sind beispielsweise Anwender, Endnutzer, Administratoren, Personen, die mit Berichten oder Daten des Sys-

[59] vgl. (Alexander & Stevens, 2002, S. 13)
[60] vgl. (Hruschka, 2014, S. 58)
[61] vgl. (Bourne, 2009, S. 30)
[62] vgl. (Leffingwell, 2011, S. 119 f.)

tems arbeiten, Entwickler oder Wartungspersonal. Diese Stakeholder sind die primäre Quelle für Anforderungen und müssen deshalb besonders intensiv analysiert werden.

- **Projekt-Stakeholder:** Zusätzlich zu den System-Stakeholdern müssen auch all jene Personen identifiziert werden, die ein grundlegendes persönliches Interesse am Projekt, im Rahmen dessen das System entwickelt wird, haben. Ein Projekt-Stakeholder ist jemand, der ein starkes Interesse am Budget und Terminplan des Projektes, oder an der Art und Weise, wie das System entwickelt wird hat, im Marketing, dem Verkauf, der Installation oder der Wartung des Gesamtsystems involviert ist. Zu dieser Gruppe gehört ein größerer Personenkreis, wie beispielsweise Geldgeber, Projektmanager, Führungskräfte, Personen aus dem Verkauf, Marketing, Kundenservice oder der Entwicklung.

Auch wenn viele der Stakeholder offensichtlich sind, bzw. durch Nachfragen oder Brainstorming mit dem Kunden festgestellt werden können, kann es sehr hilfreich sein sich beispielsweise folgende Fragen zu stellen, um eventuell noch weitere Stakeholder zu identifizieren:[63]

System-Stakeholder

- Wer wird das System direkt nutzen?
- Wer wird die Ergebnisse derer, die mit dem System arbeiten nutzen?
- Mit welchen anderen Systemen interagiert das System?
- Welche Schnittstellen muss das System bereitstellen?

Projekt-Stakeholder

- Wer weiß über den Umfang des Projektes Bescheid?
- Wer bestimmt das Budget und den Terminplan?
- Wer gilt als Schnittstelle zwischen den Teams und dem Kunden?
- Wer entscheidet wie und wann das System an den Kunden ausgeliefert wird?
- Wer kann das Projekt politisch unterstützen oder schädigen?
- Welche Partner sind von dem System abhängig?

[63] vgl. (Leffingwell, 2011, S. 123 f.), (Herrmann & Knauss, 2013, S. 138)

Es ist sehr empfehlenswert die Ergebnisse einer systematischen Analyse der Stakeholder von Beginn an schriftlich festzuhalten, beispielsweise in einer simplen Tabelle. Zumindest die folgenden Attribute sollten für jeden Stakeholder festgehalten werden:[64]

- **Kontaktdaten:** Name, Telefonnummer(n), E-Mail-Adresse etc.
- **Rolle:** Grobe Klassifizierung wie zum Beispiel Projekt- oder System-Stakeholder, oder genauere Rolle wie beispielsweise Projektmanager, Entwickler, Berater.
- **Beschreibung:** Die Beschreibung soll den Stakeholder näher charakterisieren. Ein Beispiel kann der Umstand sein, dass die Person in mehreren Fachgebieten kompetent ist.
- **Verfügbarkeit:** Angaben über Kurzarbeit, Urlaubstage, geplante Abwesenheiten, spezielle Feiertage oder darüber, wie viele Ressourcen der Stakeholder für das Projekt bereitstellen kann.
- **Begründung für die Auswahl:** Kurze und präzise Begründung für die Aufnahme in die Liste.
- **Kompetenz:** Fachgebiete, in denen der Stakeholder besonders kompetent ist. Damit ist schnell erkennbar, welche Stakeholder für eine bestimmte Aufgabe geeignet sind und welche Verantwortung übertragen werden kann.

Diese Liste kann jedoch noch beliebig erweitert werden. Beispielsweise könnten noch Informationen über die Entscheidungsbefugnis oder das persönliche Interesse am Projekt angefügt werden. Im Anschluss können die Stakeholder klassifiziert werden. Als Beispiel für eine solche Klassifikation dient das Stakeholder-Analysemodell des Projektmanagers Babou Srinivasan, das Stakeholder je nach Höhe von Einfluss und Motivation in vier verschiedene Gruppen einteilt (siehe Abbildung 4).[65] Die Klassifikation hilft bei der Pflege der Stakeholder indem die Bemühungen des Requirements Engineers zielgerichteter eingesetzt werden können. Die Klassifizierung sollte jedoch nicht öffentlich zugänglich gemacht werden, da sich die Stakeholder aufgrund ihrer Einteilung möglicherweise ungerecht behandelt fühlen könnten.[66]

[64] vgl. (Herrmann & Knauss, 2013, S. 30)
[65] vgl. (Srinivasan, 2008)
[66] vgl. (Rupp & die SOPHISTen, 2014, S. 82)

Abbildung 4: Stakeholder-Analysemodell nach Srinivasan[67]

Srinivasan empfiehlt die folgende Vorgehensweise für die jeweilige Gruppe von Stakeholdern:[68]

- **Hohe Motivation, hoher Einfluss:** Diese Stakeholder sollten so stark wie möglich in das Projekt eingebunden werden, denn sie haben nicht nur den Willen, sondern auch die Macht das Projekt zur allseitigen Zufriedenheit voranzutreiben.
- **Geringe Motivation, hoher Einfluss:** Es sollte genug Aufwand investiert werden, um diese Stakeholder zufriedenzustellen. Der Kontakt sollte gepflegt werden, allerdings sollten diese Stakeholder nicht mit zu vielen Informationen oder Details gelangweilt werden.
- **Hohe Motivation, geringer Einfluss:** Zu diesen Stakeholdern sollte eine gute Kommunikation gepflegt werden. Sie sollten stets über Entwicklungen im Projekt am Laufenden gehalten werden, denn sie können rechtzeitig auf das Entstehen gravierender Probleme hinweisen. Außerdem sind sie häufig Multiplikatoren, die trotz geringen Einflusses andere mit ihrer Begeisterung anstecken und eine positive Grundstimmung verbreiten.
- **Geringe Motivation, geringer Einfluss:** Diese Stakeholder sollten nicht vergessen werden, jedoch sollte nicht zu viel Aufwand in die Pflege investiert werden. Meist genügt es, die wichtigsten Informationen zur Verfügung zu stellen.

[67] (Srinivasan, 2008)
[68] vgl. (Srinivasan, 2008)

Um Stakeholder noch besser kennen zu lernen, können auch weiche Faktoren wie kulturelle Aspekte oder Vorbehalte und Motivation analysiert werden, auch wenn diese häufig nur schwer zu ergründen sind. In letzter Konsequenz erhält man durch die gründliche Stakeholderanalyse Gewissheit darüber, mit wem man es zu tun hat und legt eine notwendige Basis für die erfolgreiche Anforderungserhebung.[69]

2.2 Zieldefinition

Neben der, im vorherigen Kapitel vorgestellten, Stakeholderanalyse und der später beschriebenen Kontextanalyse sollten vor der Ermittlung der eigentlichen Anforderungen auch die Ziele des Systems aufgestellt werden.

2.2.1 Zielbegriff

Zielgerichtetes Arbeiten ist eine notwendige Voraussetzung für jegliche Art von systematischer Entwicklung. Nur wenn die Ziele bekannt sind, können diese auch erreicht werden.[70] Diese Erkenntnis wird auch in der folgenden Definition des Zielbegriffes widergegeben:

Unter einem Ziel versteht man die intentionale Beschreibung eines, von Stakeholdern gewünschten, charakteristischen Merkmals des zu entwickelnden Systems bzw. des zugehörigen Entwicklungsprojekts.[71] Im Unterschied zu den Anforderungen des Projekts sollten die Ziele stabil sein und sich nicht ändern.[72]

Die definierten Ziele dienen im weiteren Verlauf des Projektes vor allem dem Zweck, die funktionalen und nicht-funktionalen Anforderungen gegen die Ziele abzugleichen um sicherzustellen, dass es sich dabei um zielkonforme Anforderungen handelt.[73] Sind die Anforderungen nicht im Einklang mit den definierten Zielen, kann es sich dabei um eine irrelevante und damit überflüssige Anforderung handeln oder die Zieldefinition ist unvollständig.[74] Außerdem können dokumentierte Ziele helfen, Anforderungskonflikte zwischen verschiedenen Stakeholdern zu identifizieren und zu lösen. Das höhere Abstraktionsniveau von Zielen fördert die Auflösung von Konflikten, indem

[69] vgl. (Herrmann & Knauss, 2013, S. 32)
[70] vgl. (Ebert, 2005, S. 88)
[71] vgl. (Pohl, 2007, S. 91),
[72] vgl. (Hruschka, 2014, S. 32)
[73] vgl. (Balzert, 2009, S. 456 f.)
[74] vgl. (Pohl, 2007, S. 90)

nicht die konfliktäre Anforderung, sondern die jeweilige Intention der Stakeholder betrachtet wird.[75]

2.2.2 Zielbeschreibung

Nachdem die Stakeholder befragt und gemeinsam die Ziele erhoben wurden, müssen diese formuliert werden. Dabei ist äußerste Sorgfalt geboten, denn genauso wie falsche und fehlende Ziele ein Projekt gefährden, erschweren schlechte, zweideutige und falsch formulierte Ziele die Projektarbeit.[76]

Die Art, wie Ziele dokumentiert werden, reicht von lockerer Prosa bis zu stark formalisierten Beschreibungen. Exemplarisch werden an dieser Stelle drei Möglichkeiten vorgestellt: die natürlichsprachliche Formulierung, „Und-Oder-Bäume" als semiformaler Ansatz und Zielschablonen.

Natürlichsprachliche Formulierung

Eine prägnante und verständliche Formulierung von Zielen verbessert ihren Nutzen im Requirements-Engineering. Die folgenden neun Regeln sollen den Anwender bei der Formulierung von natürlichsprachlichen Zielen unterstützen.[77]

1. Kurz und prägnant formulieren
   Ziele sollten so kurz und prägnant wie möglich, ohne unnötige Füllwörter und Floskeln formuliert werden.
2. Aktivformulierung verwenden
   Ziele sollten in der Aktivformulierung beschrieben werden, da bei dieser Formulierungsform der Akteur benannt werden muss.
3. Überprüfbare Ziele definieren
   Wenn möglich, sollten die Ziele im späteren System überprüfbar sein.
4. Nicht überprüfbare Ziele verfeinern
   Für nicht überprüfbare Ziele sollten im weiteren Verlauf des Requirements-Engineerings überprüfbare Teilziele definiert werden.
5. Den Mehrwert des Ziels hervorheben
   Der angestrebte Mehrwert sollte möglichst präzise angegeben sein.

[75] vgl. (Nuseibeh, Kramer, & Finkelstein, 1994)
[76] vgl. (Rupp & die SOPHISTen, 2014, S. 83)
[77] Regeln 1-7: vgl. (Pohl, 2007, S. 99 ff.)

6. Das Ziel begründen
   Die Zieldefinition sollte auch eine kurze Begründung des Ziels beinhalten. Dies fördert die Diskussion und die Identifikation weiterer Ziele.
7. Keine Lösungsansätze angeben
   Lösungsansätze schränken den Lösungsraum zu früh ein, der maximal mögliche Lösungsraum sollte offen gelassen werden.
8. Einschränkende Rahmenbedingungen anführen[78]
   Rahmenbedingungen, die den Lösungsraum einschränken, müssen angeführt werden, damit die spätere Lösung nicht außerhalb des Lösungsraumes liegt.
9. Realistische Ziele formulieren
   Unrealistische, nicht erreichbare Ziele diskreditieren die gesamten Anforderungen und werden nicht ernst genommen oder als ärgerlich und frustrierend empfunden.

Diese Regeln können bereits bei der Ermittlung von Zielen berücksichtigt werden, Hauptaugenmerk liegt jedoch auf der Dokumentation und der regelmäßigen Prüfung der bereits dokumentierten Ziele auf ihre Regelkonformität.[79]

Und-Oder-Bäume

Üblicherweise wird im Zuge der Zielermittlung nicht nur ein Ziel, sondern eine Vielzahl an Zielen identifiziert. Eine sehr häufig angewendete Möglichkeit, diese Ziele in Zusammenhang zu bringen sind Und-Oder-Bäume. Mit Und-Oder-Bäumen können hierarchische Beziehungen zwischen einzelnen Teilzielen beschreiben werden, dabei wird die Dekomposition der Ziele im Baum von oben nach unten immer feiner.[80]

Die Und-Dekomposition bedingt, dass alle Teilziele erfüllt sein müssen, damit das Hauptziel erfüllt wird. Bei der Oder-Dekomposition muss lediglich eines der Teilziele erfüllt werden, um das übergeordnete Ziel zu erfüllen.[81] Die beiden Dekompositions-Arten sind in der folgenden Abbildung dargestellt. Natürlich können beide Dekompositions-Arten auch in einem Baum kombiniert werden.

[78] Regeln 8,9: vgl. (Rupp & die SOPHISTen, 2014, S. 84)
[79] vgl. (Pohl, 2007, S. 101 f.)
[80] vgl. (Rupp & die SOPHISTen, 2014, S. 180)
[81] vgl. (Rolland & Salinesi, 2005)

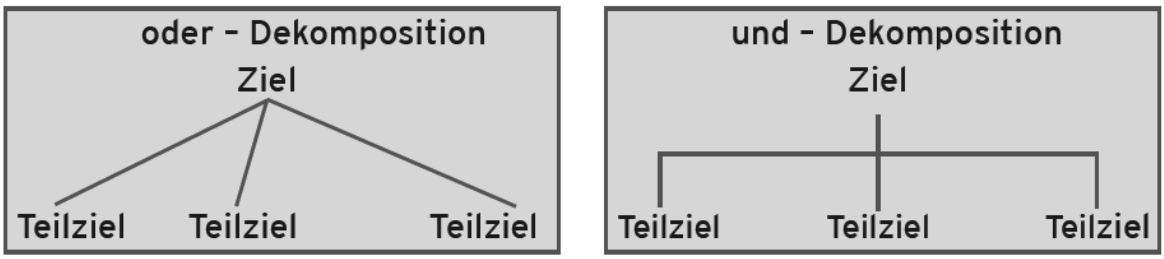


Abbildung 5: Modellierung von Und-Oder-Bäumen[82]

Zielschablonen

Eine sehr einfache, aber effektive Möglichkeit, Ziele zu dokumentieren, ist die Beschreibung anhand von Zielschablonen. Die Verwendung dieser strukturierten Dokumentationsform hat den Vorteil, dass genau definiert ist, welche Informationen für das jeweilige Ziel ausgefüllt werden müssen. Außerdem sind auf diese Weise dokumentierte Ziele ohne einführende Erklärung für jeden lesbar. Chris Rupp und die SOPHISTen empfehlen die folgenden fünf Kategorien für jedes Ziel:[83]

1. Ziel
   Welches Ziel soll durch das neue System erreicht werden?
2. Anforderungsquelle
   Wer oder was fordert die Erreichung dieses Ziels?
3. Auswirkung auf den Stakeholder
   Welche Arbeitsprozesse sind von der Zielerreichung betroffen?
4. Einschränkungen
   Welche Faktoren können eine Lösung einschränken?
5. Sonstiges
   Gibt es weitere Anmerkungen, die für das Verständnis des Ziels wichtig sind?

Hruschka reduziert die Anzahl der zu dokumentierenden Kategorien pro Ziel auf lediglich drei, die basierend auf den englischen Bezeichnungen als Akronym „PAM“ bekannt sind:[84]

1. Purpose – Ziel
   Kurze und prägnante Beschreibung des Ziels, das durch das System erreicht werden soll.

[82] (Rupp & die SOPHISTen, 2014, S. 180)
[83] vgl. (Rupp & die SOPHISTen, 2014, S. 84)
[84] vgl. (Hruschka, 2014, S. 32)

2. Advantage – Vorteil
   Wer oder was profitiert von diesem Ziel und in welcher Weise?
3. Measure – Metrik
   Mit welcher Metrik kann man nach Ende des Projektes feststellen, ob das Ziel erreicht wurde?

Die Ziele eines Projektes gelten als abstrakteste Form von dessen Anforderungen und Randbedingungen. Allerdings sollte ein Projekt wesentlich weniger Ziele als Anforderungen aufweisen. Im Extremfall kann die Anzahl auf maximal fünf Ziele beschränkt werden, um sicherzustellen, dass nur die wirklich grundlegenden Ziele identifiziert werden.[85] Die Definition, Überprüfung und Abstimmung der Ziele muss im Vorfeld der eigentlichen Projektabwicklung erfolgen, damit es weitgehende Gewissheit darüber gibt, dass die verschiedenen Stakeholder hinsichtlich der Erwartungen an das Projekt wirkliche Übereinkunft erzielt haben. Im weiteren Verlauf des Projektes ist es dann mitunter erfolgsentscheidend, dass alle Projektbeteiligten regelmäßig an die Ziele des Projektes erinnert werden, um den Hauptzweck des Projektes nicht aus den Augen zu verlieren.[86]

2.3 Kontextanalyse

Nachdem die am Projekt beteiligten Stakeholder analysiert und gemeinsam die Ziele des Systems definiert wurden, werden im nächsten Schritt der genaue Umfang und die Umgebung des Systems betrachtet.

2.3.1 System und Systemkontext

Im Wesentlichen behandelt die Kontextanalyse die Identifikation und Dokumentation des groben Systemumfangs[87], des Systemkontexts und der damit verbundenen Systemgrenzen, um genau jenen Teil zu identifizieren, der die Anforderungen an das zu entwickelnde System bestimmt. Diese Teilaufgabe des Requirements-Engineerings ist deshalb so wichtig, da hier der Leistungsumfang beschrieben und damit der Arbeits-

[85] vgl. (Hruschka, 2014, S. 33)
[86] vgl. (DeMarco, et al., 2007, S. 177 f.)
[87] engl.: Scope

auftrag definiert wird. [88] Partsch definiert in diesem Zusammenhang den Begriff „System“ wie folgt:

„Unter einem System versteht man eine Einheit von Komponenten, die nach einem bestimmten Kriterium von ihrer Umgebung abgegrenzt sind, über Beziehungen verknüpft sind oder miteinander interagieren und der Erreichung eines bestimmten Ziels dienen.“[89]

Der Systemumfang ist jener Bereich des Systems, der bei der Entwicklung bewusst gestaltet und beeinflusst werden kann. Rund um diesen Systemumfang befindet sich der Kontext, die relevante Umgebung des zu analysierenden Systems. Der Kontext ist relevant, da er das System betrifft und oft wesentlich beeinflusst, auch wenn dieser meist nicht aktiv gestaltet oder verändert werden kann. Außerhalb des Kontexts befindet sich die irrelevante Umgebung, die keinen Einfluss auf das zu entwickelnde System hat.[90] Die folgende Grafik zeigt, wie ein System und seine Umgebung abgegrenzt werden können:

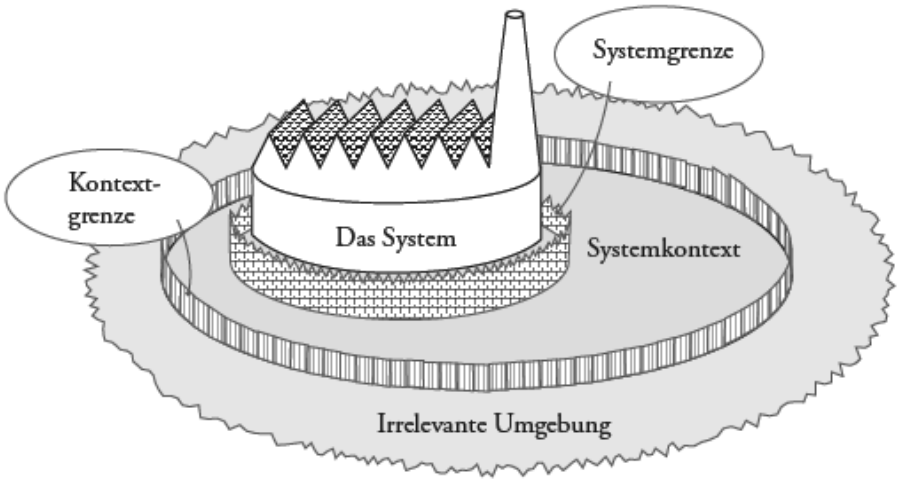


Abbildung 6: System, Systemkontext und Grenzen des Systems[91]

Für die erfolgreiche Systementwicklung ist das Verständnis des Systemkontexts essentiell. Um dieses Verständnis zu erreichen, müssen alle materiellen (z. B. Stakeholder, Dokumente, Nachbarsysteme) und immateriellen (z. B. Prozesse, Ereignisse) Objekte identifiziert werden, die eine Beziehung zu dem System haben. Um den Systemkontext exakt zu definieren, ist es außerdem notwendig, ihn in zwei Richtungen abzugrenzen. Die erste Abgrenzung erfolgt zum zu entwickelnden System hin (System-

[88] vgl. (Rupp & die SOPHISTen, 2014, S. 85)
[89] (Partsch, 2010, S. 23)
[90] vgl. (Hruschka, 2014, S. 42)
[91] (Rupp & die SOPHISTen, 2014, S. 86)

grenze), die andere nach außen zur irrelevanten Umgebung hin (Kontextgrenze). Besonders am Beginn des Requirements-Engineering-Prozesses kann diese Abgrenzung nicht immer eindeutig erfolgen. Oft sind einige oder mehrere Schnittstellen bzw. gewünschte Funktionen und Qualitäten des geplanten Systems unvollständig oder überhaupt nicht bekannt. Diese Unschärfe führt zu Grauzonen in der Kontextabgrenzung. Im Optimalfall werden diese Grauzonen beseitigt, indem sie mit den Stakeholdern besprochen und geklärt werden. Ist das nicht möglich, müssen Annahmen über die Kontextabgrenzung getroffen werden. Diese sollten explizit gekennzeichnet werden, um sie im Laufe der Zeit konkretisieren zu können. Grauzonen in der Systemabgrenzung müssen im Verlauf des Requirements-Engineerings aufgelöst werden, Grauzonen in der Kontextabgrenzung müssen nicht restlos aufgeklärt werden.[92]

2.3.2 Kontextvisualisierung

Ähnlich wie bei der Zieldefinition und der Stakeholderanalyse gibt es auch bei der Kontextanalyse zahlreiche Möglichkeiten zur Visualisierung. Neben der natürlichsprachlichen Beschreibung des Kontexts oder der Abbildung in Tabellen, ist die grafische Darstellung, beispielsweise in Form von Klassendiagrammen oder Komponentendiagrammen, möglich. Nachfolgend wird jedoch auf die zwei am häufigsten angewendeten Darstellungsformen kurz eingegangen: das Use-Case-Diagramm der UML[93] und das Kontextdiagramm der Strukturierten Analyse.[94]

Use-Case-Diagramm zur Kontextvisualisierung

Das Use-Case-Diagramm mit seinem Fokus auf das System und die Akteure (Personen und Nachbarsysteme) außerhalb des Systems eignet sich von Natur aus gut zur Kontextabgrenzung. Zu beachten gilt, dass nur System-Use-Cases dargestellt werden, wobei die Nachbarn und Schnittstellen durch Assoziationen mit den entsprechenden System-Use-Cases an denen sie beteiligt sind, verbunden werden. Bei diesen Diagrammen liegt der Fokus darauf, die groben Funktionalitäten des Systems voneinander abzugrenzen und zu zeigen, welche Akteure an den Funktionalitäten beteiligt sind. Besonders wenn Use-Cases auch im weiteren Verlauf des Projektes verwendet werden, eignen sich diese gut zur Kontextvisualisierung.[95] Im Detail wird auf Use-Cases im Zuge der Anforderungsspezifikation in Kapitel 4.1.2 eingegangen. Die fol-

[92] vgl. (Rupp & die SOPHISTen, 2014, S. 85 ff.)
[93] Unified Modeling Language
[94] vgl. (Rupp & die SOPHISTen, 2014, S. 181)
[95] vgl. (Hruschka, 2014, S. 55 f.)

gende Abbildung zeigt, in vereinfachter Form, die wichtigsten Anwendungsfälle eines Lageverwaltungssystems („KiSoft"). Jene Personen, die mit dem System interagieren sind als Strichmännchen dargestellt, externe Systeme in Form von Rechtecken.

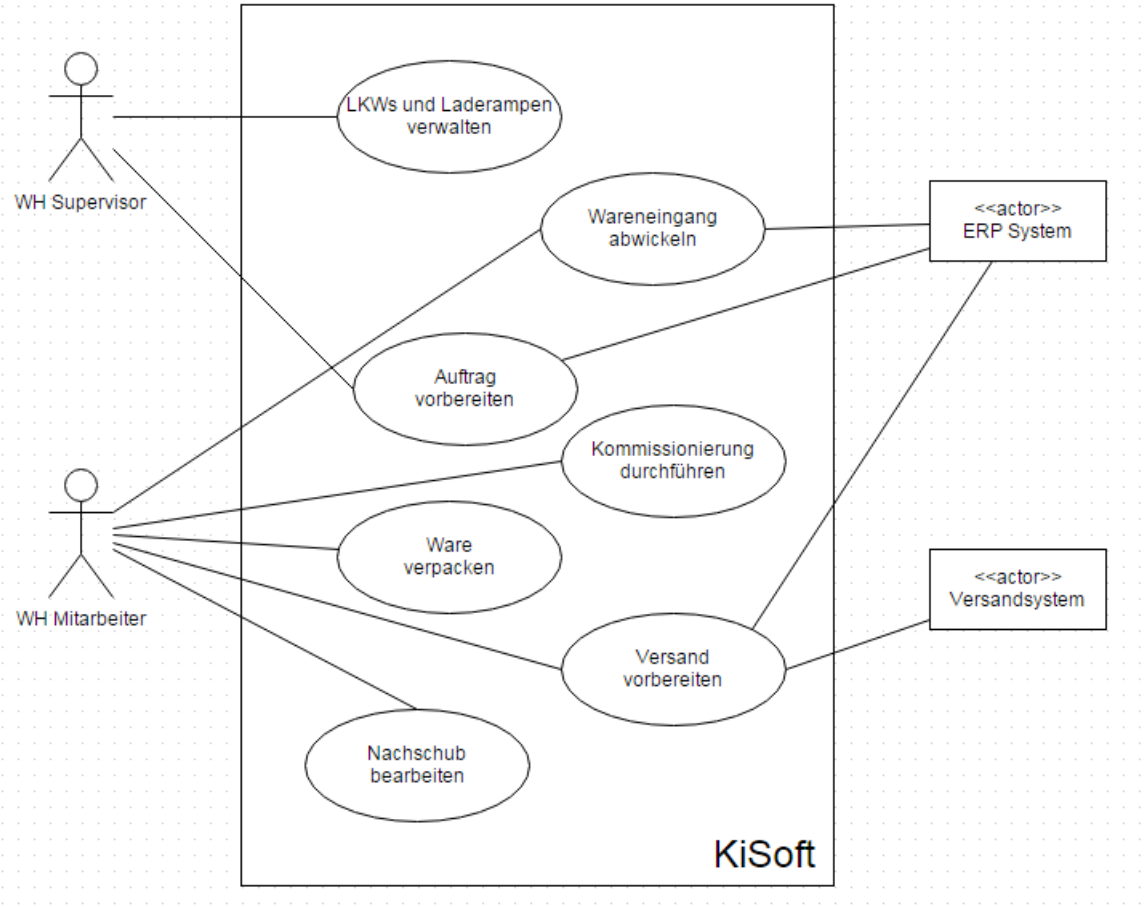


Abbildung 7: Use-Case-Diagramm eines Lagerverwaltungssystems[96]

Kontextdiagramm der „Strukturierten Analyse"

Der klassische Weg die Systemgrenzen und den Systemkontext zu beschreiben, ist das Kontextdiagramm der „Strukturierten Analyse" von Tom DeMarco. Im Prinzip handelt es sich dabei um ein Datenflussdiagramm, dass im Mittelpunkt das System ohne jede Zerlegung als Blackbox darstellt. Um das System herum werden die Nachbarsysteme (Menschen, andere IT-Systeme, Software, Sensoren etc.) angeordnet und die Schnittstellen inklusive sämtlicher ein- und ausgehender Datenflüsse explizit benannt. Der besondere Fokus des Kontextdiagrammes liegt auf dem Informationsfluss sowohl in das System als auch aus dem System heraus.[97] Die folgende Grafik zeigt exemplarisch in vereinfachter Form das Kontextdiagramm eines Bibliotheksys-

[96] eigene Darstellung
[97] vgl. (DeMarco, 1979, S. 75 f.)

tems. Im Zentrum steht dabei das System, außerhalb sind die Nachbarsysteme „Kunde", „Bibliothekar" und „Kundendatenbank" angeordnet. Die Datenflüsse sind durch Pfeile dargestellt, die Informationen, die über diese Schnittstellen ausgetauscht werden, sind am jeweiligen Pfeil angegeben.

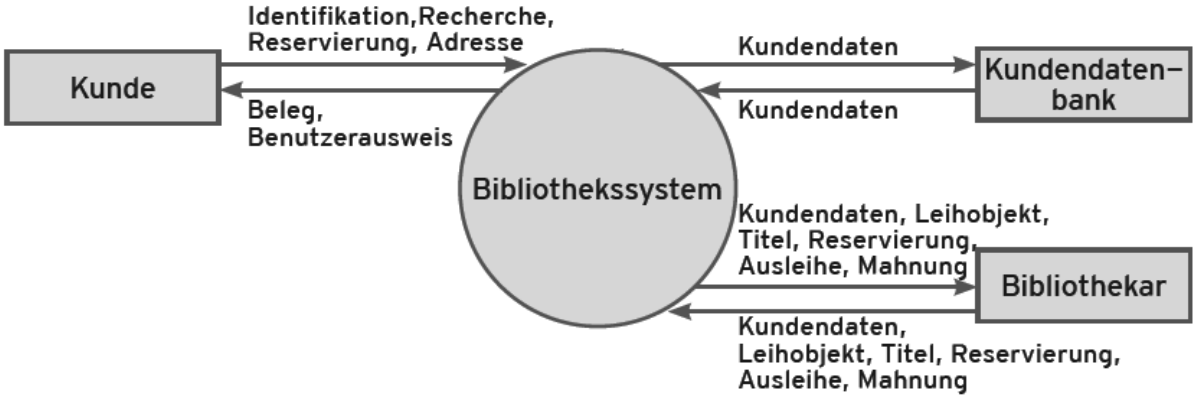


Abbildung 8: Kontextdiagramm eines Bibliotheksystems[98]

Als großer Vorteil des Kontextdiagrammes gilt die einfache Les- und Handhabbarkeit dieser Visualisierungsform. Diese wird jedoch gefährdet, wenn zu viel Information dargestellt werden soll. Die folgenden Elemente sollten nicht im Kontextdiagramm enthalten sein:[99]

- Alles, was innerhalb des Systems abläuft. Dies beinhaltet auch Schnittstellen zwischen den Komponenten des Systems.
- Eine Zerlegung des Gesamtsystems in Funktionen. Dies betrifft eine Zerlegung in Geschäftsprozesse oder in Use-Cases oder eine Zerlegung nach beliebigen anderen Kriterien.
- Alle Stakeholder, die nicht als Nachbarsystem entweder Informationen liefern oder Informationen bekommen.

Die angeführten Informationen werden mit anderen Notationsformen festgehalten, denn die Aufgabe des Kontextdiagramms besteht vorrangig aus der Darstellung der Schnittstellen zu Nachbarsystemen.[100]

[98] (Rupp & die SOPHISTen, 2014, S. 183)
[99] vgl. (Hruschka, 2014, S. 54)
[100] vgl. (Hruschka, 2014, S. 54)

3 Ermittlung von Anforderungen

Nachdem nun die Grundlagen des Requirements-Engineerings definiert wurden, kann nun mit der Anforderungsermittlung begonnen werden. In der Anforderungsermittlung steht der Wissenserwerb im Vordergrund. Im Wesentlichen dienen alle Aktivitäten dieser Phase den folgenden Zielen:[101]

- Die wahren Bedürfnisse der Stakeholder in Bezug auf das neue System ergründen,
- die Probleme und Beweggründe erkennen, die als Basis für die Entscheidung gelten, ein neues System einzuführen und
- tiefgreifendes Verständnis über die IST-Situation und ihren Kontext gewinnen.

Um diese Ziele zu erreichen, sind in der Anforderungsermittlung zahlreiche komplexe Einzelaktivitäten nötig. Diese lassen sich in drei Hauptaktivitäten gliedern:[102]

Abbildung 9: Hauptaktivitäten der Anforderungsermittlung[103]

1. **Identifikation relevanter Anforderungsquellen:** Neben den bekannten Anforderungsquellen (z. B. Altsysteme, existierende Dokumente) existieren zunächst unbekannte Anforderungsquellen. Diese gilt es systematisch zu ermitteln, da übersehene Anforderungsquellen zu unvollständigen Anforderungsdefinitionen und damit zu verminderter Systemqualität und erhöhten Aufwänden durch die spätere Integration der fehlenden Anforderungen führen.
2. **Gewinnung von existierenden Anforderungen:** Existierende Anforderungen sind in den Anforderungsquellen enthalten. Dazu zählen beispielsweise Vorstellungen der Stakeholder über das zukünftige System, definierte Schnittstellen, sowie Anforderungen, die in Dokumenten enthalten sind.

[101] vgl. (Lamsweerde, 2009, S. 61)
[102] vgl. (Pohl, 2007, S. 312 ff.)
[103] eigene Darstellung

3. **Entwicklung von innovativen Anforderungen:** Innovative Anforderungen sind in keiner Anforderungsquelle enthalten und müssen mit den verschiedenen Stakeholdern in einem kreativen und häufig iterativen Prozess erarbeitet werden.

Das Ziel aller Aktivitäten der Anforderungserhebung ist es, ein möglichst vollständiges inhaltliches Verständnis aller Anforderungen an das geplante System zu entwickeln. Es gilt neben den Funktionen auch alle gewünschten Eigenschaften, Einschränkungen und Erwartungen, die im Zusammenhang mit den Projektzielen stehen, zu erfassen, der Kunde steht dabei im Mittelpunkt.[104]

Zum Zweck der Anforderungsermittlung gibt es zahlreiche Techniken, die den Requirements-Engineer unterstützen können. Da diese Techniken jedoch bereits ausführlich in einer Bachelorarbeit[105], von der Autorin der vorliegenden Masterarbeit behandelt wurden, wird auf eine detaillierte Vorstellung der einzelnen Methoden verzichtet.

[104] vgl. (Ebert, 2005, S. 95)
[105] siehe (Grojer, 2014)

4 Spezifikation von Anforderungen

Den in Kapitel 1.1 vorgestellten Haupttätigkeiten des Requirements-Engineerings folgend befasst sich der folgende Abschnitt der vorliegenden Arbeit mit der zweiten Haupttätigkeit: dem Dokumentieren der Anforderungen. In dieser Phase des Requirements-Engineerings werden die Ergebnisse der Anforderungsermittlung möglichst präzise spezifiziert und ausformuliert, um am Ende dieser Phase die Erstversion einer Anforderungs-Spezifikation vorliegen zu haben. Wie alle Tätigkeiten des Requirements-Engineerings ist jedoch auch diese nicht zeitlich isoliert, sondern stets mit den anderen Phasen, speziell der Anforderungsermittlung, verknüpft.

Im nachfolgenden Kapitel werden zunächst sowohl natürlichsprachliche als auch graphische Techniken zur Anforderungsspezifikation beschrieben, bevor Kapitel 4.2 eine Empfehlungsmatrix zeigt, die bei der Wahl der richtigen Dokumentationstechnik unterstützen soll. Schlussendlich befasst sich Kapitel 4.3 mit nicht-funktionalen Anforderungen.

4.1 Dokumentationstechniken

In den nachfolgenden Kapiteln werden unterschiedliche Techniken, von natürlicher Sprache bis hin zu verschiedensten Diagrammarten, von informell bis semi-formal oder formal, vorgestellt. An dieser Stelle sei darauf hingewiesen, dass aufgrund der Vielzahl an möglichen Dokumentationsformen, eine Beschränkung auf die, in der Praxis am häufigsten eingesetzten Techniken, erfolgen muss.

4.1.1 Natürliche Sprache

Die offensichtlichste und in der Praxis am häufigsten angewendete Art und Weise die Anforderungen zu dokumentieren, ist natürliche Sprache. Diese Dokumentationsart hat zahlreiche Vorteile. Keiner der Stakeholder muss eine bestimmte Notation beherrschen oder verstehen, wichtig ist ausschließlich, dass die verwendete Sprache allen Beteiligten hinreichend geläufig ist. Natürliche Sprache ist vielseitig einsetzbar und für jeden, der diese Sprache spricht, gut verständlich.[106]

Allerdings birgt diese Art der Anforderungsdokumentation auch viele Tücken, die auf den ersten Blick schwer zu erkennen sind. Denn wenn Menschen unstrukturiert Anforderungen formulieren, finden bewusst und unbewusst Transformationen statt, die zu

[106] vgl. (Lamsweerde, 2009, S. 120)

unvollständigen, verzerrten oder sogar falschen Informationen führen. Wenn Menschen ein bestehendes oder zu entwickelndes System beschreiben, werden zwischen der zu beschreibende Realität und der formulierten Anforderung aufgrund von meist unterbewusst ablaufenden Transformationseffekten Divergenzen entstehen.[107] Diese Transformationseffekte resultieren aus unserer menschlichen Natur. Der Mensch wird durch seine soziale Prägung, sein Vorwissen und seine Erfahrungen in seiner Wahrnehmung beeinflusst, was dazu führt, dass jeder Mensch seine eigene Interpretation der Realität hat. Transformationen treten außerdem noch auf, wenn das persönliche Wissen in Sprache ausgedrückt wird. Demnach werden zwei Arten von Transformationsprozessen unterschieden:[108]

1. **Wahrnehmungstransformationen** (Fokussierungen) treten auf, weil jeder Mensch die Realität anders wahrnimmt und sich ein individuelles Bild davon macht.
2. **Darstellungstransformationen** (Vereinfachungen) treten auf, weil eine Wandlung erfolgt, sobald ein Mensch sein Wissen (das individuelle Bild) in Sprache ausdrückt.

Basierend auf diesen Erkenntnissen unterscheiden Brandler und Grinder im „Neurolinguistischen Programmieren“ drei Transformationskategorien: die Tilgung, die Generalisierung und die Verzerrung:

Tilgung ist ein Prozess, durch den wir unsere Aufmerksamkeit selektiv bestimmten Dimensionen unserer (im Moment möglichen) Erfahrungen zuwenden und andere Dimensionen ausschließen. Tilgung reduziert die Welt in Ausmaße, mit denen wir umgehen können.[109]

Generalisierung ist der Prozess, durch den Elemente oder Teile eines persönlichen Modells von der ursprünglichen Erfahrung abgelöst werden, um dann die gesamte Kategorie, von der diese Erfahrung ein Beispiel darstellt, zu verkörpern.[110] Durch zu starke Generalisierung entstehen globale Anforderungen an das System, die möglicherweise nur für einen Teilbereich des Systems richtig und sinnvoll sind. Sonder- und Fehlerfälle gehen hierbei häufig verloren.[111]

[107] vgl. (Rupp & die SOPHISTen, 2014, S. 125 f.)
[108] vgl. (Chomsky, 1965, S. 8 ff.)
[109] vgl. (Bandler & Grinder, 1975, S. 58 ff.)
[110] vgl. (Bandler & Grinder, 1975, S. 80 ff.)
[111] vgl. (Rupp & die SOPHISTen, 2014, S. 130)

Verzerrung ist der Prozess, der es uns ermöglicht, in unserer Erfahrung sensorischer Einzelheiten eine Umgestaltung vorzunehmen.[112] Verzerrung findet statt, wenn eine Situation mit Ausdrücken beschrieben wird, die nicht entsprechend dieser Situation sind. Dies soll helfen, neue Informationen leichter in die eigene Vorstellung integrieren zu können, indem Details nötigenfalls ein wenig verändert werden. Durch Verzerrungen können jedoch wichtige Informationen verlorengehen, es handelt sich dabei implizit um Tilgungsdefekte. Verzerrungen in Anforderungen sind häufig schwer zu vermeiden, da die Entscheidung, ob ein Sachverhalt korrekt oder verzerrt dargestellt wurde oder durch die eigene Wahrnehmung verzerrt wird, oft schwer zu treffen ist.[113]

Im alltäglichen Sprachgebrauch sind diese Transformationen sinnvoll und notwendig, bei der Beschreibung von Anforderungen stellen sie aber ein Problem dar, da möglicherweise essentielle Informationen verlorengehen. Die Wahrnehmungstransformation ist grundsätzlich nicht mehr auflösbar, sie kann höchstens durch die Befragung mehrerer Stakeholder zum gleichen Sachverhalt kompensiert werden. Die Darstellungstransformation lässt sich dagegen sehr gut auflösen, indem gezielt sprachliche Effekte (Tilgung, Generalisierung und Verzerrung) erkannt und hinterfragt werden, oder diese von Anfang an durch die Formulierung anhand von Richtlinien oder strikten syntaktischen Anforderungsmustern weitgehend vermieden werden.[114]

Als Beispiel für Richtlinien, die die Spezifikation in natürlicher Sprache unterstützen sollen, seien hier die, von Sommerville[115] beschriebenen, angeführt, die in ähnlicher Art und Weise häufig in der Literatur[116] zu finden sind:

- Alle Anforderungen sollten gemäß einem entwickelten Standardformat formuliert werden. Das Standardisieren macht Versäumnisse weniger wahrscheinlich und Anforderungen einfacher zu überprüfen. Die Anforderungen sollten in einfachen, direkten Sätzen im Aktiv formuliert werden.
- Eine einheitliche Ausdrucksweise signalisiert, ob es sich um verbindliche oder wünschenswerte Anforderungen handelt. Verbindliche Anforderungen sind Anforderungen, die das System unterstützen muss. Diese werden üblicherweise mit „muss“ oder „soll“ ausgedrückt. Wünschenswerte Anforderungen sind nicht unbedingt notwendig und werden mit „sollte“ formuliert.

[112] vgl. (Bandler & Grinder, 1975, S. 74 ff.)
[113] vgl. (Rupp & die SOPHISTen, 2014, S. 130)
[114] vgl. (Rupp & die SOPHISTen, 2014, S. 127 f.)
[115] vgl. (Sommerville I. , 2012, S. 128 f.)
[116] vgl. (Lamsweerde, 2009, S. 121 f.), (Wiegers & Beatty, 2013, S. 51 f.), (Sommerville & Sawyer, 1997, S. 147 f.), (Alexander & Stevens, 2002, S. 68 f.)

- Hervorhebungen im Text (fett, kursiv oder farbig) helfen, wichtige Teile der Anforderungen herauszustellen.
- Fachjargon, Abkürzungen und Akronyme sollen weitestgehend vermieden werden. Auch kann nicht davon ausgegangen werden, dass alle Leser die technische Sprache des Software-Engineerings verstehen. Werden Fachbegriffe verwendet, müssen diese in den Anforderungen oder dem Glossar erklärt werden.
- Zu jeder Anforderung sollten eine Begründung und eventuell auch ihre Quelle angeführt werden. Dieses Vorgehen ist besonders dann hilfreich, wenn Anforderungen verändert werden und entschieden werden muss, welche Änderungen nicht wünschenswert sind.

Angepasst an die jeweiligen Gegebenheiten, kann schon eine halbformale Spezifikation, wie die angeführten Richtlinien helfen, mehrdeutige, unvollständige und widersprüchliche Aussagen in den Anforderungen präventiv, im Rahmen der konstruktiven Qualitätssicherung zu vermeiden oder im Zuge der analytischen Qualitätssicherung (Validierung von Anforderungen) aufzudecken. Sollen häufige Fehler in natürlichsprachlichen Anforderungen von Beginn an vermieden werden, kann die Spezifikation mithilfe von stark formalisierten syntaktischen Anforderungsmustern (Anforderungsschablonen), welche nachfolgend kurz erläutert werden, erfolgen.[117]

Syntaktische Anforderungsmuster

Durch die Einhaltung von mehr oder weniger strikten Regeln bei der Anforderungsspezifikation lassen sich qualitativ hochwertige Anforderungen erhalten. Allerdings kann der Aufwand, jede Anforderung auf eine gewisse Anzahl von Regeln hin zu prüfen, hoch sein und in der Projektrealität oftmals nicht geleistet werden. Ein Mittel, um auf direktem Weg zu qualitativ hochwertigen Anforderungen zu gelangen, sind syntaktische Anforderungsmuster, auch Anforderungs- oder Satzschablonen genannt.[118]

Bei der Konstruktion der Anforderungen nach der Anforderungsschablone wird jeder Anforderung eine ähnliche Struktur aufgeprägt. Als Beispiel für eine solche Struktur wird nun die, in der folgenden Abbildung dargestellte, Schablone für funktionale An-

[117] vgl. (Pohl, 2007, S. 245)
[118] vgl. (Rupp & die SOPHISTen, 2014, S. 216 ff.)

forderungen von Chris Rupp und den SOPHISTen[119] beschrieben, die analog auch bei Pohl[120] zu finden ist.

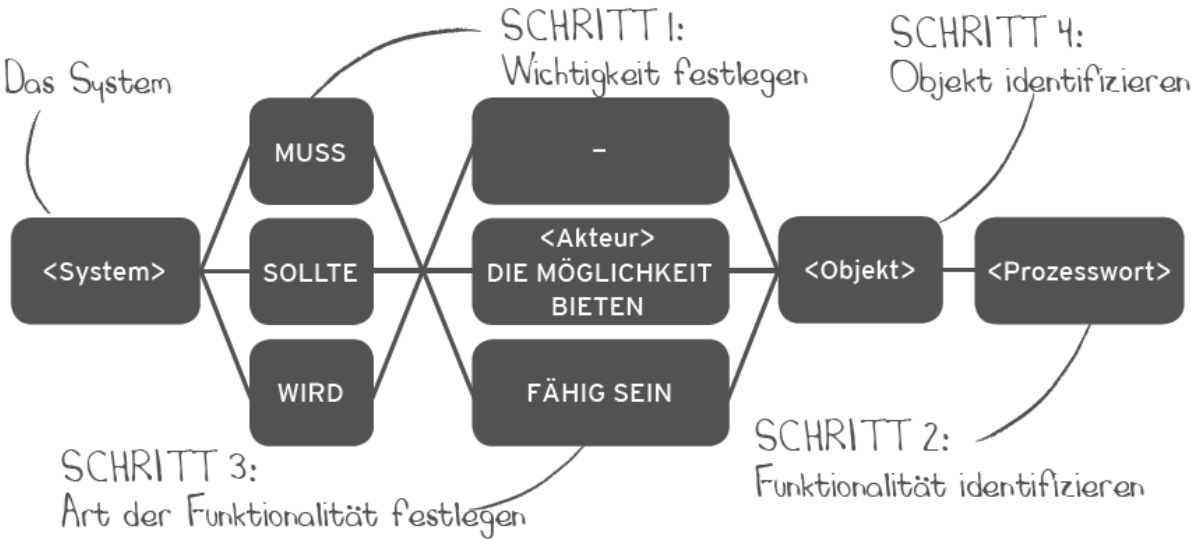


Abbildung 10: Anforderungsschablone für funktionale Anforderungen ohne Bedingung[121]

Im ersten Schritt der Anforderungsformulierung wird die Wichtigkeit bzw. rechtliche Verbindlichkeit der Anforderung festgelegt. Ist die Anforderung juristisch verbindlich, wird das Modalverb „muss" eingesetzt. Handelt es sich lediglich um eine juristisch unverbindliche Intention, wie beispielsweise einen Umsetzungsvorschlag, fällt die Wahl auf „sollte". Sollen mit der Anforderung nur zukünftige Entwicklungen angedeutet werden (z. B. kommende Erweiterungen), die es juristisch verbindlich zu berücksichtigen gilt, ist „wird" das richtige Modalverb. Danach wird die geforderte Funktionalität (z. B. drucken, speichern, übertragen, berechnen), die bei jeder Anforderung im Mittelpunkt steht, identifiziert. Diese Prozesswörter sollten ausschließlich durch Vollverben definiert sein, da sich die gesamte Anforderung an das Prozesswort bindet. Eng mit dem Prozesswort verknüpft ist Schritt 3, die Festlegung der Systemaktivität bzw. der Art der Funktionalität. Grundsätzlich lassen sich hierbei drei verschiedene Arten von Systemaktivitäten unterscheiden:

1. Selbstständige Systemaktivität: Das System startet den Prozess selbstständig und führt den Prozess selbstständig durch.
2. Benutzerinteraktion: Das System stellt dem Nutzer eine Interaktionsmöglichkeit zur Verfügung.

[119] vgl. (Rupp & die SOPHISTen, 2014, S. 220 ff.)
[120] siehe (Pohl, 2007, S. 245 f.)
[121] (Rupp & die SOPHISTen, 2014, S. 220)

3. Schnittstellenanforderung: Das System führt einen Prozess in Abhängigkeit von einem Dritten (z. B. Fremdsystem, kein Benutzer) aus, ist an sich passiv und wartet auf ein externes Ereignis.

Im vierten Schritt gilt es das Objekt zu identifizieren, für das die Funktionalität gefordert wird. Meist handelt es sich dabei um die nähere oder ergänzende Bestimmung des Prozesswortes.

Die folgende Anforderung dient als Beispiel für eine nach der beschriebenen Schablone spezifizierte System-Anforderung:

„*Das Lagerverwaltungssystem muss dem Verpacker die Möglichkeit bieten, die Packliste zu drucken.*"

Da es für Anforderungen typisch ist, dass die geforderte Funktionalität nur unter bestimmten zeitlichen oder logischen Bedingungen ausgeführt oder zur Verfügung gestellt wird, kann die Schablone in einem weiteren Schritt um eine vorangestellte Bedingung erweitert werden. Ferner können die Schablonen weiter detailliert oder separate Schablonen für nicht-funktionale Anforderungen, Begriffsdefinitionen oder Bedingungen definiert werden.[122]

Durch die Formulierung mithilfe der Schablonen können syntaktische Fehler in den Anforderungen ausgeschlossen werden. Semantische Fehler, die aufgrund der stark reduzierten, jedoch immer noch vorhandenen Gestaltungsfreiheit des Autors vorkommen können, werden aber auch durch die Verwendung von Anforderungsschablonen nicht restlos ausgeschlossen. Die semantische Präzision kann gesteigert werden, indem nur definierte Bestandteile (z. B. nur Prozesswörter aus einer definierten Liste von Vollverben) zur Formulierung herangezogen werden dürfen und die Anforderungen abschließend auf sprachliche Effekte überprüft werden.[123]

4.1.2 System-Use-Case-Diagramm

Üblicherweise werden die relevanten Geschäftsprozesse des Unternehmens bereits in der Anforderungsermittlung erhoben. Um diese Prozesse übersichtlich darzustellen, eignet sich das Business-Use-Case-Diagramm, in dem alle möglichen Interaktionen des Anwenders mit dem Gesamtsystem (z. B. Lager) erfasst und abgebildet werden. Im Zuge der Anforderungs-Dokumentation muss nun entschieden werden, welches

[122] vgl. (Rupp & die SOPHISTen, 2014, S. 220 ff.)
[123] vgl. (Rupp & die SOPHISTen, 2014, S. 220 ff.)

System zukünftig welche (Teil-) Prozesse unterstützen soll und dafür die Business-Use-Cases in spezifischere System-Use-Cases umgewandelt werden. So können aus einem Business-Use-Case durchaus mehrere Use-Cases auf Systemebene entstehen oder einzelne Business-Use-Cases nicht in System-Use-Cases umgewandelt werden, da die Entscheidung getroffen wird, diesen Prozess nicht mit dem zukünftigen System zu unterstützen.[124]

Erstmals wurden Use-Case-Diagramme von Ivar Jacobson[125] im Jahr 1992 vorgeschlagen, um die Funktionalität des zu entwickelnden Systems durch einfache Modelle analysieren und dokumentieren zu können. Mittlerweile sind Use-Case-Diagramme ein Standard der UML und gehören zu den am weitesten verbreiteten Modellierungstechniken, denn mit kaum einem anderen Notationsmittel können Sachverhalte schneller dokumentiert, strukturiert und diskutiert werden. Da funktionale Anforderungen in System-Use-Cases aus der Sicht der Benutzer beschrieben werden, sind sie auch für den Anwender gut verständlich und werden daher vor allem in frühen Stadien der Analyse eingesetzt.[126]

Ein System-Use-Case beschreibt eine typische Interaktion eines Anwenders mit einem System, die zu einem, vom Anwender gewünschten, Ereignis führt. In einem Use-Case-Diagramm werden die einzelnen Use-Cases, ihre Verbindungen untereinander und die Beteiligung der Nachbarsysteme mit den Use-Cases zusammengefasst.[127] Use-Case-Diagramme haben den Zweck, die potentiellen Fähigkeiten (Anwendungsfälle) des Systems zu finden und übersichtlich darzustellen. Die Abläufe innerhalb der Use-Cases können dann mittels natürlicher Sprache oder anderer Modellierungsmethoden (z. B. Aktivitätsdiagramm, Sequenzdiagramm, BPMN 2.0) beschrieben werden.[128] Die wichtigsten Modellierungskonstrukte von Use-Case-Diagrammen sind in der folgenden Abbildung dargestellt und werden nachfolgend kurz beschrieben.

[124] vgl. (Hruschka, 2014, S. 78)
[125] siehe (Jacobson, Christerson, & Jonsson, 1992)
[126] vgl. (Rupp & die SOPHISTen, 2014, S. 190)
[127] vgl. (Rupp & die SOPHISTen, 2013, S. 67)
[128] vgl. (Rupp, Queins, & die SOPHISTen, 2012, S. 239)

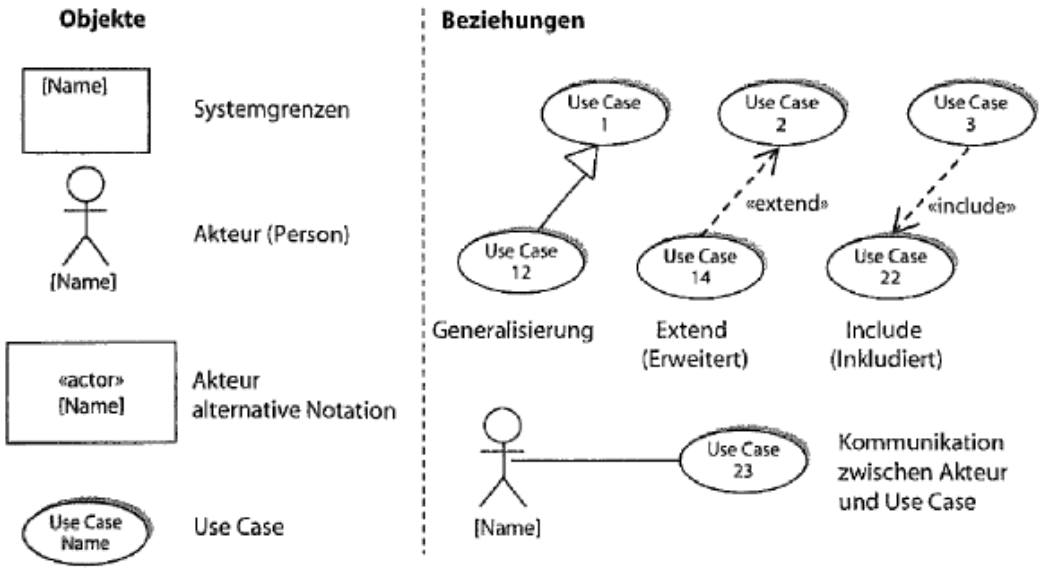


Abbildung 11: Modellierungskonstrukte von Use-Case-Diagrammen[129]

- **Akteur (System oder Personen):** Akteure liegen außerhalb der Systemgrenzen und stehen jeweils stellvertretend für Stakeholder oder Systeme, die mit dem betrachteten System interagieren. Üblicherweise wird der Akteur als Strichmännchen dargestellt, wenn es sich um eine Person handelt. Ist der Akteur ein System, so wird dieses durch ein beschriftetes Rechteck repräsentiert.[130] Handelt es sich beim auslösenden Akteur um ein Zeitereignis, wird dies meist durch eine Uhr dargestellt.[131]
- **Use-Cases:** Die für das System definierten Anwendungsfälle werden durch Ovale zusammen mit ihrem Namen dargestellt. Use-Cases beschreiben eine Menge von Aktionen die, schrittweise ausgeführt, ein spezielles Verhalten formen.[132]
- **Systemgrenzen:** Die Systemgrenzen separieren in einem Use-Case-Diagramm die Teile des Use-Cases, die zum System gehören von den Teilen (Personen und Systemen), die außerhalb der Systemgrenze liegen.[133]
- **Beziehungen zwischen Akteuren und Use-Cases:** Die Kommunikation zwischen einem Akteur und einem oder mehreren Use-Cases wird mit einer einfachen Kante modelliert.[134]

[129] (Pohl, 2007, S. 163)
[130] vgl. (Pohl, 2007, S. 162)
[131] vgl. (Rupp, Queins, & die SOPHISTen, 2012, S. 254)
[132] vgl. (Rumbaugh, Jacobson, & Booch, 2005, S. 78)
[133] vgl. (Pohl, 2007, S. 162)
[134] vgl. (Hruschka, 2014, S. 72)

- **Beziehungen zwischen Use-Cases:** Zwischen Use-Cases können drei verschiedene Arten von Assoziationen existieren:
  - **Generalisierung:** Diese Beziehung sagt aus, dass die spezialisierten Use-Cases, die im generalisierenden Use-Case enthaltenen Interaktionsschritte erben und diese gegebenenfalls um zusätzliche Schritte erweitern können.[135]
  - **Extend:** Eine Extend-Beziehung von einem Use-Case „A" zu einem Use-Case „B" sagt aus, dass die in „A" enthaltene Interaktionsfolge die in „B" definierte Interaktionsfolge an einem definierten Punkt (Extension Point) erweitert. Diese Erweiterung ist immer abhängig vom Eintreten einer bestimmten Bedingung.[136]
  - **Include:** Die Include-Beziehung von einem Use-Case „A" zu einem Use-Case „B" sagt aus, dass „A" die in „B" definierte Interaktionsabfolge inkludiert.

Die genannten Elemente, mit Ausnahme der Generalisierung, sind im nachfolgend abgebildeten Beispiel eines Use-Case-Diagramms exemplarisch dargestellt:

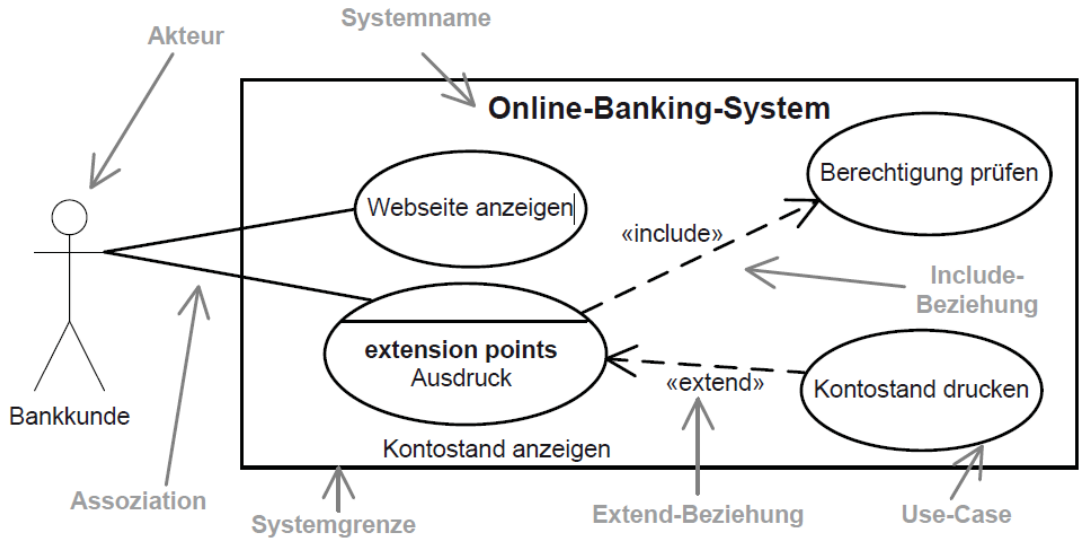


Abbildung 12: Use-Case-Diagramm eines Online-Banking-Systems[137]

Das modellierte Online-Banking-System besteht im Prinzip aus lediglich zwei Anwendungsfällen, auf die der Bankkunde direkten Einfluss hat: dem Anzeigen der Website und dem Anzeigen des Kontostandes. Die „Include"-Beziehung zeigt an, dass immer, wenn der Kontostand angezeigt wird, auch der Use-Case „Berechtigung prüfen"

[135] vgl. (Rumbaugh, Jacobson, & Booch, 2005, S. 79)
[136] vgl. (Pohl, 2007, S. 162)
[137] (Rupp, Queins, & die SOPHISTen, 2012, S. 243)

aktiviert wird. Wird im Zuge des Anzeigens des Kontostandes ein „Ausdruck“ (Extension point) ausgelöst, wird der Use-Case „Kontostand drucken“ aktiviert.

Der große Vorteil des Use-Case-Diagrammes besteht in seiner Verständlichkeit und der Anschaulichkeit für den Benutzer. Dies macht das Diagramm vor allem im Zuge der Anforderungsermittlung und für die Nutzung als Übersicht über die Funktionen des zu entwickelnden Systems wertvoll. Sehr gut können Use-Cases auch für die Gliederung von Anforderungen verwendet werden. Allerdings können Redundanzen oder Inkonsistenzen in den Diagrammen nur schwer vermieden werden, da einzelne Sachverhalte oft in mehreren Use-Cases eine Rolle spielen. Generell haben Use-Case-Diagramme den Nachteil, dass nicht-funktionale Anforderungen nicht direkt dargestellt werden können.[138]

4.1.3 Aktivitätsdiagramm

Aktivitätsdiagramme zählen zu den in der Praxis am häufigsten angewendeten Diagrammen, da sie sich sehr gut eignen, um Abläufe jeglicher Art und deren Regeln darzustellen. Sie können in jeder Phase des Projektes angewendet werden, da ihr Detaillierungsgrad stark variabel ist. So können beispielsweise komplette Geschäftsprozesse übersichtlich dargestellt oder aber einzelne Funktionen detailliert aufgezeichnet werden. Aktivitätsdiagramme eignen sich besonders gut, um komplexe Verläufe unter Berücksichtigung von Nebenläufigkeiten und alternativen Entscheidungswegen zu modellieren. Die zentrale Frage, die das Aktivitätsdiagramm beantwortet lautet: „Wie realisiert das System ein bestimmtes Verhalten?“[139]

Als Beispiel für ein Aktivitätsdiagramm dient der, in der folgenden Abbildung dargestellte, Prozess einer Bibliothek. Es handelt sich dabei um die Verleihung eines Leihobjektes:

[138] vgl. (Rupp & die SOPHISTen, 2014, S. 192)
[139] vgl. (Rupp, Queins, & die SOPHISTen, 2012, S. 263 ff.)

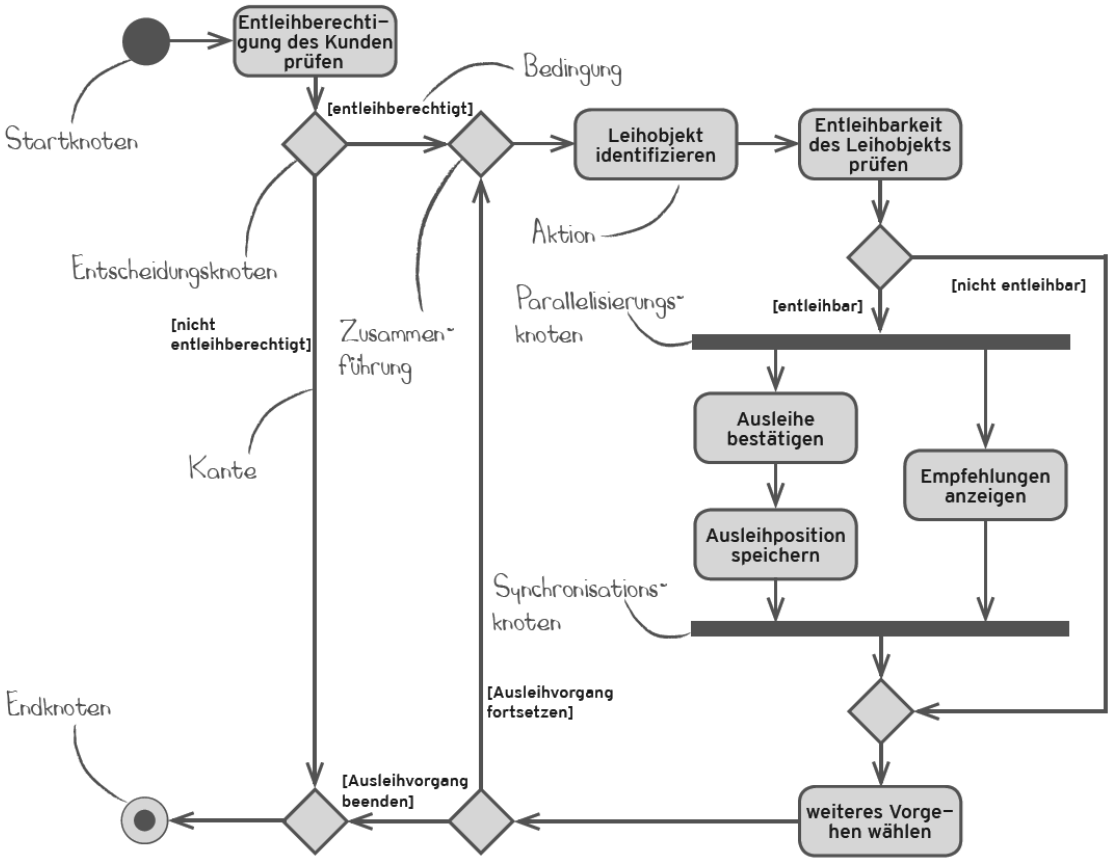


Abbildung 13: Aktivitätsdiagramm „Leihobjekt verleihen“[140]

Ein Aktivitätsdiagramm besteht im Wesentlichen aus den folgenden, auch im abgebildeten Diagramm, enthaltenen Elementen:[141]

- **eine oder mehrere Aktivitäten**
  Im abgebildeten Diagramm wird lediglich eine Aktivität behandelt: „Leihobjekt verleihen“.

- **Aktionen**
  Eine Aktion wird immer durch ein Rechteck mit abgerundeten Ecken dargestellt. Dieses wird mit einem Namen oder einem ausführlichen Text für die Aktion versehen. Das dargestellte Beispiel enthält unter anderen die Aktionen „Entleihberechtigung des Kunden prüfen“ und „Leihobjekt identifizieren“. Eine Aktion steht für den Aufruf eines Verhaltens oder die Bearbeitung von Daten, die innerhalb der jeweiligen Aktivität nicht weiter zerlegt wird. Aktivitäten sind die zentralen Elemente des Aktivitätsdiagramms, alle anderen Elemente existieren nur zu dem Zweck, den Ablauf und die Kontrolle der aufeinander folgenden Aktionen zu

[140] (Rupp & die SOPHISTen, 2014, S. 196)
[141] (Rupp, Queins, & die SOPHISTen, 2012, S. 264 ff.)

steuern. Aktionen sind über gerichtete Kanten mit anderen Elementen des Diagramms verbunden.

- **Objektknoten**
  Ein Objektknoten wird durch ein Rechteck dargestellt und repräsentiert innerhalb einer Aktivität Ausprägungen eines bestimmten Typs. In den meisten Fällen handelt es sich dabei um primitive Werte oder Objekte von Klassen, es können damit aber auch Variablen, Konstanten oder Speicher jeglicher Art modelliert werden. Im Bibliotheken-Beispiel wäre ein „Leihobjekt" ein Objektknoten.

- **Kontrollelemente zur Ablaufsteuerung**
  In jedem Aktivitätsdiagramm finden sich mehrere Kontrollelemente (Knoten), deren Aufgabe es ist, den Ablauf der Aktivität zu steuern. Zu dieser Art von Elementen zählen Start-, Verzweigungs-, Verbindungs-, Parallelisierungs-, Synchronisations- und Endknoten. Alle genannten Knotenarten finden sich auch im dargestellten Beispiel. Mit dem Startknoten beginnt die Aktivität, der erste Verzweigungsknoten bestimmt die nächste Aktivität, abhängig vom Wert der Bedingung „entleihberechtigt". Der erste Verbindungsknoten des Diagramms bewirkt, dass die Aktion „Leihobjekt identifizieren" ausgeführt wird, wenn eine Entleihberechtigung vorliegt und auch wenn der Entleihvorgang fortgesetzt wird. Der Parallelisierungsknoten zeigt, dass wenn das Leihobjekt entleihbar ist, sowohl die Ausleihe bestätigt wird, als auch Empfehlungen angezeigt werden. Sind alle Parallelisierungsaktionen abgearbeitet, werden die Ablaufstränge über den Synchronisationsknoten wieder zusammengeführt. Der Endknoten markiert schließlich das Ende der Aktivität.

- **verbindende Kanten**
  Kanten sind gerichtete Verbindungen zwischen den übrigen Elementen des Ablaufdiagramms. Kanten können mit Bedingungen versehen oder gewichtet werden. Beispiele für Bedingungen sind die booleschen Variablen „entleihberechtigt" oder „entleihbar".

Aktivitätsdiagramme zählen zu den leicht erlern- und lesbaren Diagrammtypen. Sie sind vielseitig einsetzbar und eignen sich gut, um grobe wie auch detaillierte Abläufe darzustellen. Allerdings zeigt das dargestellte Beispiel nur einen kleinen Teil der Mo-

dellierungsmöglichkeiten und bei Einsatz der weniger geläufigen Elemente können auch Aktivitätsdiagramme für manche Stakeholder schnell unverständlich werden.[142]

4.1.4 Business Process Model and Notation 2.0

Eine moderne Alternative zur Prozessdarstellung ist die Abbildung mithilfe der Business Process Model and Notation. Die erste Version dieser Modellierungsnotation (BPMN 1.0) wurde 2004 von der Business Process Management Initiative vorgestellt mit dem Ziel, eine Darstellungsart einzuführen, die leicht verständlich ist und eine grobe bis detaillierte Darstellung von Prozessen ermöglicht. 2011 wurde von der Object Management Group (OMG) die nächste Generation der BPMN vorgestellt. BPMN füllt mithilfe einer standardisierten Notation die Lücke zwischen Business Process Design und der Implementierung. Die BPMN definiert ein Business Process Diagram (BPD), das auf die Darstellung eines graphischen Modells eines Geschäftsfalles zugeschnitten ist. Ein Business Process Diagram besteht aus einem Set von graphischen Elementen, die optisch den bekannten Elementen aus Ereignisgesteuerten Prozessketten ähneln. Als Ganzes listet die BPMN eine große Anzahl von Elementen, die zur Beschreibung eines Geschäftsfalles verwendet werden können. Die Elemente sind in vier Basiskategorien eingeteilt, die wiederum jeweils Elemente enthalten:[143]

- **Flussobjekte:** Diese sind die Knoten in den Geschäftsprozessdiagrammen (Aktivität, Gateway und Ereignis).
- **Verbindende Objekte:** Die verbindenden Kanten (Sequenzfluss, Nachrichtenfluss und Assoziation) in den Geschäftsprozessdiagrammen setzen die Flussobjekte in Beziehung zueinander.
- **Pools und Swimlanes:** Swimlanes sind die Verantwortlichen für die Durchführung der Flussobjekte, jede Swimlane liegt innerhalb einer übergeordneten Instanz, den Pools. Pools sind die Grenzen des Prozesses.
- **Artefakte:** Dies sind weitere Elemente wie Datenobjekte oder Anmerkungen zur weiteren Dokumentation.

In der BPMN können Ereignisse drei Informationen tragen. Erstens eine Aussage wann (Start, Zwischen, Ende) sie in einem Prozess auftreten, zweitens, ob das Ereignis auf etwas Bestimmtes wartet oder etwas auslöst und drittens, um welchen inhaltlichen

[142] vgl. (Rupp & die SOPHISTen, 2014, S. 197)
[143] vgl. (White, 2004)

Typ (Nachricht, Fehler, Zeit, Bedingung, etc.) es sich handelt. Zunächst sollte nur der Normalablauf modelliert werden, danach können Schritt für Schritt Alternativen und Ausnahmen hinzugefügt werden. Außerdem sollte das BPD am Beginn möglichst abstrakt modelliert werden, indem Regeln, wie beispielsweise „modelliere den Prozess in maximal fünf Schritten" gesetzt werden. Erst danach werden die Details eingearbeitet.[144] Im Folgenden ist beispielhaft ein BPD dargestellt, es handelt sich um das Verleihen eines Leihobjektes in einer Bibliothek:

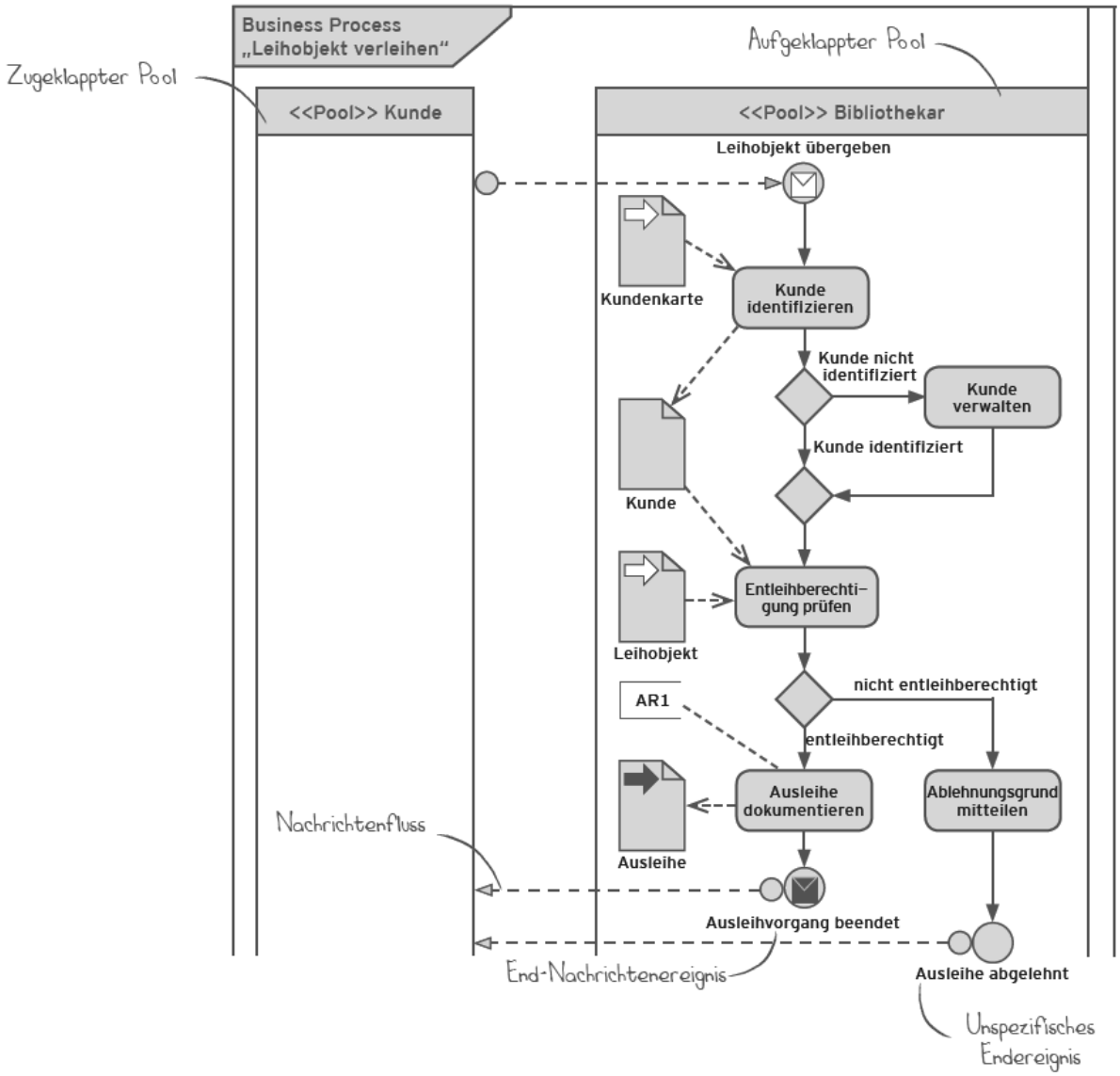


Abbildung 14: BPMN-Diagramm „Leihobjekt verleihen"[145]

Die Verwendung von Business Process Diagrammen in Workshops ermöglicht die aktive Einbeziehung aller Teilnehmer in die Gestaltung der Prozesse. Die BPMN Symbolik

[144] vgl. (Freund & Rücker, 2014, S. 171 f.)
[145] (Rupp & die SOPHISTen, 2014, S. 176)

kann sehr schnell und einfach erklärt werden und die klare Symbolik erleichtert eine gemeinsame, dynamische Prozessdarstellung.[146]

4.1.5 Sequenzdiagramm

Sequenzdiagramme dienen vorrangig dazu, Interaktionen zwischen beliebigen Systemen bzw. zwischen Systemen und ihrem Kontext zu modellieren. Mithilfe dieses Diagrammtyps der UML lassen sich durch die Notationen von Lebenslinien das zu spezifizierende System, sowie externe Systeme oder Personen als Kommunikationspartner darstellen. Darüber hinaus liegt die große Stärke dieses Diagramms in der Modellierung von zeitlichen Reihenfolgen des Informationsaustauschs zwischen diesen Kommunikationspartnern.[147] Grundsätzlich kann das Sequenzdiagramm in jeder Projektphase eingesetzt werden, es eignet sich beispielsweise dafür Use-Cases zu verfeinern und unter Interaktionsgesichtspunkten zu betrachten oder um Szenarien formaler darzustellen. Meist werden Sequenzdiagramme aber für detaillierte Anforderungen verwendet.[148] Die wesentlichen Elemente des Sequenzdiagrammes werden nun anhand des nachfolgend abgebildeten Beispiels erläutert. Es handelt sich bei diesem Prozess um die Rücknahme eines Leihobjektes in einer Bibliothek.

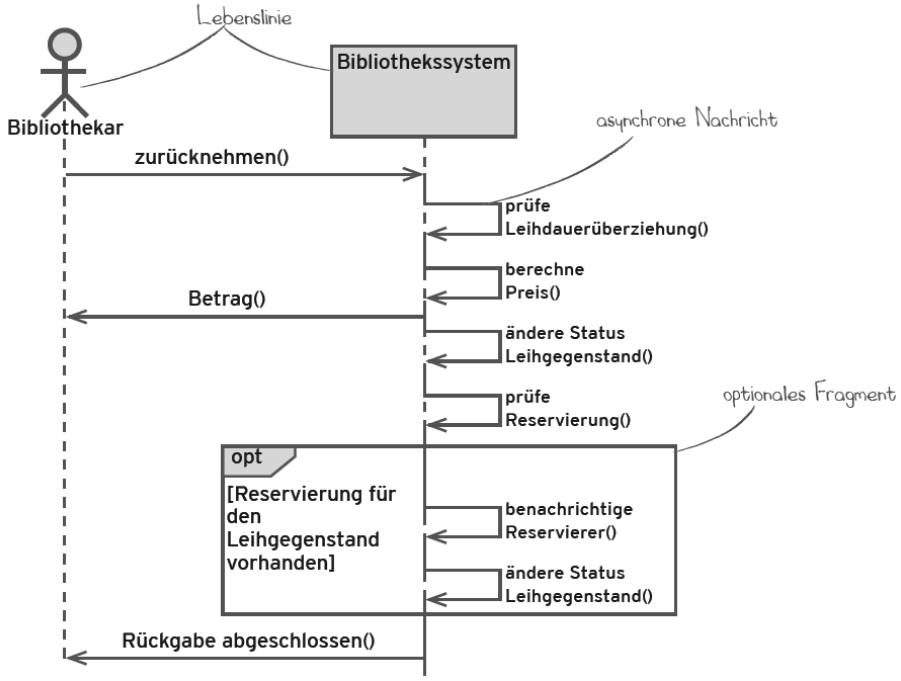


Abbildung 15: Sequenzdiagramm „Leihobjekt zurücknehmen"[149]

[146] vgl. (Niebisch, 2013, S. 88)
[147] vgl. (Pohl, 2007, S. 156 ff.)
[148] vgl. (Rupp & die SOPHISTen, 2014, S. 197)
[149] (Rupp & die SOPHISTen, 2014, S. 199)

Das Sequenzdiagramm zeigt Interaktionen in einem zweidimensionalen Modell. Die vertikale Dimension entspricht der Zeitachse, wobei das Diagramm von oben nach unten zu lesen ist. Die horizontale Dimension zeigt die Kommunikationspartner und deren Lebenslinien. Die beteiligten Kommunikationspartner sind in diesem Beispiel der Bibliothekar, dargestellt als Strichmännchen, und das System, dargestellt als Rechteck. Die Interaktionen zwischen den Kommunikationspartnern sind im Diagramm als Pfeile dargestellt. Mittels synchroner Kommunikation, die als Pfeil mit ausgefüllter Spitze dargestellt wird, werden Informationen übertragen, bei denen der Sender auf eine Antwort warten muss, bevor er weitere Aktionen vornehmen oder Nachrichten senden kann. Asynchrone Nachrichten (Pfeile mit offener Spitze) warten nicht auf eine Antwort. Im abgebildeten Beispiel gibt der Bibliothekar in das System ein, dass ein Leihobjekt zurückgegeben wurde. Er wartet nicht auf eine Bestätigung des Systems, dass die Bearbeitung durchgeführt wurde, es handelt sich bei dieser Kommunikation um eine asynchrone Nachricht. Mittels Fragmenten können Alternativen oder optionale Abläufe modelliert werden. Im angegebenen Beispiel wird geprüft, ob das zurückgegebene Leihobjekt reserviert ist oder nicht. Nur falls es reserviert ist, wird der optionale Teil durchlaufen.[150]

Sequenzdiagramme eignen sich besonders gut, um zeitliche Abfolgen besonders intuitiv zu spezifizieren. Neben dem Kommunikationsprotokoll (wann fließen welche Daten unter welchen Bedingungen) werden auch die Verantwortlichkeiten der beteiligten Kommunikationspartner klar dargestellt. Allerdings müssen Sequenzdiagramme möglichst einfach gehalten werden, da sie für die meisten Stakeholder eher ungewohnt sind. Gerade dieser Diagrammtyp neigt jedoch dazu schnell zu groß und unlesbar zu werden, bereits kleine Kommunikationsabläufe können zu unüberschaubaren Diagrammen führen. Eine Möglichkeit um diesem Problem entgegenzuwirken ist es, nur ein mögliches Szenario (z. B. Normalablauf oder spezieller Fehlerfall) darzustellen und für den übrigen Ablauf eine andere Dokumentationsart (natürliche Sprache oder einen anderen Diagrammtyp) zu wählen.[151]

4.1.6 Zustandsdiagramm

Zustandsdiagramme beschreiben ein System als eine Menge von Zuständen und deren Übergänge, also den Ereignissen, die zu Zustandsänderungen führen und so das

[150] vgl. (Rumbaugh, Jacobson, & Booch, 2005, S. 102 ff.)
[151] vgl. (Rupp & die SOPHISTen, 2014, S. 198 f.)

Verhalten des Systems bestimmten. Ein Zustand ist dabei eine Situation (ein Zeitraum) in der eine bestimmte Bedingung für das System gilt und in der es so lange bleibt, bis ein definiertes Ereignis eintritt. Dieses Ereignis führt zu einem Zustandsübergang, welcher das System vom vorherigen in den nächsten Zustand versetzt.[152]

Neben der Beschreibung der Zustände des Gesamtsystems eignen sich Zustandsdiagramme vor allem zur Darstellung der Zustände von Objekten, um deren Werdegang im System zu beschreiben.[153] Das nachfolgend abgebildete Beispiel zeigt ein Zustandsdiagramm für ein Leihobjekt einer Bibliothek.

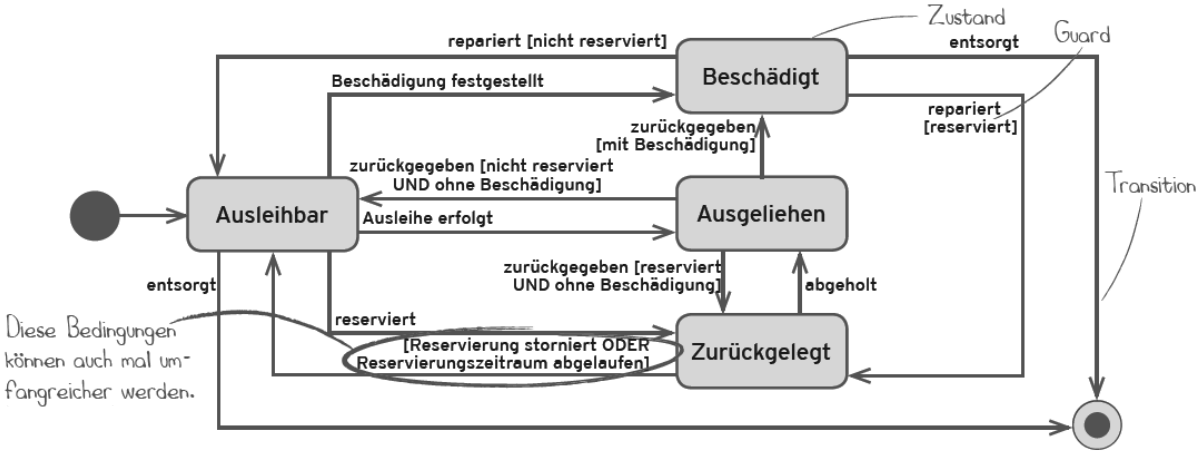


Abbildung 16: Zustandsdiagramm „Leihobjekt"[154]

Die möglichen Zustände werden in beschrifteten Rechtecken dargestellt. Für jeden Zustandsübergang (Transition, gerichtete Kanten) sind Regeln definiert: Das Leihobjekt wechselt beispielsweise aus dem Zustand „Ausgeliehen" in den Zustand „Ausleihbar", wenn es zurückgegeben wurde und weder beschädigt noch reserviert ist. Die Auslöser für den Zustandsübergang können von außen kommen, z. B. indem ein Benutzer das Leihobjekt „reserviert". Eine Zustandsänderung kann aber auch durch interne Bedingungen (Guards) verursacht werden. Dies wäre beispielsweise der Fall, wenn das Leihobjekt „beschädigt" ist. Sind die Bedingungen eines Zustandsüberganges sehr umfangreich, sollten diese zum Zweck der Übersichtlichkeit außerhalb des Diagramms abgebildet werden. Zustände können auch verschachtelt sein, so kann ein Zustand mehrere Unterzustände beinhalten (zusammengesetzter Zustand).[155]

[152] vgl. (Ebert, 2005, S. 147)
[153] vgl. (Rupp & die SOPHISTen, 2014, S. 199)
[154] (Rupp & die SOPHISTen, 2014, S. 200)
[155] vgl. (Rupp, Queins, & die SOPHISTen, 2012, S. 330 ff.)

Mit Zustandsdiagrammen kann das Verhalten des Systems, einzelner Komponenten oder einzelner Objekte verständlich dargestellt werden. Besonders gut lassen sich die Reaktionen auf verschiedene Ereignisse und die Regeln für ein bestimmtes Verhalten beschreiben. Zustandsdiagramme sind außerdem sehr gut geeignet, um das Gesamtsystemverhalten aus der Kombination mehrerer Anwendungsfälle darzustellen. Allerdings gilt es zu beachten, dass manche Menschen intuitiv lieber in Abläufen als in Zuständen denken und das Diagramm ab einer gewissen Detailebene nur mehr sehr schwer lesbar ist. Zudem erfordert gute Zustandsmodellierung viel Erfahrung und gute Modellierungskenntnisse des Erstellers.[156]

4.1.7 Klassendiagramm als Begriffsmodell

Wie bereits häufiger angeführt, ist die klare Beschreibung der verwendeten Begriffe ein zentrales Erfolgskriterium der Spezifikation. Das Klassendiagramm der UML wird zwar in der Fachliteratur[157] meist als Designdiagramm für die Softwareentwicklung beschrieben, es eignet sich jedoch auch sehr gut, um Begriffe und deren Beziehungen zueinander zu beschreiben. Wird das Klassendiagramm als Begriffsmodell verwendet, genügt es, eine vereinfachte Version dieser sehr umfangreichen Modellierungsmethode anzuwenden. Üblicherweise reichen die anhand des nachfolgend abgebildeten Beispiels erläuterten Elemente des Klassendiagramms zur Darstellung der Begriffs-Interdependenzen aus.[158]

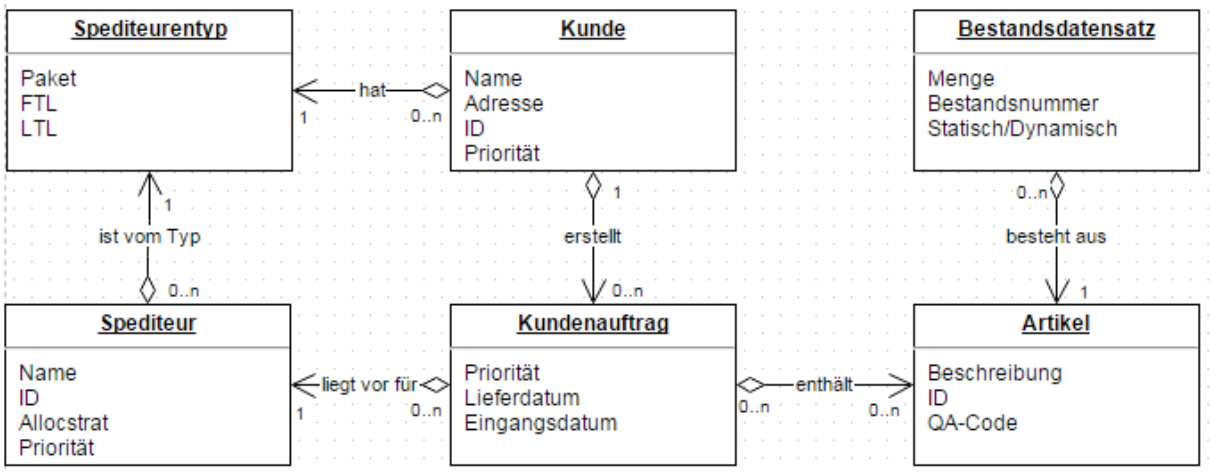


Abbildung 17: Ausschnitt Klassendiagramm als Begriffsmodell in einem Lagerverwaltungssystem[159]

[156] vgl. (Rupp & die SOPHISTen, 2014, S. 201)
[157] siehe z. B. (Balzert, 2009, S. 131 ff.)
[158] vgl. (Rupp & die SOPHISTen, 2014, S. 201)
[159] eigene Darstellung

Eine Klasse wird durch ein Rechteck dargestellt, das in drei Abschnitte untergliedert ist. Im obersten Abschnitt wird der Name der Klasse notiert (z. B. „Kunde"). Im mittleren Abschnitt werden die Attribute notiert, durch welche die Objekte dieser Klasse genauer beschrieben werden. Sie stellen die Eigenschaften der Klasse dar. Im untersten Abschnitt werden die Methoden angegeben, die von der Klasse angeboten werden. Auf die Angabe der Methoden wird bei der Verwendung des Klassendiagramms als Begriffsmodell verzichtet.[160]

Assoziationen zwischen den Klassen werden durch Kanten dargestellt, die mit dem Prozesswort beschriftet werden können. Im abgebildeten Beispiel ist exemplarisch das Prozesswort „erstellt" zwischen „Kunde" und „Kundenauftrag" angegeben. Die Assoziationen zwischen den Klassen haben Multiplizitäten, die die Mengenverhältnisse wiedergeben. So legen z. B. die Multiplizitäten der Assoziation zwischen „Kunde" und „Kundenauftrag" fest, dass ein Kunde beliebig(Null bis unendlich) viele Kundenaufträge erstellen kann. Umgekehrt gehört jedoch jeder Kundenauftrag zu genau einem Kunden. Ein weiteres Element, das für die Begriffsmodellierung sehr hilfreich ist, ist die Generalisierung (dargestellt durch ein Dreieck). Ein Subtyp einer Generalisierungsbeziehung erbt die Eigenschaften des Supertyps und kann diese gegebenenfalls adaptieren oder erweitern. Im angeführten Klassendiagramm eines Lagerverwaltungssystems ist keine Generalisierung abgebildet, ein Beispiel dafür wären jedoch die speziellen Gruppen, „Tiefkühlware", „Trockenware" und „Non-Food-Artikel", der Generalisierung „Artikel". So hat beispielsweise die „Tiefkühlware" alle Attribute des „Artikels", zusätzlich hat jedes Objekt der Klasse „Tiefkühlware" noch einen „Lagertemperaturbereich".[161]

Das Klassendiagramm als Begriffsmodell erleichtert das Verständnis der Zusammenhänge zwischen den einzelnen Begriffen sehr. Neben seiner Verwendung im Rahmen der Spezifikation, bei der es auch hilft Missverständnisse und Doppelbelegungen von Begriffen bei unterschiedlichen Stakeholdern aufzufinden, kann es sehr gut als Analysemodell beim Validieren der Anforderungen eingesetzt werden. Allerdings gilt es zu beachten, dass das Begriffsmodell für das gesamte Projekt schnell zu groß und unübersichtlich werden kann und es zudem keine Definition der Bedeutung der Begriffe liefert.[162]

[160] vgl. (Pohl, 2007, S. 201 f.)
[161] vgl. (Rumbaugh, Jacobson, & Booch, 2005, S. 52 ff.)
[162] vgl. (Rupp & die SOPHISTen, 2014, S. 202 f.)

4.2 Die Wahl der richtigen Dokumentationstechnik

Auch wenn es sich lediglich um einen Ausschnitt aus den zahlreichen Möglichkeiten handelt, haben die vorangegangenen Kapitel gezeigt, dass man Anforderungen auf viele Arten und Weisen dokumentieren kann. Die Frage, welche dieser Dokumentationstechniken die richtige ist, ist nicht trivial zu beantworten, denn diese Entscheidung hängt von vielen unterschiedlichen Faktoren ab. Neben den spezifischen Vor- und Nachteilen der jeweiligen Methode ist zu bedenken, dass kein Diagramm alle wichtigen Aspekte des Systems abbilden kann. Dennoch sollten nicht zu viele verschiedene Dokumentationstechniken verwendet werden, da mit der Anzahl der Techniken in der Regel auch der Aufwand für die Sicherstellung der Nachverfolgbarkeit und der Konsistenz steigt.[163] In der Praxis hat sich eine Kombination aus natürlichsprachlichem Text und Diagrammen bewährt, wobei der Text dazu dient, die graphische Darstellung zu verfeinern, Beispiele oder den Kontext zu beschreiben oder den dargestellten Sachverhalt nochmals textuell abzubilden.[164] Bei der Wahl der Dokumentationstechniken sollten zumindest die folgenden Einflussfaktoren berücksichtigt werden:[165]

- **Akzeptanz:** Akzeptieren die Stakeholder die Dokumentationstechnik? Wird die Akzeptanz nicht berücksichtigt, kann es passieren, dass Anforderungen von einzelnen Stakeholdern ignoriert oder sogar abgelehnt werden. Je stärker formalisiert eine Darstellungsmethode ist, desto stärker sollte dieses Kriterium berücksichtigt werden.
- **Notationskenntnisse notwendig:** Inwieweit verstehen die Stakeholder die Dokumentationstechnik? Ist die Methode den Requirements-Engineers bekannt? Auch stellt sich hier die Frage, wie schwierig das Erlernen der jeweiligen Technik ist.
- **Konsistenzsicherung:** Während der Anforderungsanalyse und darüber hinaus können sich Anforderungen ändern. Diese Änderungen in Teilen der Spezifikation sollten immer konsistent zu anderen Anforderungen durchgeführt werden. Wie einfach ist es bei der jeweiligen Technik, Änderungen einzupflegen?
- **Detaillierungsgrad:** Einige Dokumentationsarten sind nur für einen bestimmten Detaillierungsgrad der Spezifikation geeignet. Für welche Detailstufe kann die jeweilige Technik angewendet werden?

[163] vgl. (Wiegers & Beatty, 2013, S. 222 f.)
[164] vgl. (Pohl, 2007, S. 304)
[165] vgl. (Rupp & die SOPHISTen, 2014, S. 210 f.)

Einen Vorschlag zur Bewertung der unter Kapitel 4.1 angegebenen Dokumentationstechniken liefern Chris Rupp und die SOPHISTen in Form der folgenden Empfehlungsmatrix:

Tabelle 1: Empfehlungsmatrix der Dokumentationstechniken[166]

Techniken	Einflussfaktoren			
	Akzeptanz	Notationskenntnisse notwendig	Konsistenzsicherung	für Detaillierungsgrad
Natürliche Sprache	+	-	- -	alle
Use-Cases	++	+	-	grob
Aktivitätsdiagramm	+	-	0	alle
BPMN 2.0	+	-	0	alle
Sequenzdiagramm	-	-	+	alle
Zustandsdiagramm	-	- -	++	alle
Klassendiagramm	-	-	++	alle

Legende	
++	sehr gut geeignet
+	gut geeignet
0	geeignet
-	weniger geeignet
- -	nicht gut geeignet

Diese Matrix kann, wenn bereits Anforderungen auf eine bestimmte Art und Weise dokumentiert wurden, dazu verwendet werden, diese zu Evaluieren und gegebenenfalls die Wahl der Technik den aktuellen Projektbedingungen anzupassen. Wurden in einer Spezifikation beispielsweise vorrangig Klassen- und Zustandsdiagramme eingesetzt und die Stakeholder geben keinerlei Rückmeldung zu den Diagrammen, zeigt ein Blick auf die Matrix, dass diese beiden Techniken mit „weniger geeignet" im Bereich „Akzeptanz" bewertet sind. In diesem Fall wäre es zielführend, eine besser akzeptierte Technik zu wählen und die Dokumentation entsprechend zu ergänzen. Sind noch keine Anforderungen dokumentiert, kann die Matrix helfen, angepasst an die jeweiligen Projektbedingungen, die richtige Kombination aus Techniken zu finden.[167]

[166] in Anlehnung an (Rupp & die SOPHISTen, 2014, S. 211)
[167] vgl. (Rupp & die SOPHISTen, 2014, S. 212)

4.3 Nicht-funktionale Anforderungen

Sowohl die Literatur als auch die Praxis haben sich traditionell auf funktionale Anforderungen fokussiert. Diese Art der Anforderungen ist die offensichtlichste, beschreibt sie doch die Funktionen, die von den Benutzern erwartet werden. Noch wichtiger wäre es jedoch, Einigkeit über die nicht-funktionalen Anforderungen (NFA) zu erlangen. Lawrence et al zählen das Vernachlässigen von nicht-funktionalen Anforderungen zu den sechs größten Risiken im Requirements-Engineering.[168] Und zahlreiche Untersuchungen[169] zeigen, dass Stakeholder über fehlende Funktionen oder Features hinwegsehen, wenn nur ihre Qualitätsanforderungen erfüllt werden. Als weiteres Beispiel für die möglicherweise sogar katastrophalen Folgen der Vernachlässigung von NFA sei die Fallstudie des London Ambulance Service genannt.[170]

Die verschiedenen Arten und Schemata für NFA wurden bereits in Kapitel 1.2.3 beschrieben, nachfolgend soll zuerst kurz auf die Ermittlung von NFA eingegangen werden, danach auf die Möglichkeiten zur Dokumentation dieser Anforderungsart. Die grundlegende Reihenfolge der durchzuführenden Aktivitäten ist in der folgenden Abbildung dargestellt:

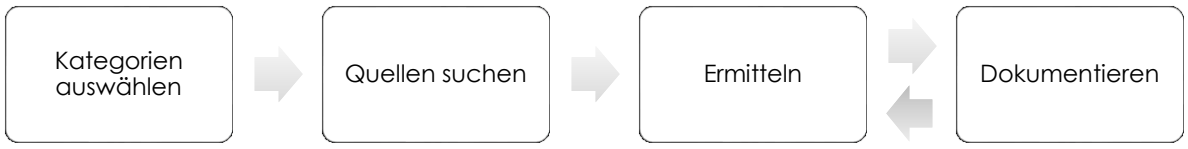


Abbildung 18: Tätigkeiten im Rahmen der Erhebung der nicht-funktionalen Anforderungen[171]

Kategorien auswählen

In einer vorbereitenden Phase muss der Prozess an die jeweilige Projektsituation angepasst werden. Dazu müssen zuerst die zu untersuchenden Inhalte und damit auch die zu betrachtenden Anforderungskategorien festgelegt werden. Hierzu kann man sich an einem bestimmten NFA-Schema oder einer Mischform von Schemata orientieren und zumindest grundlegend bestimmen, welche Hauptkategorien von NFA für das Projekt relevant sind.[172]

[168] vgl. (Lawerence, Wiegers, & Ebert, 2001)
[169] siehe z. B. (Charette, 1990)
[170] siehe (Finkelstein & Dowell, 1996)
[171] vgl. (Rupp & die SOPHISTen, 2014, S. 271)
[172] vgl. (Dörr, 2011)

Quellen suchen

Anschließend wird mit der Erhebung der Abhängigkeiten zwischen den NFA-Kategorien und der Erstellung einer Liste der möglichen Anforderungsquellen die Ermittlung der NFA vorbereitet. Allerdings gilt es zu beachten, dass die Stakeholder, die die jeweilige funktionale Anforderung liefern, nicht unbedingt jene Stakeholder sind, die auch für die dazugehörigen NFA zuständig sind. Meist gibt es für die jeweilige NFA-Kategorie zumindest einen bestimmten Ansprechpartner (z. B. Domänenexperten wie Ergonomie-Experten, Sicherheits-Experten oder Mitarbeiter mit Erfahrung mit einem ähnlichen System bzw. Vorgängersystem).[173]

Ermittlung

Anschließend an diese vorbereitende Phase sollten die NFA im Kontext des jeweiligen Bezugselements erhoben werden. Anhand der erstellen Liste werden die potentiellen Anforderungsquellen systematisch, hinsichtlich nicht-funktionaler Anforderungen befragt. Darüber hinaus können grundlegende Überlegungen zur Dokumentationsform der jeweiligen NFA-Kategorie getroffen werden, z. B. ob Anforderungen zur Benutzungsoberfläche mit Sprache oder Skizzen dokumentiert werden sollen. Außerdem sollte bereits zu diesem Zeitpunkt festgelegt werden, bis zu welchem Detaillierungsgrad nicht-funktionale Anforderungen erhoben werden sollen.[174]

Spezifikation

Nicht-funktionale Anforderungen sind meist schwieriger zu spezifizieren als funktionale Anforderungen. Dies liegt vor allem daran, dass Qualitätsattribute von den Stakeholdern oft in Form von vagen, abstrakten Zielen, die schwer mess- und testbar sind, formuliert werden.[175] In der Praxis hat sich daher die folgende Vorgehensweise bewährt: die nicht-funktionalen Anforderungen werden zunächst pro Anforderungskategorie grob erhoben, wobei die konkrete Zuordnung zu einer bestimmten Kategorie sekundär ist, denn die Orientierung an der NFA-Kategorie dient vorrangig dazu, keine Art von NFA zu übersehen. Im nächsten Schritt werden die Anforderungen dann bis zum festgelegten Detaillierungsgrad verfeinert, zumindest so weit, bis sie eindeutig mess- und testbar sind.[176]

[173] vgl. (Hruschka, 2014, S. 211 ff.)
[174] vgl. (Dörr, 2011)
[175] vgl. (Alexander & Beus-Dukic, 2009, S. 141 f.)
[176] vgl. (Rupp & die SOPHISTen, 2014, S. 275 f.)

Für die Dokumentation von nicht-funktionalen Anforderungen eignet sich grundsätzlich die natürliche Sprache. Die in Kapitel 4.1.1 vorgestellten Methoden und Hilfsmittel zur natürlichsprachlichen Dokumentation können auch bei nicht-funktionalen Anforderungen angewendet werden. Zusätzlich zu den genannten Techniken wird nachfolgend eine Spezifikationstechnik, die speziell für NFAs entwickelt wurde, vorgestellt: „Planguage".

Qualitative Begriffe, wie zum Beispiel „einfach", „schnell", „verlässlich", „sicher" oder „effizient" führen häufig zu Missverständnissen zwischen allen Projektbeteiligten. Planguage wurde von Tom Gilb entwickelt, um diesem Problem entgegen zu wirken, indem qualitative Begriffe quantifiziert werden. Planguage steht für „Planning Language" und bezeichnet zugleich eine formelle Spezifizierungssprache und eine Sammlung von Methoden und Prozessen für das System Engineering.[177] Obwohl sich Planguage hauptsächlich mit der Beschreibung von Anforderungen auseinandersetzt, kann es auch einen wichtigen Beitrag in der Erhebung leisten, indem die semiformale Notation und die vorgeschlagenen Metriken helfen können, nichtfunktionale Anforderungen vollständig und korrekt zu erheben und eine Sensibilität für ihre Wichtigkeit zu erlangen.[178]

Planguage versucht durch die Verwendung eines umfangreichen Glossars, einem Set von Parametern und einigen Icons, die natürlichsprachliche Spezifikation durch zusätzliche Aspekte zu verfeinern. Bei der Spezifikation von skalaren Attributen von Performance-Anforderungen sind die folgenden beiden Aspekte am wichtigsten: [179]

1. Die Sicherstellung von Mess- und Testbarkeit
2. Die Festlegung von Erfolg und Misserfolg

Die Mess- und Testbarkeit wird sichergestellt, indem die Messmethode und der Maßstab bereits zum Spezifikationszeitpunkt festgelegt werden. Es werden Zielwerte und mögliche Einschränkungen definiert, anhand deren Erreichung der Erfolg der Umsetzung gemessen werden kann. Es ist auch zulässig, anstatt eines genauen Wertes, einen Wertebereich zu definieren. Damit klar ist, wie skalare Attribute spezifiziert werden liefert Planguage ein Regelwerk und ein dazugehöriges Spezifikationstemplate, das beispielsweise folgende Elemente beinhaltet:[180]

[177] vgl. (Gilb, 2005, S. 9)
[178] vgl. (Herrmann & Knauss, 2013, S. 156)
[179] vgl. (Simmons, 2001)
[180] vgl. (Simmons, 2001)

- Tag:[181] Eindeutiger Name der Anforderung
- Definition: Beschreibung der Anforderung
- Scale: Maßeinheit, in der die Anforderung gemessen wird
- Meter: Messmethode
- Benchmark: Vergleichswerte zur Orientierung
- Target: Vorgabe der Anforderung, das angestrebte Ziel
- Constraint: Zulässige Grenzen der Anforderung

Zum besseren Verständnis, wird im Folgenden ein vereinfachtes Beispiel angegeben. Es handelt sich hierbei um die häufige nicht-funktionale Anforderung der Wartbarkeit:[182]

- Tag: Wartbarkeit
- Definition: Die Leichtigkeit, mit der ein System verändert oder erweitert werden kann
- Scale: Stunden
- Meter: Durchschnittliche Dauer für die Behebung eines Fehlers; Gemessen wird von Beginn der Behebung bis zu dessen Lösung
- Target: 2 Stunden
- Constraint: 4 Stunden

Alle nachfolgenden Phasen der Validierung und Verwaltung werden analog für funktionale und nicht-funktionale Anforderungen ausgeführt. Aus diesem Grund werden diese Phasen nicht getrennt nach Anforderungsart behandelt.

[181] Für die Planguage-Elemente und Begriffe gibt es bis dato keine einheitliche deutsche Übersetzung, weshalb in der vorliegenden Arbeit die englischen Bezeichnungen verwendet werden.
[182] vgl. (Emmanuel, 2010)

5 Prüfung von Anforderungen

Anforderungen sind immer das Ergebnis eines komplexen Erstellungsprozesses. So können beispielsweise unterschiedliche Fachsprachen oder Schwächen in der Anforderungsspezifikation zu mangelnder Produkt- oder Kommunikationsqualität führen. Nur mittels systematischer Qualitätssicherungsmaßnahmen kann der Nutzen der Anforderungen garantiert werden. Qualitätssicherung als Teilbereich des Requirements-Engineerings beschäftigt sich mit der Qualität von Anforderungen und Anforderungsdokumenten. Die Qualitätssicherung nimmt in diesem Kontext eine besonders wichtige Stellung ein, da Anforderungen die Basis der Systementwicklung und der weiteren Aktivitäten im Projekt sind. Ihre Qualität bestimmt damit maßgeblich die Qualität des zu entwickelnden Systems.[183]

Die „Internationale Organisation für Normung" definiert den Begriff der „Qualität" als den Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt.[184] Im Kontext der Anforderungsanalyse gibt die Qualität also an, in welchem Maß eine Anforderung oder ein Anforderungsdokument den objektiv messbaren und fest definierten Qualitätskriterien, die in den Kapiteln 1.2.4 und 1.3.2 genannt wurden, entspricht. Ziel der Qualitätssicherung ist es demnach, Anforderungen zu entwickeln, die den von den Stakeholdern geforderten Qualitätsmaßstäben angemessen entsprechen.[185] Das Prüfen im Sinne der Qualitätssicherung soll Auffälligkeiten in den Anforderungen und Anforderungsdokumenten finden und gegebenenfalls beheben. Im Wesentlichen werden die folgenden drei Qualitätsaspekte untersucht:[186]

- **Dokumentation:** Prüfung auf Mängel in der Dokumentation hinsichtlich Konformität mit Dokumentations- und Spezifikationsvorschriften.
- **Inhalt:** Prüfung auf inhaltliche Fehler hinsichtlich der Vollständigkeit der Anforderung und des Anforderungsdokuments, sowie Prüfung auf Verfolgbarkeit, Korrektheit bzw. Adäquatheit, Konsistenz, Überprüfbarkeit, Eindeutigkeit, etc.
- **Abgestimmtheit:** Prüfung der Anforderungen auf Mängel in der Anforderungsabstimmung unter den relevanten Stakeholdern.

[183] vgl. (Denger & Olsson, 2005)
[184] (ISO/IEC 9000 Norm, 2005)
[185] vgl. (Rupp & die SOPHISTen, 2014, S. 302)
[186] vgl. (Pohl & Rupp, 2011)

5.1 Qualitätssicherungsprozess

Grundsätzlich gliedert sich die Qualitätssicherung in zwei Bereiche: die konstruktive und die analytische Qualitätssicherung.[187]

Die **konstruktive Qualitätssicherung** hat zum Ziel, Fehler in den Anforderungen und Anforderungsdokumenten gar nicht erst entstehen zu lassen. Die Methoden der konstruktiven Qualitätssicherung können bereits ab der Phase der Anforderungsermittlung eingesetzt werden, um von Beginn an qualitativ möglichst hochwertige Anforderungen zu dokumentieren. Zu den Techniken der konstruktiven Qualitätssicherung zählen die Definition von Zielen und die Kontextanalyse, die Berücksichtigung sämtlicher Anforderungsquellen oder die Spezifikation der Anforderungen anhand definierter Spezifikationsvorschriften und Standards.[188]

Die **analytische Qualitätssicherung** befasst sich mit bereits dokumentierten Anforderungen mit dem Ziel, Auffälligkeiten in den Anforderungen zu finden und zu beheben. Die Qualität der Anforderungen soll durch die Techniken und Hilfsmittel der analytischen Qualitätssicherung noch vor der Nutzung in der nachgelagerten Entwicklungsphase geprüft und falls erforderlich, verbessert werden.[189] Die analytische Qualitätssicherung kann ihrerseits wiederum in zwei Teilbereiche gegliedert werden: die Verifikation (Prüfung auf Form und Syntax) und die Validierung (inhaltliche Prüfung).[190, 191]

Für das Vorgehen bei der analytischen Qualitätssicherung hat sich in der Praxis die Orientierung am nachfolgend abgebildeten Plan-Do-Check-Act-Zyklus von W. E. Deming[192] bewährt. Die konkrete Umsetzung des Zyklus im Zuge der Qualitätssicherung ist in den nachfolgenden Kapiteln kurz beschrieben.

[187] vgl. (Sommerville & Sawyer, 1997, S. 190)
[188] vgl. (Denger & Olsson, 2005)
[189] vgl. (Pohl, 2007, S. 420 f.)
[190] vgl. (Hruschka, 2014, S. 283)
[191] Da die potentiellen Maßnahmen zur konstruktiven Qualitätssicherung in den Kapiteln 2-4 beschrieben wurden, befassen sich alle weiteren Unterkapitel ausschließlich mit den Aspekten der analytischen Qualitätssicherung.
[192] siehe (Deming, 2000)

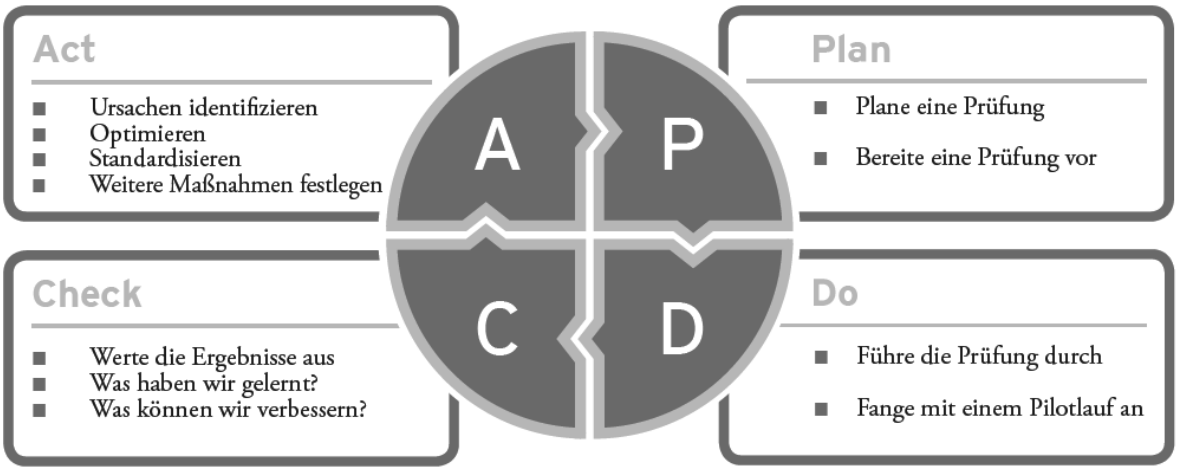


Abbildung 19: Die Phasen des PDCA-Zyklus[193]

5.1.1 Vorbereitung

Bevor die eigentliche Qualitätssicherungsmaßnahme im Sinne des PDCA-Zyklus durchgeführt werden kann, müssen Entscheidungen in den folgenden Bereichen getroffen werden:

Festlegung der Qualitätsziele

Die Qualitätsziele müssen an die Anforderungen des jeweiligen Projektes angepasst werden, da es nicht immer erstrebenswert oder möglich ist, alle in Kapitel 1.2.4 und 1.3.2 genannten Qualitätskriterien gleichermaßen zu optimieren. Es gilt daher den Fokus der Qualitätssicherung auf jene Kriterien zu legen, die für den Erfolg des Projektes wichtig sind. Werden im Zuge der Prüfung Qualitätsmetriken für die Anforderungen oder die Anforderungsdokumente erhoben, können Zielwerte definiert werden. Bei der Festlegung der Qualitätsziele gilt es stets zu beachten, den Systemanalyseprozess durch einen zu hohen Qualitätsanspruch nicht unnötig umfangreich und damit teuer werden zu lassen, ohne einen messbaren Nutzen beizutragen. Idealerweise sind die formalen Kriterien, die im Zuge der Qualitätssicherung geprüft werden sollen auch jene, die von den Anforderungsautoren explizit gefordert werden.[194]

Auswahl der Qualitätssicherungsmethoden

Nachdem die Ziele der Qualitätssicherung definiert und dokumentiert wurden, müssen die Methoden ausgewählt werden, die eine Aussage über den Erreichungsgrad der Ziele liefern können. Dabei können die folgenden Überlegungen vorgenommen

[193] (Deming, 2000)
[194] vgl. (Rupp & die SOPHISTen, 2014, S. 307)

werden: Werden Spezifikationen verglichen oder soll nur eine Momentaufnahme einer einzelnen Spezifikation erfolgen? Orientieren sich die Messziele an quantitativen Fragestellungen oder steht die inhaltliche Qualität im Mittelpunkt? Steht die Gesamtspezifikation als Prüfungsunterlage zur Verfügung oder nur ein Teilausschnitt? Alle diese Fragen bilden die Grundlage für die Auswahl einer oder mehrerer Prüftechniken für Anforderungen, wie sie im nachfolgenden Kapitel vorgestellt werden.[195]

Definition der Qualitätszeitpunkte

Wann welche Qualitätssicherungsmethode eingesetzt werden soll ist von mehreren Randbedingungen abhängig. Übereinstimmend empfehlen jedoch viele Autoren so früh wie möglich konstruktive Qualitätssicherungsmaßnahmen einzusetzen und somit Auffälligkeiten in den Anforderungen von vornherein zu vermeiden. Auch die analytischen Prüfmaßnahmen können eingesetzt werden, sobald die Anforderungen oder zumindest ein Teil der Anforderungen einigermaßen stabil und die konstruktiven Maßnahmen abgeschlossen sind. Dadurch können zwar mehrere Qualitätsprüfungszyklen für dieselbe Funktionalität entstehen, allerdings können sich dadurch die Stakeholder intensiver mit den Wünschen an das zu entwickelnde System auseinandersetzen und späteren Änderungen wird somit vorgebeugt. Ist ein PDCA-Zyklus für eine Anforderung abgeschlossen und sind die festgelegten Qualitätsziele erreicht, passiert die Anforderung ein Qualitätstor[196]. Alle Anforderungen müssen spätestens dann geprüft sein, wenn sie der Entwicklung übergeben werden.[197]

Ernennung der Prüfer

Nachdem nun festgelegt wurde wann welche Qualitätsmethoden eingesetzt werden sollen, gilt es zu entscheiden, welche Personen bzw. Rollen an der Prüfung beteiligt sein sollten. In der Praxis hat sich eine Kombination aus unterschiedlichen Rollen als effizient herausgestellt. Methodisch versierte Requirements-Engineers beispielsweise sind darauf geschult, logische Zusammenhänge zu erfassen und formale Mängel wie Widersprüche oder Lücken in den Anforderungen zu finden. Ohne den nötigen Einblick in die fachlichen Hintergründe des Projektes, können diese aber keine inhaltlichen Fehler identifizieren. Der Auftraggeber bzw. ein Anwendervertreter wiederum eignen sich gut um die Anforderungen auf ihren Inhalt hin zu prüfen. Unbedingt notwendig ist es, alle diejenigen an der Prüfung teilnehmen zu lassen, die im weiteren

[195] vgl. (Denger & Olsson, 2005)
[196] engl.: Quality Gate
[197] vgl. (Hruschka, 2014, S. 276 ff.)

Verlauf des Projektes mit den Anforderungen arbeiten müssen. Das betrifft z. B. die Entwickler, die Softwarearchitekten und die Tester. Sehr empfehlenswert ist außerdem die Beteiligung von externen Auditoren wie beispielsweise Personen aus der Qualitätssicherung oder aus Standardisierungsgruppen.[198] Das wesentliche Ziel der Beteiligung mehrerer unterschiedlicher Rollen ist es, die Anforderungen aus verschiedenen Perspektiven zu prüfen. Diese unterschiedlichen Perspektiven können durch die Betrachtung desselben Sachverhalts durch verschiedene reale Personen erfolgen. Mehrere Autoren haben auf Basis der von Edward de Bono erstmals vorgestellten Methode der „Six Thinking Hats“[199] aber auch Modelle entwickelt, bei denen dieselbe Person die Anforderungen aus unterschiedlichen Sichten prüfen kann. Als weitere Beispiele seien an dieser Stelle „CORE“[200], „PREview“[201] oder „Viewpoints“[202] genannt.

5.1.2 Plan – Qualitätsprüfung vorbereiten

Sobald die, im vorherigen Kapitel beschriebenen, allgemeinen Festlegungen getroffen wurden, kann mit der Vorbereitung einer Qualitätsprüfung begonnen werden. Zu dieser Vorbereitung zählen die nachfolgend angeführten Maßnahmen, deren Durchführung die Voraussetzung für die Anwendung einer Methode ist.

Prüfbarkeit feststellen

Bevor mit der Prüfung begonnen wird, sollte festgestellt werden, ob der Aufwand für die Qualitätsprüfung per Prüftechnik oder Metrik zum gegenwärtigen Zeitpunkt sinnvoll und damit gerechtfertigt ist. Dazu zählen rein formale Gesichtspunkte wie das Format des Anforderungsdokuments oder die Vergewisserung, dass alle benötigten Informationen vorhanden sind.[203]

Prüfgegenstand definieren

Häufig ist es vorteilhaft, den Aufwand für die Qualitätssicherung auf ein sinnvolles Maß zu reduzieren, indem die Prüfung auf eine Stichprobe der Anforderungen angewendet wird. Als Stichprobe werden bei einer Prüfmethode gezielt Anforderungen ausgewählt, die einem Kriterium entsprechen. Zum Beispiel können alle neu hinzugefügten Anforderungen betrachtet werden oder ein bestimmtes Kapitel. Bei Metriken

[198] (Hruschka, 2014, S. 282 f.)
[199] siehe (Bono, 1986)
[200] siehe (Mullery, 1979)
[201] siehe (Sommerville, Sawyer, & Viller, 1998)
[202] siehe (Sommerville I. , Software Engineering, 2012)
[203] vgl. (Rupp & die SOPHISTen, 2014, S. 312)

wird eine zufällig gezogene Teilmenge von Anforderungen geprüft, um einen neutralen Überblick über den Zustand der Spezifikation zu erhalten.[204]

Prüfgegenstand extrahieren und dokumentieren

Nachdem nun definiert wurde, welche Anforderungen geprüft werden sollen, ist es empfehlenswert, diese in eine Messvorlage einzutragen. Dieses Dokument dient im Wesentlichen dazu, die Qualitätssicherungsmaßnahme inklusive der gefundenen Auffälligkeiten bzw. Messwerte zu dokumentieren. In diesem Dokument sind eindeutige Referenzen auf die geprüften Originalanforderungen sowie methodenspezifische Felder für die Eintragungen des Prüfers enthalten. Eine Prüf- oder Messvorlage stellt sicher, dass der Betrachtungsgegenstand immer gleich geprüft wird und beschleunigt zudem die Auswertung und die Korrektur, durch die Verwendung einer standardisierten Dokumentenvorlage.[205]

5.1.3 Do – Qualitätsprüfung durchführen

Sind nun alle vorbereitenden Maßnahmen durchgeführt, kann die eigentliche Qualitätsprüfung vorgenommen werden. Die Prüfer bewerten die ausgewählten Anforderungen anhand der vorab definierten Kriterien und tragen die Ergebnisse der Prüfung in die Messvorlage ein. Parallel dazu sollte begonnen werden einen Prüfbericht zu verfassen. Dieser beinhaltet sämtliche wichtigen Informationen zum Ablauf des aktuellen Qualitätssicherungszyklus und Kommentare zu Auffälligkeiten in der bewerteten Anforderungsspezifikation. Dieser Bericht erleichtert die gemeinsame Retrospektive des Qualitätssicherungszyklus und hilft einzuschätzen, ob die Qualität der ausgewählten Stichprobe vom Gesamteindruck der Spezifikation abweicht.[206]

5.1.4 Check – Ergebnisse beurteilen

Nach Abschluss der Qualitätsprüfung müssen die Ergebnisse mit den zuvor festgelegten Qualitätszielen verglichen werden. Dabei sollte festgehalten werden, wo die Ergebnisse die Ziele erfüllen, verfehlen oder sogar übertreffen. Zudem sollte nun auch der subjektive Gesamteindruck des Prüfers mit der Aussage der Prüfungsergebnisse abgeglichen werden. Gibt es Differenzen, sollten entweder die gesetzten Qualitätsziele oder die angewendeten Methoden hinterfragt werden. Außerdem können durch den Vergleich der aktuellen Prüfung mit vorhergegangenen Prüfungen Ten-

[204] vgl. (Kamiske & Brauer, 2002)
[205] vgl. (Brüggemann & Bremer, 2012)
[206] vgl. (Rupp & die SOPHISTen, 2014, S. 314)

denzen hinsichtlich des Erfolges durchgeführter Qualitätsmaßnahmen festgestellt werden.[207]

5.1.5 Act – Maßnahmen initiieren

Entspricht die ermittelte Qualität der Spezifikation nicht den gesetzten Qualitätszielen oder zeigt sich eine generelle Tendenz zu sich verschlechternder Qualität, ist es wichtig, zeitnah Maßnahmen zur Verbesserung der Qualität durchzuführen. Solche Maßnahmen können beispielsweise die Aus- und Weiterbildung der Requirements-Engineers oder den Einsatz bestimmter Notationstechniken, die die Dokumentation formalisieren betreffen. Als Ergebnis der Prüfung liegt dann eine Liste von Mängeln, Fehlern und Anmerkungen zu den Anforderungen vor, die im nächsten Schritt korrigiert und in die Anforderungen eingearbeitet werden müssen. Hier gilt es zu beachten, dass die überarbeiteten Anforderungen einem weiteren Qualitätsprüfungszyklus unterzogen werden müssen.[208]

5.2 Prüftechniken für Anforderungen

Nachdem nun die Schritte des Qualitätsprüfungszyklus beschrieben wurden, werden in den nachfolgenden Kapiteln die erwähnten Prüftechniken einer näheren Betrachtung unterzogen.

5.2.1 Reviews

Reviews zählen zu den am häufigsten durchgeführten Prüftechniken, mit denen Auffälligkeiten in den Anforderungen identifiziert werden können. Zu dem Überbegriff des „Reviews“ gehören die Stellungnahme, der Walkthrough und die Inspektion. Diese Prüftechniken sollen nun kurz beschrieben werden.

Stellungnahme

Stellungnahmen oder Gutachten zählen zu den wenig formalen Prüftechniken, die zeitnah und unkompliziert durchgeführt werden können. Die zu prüfenden Anforderungen werden einem Gutachter übergeben, welcher die Anforderungen gemäß definierter Qualitätskriterien prüft und Auffälligkeiten in den Anforderungen markiert oder in eine Mängelliste einträgt. Das erstellte Gutachten bzw. die Mängelliste werden dem Anforderungsautor übergeben und dieser pflegt die daraus resultierenden

[207] vgl. (Rupp & die SOPHISTen, 2014, S. 314 f.)
[208] vgl. (Rupp & die SOPHISTen, 2014, S. 315 f.)

Änderungen in die Anforderungen ein. Zu beachten ist bei dieser Prüfmethode jedoch, dass das Prüfergebnis sehr von den Prüfern und deren Qualitätsbewusstsein abhängig ist.[209]

Walkthrough

Bei einem Walkthrough leitet der Anforderungsautor die Prüfer schrittweise z. B. anhand einer Präsentation, durch den Prüfgegenstand. Dabei werden die Prozesse primär nach logischen Gesichtspunkten vorgestellt und der Autor erläutert auch den Entstehungsprozess und seine Gedankengänge im Zusammenhang mit der Anforderungserstellung. Im Zuge der Erläuterungen wird der Anforderungsautor einerseits selbst Auffälligkeiten entdecken, zusätzlich stellen die Prüfer Fragen und versuchen so, mögliche Probleme zu identifizieren. Offene Punkte können dann gemeinsam eingearbeitet oder zumindest protokolliert werden. Bei dieser Prüfmethode gilt es allerdings zu beachten, dass der Anforderungsautor gezielt von Schwächen in der Spezifikation ablenken kann. Deshalb ist es umso wichtiger, kritische und aufmerksame Prüfer in den Reviews zu beteiligen.[210]

Inspektion

Inspektionen haben ihre Ursprünge in der reinen Code-Inspektion, bei der der Programmiercode vor dem Kompilieren Zeile für Zeile von den Inspektoren geprüft wurde, um durch das Auffinden von Fehlern wertvolle Computerzeit zu sparen.[211] Heute wird dieses Verfahren vorrangig für Anforderungsdokumente verwendet. Inspektionen sind die formalste Art des Reviews und werden von einer kleinen Gruppe von Personen durchgeführt. Den beteiligten Stakeholdern werden dabei die folgenden Rollen zugeordnet:[212]

- Der **Autor** ist jene Person, die das zu untersuchende Anforderungsdokument erstellt hat. Im Gegensatz zum Walkthrough nimmt der Autor bei der Inspektion eine passivere Rolle ein und bringt sich bei Kommentaren der Inspektoren erklärend ein.
- Der **Moderator** organisiert und leitet die Inspektion. Er verteilt alle nötigen Dokumente rechtzeitig an die beteiligten Personen und moderiert die Inspektion, in-

[209] vgl. (Hruschka, 2014, S. 279 f.)
[210] vgl. (Pohl, 2007, S. 451 f.)
[211] vgl. (Fagan, 1976)
[212] vgl. (Wiegers & Beatty, 2013, S. 334 f.)

dem er für einen Ausgleich zwischen den gegensätzlichen Interessen des Autors und der Inspektoren sorgt.

- Der **Vorleser** stellt den Inspektionsgegenstand sukzessive vor und führt die Inspektoren durch das Anforderungsdokument. Diese Rolle sollte nicht vom Anforderungsautor übernommen werden, kann aber auf einen der Inspektoren oder den Moderator übertragen werden. Auf einen Vorleser kann jedoch auch gänzlich verzichtet werden, indem nicht der gesamte Text, sondern lediglich die Kommentare der Inspektoren zur jeweiligen Seite des Dokuments besprochen werden.[213]
- Die **Inspektoren** sind für das Auffinden von Auffälligkeiten in den Anforderungen verantwortlich. Für den Erfolg der Inspektion ist es essentiell, dass sich die Inspektoren bereits im Vorfeld der Inspektion mit dem zu inspizierenden Dokument auseinandergesetzt und ihre persönlichen Anmerkungen zu den einzelnen Anforderungen vermerkt haben. Die Inspektoren sollten so gewählt werden, dass möglichst viele Facetten der Anforderungen betrachtet werden.
- Der **Protokollant** dokumentiert die Ergebnisse der Inspektion, insbesondere die aufgedeckten Fehler.

Die Inspektion wird von Moderator und Autor gemeinsam geplant. Die notwendigen Dokumente, die von den Inspektoren vorab begutachtet werden sollen, müssen vom Moderator bereits einige Tage vor dem angesetzten Inspektionstermin verteilt werden, um den Inspektoren ausreichend Zeit für eine gründliche Auseinandersetzung mit den Anforderungen zu geben.[214] Bei der Durchführung der Inspektion im Rahmen des Inspektionstermins, führt der Moderator bzw. der Vorleser seitenweise durch das Anforderungsdokument. Einwände oder Fragen der Inspektoren zum jeweiligen Abschnitt werden besprochen und vom Protokollanten in eine Mängelliste aufgenommen. In diesen Diskussionsphasen der Inspektion ist es die zentrale Aufgabe des Moderators, das Gespräch nicht vom eigentlichen Thema abschweifen zu lassen und für einen Ausgleich zwischen der Position des Autors und jener der Inspektoren zu sorgen. Im Anschluss an die Inspektion arbeitet der Anforderungsautor die besprochenen Änderungen in die Anforderungen ein. Danach erfolgt ein weiterer Review-Zyklus der überarbeiteten Anforderungen.[215]

[213] vgl. (Hruschka, 2014, S. 280)
[214] vgl. (Wiegers & Beatty, 2013, S. 336)
[215] vgl. (Pohl, 2007, S. 443 f.)

Neben den genannten Erfolgsfaktoren ist es bei allen Review-Techniken wichtig, die Inspektoren hinsichtlich der Durchführung eines Reviews zu schulen. Den Inspektoren sollte bewusst sein, welchen Zweck ihr Review verfolgt und was bzw. welche Sicht konkret vom jeweiligen Inspektor benötigt wird. Zudem sollten nicht zu viele Anforderungen in einem Review inspiziert werden, da sonst die Gefahr besteht, die einzelnen Anforderungen zu oberflächlich zu betrachten. Insbesondere bei einer Inspektion müssen die Beteiligten ihre Kritik sachlich, konstruktiv und wertschätzend äußern. Wird dieser zwischenmenschliche Aspekt vernachlässigt kann es passieren, dass die Anforderungsautoren versuchen werden, sich dieser stressvollen Situation zu entziehen und Änderungsvorschläge nicht offen annehmen. Deshalb ist es essentiell, dass alle Beteiligten des Reviews das Gefühl mitnehmen, einen konstruktiven Beitrag zur Verbesserung des Anforderungsdokuments geleistet zu haben. Werden diese Erfolgsfaktoren beachtet, sind Inspektionen eine sehr effektive Form des Anforderungs-Reviews und offenbaren viele tiefsitzende Auffälligkeiten in den Anforderungen.[216]

5.2.2 Prototypen

In einem Prototyp werden Anforderungen teilweise umgesetzt, um ihre Realisierbarkeit zu prüfen oder dem Auftraggeber einen Eindruck davon zu geben, wie das zu entwickelnde System aussehen oder sich verhalten wird. Prinzipiell wird zwischen zwei Arten von Prototypen unterschieden:[217]

- **Horizontale Prototypen[218]** zeigen die Systemfunktionalität und die Benutzungsoberfläche. Sie zeigen das Verhalten der Software ohne konkrete Implementierung, d. h. der Benutzer kann die Masken betrachten und eventuell zwischen ihnen Navigieren, aber keine oder nur sehr wenige tatsächliche Aktionen ausführen. Mit dieser Art von Prototyp kann sehr schnell überprüft werden, ob das System von den zukünftigen Nutzern akzeptiert wird und ob die, in den Anforderungen beschriebenen, Abläufe praktikabel sind.
- **Vertikale Prototypen[219]** werden eingesetzt, wenn neue Technologien, wie z. B. neue Frameworks, unbekannte Datenbanken oder eine neue Programmiersprache im zukünftigen System verwendet werden sollen. Der Prototyp besteht in diesem Fall aus einem Durchstich durch alle Schichten der Software und nicht nur

[216] vgl. (Wiegers K. E., 2006, S. 57 ff.)
[217] vgl. (Hruschka, 2014, S. 281 f.)
[218] engl.: Mock-up
[219] engl.: Proof of concept

aus der Präsentation der Oberfläche. Vertikale Prototypen dienen der Risikominimierung indem sofort geprüft wird, ob das in den Anforderungen beschriebene Verhalten umsetzbar und funktionell ist. Mit vertikalen Prototypen können zudem Prüfungen auf Durchsatz oder Erweiterbarkeit von Teilfunktionen durchgeführt werden.

Neben der Unterscheidung in horizontale und vertikale Prototypen muss vor Beginn der Entwicklung des Prototyps die Entscheidung getroffen werden, ob es sich um einen **Wegwerf-** oder **evolutionären Prototypen** handeln soll. Wegwerf-Prototypen dienen dazu, Unsicherheiten oder Fragen zum zukünftigen System zu beantworten oder die Qualität der Anforderungen zu erhöhen. Da der Prototyp verworfen wird, sobald er seinen Zweck erfüllt hat, sollte er so günstig und einfach wie möglich entwickelt werden. Im Gegensatz zum Wegwerf-Prototyp bietet der evolutionäre Prototyp eine solide architektonische Basis für die inkrementelle Entwicklung des Systems basierend auf immer klareren Anforderungen. Aus diesem Grund muss der evolutionäre Prototyp von Anfang an qualitativ hochwertig implementiert und häufige Erweiterungen und Verbesserungen im Design berücksichtigt werden.[220]

Jegliche Art von Prototyp hilft, die Vorstellungskraft bei komplexen Systemen zu steigern und dient als gute Diskussionsgrundlage für die Anforderungskommunikation mit den Anwendern und der Entwicklung. Speziell bei Oberflächenprototypen kann jedoch leicht ein falscher Eindruck von der Reife des Produkts oder dem tatsächlichen Aussehen des Endprodukts entstehen. Zudem sollte ihre Anwendung gut überdacht werden, da die Erstellung von Prototypen aufwendig und zeitraubend ist.[221]

5.2.3 Testfälle

Als Softwaretest wird üblicherweise das reine Ausführen der definierten Testfälle am Ende des Entwicklungsprozesses verstanden. Die Testfälle prüfen dabei, ob die entwickelte Software den dokumentierten Anforderungen entspricht. Dieses Verständnis hat zu dem Mythos geführt, dass das Testen erst am Ende des Entwicklungsprozesses begonnen werden kann.[222] Im Kontext des Requirements-Engineerings kann das Testen jedoch mehr leisten als das Durchlaufen von Testfällen und das Auffinden von Fehlern in der Software. Insbesondere die Testfallerstellung kann bereits durchgeführt werden, sobald ein Teil Anforderungen stabil ist. Die frühe Testfallerstellung gibt den

[220] vgl. (Wiegers & Beatty, 2013, S. 299 f.)
[221] vgl. (Rupp & die SOPHISTen, 2014, S. 323)
[222] vgl. (Graham, 2002)

Testern die Möglichkeit sich schon frühzeitig mit den Anforderungen des Systems auseinanderzusetzen und Auffälligkeiten in den Anforderungen aufzudecken.[223] Denn kaum jemand liest die Anforderungen so intensiv wie derjenige, der die dazugehörigen Testfälle erstellt. Die frühe Testfallerstellung ermöglicht eine einfachere Änderung der Anforderungen, da keine zusätzliche Freigabe nötig ist. Je später die Testfälle erstellt werden, desto geringer ist ihr Nutzen für das Prüfen der Anforderungen.[224]

Das IREB definiert einen Testfall als eine Anweisung für den Test bezüglich einer Anforderung (oder eines Anforderungsteils), welche die Erfüllung der Anforderung (oder des Teils) im erstellten Produkt oder durchgeführten Prozess beschreibt.[225] Die Literatur[226] kennt zahlreiche Arten von Testfällen, im Zuge der Prüfung von Anforderungen sind jedoch vorrangig natürlichsprachliche von Bedeutung.

Ein typischer natürlichsprachlicher Testfall besteht aus drei Elementen: der Ausgangssituation, dem Testereignis und dem erwarteten Ergebnis. Im Zuge der Testfallerstellung wird oft klar, dass das tatsächlich auslösende Ereignis für eine Aktion oder für allgemeine Vorgänge nicht oder nur undeutlich beschrieben ist. Wird ein derartiger Defekt erkannt, sollte neben der vollständigen Testfallbeschreibung auch die Anforderung entsprechend vervollständigt werden. Diese Art der Testfallbeschreibung ist mit relativ hohem Aufwand verbunden, liefert aber auch sehr wichtige Ergebnisse und qualitativ hochwertige Anforderungen. Zudem sind gute und ausführliche Testfälle für die spätere Durchführung der Tests äußerst hilfreich und insbesondere die Testbarkeit von nicht-funktionalen Anforderungen ist nach der Erstellung der Testfälle gegeben.[227]

5.2.4 Checklisten

Checklisten zählen zu den Assistenztechniken der Anforderungsprüfung. Sie beinhalten Fragestellungen, die ein Prüfer bei der Prüfung von Anforderungen berücksichtigen soll, um Fehler in den Anforderungen zu finden. Checklisten sind in der Praxis weitverbreitet und können bei allen vorgestellten Prüftechniken eingesetzt werden.[228] Damit ein Prüfer eine Checkliste effektiv bei der Prüfung von Anforderungen einset-

[223] vgl. (Denger & Olsson, 2005)
[224] vgl. (Rupp & die SOPHISTen, 2014, S. 323)
[225] vgl. (Pohl & Rupp, 2011)
[226] siehe z. B. (Alper, 1994)
[227] vgl. (Wiegers & Beatty, 2013, S. 342 ff.)
[228] vgl. (Pohl, 2007, S. 463)

zen kann, sollte der Umfang der Checkliste auf wenige Fragestellungen beschränkt sein. Im Idealfall sollte der Umfang zehn Fragen nicht überschreiten und der Prüfer die Checkliste während der Prüfung in Erinnerung behalten können.[229]

Die in der Anforderungsprüfung verwendeten Checklisten können grundsätzlich einem von zwei Prüfbereichen zugeordnet werden: der **Prüfung auf Form und Syntax** oder der **inhaltlichen Prüfung.** Im ersten Fall befassen sich die Fragestellungen der Checkliste mit der Gliederung des Dokuments und der Konformität der zu prüfenden Anforderungen mit dem Firmenstandard. Beispielsweise könnte die Checkliste prüfen, ob vorgegebene Kapitel, ein Glossar oder ein Abkürzungsverzeichnis vorhanden sind. Checklisten für die inhaltliche Prüfung formulieren die inhaltlichen Vorgaben zur Anforderungsdokumentation als Testfragen. Eine derartige Fragestellung könnte beispielsweise prüfen, ob alle Eingaben und Ausgaben eines Prozesses definiert sind oder ob alle Ausnahmen erkannt wurden. Bei der Prüfung des Glossars könnte mit der Checkliste überprüft werden, ob der Begriff qualitativ gut definiert und durch ein anschauliches Beispiel präzisiert worden ist. Auch Diagramme sollten mit den Fragestellungen inhaltlich auf semantische Aussagekräftigkeit und Korrektheit geprüft werden. Auf dieselbe Art und Weise gilt es, geeignete Fragestellungen für nichtfunktionale Anforderungen und Randbedingungen zu formulieren.[230]

Neben dem Umfang einer Checkliste ist auch das Abstraktionsniveau der Fragestellung erfolgsentscheidend. Werden die Fragestellungen zu generisch gehalten, können sie zwar für unterschiedliche Systeme und Anforderungsarten wiederverwendet werden, jedoch ist ihr Nutzen für die Prüfung gering. Beispielsweise bietet die Fragestellung „Sind die Anforderungen vollständig?“ keinen Hinweis darauf, wie der Prüfer fehlende Anforderungen identifizieren soll. Werden Checklisten jedoch gemäß dem zu prüfenden Sachverhalt, dem Fachwissen und der Erfahrung des prüfenden Stakeholders angepasst, können sie sehr effektiv und effizient eingesetzt werden.[231]

[229] vgl. (Sommerville & Sawyer, 1997, S. 119)
[230] vgl. (Hruschka, 2014, S. 283 ff.)
[231] vgl. (Pohl, 2007, S. 467)

5.3 Die Wahl der richtigen Prüftechnik

Wie bei der Wahl der richtigen Dokumentationstechnik für Anforderungen, ist auch die Wahl der richtigen Prüftechnik von der konkreten Projektsituation abhängig. Einige Projektrandbedingungen haben großen Einfluss auf die Einsetzbarkeit einer Technik, dieser Zusammenhang ist in der nachfolgenden Matrix dargestellt.

Tabelle 2: Prüftechniken und Einflussfaktoren[232]

Techniken	Einflussfaktoren						
	Geringe Motivation Stakeholder	Geringes Abstraktionsvermögen Stakeholder	Schlechte Verfügbarkeit Stakeholder	Fixiertes, knappes Budget	Hohe Kritikalität des Systems	Großer Systemumfang	Viele nicht-funktionale Anforderungen
Stellungnahme	- -	0	-	++	-	- -	- -
Walkthrough	-	0	-	+	-	-	+
Inspektion	- -	0	++	-	++	++	++
Prototyp	++	++	+	+/-	+	0	- -
Testfälle	++	0	++	+	++	++	++

Legende	
++	sehr gut geeignet
+	gut geeignet
0	geeignet
-	weniger geeignet
- -	nicht gut geeignet

Sind beispielsweise die Stakeholder auch in andere Projekte eingebunden und deshalb schlecht verfügbar, ist der Einsatz der Inspektion nicht ratsam, da jeder Prüfer neben der Vorbereitungszeit auch an der Inspektion teilnehmen muss. Aus demselben Grund ist die Inspektion auch bei geringer Motivation der Teilnehmer nicht das optimale Mittel der Wahl. Prototypen sind bei einem fixierten knappen Budget in Abhängigkeit davon, mit welchem Aufwand sie erstellt werden, gut oder weniger gut geeignet. Einfache Papierprototypen können beispielsweise auch bei einem knappen Budget gewinnbringend eingesetzt werden.[233]

[232] in Anlehnung an (Rupp & die SOPHISTen, 2014, S. 332)
[233] vgl. (Rupp & die SOPHISTen, 2014, S. 332)

Teil II Identifikation und Definition von Verbesserungsmaßnahmen für das Requirements-Engineering

Nachdem in Teil I die, für die vorliegende Arbeit benötigten, theoretischen Grundlagen erklärt bzw. beschrieben wurden, befasst sich Teil II der Arbeit mit der praktischen Anwendung des erlangten Wissens. Am Beginn dieses Abschnitts wird in Kapitel 6 beschrieben, wie bei der Identifikation von Verbesserungspotentialen und der anschließenden Ableitung von Verbesserungsmaßnahmen vorgegangen wurde. Anschließend erläutert Kapitel 7 die aktuelle Situation und Arbeitsweise in der Abteilung „Software Engineering“ und in den betroffenen Bereichen der angrenzenden Abteilungen der KNAPP Systemintegration, um ein Verständnis für die Notwendigkeit der Thematik zu erhalten. Die nachfolgenden Kapitel beschreiben die Schritte, die zur Identifikation der Verbesserungspotentiale geführt haben, die Auswahl geeigneter Methoden des Requirements-Engineerings für die Nutzung der Verbesserungspotentiale und die konkret abgeleiteten Verbesserungsmaßnahmen.

6 Vorgehensweise Verbesserungsprojekt

Wie bereits in der Einleitung kurz beschrieben wurde, ist das Ziel der vorliegenden Arbeit, den bestehenden Requirements-Engineering-Prozess bei der KNAPP Systemintegration hinsichtlich Verbesserungspotentialen zu durchleuchten und Maßnahmen abzuleiten, die die Qualität der Ergebnisse dieses Prozesses verbessern können.

Um dieses Ziel zu erreichen, wurde ein Projektteam nominiert, in dem möglichst alle, vom Ergebnis der Arbeit der Abteilung „Software Engineering" betroffenen Abteilungen, repräsentiert werden sollten. Zudem sollten noch Kollegen aus der Abteilung „Software Engineering" ihre Eindrücke und Erfahrungen in den Verbesserungsprozess einfließen lassen. Dieses Projektteam bestand demnach aus den folgenden Rollen:

- Abteilungsleiter „Software Engineering" und „Software Test & Installation"
- Abteilungsleiterin „Project Development Nord"
- Gruppenleiter „Software Engineering"
- Technischer Projektleiter
- Entwicklungsprojektleiter

Außerdem wurden, nach Bedarf, noch zusätzliche Personen befragt bzw. miteinbezogen.

Ende August 2015 fand ein Projekt-Kick-Off statt, bei dem das Verbesserungsprojekt erstmals dem Projektteam vorgestellt wurde. Grundlegendes Ziel dieser Veranstaltung war es, allen Projektbeteiligten die Notwendigkeit von professionellem Requirements-Engineering und die Ziele des Projektes näher zu bringen. Neben dem Senken des Risikos, Zeit und Ressourcen aufgrund semantischer und syntaktischer Fehler in der Spezifikation zu verschwenden, bestanden die Ziele des Projektes auch wesentlich daraus, neuen Mitarbeitern den Einstieg in die Projekte zu erleichtern. Um diese Ziele zu erreichen, sollte die Anforderungsspezifikation im Zuge des Verbesserungsprojektes stärker formalisiert und prüfbarer werden. Durch die Auswahl von konkreten Methoden, die bei der Spezifikation angewendet werden sollten, sollte zudem die Basis für tiefergehende Schulungsmaßnahmen gelegt und dadurch die Qualität der Spezifikationen verbessert werden. Dies sollte Kunden und Entwicklern helfen, sich in den Projekten schneller zurechtzufinden, neue Mitarbeiter sollten schneller eingearbeitet werden können und Tester einfacher und schneller Testfälle aus den Anforderungen ableiten können. Neben der detaillierten Darstellung der Ziele des Verbesse-

rungsprojektes, wurden auch die wichtigsten Elemente des Requirements-Engineerings vorgestellt, um dem Projektteam einen Überblick über die zu bearbeitenden Themen zu geben.

Die Vorgehensweise bei der Bearbeitung der Themen des Verbesserungsprojektes wurde dann wie folgt festgelegt. Zuerst wurde die Gesamtbetrachtungseinheit des Verbesserungsprozesses gemäß den, im Zuge der vorliegenden Masterarbeit behandelten, drei Haupttätigkeiten des Requirements-Engineerings unterteilt. Dadurch ergaben sich drei Themenblöcke, die nacheinander behandelt wurden:

1. Grundlagen dokumentieren und Anforderungen ermitteln,
2. Anforderungen formulieren und
3. Anforderungen validieren.

Die Verfasserin der vorliegenden Arbeit sollte nach der Analyse der IST-Situation der jeweiligen Haupttätigkeit eine intensive Literaturrecherche durchführen, um Möglichkeiten zur Realisierung der Aktivitäten in der jeweiligen Projekt- bzw. Spezifikationsphase und Best Practices zu identifizieren. Die Erkenntnisse der Recherche wurden dann im Rahmen von Workshops vorgestellt und innerhalb der Gruppe diskutiert. Ziel der Diskussion war es, konkrete Verbesserungsmaßnahmen für die KNAPP Systemintegration zu entwickeln. Wie dabei im Detail vorgegangen wurde, wird in den nachfolgenden Kapiteln beschrieben. In der folgenden Tabelle ist der zeitliche Ablauf des Projektes übersichtlich dargestellt:

Tabelle 3: Zeitlicher Ablauf des Projektes

Phase	Zeitraum					
	Aug 15	Sep 15	Okt 15	Nov 15	Dez 15	Jan 16
Kick Off, Vorbereitung	■					
IST-Analyse, Literaturrecherche		■				
Literaturrecherche, Workshop: Grundlagen/Anforderungen ermitteln			■			
Literaturrecherche, Workshop: Anforderungen formulieren				■		
Literaturrecherche, Workshop: Anforderungen prüfen						■
Erstellung der Arbeit		■	■	■	■	■

7 IST-Situation

Ziel eines Intralogistikanbieters, wie der KNAPP Systemintegration ist es, ein Logistiksystem zu planen, dass es dem Anlagenbetreiber ermöglicht, seine Logistikprozesse effizient abzuwickeln und dadurch die stetig steigenden Erwartungen seiner Kunden an die Schnelligkeit, die Qualität, die Kosten und die Variantenvielfalt der Logistikleistung zu erfüllen. Der Planung der Systemstruktur der Intralogistikanlage kommt damit besondere Bedeutung zu, da nicht nur die aktuellen, sondern auch die mittel- bis langfristigen Leistungsanforderungen des Kunden an die Lagerprozesse berücksichtigt werden müssen. Insbesondere die Intralogistiksoftware steht dabei vor der Herausforderung, immer komplexere Prozesse abbilden und unterstützen zu müssen, dabei soll jedoch die Bedienung der Software durch den Lagermitarbeiter, immer einfacher und intuitiver, erfolgen können.[234] Die diesbezüglichen Haupttätigkeiten der KNAPP Gruppe im Allgemeinen und der KNAPP Systemintegration sowie der Abteilung „Software Engineering" im Speziellen, wurden bereits in der Einleitung der vorliegenden Masterarbeit kurz beschrieben. Nun soll eine nähere Betrachtung der Aufgabenbereiche und der Arbeitsabläufe in Bezug auf die Erhebung, der Spezifikation und der Prüfung der Anforderungen an die zu entwickelnde Intralogistiksoftware erfolgen.

7.1 Vorbereitung

In der Vorverkaufsphase ermittelt die Projektierung anhand der Ausschreibungsunterlagen des Kunden und im Rahmen von Software- und Layout-Workshops bei bzw. mit dem Kunden die ersten, groben Anforderungen an die zu entwickelnde Intralogistiksoftware, mit dem Ziel, einen Angebotspreis abschätzen und ermitteln zu können.

Akzeptiert der Kunde den, von der Projektierung kalkulierten, Angebotspreis und entscheidet er sich für den Kauf, beginnt mit dem formellen Auftragsstart die Feinplanung der Intralogistik-Anlage und der Software. Die Projektierung übergibt die Grobplanung der Anlage in Form des Layouts, der technischen Spezifikation und eines logistischen Konzeptes, welches die geplanten Kapazitäten und die Durchsätze der Anlage enthält, an die jeweiligen Fachabteilungen (Software, Mechanik, Elektrik etc.) und das Projektmanagement.

[234] vgl. (Hompel & Schmidt, 2008, S. 8 ff.)

Der technische Projektleiter beginnt mit der Erhebung der detaillierten Anforderungen an die zu entwickelnde Software, indem er, die von der Projektierung und dem Vertrieb zur Verfügung gestellten Dokumente und das Layout analysiert, und sich so mit den benötigten Prozessen im Lager vertraut macht. Auf Basis der Unterlagen wird auch bereits mit dem Erstellen der Spezifikation begonnen. Die Verfeinerung der Prozesse und die Erhebung der noch unbekannten Anforderungen erfolgt dann hauptsächlich über Workshops mit dem Kunden. Der Umfang und Inhalt dieser Workshops variiert von Projekt zu Projekt und ist im Wesentlichen von der Projektgröße und -komplexität, dem Projektterminplan, dem Bewusstsein des Kunden über seine Anforderungen, der Anzahl der Workshop-Teilnehmer sowie deren Erfahrung und Entscheidungsbefugnis abhängig.

7.2 Spezifikation

Alle identifizierten Anforderungen werden in weiterer Folge sukzessive in der funktionalen Spezifikation ergänzt bzw. angepasst und in den Schnittstellen-Spezifikationen beschrieben. Diese Dokumentation und die Verwaltung der Anforderungen erfolgt durch den technischen Projektleiter und den, bei den meisten Projekten definierten technischen Co-Projektleiter, im Requirements-Engineering- und -Management-Tool „Polarion ALM Requirements“. Abhängig davon, welches KNAPP Haupt-Softwareprodukt in der zukünftigen Anlage verwendet wird, ist die jeweilige Standarddokumentenvorlage auszuwählen. Diese gibt einen Anhaltspunkt für den Aufbau der Spezifikation und die möglichen Inhalte. Übergreifend geben alle Vorlagen vor, jene Kapitel, in denen Prozesse beschrieben werden (z. B. Wareneingang, Wareneinlagerung, Kommissionierung im manuellen Bereich), in die Unterkapitel „Allgemeines“, „Voraussetzungen“, „Materialfluss“, „Funktionalität“, „Resultate“ und „Fehlerfälle“ aufzuteilen. Die meisten technischen Projektleiter halten sich bei der Dokumentation grob an diese Vorgabe, sehr oft werden aber Unterkapitel wie z. B. „Allgemein“ nicht verwendet oder zusätzliche, vom Inhalt des Hauptkapitels abhängige, Unterkapitel eingefügt. Die Reihenfolge der Hauptkapitel wird außerdem dem Prozessfluss durch das Lager angepasst. Die folgende Abbildung zeigt einen Ausschnitt aus der, von der KNAPP Systemintegration definierten, Standardgliederung der Spezifikation, für Projekte, bei denen der Kunde sich für ein Warehouse Control System (WCS) entschieden hat.

6 Primary Processes
- *6.1 Receiving*
- *6.2 Decanting*
- *6.3 Goods-In*
  - 6.3.1 General
  - 6.3.2 Material Flow
  - 6.3.3 Preconditions
  - 6.3.4 Functionality
  - 6.3.5 Result
  - 6.3.6 Error handling
- *6.4 Quality check incoming goods*
  - 6.4.1 General
  - 6.4.2 Material Flow
  - 6.4.3 Preconditions
  - 6.4.4 Process
  - 6.4.5 Result
  - 6.4.6 Error handling
- *6.5 Put away*
  - 6.5.1 General
  - 6.5.2 Material Flow
  - 6.5.3 Preconditions
  - 6.5.4 Functionality
  - 6.5.5 Result
  - 6.5.6 Error handling

Abbildung 20: Standardgliederung für WCS-Spezifikationen[235]

Für die Spezifikation der Anforderungen wird in der Dokumentenansicht von „Polarion" der Anforderungstext verfasst, anschließend können einzelne oder mehrere Absätze innerhalb eines Kapitels als „Requirement" markiert werden. Der Inhalt dieser Anforderungen ist also meist nur im Kontext der vorstehenden und nachfolgenden Anforderungen schlüssig. Teils werden gesamte Lagerprozesse, inklusive dem dazugehörigen Fehlerhandling, als einzelne Anforderung markiert, teils werden die Anforderungen auf Basis der Absätze gebildet. Darüber hinaus gab es den Versuch, die Anforderungen auf Satzebene heruntergebrochen zu markieren, dies führte jedoch zu einer unüberschaubar großen Menge von unzusammenhängenden Anforderungen, die in der Entwicklung in dieser Form nicht praktikabel weiterverwendet werden konnten. Wie granular die Markierung erfolgt, ist dem technischen Projektleiter überlassen und von den Präferenzen des Entwicklungsprojektleiters abhängig, der die Anforderungen für die Entwicklung übernimmt.

Das Requirements-Engineering-Tool vergibt bei der Erstellung einer Anforderung automatisch eine eindeutige Identifikationsnummer für jede Anforderung. Zusätzlich muss vom Ersteller noch die „Verbindlichkeit" und die „Quelle" angegeben werden. Die auswählbaren Werte erstrecken sich bei der Verbindlichkeit von „muss enthalten", über „sollte enthalten" und „kann enthalten" bis „wird nicht enthalten". Stan-

[235] (KNAPP Systemintegration GmbH, 2016)

dardmäßig ist dieser Wert mit „muss enthalten“ belegt, die anderen Werte werden bei diesem Pflichtfeld nie verwendet. Die „Quelle“ kann aus den folgenden Werten ausgewählt werden: „Standardanforderung“, „modifizierte Standardanforderung“, „neue projektspezifische Anforderung“ oder „kopierte Projektanforderung“. Welcher Wert angegeben wird, ist vom technischen Projektleiter subjektiv zu entscheiden. Bei vielen Projekten wird vom technischen Projektleiter außerdem das Feld „Funktionale Beschreibung“ ausgefüllt. Hierbei handelt es sich um Zusatzinformation für den Entwickler der Anforderung, welche im Anforderungstext zu viel Detailinformation für den Kunden darstellen würde.

Sowohl für die Syntaktik als auch die Semantik der Prosatexte in der Spezifikation gibt es, abgesehen von der groben Kapitelstruktur, keine Regeln oder Vorgaben, die von den Software-Engineers einzuhalten wären.

Die grafische Darstellung der Prozessabläufe erfolgte in der Vergangenheit hauptsächlich mit Ablaufdiagrammen. Meist wurden bzw. werden Prozesse zuerst in Form eines Prosatextes beschrieben und erst danach, wenn es der zeitliche Rahmen des Projektes zulässt, zusätzlich grafisch dargestellt. Vor einigen Jahren hat sich KNAPP dazu entschlossen, Prozesse zukünftig mit der BPMN, in Form von BPDs, darzustellen. Für die Einführung der BPMN wurde bei der KNAPP AG in Graz ein Projektteam formiert, welches die, bei der gesamten KNAPP Gruppe zu verwendenden Symbole und die anzuwendenden Regeln für die Erstellung der BPDs definierte. Danach wurde mit den Software-Fachabteilungen in Graz und Leoben eine Reihe von Standardlagerprozessen anhand dieser Regeln dargestellt. Wie bei der KNAPP AG bereits realisiert, sollen in Zukunft auch in Leoben die, von der Projektierung gemeinsam mit dem Kunden in der Grobplanung definierten und mithilfe der BPMN dargestellten Lagerprozesse in der Feinplanung, von den Software-Engineers weiter verfeinert und detailliert werden. Zum momentanen Zeitpunkt fehlt den Software-Engineers jedoch das methodische Wissen, da sie noch nicht für die richtige Anwendung der BPMN im Allgemeinen und die für KNAPP geltenden Regeln und Symbole im Speziellen ausgebildet wurden. Schnittstellen mit anderen Systemen werden üblicherweise in Form von Tabellen spezifiziert. Begriffe und deren Zusammenhänge werden bis dato nicht graphisch dargestellt.

7.3 Qualitätssicherung

Während der Erstellung der Spezifikation finden regelmäßig Reviews, meist in Form von Stellungnahmen, statt. Diese Reviews erfolgen hauptsächlich durch den Kunden, grundsätzlich wäre auch eine Prüfung der Anforderungen durch den KNAPP Entwicklungsprojektleiter in dieser Phase vorgesehen, in der Projektrealität erfolgt dies jedoch fast immer erst nach der Abnahme der Spezifikation bzw. vor dem Entwicklungsstart. Außerdem sollte die Spezifikation von zumindest einem „projektfremden" Kollegen der Abteilung „Software Engineering" geprüft werden. Bisher konnte diese Vorgabe nur bei einem Teil der Projekte erfüllt werden. Die finale Version der Spezifikation muss vom Kunden formal abgenommen werden.

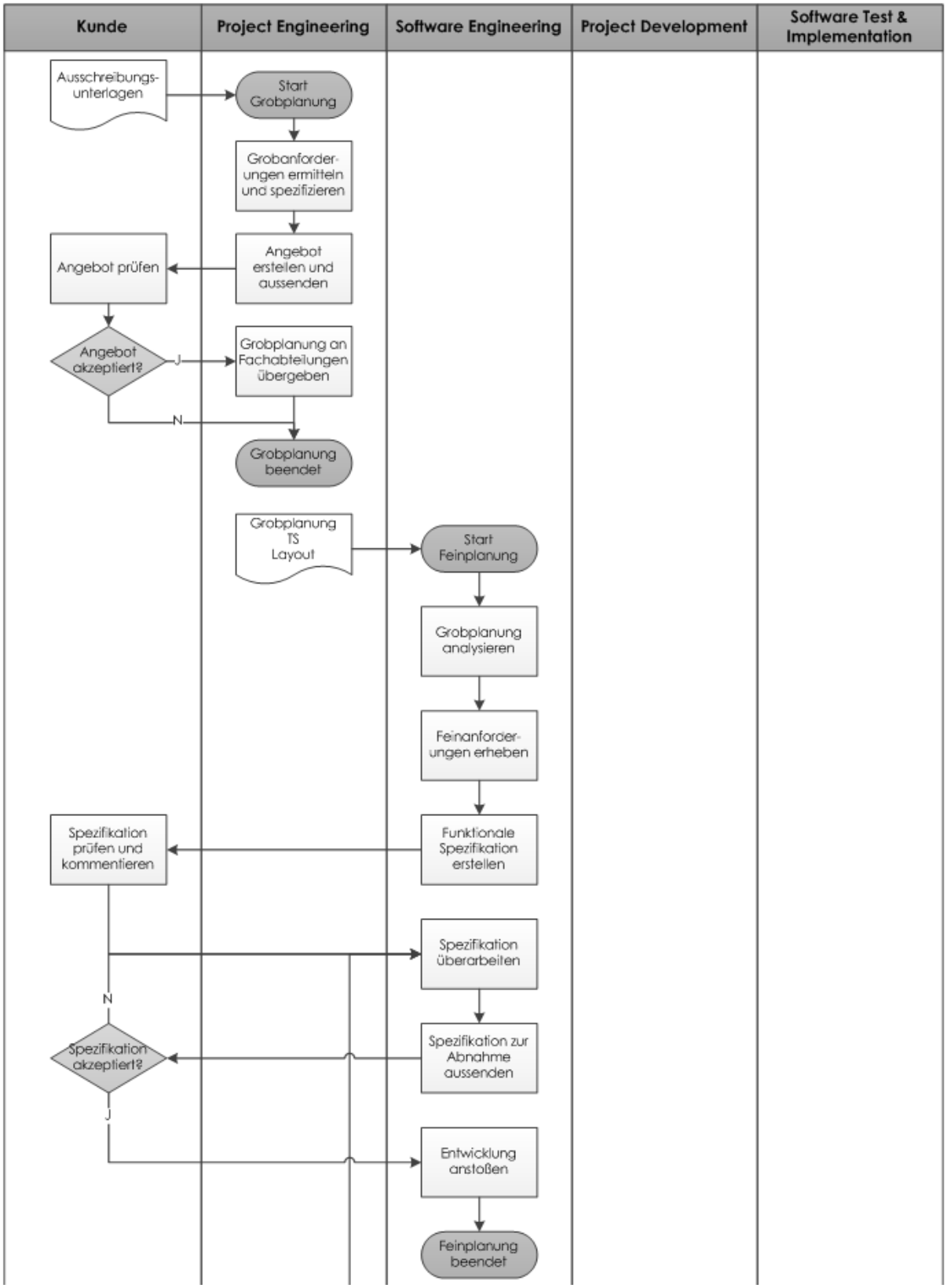
Alle diese Prüfungen dienen der Verbesserung der semantischen Qualitätskriterien der Anforderungen, es sollen vorrangig Lücken und Widersprüche oder Unklarheiten aufgedeckt werden. Für die Reviews gibt es keine formalen Vorgaben oder Prüfkriterien, auch die Ergebnisse der Reviews werden nicht dokumentiert oder ausgewertet.

7.4 Weiterer Projektverlauf

Sobald der Kunde die Spezifikation abgenommen hat, beginnt die Entwicklung der Software. Dazu werden die Anforderungen vom Requirements-Engineering-Tool „Polarion ALM Requirements" in das Software-Projektmanagement-Tool „Atlassian JIRA Software" übertragen. In diesem Tool werden dann vom Entwicklungsprojektleiter die auf den Anforderungen basierenden Arbeitspakete für die Softwareentwickler definiert und verwaltet. Änderungen oder große Lücken, die erst im Nachhinein identifiziert werden, können nach der Spezifikationsabnahme durch den Kunden nur mehr mit vermehrtem Aufwand eingearbeitet werden, was sich wiederum in den Zeitaufwänden und somit in weiterer Folge auch in den Projektkosten niederschlägt. Abweichungen zum verkauften Projektumfang werden als Änderungsanträge[236] ausgearbeitet, abgeschätzt und dem Kunden zum Kauf angeboten. Werden Änderungen vom Kunden gekauft, werden diese manchmal in die Spezifikationen eingearbeitet, manchmal versuchen die technischen Projektleiter die Spezifikation in ihrer abgenommenen Form zu erhalten und Änderungen existieren nur in separaten Dokumenten oder werden lediglich per E-Mail an die Entwicklung weitergegeben. Hinsichtlich der Verwaltung von Änderungen gibt es keinen definierten einzuhaltenden Prozess und auch keine einheitliche Vorgehensweise. Die wesentlichen Schritte des in den

[236] engl.: Change Requests

vorherigen Kapiteln beschriebenen Prozessablaufs sind im nachfolgenden Diagramm dargestellt:



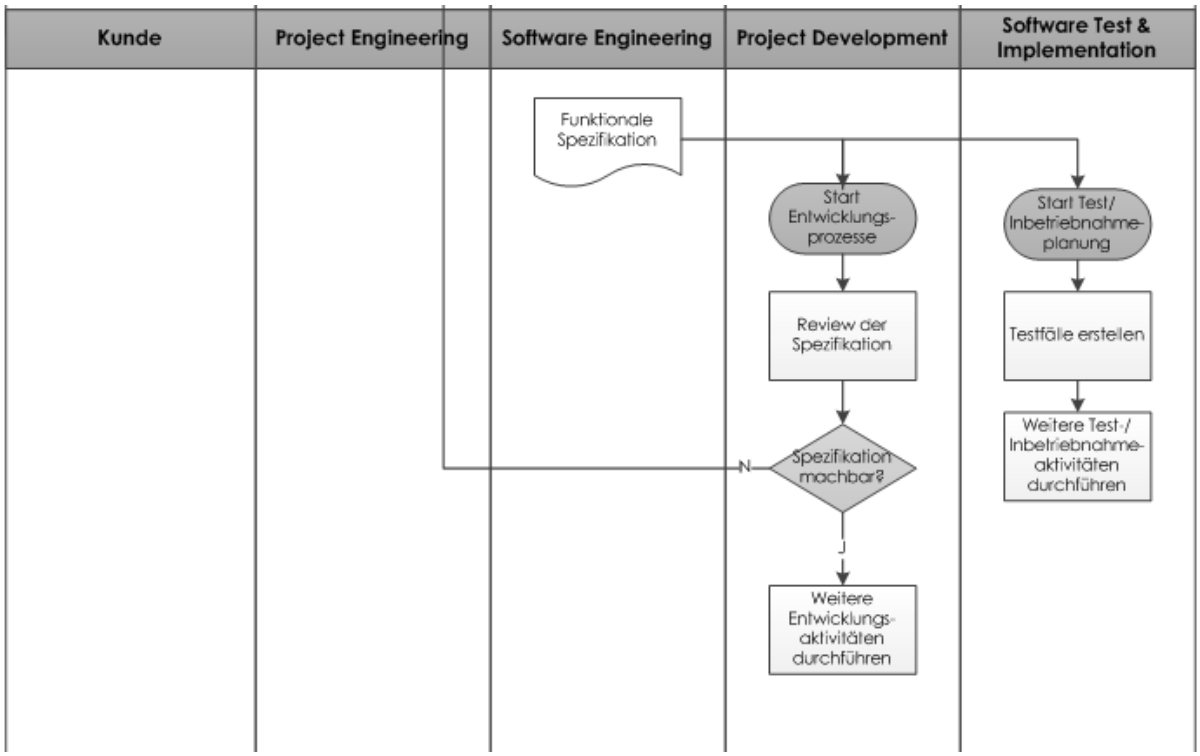


Abbildung 21: IST-Prozess Grob-/Feinplanung, Entwicklungs- und Test-/Inbetriebnahme-Start[237]

[237] Quelle: Eigene Darstellung

8 Grundlagen dokumentieren und Anforderungen ermitteln

Die Dokumentation der Projektgrundlagen und die verschiedenartige Ermittlung der Anforderungen des Kunden an die zu erstellende Software stellten den ersten Themenblock des Verbesserungsprojektes dar. Im Detail wurden dabei die folgenden Themenbereiche ausführlich behandelt:

1. Stakeholder
2. Projektziele
3. Systemkontext
4. Glossar
5. Anforderungsermittlung

Die Ergebnisse der umfassenden Literaturrecherche der Verfasserin der vorliegenden Arbeit wurden dem Projektteam in einem halbtägigen Workshop am 30. Oktober 2015 vorgestellt. Anschließend wurden Verbesserungspotentiale und mögliche Maßnahmen diskutiert. Diese Arbeit ist in den nachfolgenden Kapiteln beschrieben.

8.1 Stakeholder

Da es bezüglich der Analyse und Dokumentation der Stakeholder eines Projektes bei der KNAPP Systemintegration bisher keine definierte Vorgehensweise gab und diese Aktivitäten deshalb nicht durchgeführt wurden, galt es zunächst die theoretischen Grundlagen der Stakeholderanalyse und ihren Nutzen dem Projektteam vor Augen zu führen. Dazu wurde dargestellt, wer die Stakeholder eines Projektes sind, wie viele Stakeholder ein Projekt möglicherweise hat und wie die Stakeholder eines Projektes identifiziert werden können. Anschließend wurden die zwei wesentlichen Möglichkeiten zur Dokumentation der Stakeholder vorgestellt:

- **Datenbank:** Umfassende Dokumentation der Stakeholder und aller Stakeholder-bezogenen Informationen in einer Datenbank.
- **Liste:** Sammlung aller Stakeholder-Daten in einer einfachen Liste. Diese wird für alle Projektbeteiligten zugänglich abgelegt.

Abschließend wurden dann Attribute vorgestellt, die für jeden Stakeholder festgehalten werden sollten und eine Beispielliste gezeigt.

8.1.1 Diskussion und Verbesserungspotentiale

Die Idee, die Stakeholder eines Projektes zentral zu dokumentieren fand bei den Workshop-Teilnehmern sofort großen Anklang. Der größte Vorteil der Stakeholder-Dokumentation wurde vor allem darin gesehen, dass schnell die Verantwortlichen auf Kunden- und KNAPP-Seite gefunden werden können, ohne erst beim Projektmanager oder technischen Projektleiter nachfragen zu müssen. Denn auch intern ist es bei den Projekten oft nicht klar ersichtlich welche Person beim jeweiligen Projekt für welches KNAPP-Produkt der Ansprechpartner ist. Zudem wurde der Aufwand für diese zusätzliche Dokumentation als relativ gering bewertet. Kritisch wurde lediglich die Aktualisierung der Stakeholderliste gesehen, da dies regelmäßig vom technischen Projektleiter durchgeführt werden muss. Als Verbesserungspotential wurde demnach klar die Einführung der Dokumentation der Projekt-Stakeholder identifiziert.

8.1.2 Verbesserungsmaßnahmen

Nachdem die Entscheidung getroffen wurde, die Projekt-Stakeholder zukünftig dokumentieren zu wollen, wurde festgelegt, auf welche Art und Weise dies erfolgen sollte. Nach Rücksprache mit der internen Systemadministration wurde entschieden, die Stakeholderliste im bereits in Verwendung befindlichen Kollaborations- und Dokumentenmanagementsystem „Microsoft SharePoint“ zu führen. Auf Basis der Social-Software-Funktionalität dieser webbasierten Anwendung können alle KNAPP-internen Stakeholder einfach zur Liste hinzugefügt werden. Externe Stakeholder wie z. B. die Stakeholder des Kunden oder externer Gewerke können nach einigen Anpassungen der Software auch in der Stakeholderliste angelegt werden. Die größten Vorteile der Nutzung von „Microsoft SharePoint“ bestehen darin, dass die angebotene Funktionalität einer bereits etablierten Software genutzt und der Zugriff auf die Stakeholderliste für alle Beteiligten gewährleistet wird.

Die Anzahl der zu dokumentierenden Stakeholder ist prinzipiell dem technischen Projektleiter überlassen, allerdings wurden von den Workshop-Teilnehmern Rollen definiert, für die verpflichtend ein Eintrag erfolgen muss. Diese sind in der nachfolgenden Aufzählung gelistet. Zusätzliche Stakeholder können je nach Bedarf hinzugefügt werden.

- Projektmanager Kunde
- HOST-Systemverantwortlicher Kunde

- Logistiker Kunde (z. B. Lagerleiter, Logistikplaner)
- Endbenutzervertreter (z. B. Super-User, pro Lagerbereich)
- Verantwortlicher „Externes Gewerk“ (falls vorhanden, für jedes externe Gewerk zumindest eine Person)
- Projektmanager KNAPP
- Technischer Projektleiter KNAPP
- Entwicklungsprojektleiter KNAPP
- Verantwortlicher „Subgewerk“ KNAPP (für jedes Subgewerk zumindest eine Person, z. B. WCS, SPS, OSR)
- Design-Lead KNAPP (Project Engineering - Layout-Planer)
- Software-Lead KNAPP (Project Engineering - Basis-Software-Design)
- Vertrieb
- Sicherheitsbewertung/Risikoanalyse (Arbeitssicherheit)
- Systemadministration IT KNAPP
- Produktmanagement KNAPP

Die folgenden Attribute sollen pro Stakeholder festgehalten werden:

- Rolle/Funktion
- Name
- Kontaktdaten und Verfügbarkeit
- Wissensgebiet (z. B. kennt das Altsystem, kennt die Lagerprozesse bis ins Detail)
- Einfluss/Relevanz/Bedeutung (z. B. Informationslieferant User Acceptance, Reviewer der Anforderungen fachlich, Reviewer der Anforderungen auf Machbarkeit)

Bei den Attributen müssen zumindest die Rolle, der Name und die Kontaktdaten beim Anlegen ausgefüllt werden, die weiteren Attribute können danach ergänzt werden.

8.2 Projektziele

Gleich wie die Stakeholder der Projekte, werden auch die Projektziele in der Feinplanung nicht explizit dokumentiert. Deshalb erfolgte im Workshop ein kurzer theoretischer Einstieg in das Thema „Projektziele“, indem auf die wichtigsten Gründe für die Definition von Zielen eingegangen wurde. Anschließend wurden die folgenden Möglichkeiten zur Ziel-Dokumentation vorgestellt:

- **Natürlichsprachliche Formulierung** unter Einhaltung der neun Regeln für natürlichsprachliche Ziele.
- **Zielschablonen** mit einer bestimmten Anzahl auszufüllender vordefinierter Attribute. Es wurden drei verschiedene Schablonen gezeigt, von 15 Attributen, über fünf Attribute, bis lediglich drei zu definierende Eigenschaften pro Ziel.
- **Zielmodellierung** mit **Und-Oder-Bäumen** als Ergänzung bzw. Darstellung der definierten Ziele.

Nach der Illustration der vorgestellten Möglichkeiten zur Ziel-Dokumentation mit Beispielen erfolgte die Diskussion zur Auffindung von Verbesserungspotentialen.

8.2.1 Diskussion und Verbesserungspotentiale

Das Thema „Zieldokumentation" wurde von den Workshop-Teilnehmern sehr kritisch betrachtet und sehr kontrovers diskutiert. Während einige Teilnehmer die Meinung vertraten, dass das Wissen über die Projektziele des Kunden für die Verfolgung der eigenen Ziele unerheblich wären, meinten andere, dass die Ziele des Kunden bereits in der Vertriebsphase ermittelt würden und dies in der Feinplanung nicht nochmalig erfolgen müsse, auch wenn diese Ziele von der Projektierung nicht explizit dokumentiert werden. Nach längerer Diskussion kamen die Teilnehmer zu dem Schluss, dass bezüglich der Zieldefinition und –dokumentation keine Verbesserungsnotwendigkeit gegeben sei.

8.2.2 Verbesserungsmaßnahmen

Aufgrund dessen, dass das Projektteam kein Verbesserungspotential hinsichtlich der Zieldefinition und –dokumentation identifizieren konnte, wurden keine Verbesserungsmaßnahmen definiert.

8.3 Systemkontext

Das dritte Thema des ersten Workshops befasste sich mit der Definition und der Dokumentation des Systemkontexts. Im Gegensatz zu den Stakeholdern und den Projektzielen wird der Systemkontext in den Projekten bereits in einfacher Art und Weise beschrieben, weshalb ein sehr kurzer theoretischer Einstieg in das Thema ausreichte. Anhand eines Beispiels wurde ein System von seinem Kontext und der irrelevanten Umgebung abgegrenzt. Danach wurden die wesentlichen Gründe für die Definition

und Dokumentation des Systemkontexts genannt. Am Ende des Theorieteils wurde erläutert, wie bei der Definition des Systemkontexts vorgegangen wird und die folgenden Alternativen für die Dokumentation im Detail beschrieben:

- Natürlichsprachliche Beschreibung
- Beschreibung in Form von Tabellen
- Grafische Darstellung in Form von:
    - Klassendiagramm/Komponentendiagramm der UML
    - Erweiterte Anwendungsfälle – USE-CASES
    - Kontextdiagramm der strukturierten Analyse

Im Anschluss an die kurzen theoretischen Erläuterungen zur Systemkontextabgrenzung erfolgte die Diskussion des Themas.

8.3.1 Diskussion und Verbesserungspotentiale

Im Zuge der Diskussion innerhalb des Projektteams stellte sich heraus, dass die grundlegende Systemkontextabgrenzung in ihrer derzeitigen Form gut funktioniert. Der Systemkontext wird dabei in der Vertriebsphase von der Projektierung definiert, um auf Basis des Arbeitsauftrages den Angebotspreis abschätzen zu können. Diese Grobplanung wird in der anschließenden Feinplanung detailliert. Die grundlegende Vorgehensweise bei der Kontextabgrenzung sollte demnach beibehalten werden, allerdings konnten bereits einige Verbesserungspotentiale hinsichtlich der Schnittstellendokumentation identifiziert werden, die jedoch im Rahmen der Anforderungsdokumentation näher betrachtet werden sollten.

8.3.2 Verbesserungsmaßnahmen

Da die Systemkontextabgrenzung in ihrer derzeitigen Form zufriedenstellend durchgeführt wird, wurden keine Verbesserungsmaßnahmen definiert. Verbesserungen hinsichtlich der inhaltlichen Systemkontextbeschreibung (Schnittstellenbeschreibung) werden im Kapitel 9.2.2 beschrieben.

8.4 Glossar

Im Zuge des ersten Workshops wurde neben den genannten Projektgrundlagen auch das Thema „Glossar" besprochen. Im Theorieteil erfolgten eine Definition des Begriffes des „Glossars" sowie die folgende Auflistung von Elementen, die in einem Glossar angeführt werden können:

- Fachbegriffe, die im Dokument vorkommen inklusive verwendeter Synonyme
- Verben, die im Dokument nicht näher erläutert werden
- erklärungswürdige Adjektive (z. B. abgeschlossener Auftrag)
- im Dokument vorkommende Rollen

Neben der üblicherweise angewendeten Form die Begriffe im Glossar frei natürlichsprachlich zu definieren, wurde auch die Möglichkeit vorgestellt, diese anhand einer Satzschablone zu formulieren.

8.4.1 Diskussion und Verbesserungspotentiale

In der, an den theoretischen Vortrag anschließenden, Diskussion zeigte sich, dass alle Workshop-Teilnehmer die Wichtigkeit und den Nutzen eines vollständigen Glossars erkennen. Das Glossar und auch das Abkürzungsverzeichnis sollen helfen, Missverständnisse aufgrund von unterschiedlichen Auffassungen über dieselben Begriffe zu vermeiden und allen Projektbeteiligten helfen, sich in den Projekten schneller zurechtzufinden. Grundsätzlich war daher auch bisher das Anlegen eines Glossars in den Spezifikationen vorgesehen, aufgrund des Zeitdrucks in den Projekten, erfolgte dies jedoch bis dato nur in den seltensten Fällen. Zudem wurde weder das Vorhandensein, noch die Qualität des Glossars bisher überprüft. Bei Projekten, bei denen ein Glossar angelegt wurde, erfolgte dies erst nach Abschluss der Spezifikation. Zudem fanden sich ausschließlich Substantive in den Glossaren.

8.4.2 Verbesserungsmaßnahmen

Aufgrund des breiten Konsenses konnten bezüglich des Glossars rasch einfache Verbesserungsmaßnahmen festgelegt werden. Das Glossar soll zukünftig von Beginn an in den Projekten geführt und sukzessive erweitert und ergänzt werden. Im Glossar können neben Substantiven auch erklärungswürdige Adjektive und Verben angeführt werden. Zudem sollten auch in der Spezifikation vorkommende projektspezifi-

sche Rollen im Glossar definiert werden. Werden Begriffe im Text der Spezifikation erklärt, müssen diese nicht in das Glossar aufgenommen werden. Um die Übersichtlichkeit und Lesbarkeit zu fördern, werden die Begriffe im Glossar alphabetisch geordnet.

Für jedes Projekt muss zukünftig ein Glossar angelegt werden. Dies gilt ebenfalls für das getrennt vom Glossar angelegte Abkürzungsverzeichnis. Der Definitionstext kann in beiden Verzeichnissen frei gewählt werden, die Verwendung einer Satzschablone wurde von den Workshop-Teilnehmern einstimmig abgelehnt.

8.5 Anforderungen ermitteln

Das fünfte und damit letzte Thema, dass beim ersten Workshop des Requirements-Engineering-Verbesserungsprojektes behandelt wurde, war die Ermittlung der Kundenanforderungen. Wie bereits im ersten Teil der vorliegenden Masterarbeit angeführt wurde, ist dieses Thema von der Verfasserin bereits in einer Bachelorarbeit ausführlich theoretisch aufbereitet und behandelt worden. Da diese Bachelorarbeit bereits vorgestellt wurde und sie den Teilnehmern vorliegt, erfolgten im Workshop nur ein kurzer theoretischer Input und die Vorstellung der Ergebnisse der Arbeit als Ausgangspunkt für die Diskussion des Themas, das Auffinden von Verbesserungspotentialen und die anschließende Ableitung von Verbesserungsmaßnahmen. Im Workshop wurden die folgenden Ermittlungstechniken vorgestellt und kurz erläutert:

- Beobachtung
- Schriftliche Befragung
- Perspektivenbasiertes Lesen und Befragen
- Workshops
- Interviews
- Szenario-Techniken/grafische Darstellungsformen (Use-Cases, BPMN 2.0)
- Assistenztechniken
    - Brainstorming
    - Mind Mapping
    - Gruppenmedien (Projektwebseite, Projektwiki)
    - Essenzbildung
    - Planguage

Anschließend wurde den Workshop-Teilnehmern das Ergebnis der Bachelorarbeit in Form einer Matrix, die die Eignung der jeweiligen Technik in Abhängigkeit von den Projektrahmenbedingungen bewertet, gezeigt.

Zusätzlich zu dem Workshop, bei dem das Thema „Anforderungserhebungstechniken" im Vordergrund stand, befragte die Verfasserin der vorliegenden Arbeit Projektierungsverantwortliche aus der Abteilung „Project Engineering" und technische Projektleiter im Rahmen separater Besprechungstermine, um ein möglichst umfassendes Bild von der IST-Situation der Anforderungsermittlung in der Grob- und in der Feinplanung zu erhalten und daraus Verbesserungspotentiale und –empfehlungen ableiten zu können.

8.5.1 Diskussion und Verbesserungspotentiale

Im Rahmen der Diskussion über das Thema „Anforderungsermittlung" waren sich die Workshop-Teilnehmer auf Basis ihrer Erfahrung einig, dass es nachteilig wäre, die technischen Projektleiter bei der Wahl der zur Verfügung stehenden Anforderungsermittlungs-Techniken zu stark einzuschränken. Dieser subjektive Eindruck wurde durch die vorgestellte und im vorherigen Absatz erwähnte Empfehlungsmatrix bestätigt, da sie zeigt, dass die Eignung einer bestimmten Ermittlungstechnik sehr stark von den jeweiligen Projektrahmenbedingungen abhängt. Darüber hinaus wird die Entscheidung für die Anwendung einer Technik wesentlich von der Erfahrung des technischen Projektleiters und seinem Wissen über den Einsatz der Technik beeinflusst. Hier zeigte sich im Zuge der Diskussion, dass sich die Teilnehmer zwar der Tatsache bewusst sind, dass sich verschiedene Techniken unter bestimmten Bedingungen besser eignen als andere, in der Praxis jedoch meist auf die gewohnten Techniken zurückgegriffen wird. Als Grund dafür, neue bzw. ungewohnte Techniken nicht anwenden zu wollen, wurde das mangelnde Wissen über den konkreten Einsatz der verschiedenen Techniken und der hohe Zeitdruck in der Analyse- und Spezifikationsphase, der die Motivation für das „Ausprobieren" einer ungewohnten Technik senkt, genannt.

Einen weiteren Diskussionspunkt im Rahmen der Anforderungserhebung stellte die Zusammenarbeit mit der Abteilung „Project Engineering" (Projektierung) dar, da diese die Kundenanforderungen bereits in der Grobplanung auf abstrakter Ebene erhebt und damit eine wesentliche Quelle der Kundenanforderungen für die Feinplanung darstellt. Wie bereits in Kapitel 7.1 beschrieben, sind die Qualität und die Quantität dieser Anforderungen stark variabel und hängt wesentlich von Ausschreibungs-

unterlagen des Kunden und der, für die Projektierung zur Verfügung stehenden Zeitdauer ab. Für die Übergabe der, von der Projektierung und dem Vertrieb erstellten Dokumente (technische Spezifikation inkl. logistischen Konzepts, Lagerlayout etc.) an das „Software Engineering" gibt es zwar einen grob definierten Standardprozess der vorgibt, diese Dokumente im Rahmen eines Besprechungstermins zu übergeben, dieser wird jedoch in der Realität noch nicht bei jedem Projekt umgesetzt. Zudem ist die Projektierung üblicherweise ab dem Zeitpunkt der Übergabe des Projektes an den technischen Projektleiter nicht mehr im vollen Umfang aktiv in das Projekt und die Anforderungserhebung miteingebunden und wird nur mehr bei offenen Fragen kontaktiert. Diese recht scharfe Trennung zwischen Grob- und Feinplanung führt immer wieder zu Verzögerungen oder Missverständnissen wenn Details über den verkauften Lieferumfang oder die in der Grobplanung definierten Prozesse vom technischen Projektleiter erst mühsam nachgefragt werden müssen bzw. der technische Projektleiter sich nicht-dokumentiertes Projektwissen, das der Projektierungsverantwortliche durch die Grobplanung der Anlage bereits gewonnen hat, erst aneignen muss.

Bezüglich der Kommunikation zwischen der Projektierungsabteilung und dem „Software Engineering" (zwischen Grob- und Feinplanung) ist außerdem anzumerken, dass es kein direktes Feedback des technischen Projektleiters an die Software- oder Layout-Projektierung gibt. Es gibt zwar ein formales Projekt-Review, das vom Projektmanager erstellt und vom technischen Projektleiter kommentiert wird, dieses wird jedoch nur für große Projekte und/oder Projekte erstellt, bei denen es zu großen technischen oder organisatorischen Problemen gekommen ist. Dieses Review-Dokument wird im Projektordner archiviert und fallweise besprochen. Bei einer zufällig ausgewählten Stichprobe von sechs abgeschlossenen Projekten, war das Review-Dokument nur für drei Projekte angelegt worden, wobei bei zwei dieser drei Projekte nur das finanzielle Ergebnis betrachtet wurde. Damit fehlt also das Feedback an die Grobplanung, in welcher Qualität diese ausgeführt wurde bzw. welche Lagerprozesse aus der Grobplanung problemlos in die Feinplanung übernommen werden konnten.

8.5.2 Verbesserungsmaßnahmen

Die aus den identifizierten Verbesserungspotentialen abgeleiteten Verbesserungsmaßnahmen gliedern sich in zwei Teilbereiche:

1. Weiterentwicklung des Know-hows der technischen Projektleiter bezüglich der Anforderungsermittlungstechniken
2. Verbesserung der Kommunikation mit der Abteilung „Project Engineering“ (Grobplanung)

In Bezug auf den ersten Teilbereich der Verbesserungsmaßnahmen sollen Schulungen das theoretische und praktische Wissen der technischen Projektleiter fördern. Aufgrund der großen Anzahl an verschiedenen Ermittlungstechniken ist es dabei sinnvoll, sich in kurzen Schulungseinheiten jeweils einer Technik zu widmen. Praxisorientiert sollen dabei die optimalen Einsatzbedingungen der jeweiligen Technik und deren richtige Anwendung anhand von praktischen Beispielen vermittelt werden. Auch wenn, wie bereits erwähnt, eine Einschränkung der anzuwendenden Techniken schwer fällt, sollten vorrangig jene Methoden geschult werden, die gut zur späteren Anforderungsspezifikation passen. Da beispielsweise BPMN 2.0 bei KNAPP für die grafische Prozessdarstellung verwendet wird, ist es sinnvoll, die Verwendung dieser Darstellungsmethode im Rahmen der Anforderungsermittlung zu schulen. Diese Schulungen sollen die technischen Projektleiter dabei unterstützen, die beste Kombination an Anforderungserhebungstechniken für die jeweilig vorliegende Projektsituation auszuwählen und anzuwenden.

Bezüglich der Verbesserung der Kommunikation mit dem „Project Engineering“ ergeben sich nach ausführlicher Diskussion mit Mitarbeitern des „Project Engineerings“, als auch des „Software Engineerings“ mehrere potentielle Verbesserungsmaßnahmen:

- Der Übergabeprozess von der Grob- an die Feinplanung muss bei jedem Projekt im Rahmen eines gemeinsamen Besprechungstermins erfolgen. Die technischen Projektleiter müssen sich mithilfe der Projektierungsunterlagen auf diesen Besprechungstermin vorbereiten, um zielgerichtet Fragen stellen zu können. Das Risiko für Missverständnisse und nicht übertragenes Projektwissen kann dadurch gesenkt werden.
- Die derzeit im KNAPP Managementsystem vorliegende Beschreibung der, von der Projektierung zu erstellenden und dem technischen Projektleiter zu übergebenden Dokumente, muss überarbeitet und aktualisiert werden. Dies betrifft die Art (z. B. Layout, technische Spezifikation, logistisches Konzept) und den Inhalt der Dokumente. Bezüglich des Inhalts muss definiert werden, welche Informationen der

technische Projektleiter von der Projektierung bzw. dem Vertrieb benötigt, um sowohl mühsame nachträgliche Informationsbeschaffung durch den technischen Projektleiter, als auch unnötige Ressourcenaufwände in der Projektierung zu vermeiden.

- Die Trennung zwischen Grob- und Feinplanung sollte im Sinne einer ganzheitlichen Prozessbetrachtung fließender verlaufen. Bei einigen Projekten hat es sich bereits als sehr vorteilhaft erwiesen, den technischen Projektleiter schon in der Vorverkaufsphase in das Projekt einzubinden, da er somit seine operative Projekterfahrung möglichst früh einbringen kann. Zumindest bei komplexen Projekten sollte dies angestrebt werden.

  Von großem Nutzen für den Projekterfolg wäre es außerdem, den bzw. die Projektierungsverantwortlichen zumindest in die Analysephase der Feinplanung miteinzubeziehen. Dadurch kann bei der Anforderungserhebung sofort geklärt werden, welche vom Kunden gewünschte Funktionalität zum verkauften Leistungsumfang gehört und das Hintergrundwissen aus der Grobplanung direkt in die Feinplanung der Anlage einfließen. Es entsteht dabei eine für beide Seiten vorteilhafte Situation, da der Projektierungsverantwortlich die, in der Feinplanung gesammelte, Erfahrung positiv in die zukünftige Grobplanung einfließen lassen kann.

  Der zusätzliche Ressourcenaufwand dieser Maßnahmen lässt sich durch die vereinfachte direktere Kommunikation der beiden Abteilungen, die Generierung von Synergieeffekten, die effektivere Nutzung der eingesetzten Ressourcen und die erhöhte Qualität des Ergebnisses der Analysephase rechtfertigen.

- Besonders wenn der fließende Übergang von der Grob- in die Feinplanung bei einem Projekt nicht erfolgt, sollte es nach dem Ende der Implementierungsphase des Projektes eine Nachbesprechung mit der Projektierung geben. Dabei sollten vorrangig jene Prozesse besprochen und dokumentiert werden, die in der Feinplanung verändert oder neu mit dem Kunden gestaltet werden mussten. Zusätzlich würde dieses Review auch die Möglichkeit bieten, zusammen mit dem Projektierungsverantwortlichen die Informationsqualität und –quantität der Grobplanung zu bewerten, um daraus Verbesserungspotentiale und –maßnahmen für zukünftige Projekte abzuleiten.

9 Anforderungen formulieren

Der zweite Themenblock des Requirements-Engineering-Verbesserungsprojektes befasste sich mit den verschiedenen Möglichkeiten der Dokumentation der Kundenanforderungen in den Spezifikationen. Im Wesentlichen wurden dabei die folgenden Themen behandelt:

- Natürlichsprachliche Anforderungen
- Grafische Darstellungsarten für Anforderungen
- Granularität von Anforderungen
- Nicht-funktionale Anforderungen

Es wurde von der Verfasserin der vorliegenden Arbeit die gleiche Vorgehensweise wie beim ersten Themenblock angewandt. Es wurde eine intensive Literaturrecherche zu den genannten Themen durchgeführt und die aufbereiteten Ergebnisse dieser Recherche den Verbesserungsprojekt-Beteiligten im Rahmen eines Workshops am 12. November 2015 vorgestellt. Ziel dieses zweiten Workshops war es, auf Basis der vorgetragenen Theorie, Verbesserungspotentiale zu identifizieren und Methoden für die zukünftige Verwendung auszuwählen.

9.1 Natürlichsprachliche Anforderungen

Die natürlichsprachliche Beschreibung der Prozessabläufe und der sonstigen Anforderungen an die zu entwickelnde Lagersoftware war, neben der tabellarischen Darstellung, bisher die häufigste Form der Spezifikation bei der KNAPP Systemintegration.

Um den Workshop-Teilnehmern die Probleme bewusst zu machen, die beim unstrukturierten niederschreiben der Kundenanforderungen auftreten können, wurde am Beginn des Theorieteils auf die sprachlichen Effekte (Transformationen) eingegangen, die bewusst oder unbewusst bei der Wahrnehmung (Wahrnehmungstransformation) und dem sprachlichen Ausdruck (Darstellungstransformation) vorkommen. Anhand von anschaulichen Beispielen wurden den Teilnehmern, die durch diese Transformation entstehenden sprachlichen Defekte, Tilgung, Generalisierung und Verzerrung nähergebracht, um eine Basis für das Verständnis der nachfolgend erläuterten Maßnahmen zur Auffindung bzw. Vermeidung dieser Defekte zu schaffen.

Im Anschluss wurden drei Ansätze präsentiert, die mit unterschiedlichem Formalisierungsgrad dem Zweck dienen, syntaktische und/oder semantische Fehler in den natürlichsprachlichen Anforderungen zu vermeiden: Selbstprüfung/Berücksichtigung von Formulierungsregeln, Spezifikation anhand eines Formulars und die Formulierung anhand von Satzschablonen.

Regeln

Zunächst wurden die wichtigsten Grundregeln für das Formulieren in natürlicher Sprache erläutert. Dazu gehören beispielsweise das Beschreiben von Anforderungen in vollständigen Sätzen oder das Vermeiden von Homonymen und Synonymen sowie das Eintragen von Begriffen in das Glossar und das Abkürzungsverzeichnis.

Danach wurden dem Projektteam zahlreiche Regeln vorgestellt, mithilfe derer die bereits formulierten Anforderungen durch den Anforderungsautor selbst oder in weiterer Folge durch einen Reviewer auf sprachliche Defekte geprüft werden können. Im Laufe der Zeit und mit der Übung werden die Regeln dann vom Anforderungsautor schon während der Spezifikation oder sogar durch gezieltes Nachfragen bereits während der Anforderungsermittlung angewandt. Die Regeln betreffen unterschiedliche Elemente der Spezifikation. Einige Regeln fokussieren Satzbestandteile (z. B. Mengen/Häufigkeiten, Eigenschaften), andere betreffen die einzelnen Sätze und wiederum andere Regeln prüfen das Gesamtbild der Spezifikation. Die nachfolgende Auflistung zeigt die im Workshop vorgestellten Regeln:[238]

1. Anforderungen im Aktiv formulieren
2. Prozesse durch eindeutige Vollverben ausdrücken
3. Nominalisierungen auflösen
4. Funktionsverbgefüge auflösen
5. Für jedes Prozesswort genau einen Anforderungssatz schreiben
6. Fehlende Informationen zum Verb analysieren
7. Fehlende Informationen zu beschriebenen Eigenschaften hinterfragen
8. Eigenschaftswörter mess- und testbar formulieren
9. Eigene Anforderungen für nicht-funktionale Aspekte formulieren

[238] vgl. (Rupp & die SOPHISTen, 2014, S. 123 ff.)

10. Verwendete Zahl- und Mengenwörter hinterfragen
11. Verwendete Zahl- und Mengenwörter klären
12. Schwammige Substantive hinterfragen
13. Mögliches und Unmögliches klären
14. Nebensätze extrahieren, die für die Anforderung nicht notwendige Information enthalten
15. Floskeln und redundante Informationen in den Anforderungen vermeiden
16. Abweichungen vom Normalverhalten klären
17. Unvollständige Bedingungsstrukturen analysieren
18. Implizite Annahmen analysieren

Für jede der genannten Regeln wurden im Workshop Beispiele aus dem Projektalltag bzw. aus vorhandenen Spezifikationen vorgestellt und durch die Verbesserung der Anforderung ihre Anwendung veranschaulicht. Des Weiteren wurde von der Verfasserin der vorliegenden Arbeit darauf hingewiesen, dass eine gleichzeitige Einführung aller 18 Regeln kontraproduktiv wäre, da die Berücksichtigung bzw. Prüfung aller Regeln in der kurzen Spezifikationsphase der Kundenprojekte, die technischen Projektleiter überfordern würde. Aufgrund dessen wurde eine schrittweise Einführung der Regeln vorgeschlagen, wobei nicht mehr als sechs Regeln gleichzeitig eingeführt werden sollten. Die nachfolgende Diskussion des Regelkonzepts und die Vorgehensweise bei der Auswahl der Regeln für den ersten Einführungsschritt ist in Kapitel 9.1.1 beschrieben.

Formular

Im Anschluss an die Formulierungsregeln wurde die Möglichkeit vorgestellt, die Anforderungen im Rahmen eines Formulars zu spezifizieren. Das Formular gibt dabei eine definierte Anzahl von Attributen vor, die vom Anforderungsautor für jede Anforderung ausgefüllt werden müssen. Diese Art der Spezifikation würde auch von dem in Verwendung befindlichen Requirements-Engineering-Tool „Polarion ALM Requirements“ unterstützt. Zum besseren Verständnis wurden den Workshop-Teilnehmern die folgenden Attribute als Beispiele genannt:

- Vorbedingung
  Vorbedingungen, die für das Ausführen/Eintreten der Anforderung erfüllt sein müssen.
- Funktionalität
  Die Beschreibung der Anforderung.
- Erwartetes Ergebnis
  Das Ergebnis/der Endzustand, der nach Ausführen der Funktion eintreten soll.
- Begründung
  Geschäftsprozess/Vorgabe, die die Anforderung begründet.

Anschließend wurde die Anwendung des Formulars anhand eines Beispiels illustriert. Dabei wurde ein Kommissionier-Prozess mit den genannten Attributen spezifiziert.

Anforderungsschablonen

Als dritte Alternative zur Beschreibung natürlichsprachlicher Anforderungen wurde die Verwendung von Satzschablonen vorgestellt. Dabei wird der Text der Anforderung Satz für Satz nach einem vorgegebenen Muster formuliert. Da diese Art der Spezifikation sehr ungewohnt ist und die syntaktische Freiheit des Anforderungsautors mehr oder minder stark beschränkt wird, stehen viele Requirements-Engineers Anforderungsschablonen eher skeptisch gegenüber. Aufgrund dieser Tatsache wurden am Beginn dieses Theoriekapitels die wichtigsten Argumente für den Einsatz dieser Spezifikationsmethode erläutert. Dazu gehört beispielsweise, dass viele sprachliche Defekte sofort vermieden werden können und die syntaktischen Unterschiede bei mehreren Autoren vermindert werden.

Anschließend wurden den Workshop-Teilnehmern verschiedene Arten von Satzschablonen vorgestellt. Begonnen wurde mit einer einfachen Funktionsbeschreibung ohne Bedingungsstruktur. Um dem Projektteam die praktische Anwendung näherzubringen, wurde die Schablone in drei Beispielsätzen angewandt. Anschließend wurde die schablonenbasierte Formulierung einer Anforderung, die an eine logische oder zeitliche Bedingung geknüpft ist, gezeigt. Auch diese Schablonentypen wurden durch zahlreiche Satzbeispiele veranschaulicht. Zusätzliche wurde den Teilnehmern noch eine kurze, mithilfe unterschiedlicher Schablonen spezifizierte Prozessbeschreibung präsentiert. Dabei sollte den Teilnehmern vor Augen geführt werden, wie die unterschiedlichen Schablonentypen zu einer vollständigen und lesbaren Prozessbe-

schreibung kombiniert werden können. Danach wurden die Satzschablonen für das englischsprachige Spezifizieren gezeigt, da ein großer Teil der Spezifikationen der KNAPP Systemintegration aufgrund des hohen Exportgrades in englischer Sprache verfasst werden. Abschließend wurde erläutert, wie die einzelnen Schablonentypen weiter verfeinert und detailliert werden können. Durch zusätzliche Adjektive und Adverbien oder der näheren Beschreibung des Prozesswortes, können die zu oberflächlich formulierten Anforderungen konkretisiert werden. Auch dieses Vorgehen wurde durch Beispiele veranschaulicht.

Damit endete der Theorievortrag zum Thema „natürlichsprachliche Formulierung von Anforderungen“ und die Diskussion über potentielle Verbesserungsmaßnahmen konnte beginnen.

9.1.1 Diskussion und Verbesserungspotentiale

Am Beginn der Diskussion zeigt sich, dass den Workshop-Teilnehmern das Konzept der Wahrnehmungs- und Darstellungstransformation und die dadurch entstehenden sprachlichen Effekte zwar bereits bekannt waren, die praktische Anwendung bzw. Berücksichtigung dieses Wissens in der Anforderungsspezifikation und auch der Anforderungsermittlung schwer fällt. Speziell dadurch, dass ein technischer Projektleiter über einen langen Zeitraum mit dem jeweiligen Kundenprojekt beschäftigt ist, eignet er sich viel implizites Wissen an. Aufgrund dessen fällt es dann sehr schwer, die Anforderungen dennoch klar, vollständig und damit auch für „projektfremde“ Personen (z. B. Entwickler, Tester, Inbetriebnehmer) verständlich zu spezifizieren.

Die zahlreichen Regeln wurden grundsätzlich mit großem Verständnis aufgenommen, allerdings tat sich das Projektteam sehr schwer mit der Idee, dass der Anforderungsautor seine spezifizierten Anforderungen selbst mithilfe der Regeln prüfen soll. Diesbezüglich waren die Teilnehmer der Meinung, dass der Autor durch das im vorherigen Absatz erwähnte implizite Projektwissen Fehler in den Anforderungen nicht auffinden kann. Die Ausführungen der Verfasserin der vorliegenden Arbeit, wonach auch der Autor selbst durch das fokussierte Prüfen der von ihm formulierten Anforderungen auf eine bestimmte Regel hin (z. B. sind die Anforderungen im Aktiv formuliert?), Fehler auffinden kann und dem Autor durch die Regeln eine Hilfestellung gegeben wird, die ihn beim qualitativ hochwertigen Spezifizieren unterstützen soll, konnten die Workshop-Teilnehmer dann jedoch vom Nutzen der Formulierungsregeln in der Spezifikationsphase überzeugen.

Dennoch waren sich alle Workshop-Teilnehmer einig, dass eine gleichzeitige Einführung aller Formulierungsregeln die Anforderungsautoren zeitlich überfordern würde. Der vorgeschlagenen schrittweisen Einführung der Regeln wurde damit zugestimmt. Für die erste Anwendung der Methode sollten nur einige wesentlichen Regeln angewendet werden, um die Aufmerksamkeit des Anforderungsautors auf besonders wichtige Defekte konzentrieren zu können. Um die wichtigsten Regeln identifizieren zu können, galt es die Regeln zu priorisieren. Die sechs Regeln mit der höchsten Priorität sollten dann im Zuge der Verbesserungsmaßnahmen in einem ersten Schritt umgesetzt werden. Um die Priorisierung der Regeln unter möglichst vielen Gesichtspunkten vornehmen zu können, wurde die Wichtigkeit der Regeln von den Workshop-Teilnehmern im Rahmen einer Umfrage anonym bewertet. Im Vorfeld der Umfrage wurde den Befragten von der Verfasserin der vorliegenden Arbeit ein umfassendes Dokument vorgelegt, in dem alle Regeln im Detail und mit Beispielen aus dem Projektalltag beschreiben wurden, um den Befragten die Möglichkeit zu bieten, sich zusätzlich zu dem Kurzvortrag im Workshop ein vollständiges Bild von jeder einzelnen Regeln machen zu können. Anschließend wurden die Teilnehmer aufgefordert, in einer für diesen Zweck erstellten Online-Umfrage ihre persönlichen, maximal sechs wichtigsten Regeln auszuwählen. Da es aufgrund der hohen Anzahl von Regeln absehbar war, dass der Rücklauf der Umfrage einige Zeit in Anspruch nehmen würde, wurde den Teilnehmern mehrere Wochen Zeit gegeben, um ihre Favoriten zu wählen. Die Wahl jener Personen, die bereits abgestimmt hatten, war für den Teilnehmer nicht einsehbar. Nach Ablauf der Frist wurde die Umfrage ausgewertet. Jene sechs Regeln mit der höchsten Übereinstimmung werden im Rahmen der Verbesserungsmaßnahmen in Zukunft umgesetzt. Die folgende Tabelle zeigt das Ergebnis der Umfrage:

Tabelle 4: Formulierungsregeln für natürlichsprachliche Anforderungen[239]

Formulierungsregel	T1[240]	T2	T3	T4	T5	T6	∑
Anforderungen im Aktiv formulieren	X		X			X	**3**
Prozesse durch eindeutige Vollverben ausdrücken				X	X	X	**3**
Nominalisierungen auflösen			X				**1**

[239] (Rupp & die SOPHISTen, 2014, S. 123 ff.)
[240] T... Teilnehmer

Funktionsverbgefüge auflösen	X					X	**2**
Für jedes Vollverb genau einen Satz schreiben							**0**
Fehlende Informationen zum Vollverb analysieren		X	X	X			**3**
Fehlende Infos zum Eigenschaftswort analysieren, welches sich aus einem Vollverb ableitet							**0**
Eigenschaftswörter mess- bzw. testbar formulieren		X		X			**2**
Eigene Anforderungen für nicht-funktionales formulieren, wenn sie eigenständig behandelt werden sollen oder übergreifend gelten		X				X	**2**
Verwendete Zahl- und Mengenwörter hinterfragen							**0**
Fehlende Zahl- und Mengenwörter klären							**0**
Schwammige Substantive hinterfragen		X	X			X	**3**
Formulierungen ersetzen, die Mögliches oder Unmögliches beschreiben					X		**1**
Nebensätze löschen, die für Anforderungen irrelevant sind	X						**1**
Floskelhafte Wörter und Redewendungen kürzen oder entfernen, die für Anforderungen irrelevant sind	X				X		**2**
Ausnahme vom Normalverhalten des Systems klären und die Anforderungen erweitern bzw. eine neue Anforderung formulieren		X		X	X	X	**4**
Anforderungen auf unvollständige Bedingungsstrukturen überprüfen und diese ausformulieren	X	X	X	X	X		**5**
Für jede nicht beschriebene implizite Annahme eine oder mehrere neue Anforderungen schreiben	X		X	X			**3**

Die aus der Abstimmung abgeleiteten konkreten Verbesserungsmaßnahmen bezüglich der Formulierungsregeln sind im nachfolgenden Kapitel angeführt.

Der nächste Diskussionspunkt betraf die Spezifizierung der Anforderungen anhand von auszufüllenden Attributen in Formularform. Hier zeigte sich, dass von einigen technischen Projektleitern bereits Attribute verwendet und ausgefüllt wurden. Insbesondere betraf dies die „Funktionale Beschreibung“, in der vertiefende Informationen

für den Software-Entwickler dokumentiert wurden. Die Idee, mehrere Attribute zu nutzen wurde von den Workshop-Teilnehmern grundsätzlich positiv aufgenommen, allerdings wurde darüber diskutiert, ob ein verpflichtendes Ausfüllen einer zu großen Anzahl von Attributen mehr Aufwand als Nutzen bedeuten würde. Zudem war es fraglich, ob das Ausfüllen eines Attributs bei jeder einzelnen Anforderung auch sinnvoll möglich ist, da nicht bei jeder Anforderung beispielsweise eine Begründung vorliegt. Da jedoch die Vorteile der Verwendung von Attributen zur näheren Beschreibung der Anforderungen vom Projektteam nicht bestritten wurde, soll im Rahmen der Verbesserungsmaßnahmen eine kleine Auswahl von Attributen definiert werden, deren Verwendung den Anforderungsautoren empfohlen wird.

Hinsichtlich der Anwendung von Satzschablonen zur Anforderungsdefinition, konnte die anfängliche Skepsis der Workshop-Teilnehmer nicht völlig ausgeräumt werden. Trotz der zahlreichen kurzen Beispiele, die im Theorieteil des Workshops vorgestellt wurden, konnten die Teilnehmer kein umfangreiches Bild von der praktischen Anwendbarkeit dieser Spezifikationsalternative gewinnen. Daher wurde beschlossen, einen vollständigen Lagerprozess aus einem bereits abgeschlossenen Kundenprojekt mithilfe der verschiedenen Anforderungsschablonen zu beschreiben, um den möglichen Mehrwert der Verwendung der Satzschablonen und die Praktikabilität dieser Dokumentationsform bewerten zu können. Ein kleiner Ausschnitt der umformulierten Spezifikation ist nachfolgend angeführt.

Order Processing

The operator executes the picking process using the KiSoft WCS RF picking system. The HOST-system determines the pick location. Based on these pick locations, KiSoft WCS CONTROL diverts the carton into the stations, where the operator needs to pick items for the order. As soon as the operator scans the license plate on the shipping label of the carton in the station with the RF terminal, KiSoft WCS CONTROL displays the required items in form of the orderlines of the items, on the RF terminal screen. KiSoft WCS CONTROL provides the operator with the functionality to confirm the picking of the items, one orderline at a time. KiSoft WCS CONTROL does not provide the operator with the functionality to increase the pick quantity.

As soon as the operator has completed all orderlines, the operator pushes the carton back on the main conveyor. This process is not supported by KiSoft WCS CONTROL.

Operator Login

KiSoft WCS CONTROL provides the operator with the functionality to login on the RF terminal. When the operator opens the login, KiSoft WCS CONTROL provides the operator with the ability to enter the operator ID, the operator password and the station ID. If all data is valid, KiSoft WCS CONTROL logs the operator in at the station and activates the picking dialog.

Operator Logout

KiSoft WCS CONTROL provides the operator with the functionality to logout the RF terminal at any point during the picking process.

Short Pick Confirm

KiSoft WCS CONTROL provides the operator with the functionality to decrease the pick quantity before the operator confirms the orderline. As soon as the operator confirms the orderline with the decreased pick quantity, KiSoft WCS CONTROL books the decreased pick quantity to the carton and confirms the decreased pick quantity to the HOST-system.

9.1.2 Verbesserungsmaßnahmen

Der erste Teil der Verbesserungsmaßnahmen bezieht sich auf die Formulierungsregeln, die den technischen Projektleitern helfen sollen, Auffälligkeiten in den von ihnen verfassten Spezifikationen selbst aufzudecken und korrigieren zu können. Wie in der im vorherigen Kapitel abgebildeten Tabelle ersichtlich, wurden die sechs Formulierungsregeln mit den meisten Stimmen für den ersten Einführungs-Schritt ausgewählt:

Tabelle 5: Finale Auswahl der Formulierungsregeln

Formulierungsregel	T1[241]	T2	T3	T4	T5	T6	∑
Anforderungen im Aktiv formulieren	X		X			X	**3**
Fehlende Informationen zum Vollverb analysieren		X	X	X			**3**
Schwammige Substantive hinterfragen		X	X			X	**3**
Ausnahme vom Normalverhalten des Systems klären und die Anforderungen erweitern bzw. eine neue Anforderung formulieren		X		X	X	X	**4**

[241] T ... Teilnehmer

Anforderungen auf unvollständige Bedingungsstrukturen überprüfen und diese ausformulieren	X	X	X	X	X		**5**
Für jede nicht beschriebene implizite Annahme eine oder mehrere neue Anforderungen schreiben	X		X	X			**3**

Die restlichen Regeln sollen, sobald sich die Anwendung der ersten Regeln etabliert hat, sukzessive eingeführt werden. Der Inhalt und die Anwendung der Regeln sollen den technischen Projektleitern im Rahmen einer Schulung näher gebracht werden. Danach werden die Projektleiter bei der konkreten Anwendung in den Projekten bei Bedarf unterstützt.

Bezüglich der Beschreibung der Anforderungen anhand von Attributen wurde vereinbart, die in der folgenden Aufzählung angeführten Attribute den Anforderungsautoren zur Verwendung zu empfehlen:

- Vorbedingung
- Funktionelle Beschreibung
- Erwartetes Ergebnis

Auf das Attribut „Begründung“ wird verzichtet, da eine Dokumentation dieses Attributs für jede Anforderung als zu aufwendig bewertet wurde.

Auf die Verwendung von Satzschablonen soll vorerst verzichtet werden, da die durch die Satzschablonen ermöglichten Verbesserungen größtenteils bereits mit der Anwendung der Formulierungsregeln generiert werden können. Trotzdem sollen sie für die Einführung im Rahmen eines zukünftigen Verbesserungszyklus in Betracht gezogen werden.

9.2 Graphische Darstellungsmethoden

Im Anschluss an die Besprechung der natürlichsprachlichen Anforderungsspezifikation wurden im Rahmen des zweiten Workshops des Verbesserungsprojektes Möglichkeiten für die graphische Darstellung von Anforderungen behandelt. Die folgenden sechs Diagramm- bzw. Darstellungsarten wurden behandelt:

- Use-Cases
- Business Process Model and Notation 2.0
- Klassendiagramm als Begriffsmodell

- Aktivitätsdiagramm
- Sequenzdiagramm
- Zustandsdiagramm

Erneut erfolgte zuerst die Vorstellung der einzelnen Methoden in der Theorie bzw. anhand von Beispielen, danach wurden die Methoden besprochen und über ihre Einsetzbarkeit und die konkrete Anwendung diskutiert.

Use-Cases

Als erster Diagrammtyp für grafische Aufbereitung von Anforderungen wurde die Verwendung von Use-Cases für die übersichtliche Darstellung der Funktionen des Systems vorgestellt. Zuerst wurde auf die möglichen Anwendungsfelder dieser Darstellungsmethode eingegangen, bevor anhand eines Beispiels die Elemente von Use-Case-Diagrammen beschrieben wurden.

Business Process Model and Notation 2.0

Danach wurde auf die Darstellung von Business Process Diagrammen mithilfe der BPMN 2.0 eingegangen. Da diese Art der grafischen Modellierung von Prozessen dem Projektteam bereits geläufig war, wurden nur die wesentlichen Charakteristika sowie die Vor- und Nachteile der Methode genannt.

Klassendiagramm als Begriffsmodell

Anschließend wurde eine, dem Projektteam unbekannte, Alternative der grafischen Darstellung von Anforderungen vorgestellt: die Verwendung des Klassendiagramms der UML zur übersichtlichen Abbildung von Begriffen, ihren Attributen und ihren Abhängigkeiten. Neben dem Hinweis, dass das bekannte Klassendiagramm der UML für seine Verwendung als Begriffsmodell vereinfacht und somit ohne Methoden verwendet wird, wurden im Workshop die wesentlichen Vor- und Nachteile der Anwendung dieses Modells genannt. Anschließend wurde die Methode durch ein konkretes Beispiel veranschaulicht.

Aktivitätsdiagramm

Als weitere Alternative zur grafischen Darstellung von Prozessen wurde das Aktivitätsdiagramm erläutert. Zuerst wurde auf die wesentlichen Eigenschaften dieses Diagrammtyps eingegangen, danach wurden die Elemente des Diagramms und seine praktische Anwendung anhand eines Beispielprozesses beschrieben.

Sequenzdiagramm

Diese, speziell für Schnittstellen geeignete Visualisierungsmethode war den Workshop-Teilnehmern bisher in ihrer praktischen Anwendung unbekannt, daher wurde das Sequenzdiagramm etwas detaillierter beschrieben. Anhand eines anschaulichen Beispiels wurde die Anwendung des Diagramms für die Darstellung der Kommunikation zwischen Systemen bzw. System und Nutzer erläutert. Danach wurde explizit auf die Vor- und Nachteile der Methode eingegangen.

Zustandsdiagramm

Das Zustandsdiagramm bildete den Abschluss des Theorieteils zum Thema „Graphische Darstellungsmethoden". Wie auch schon bei den vorhergegangenen Diagrammtypen wurden zuerst die wesentlichen Charakteristika von Zustandsdiagrammen erläutert, bevor die praktische Anwendung dieser Darstellungsalternative anhand eines Beispiels illustriert wurde. Zum Schluss wurden den Workshop-Teilnehmern noch die Vor- und Nachteile der Anwendung dieser eher unüblichen Darstellungsmethode erläutert.

9.2.1 Diskussion und Verbesserungspotentiale

Der Großteil der vorgestellten graphischen Darstellungsmethoden wurde von den Workshop-Teilnehmern sehr positiv aufgenommen. Die erläuterten Vorteile der Visualisierung von Anforderungen wurden von den Teilnehmern, auch auf Basis ihrer bisherigen Projekterfahrung, bekräftigt, demnach wurde konstruktiv diskutiert, wie die Diagramme in den Projekten praktisch angewendet werden könnten.

Das Use-Case-Diagramm wurde schnell für die konkrete Verwendung ausgeschlossen, da die damit visualisierbaren Anforderungen auch mit anderen Diagrammtypen abgebildet werden können. Generell wurde versucht, die Auswahl der zukünftig zu verwendenden Diagrammtypen nicht zu groß werden zu lassen und sich bei Sachverhalten, die mit mehreren verschiedenen Diagrammen dargestellt werden können, auf einen möglichst vielseitigen Diagrammtyp zu einigen, um eventuell nötige Schulungsmaßnahmen auf wenige Diagrammtypen fokussieren zu können. Im konkreten Fall der Use-Case-Diagramme entschieden sich die Teilnehmer dafür, den Überblick über die Funktionen der Software mithilfe der BPMN 2.0 darzustellen.

Die anschließende Diskussion bezüglich der Verwendung der BPMN 2.0 zur Darstellung von Prozessen hielt sich kurz, da seit einigen Jahren angestrebt wird, BPDs als

Firmenstandard für die Prozessvisualisierung zu etablieren. Im Zuge der Diskussion stellte sich heraus, dass die technischen Projektleiter noch nicht im Hinblick auf die korrekte Anwendung der BPMN und den firmeninternen Standard der BPMN geschult wurden.

Die Verwendung von Aktivitätsdiagrammen wurde, zugunsten der BPDs, mit denen Prozessabläufe sehr gut modelliert werden können, verzichtet.

Die Vorstellung des Sequenzdiagrammes wurde von den Workshop-Teilnehmern sehr positiv aufgenommen und es wurde sehr schnell entschieden, diesen Diagrammtyp zukünftig für Schnittstellenbeschreibungen einsetzen zu wollen. Den größten Vorteil des Diagramms sahen die Teilnehmer im chronologisch dargestellten Nachrichtenfluss zwischen den Beteiligten des Prozesses.

Auch das Klassendiagramm in seiner Verwendung als Begriffsmodell wurde von der Gruppe intensiv diskutiert. Auf Basis des vorgestellten Beispiels konnten sich die Teilnehmer des Workshops seinen positiven Beitrag für das Verständnis der Zusammenhänge und Abhängigkeiten von Begriffen erkennen und sie entschieden sich, auch diesen Diagrammtyp zukünftig in die Spezifikation mit aufzunehmen.

Da das Zustandsdiagramm zu den eher unüblichen Darstellungsarten zählt, wurde seine Verwendung auch im Workshop kontrovers diskutiert. Auch wenn mithilfe des Diagramms die möglichen Zustände einer Betrachtungseinheit und ihre Übergänge sehr gut modelliert werden können, wurde aufgrund der geringen Akzeptanz dieser Beschreibungsart auf eine verpflichtende Verwendung in den Spezifikationen von der Gruppe verzichtet.

9.2.2 Verbesserungsmaßnahmen

Aus der Diskussion zum Thema „graphische Darstellungsmethoden“ ergaben sich die folgenden konkreten Verbesserungsmaßnahmen:

Die Etablierung des KNAPP-Standards der BPMN 2.0 wird stärker forciert. Alle technischen Projektleiter werden im Rahmen von Weiterbildungsmaßnahmen für die richtige Anwendung der BPMN geschult. Zukünftig sollen sämtliche Prozessabläufe ausschließlich in BPDs abgebildet werden, bei zwei momentan laufenden Projekten wird dies bereits durchgeführt. Damit die Diagramme dennoch auch für Kunden, die noch keine Erfahrung mit der BPMN haben, lesbar sind, wird den Spezifikationen ein, von der KNAPP AG zur Verfügung gestellter, Leseleitfaden angehängt.

Auch für die korrekte Erstellung von Klassen- und Sequenzdiagrammen sind Schulungsmaßnahmen geplant. Danach wird die Verwendung des Klassendiagramms zur übersichtlichen Darstellung der begrifflichen Zusammenhänge in der Schnittstelle zum HOST-System verpflichtend. Ebenso sollen zukünftig alle Schnittstellen zu sämtlichen externen Systemen (z. B. Versandsystem) mit Sequenzdiagrammen visualisiert werden. Die Anwendung von Zustandsdiagrammen bleibt optional.

9.3 Granularität von Anforderungen

Nachdem der konkrete Inhalt der Anforderungen untrennbar mit ihrer Strukturierung und ihrem Detaillierungsgrad verbunden ist, wurde auch dieses Thema im Zuge des zweiten Workshops behandelt. Die Bearbeitung des Themas erfolgte jedoch nur in einem Ausmaß, wie es für die natürlichsprachliche und graphische Spezifikation von Anforderungen notwendig ist. Themen wie die Versionierung und Zustandsverwaltung von Anforderungen wurden demnach nicht behandelt, viel mehr ging es darum, wie die Struktur der Spezifikation prinzipiell aufgebaut werden kann.

Im Theorieteil des Workshops wurden den Teilnehmern daher zuerst die jeweiligen Vor- und Nachteile der Verwendung von natürlicher Sprache und von Diagrammen zur Beschreibung von Anforderung erläutert. Dadurch konnte die darauf folgende Erklärung, dass die Kombination dieser Beschreibungsarten die beste Variante der Anforderungsspezifikation ist, den Teilnehmern gut vermittelt werden.

Anschließend wurde anhand eines konkreten Beispiels aus dem Projektalltag eine Möglichkeit zur Gestaltung der Kapitelstruktur der Spezifikation gezeigt. Dabei wird die erste Hierarchieebene ausgehend von den Grobprozessen (z. B. Wareneingang, Kommissionierung) definiert. Anschließend werden die Prozesse immer feiner aufgeteilt und in weitere Hierarchieebenen unterteilt, bis ein, für die Entwicklung ausreichender, Detaillierungsgrad erreicht ist. Ziel ist es, so wenige Anforderungen wie möglich in der untersten Ebene unter einer Kapitelüberschrift zu versammeln. So soll klar erkennbar sein, in welche Teilprozesse sich die Abläufe gliedern und welche Inhalte in welchem Kapitel zu finden sind. Anhand des Beispiels wurde auch die sinnvolle Verknüpfung von natürlichsprachlicher Beschreibung und graphischer Darstellung von Anforderungen demonstriert. Hier wurde jedoch darauf hingewiesen, dass

Redundanzen durch Beschreibung des gleichen Sachverhalts in Text und Diagramm generell zu vermeiden sind, da dies sonst zu Konsistenzproblemen führen kann.[242]

9.3.1 Diskussion und Verbesserungspotentiale

Da der strukturelle Aufbau der Spezifikation und die Granularität von Anforderungen schon länger Gegenstand von Diskussionen bei der KNAPP Systemintegration ist, wurde auch im Workshop sehr kontrovers darüber diskutiert. Die Einigung auf eine klare Vorgabe bezüglich der Granularität von Anforderungen wird einerseits dadurch erschwert, dass die Freiheit der technischen Projektleiter nicht zu stark eingeschränkt werden soll, zudem variieren auch die Präferenzen des Entwicklungsprojektleiters, der in weiterer Folge mit den einzelnen Anforderungen arbeitet, relativ stark. Während einige Entwicklungsprojektleiter möglichst viele Funktionen eines Themas in einer einzelnen Anforderung gesammelt haben möchten, bevorzugen andere die strikte Aufteilung der einzelnen Funktionen eines Prozesses in getrennte Anforderungen.

Zudem fällt den technischen Projektleitern die Entscheidung schwer, wie die in der Spezifikation beschriebenen Prozesse in einzelne Anforderungen aufzuteilen sind. Diese Aufgabe wird auch durch die bisher angewandte Vorgehensweise bei der Erstellung der Spezifikation erschwert, bei der zuerst das Dokument verfasst und erst danach einzelne Absätze oder ganze Kapitel als Anforderung markiert werden. Daher bestanden die Anforderungen bisher in den meisten Fällen sowohl aus der Beschreibung des Normalverhaltens als auch den Abweichungen des Normalverhaltens und den möglichen Fehlerfällen und deren Behandlung. Diese Vorgehensweise führt zu einer Reihe von Problemen. Zum einen leidet die Übersichtlichkeit, denn es liegt dann am jeweiligen Leser die eigentliche Kern-Anforderung aus der Fülle an beschriebenen Funktionen herauszulesen. Ein weiteres Problem dieser hohen Granularität sind die zwangläufig vorkommenden Redundanzen. Da beispielsweise die potentiell auftretenden Fehlerfälle in jeder Anforderung angeführt werden, anstatt die identischen Fehlerfälle in einer gemeinsamen Anforderung zu sammeln und mit den betreffenden Anforderungen zu verknüpfen, müssen alle Anforderungen, die diese Fehlerfälle beinhalten bei Änderungen überprüft werden, um Inkonsistenzen zu vermeiden. Zudem wird die Validierung der Anforderung erschwert, da die verschiedenen Funktionen innerhalb der Anforderung meist vermischt beschrieben werden und so die Überprüfung z. B. auf vollständige Bedingungsstrukturen schwerer fällt.

[242] vgl. (Rupp & die SOPHISTen, 2014, S. 432 ff.)

Die Anforderungen werden bei der KNAPP Systemintegration nicht in Spezifikationsebenen oder –hierarchien gegliedert. Damit steht jede Anforderung auf derselben Ebene und es ist somit ohne den Blick in das Spezifikationsdokument nicht klar, ob die entsprechende Anforderung einen Überblick über das jeweilige Thema beschreibt oder schon die zu entwickelnde Feinanforderung. Auch für den technischen Projektleiter ist damit nicht klar, an welchen Stellen der Spezifikation er bereits in ausreichendem Detaillierungsgrad spezifiziert hat.

9.3.2 Verbesserungsmaßnahmen

Grundlegend wurde im Rahmen des Workshops die Entscheidung getroffen, auch weiterhin den technischen Projektleitern keine verpflichtenden Richtlinien zur Granularität der Anforderungen vorzugeben, allerdings wird ihnen empfohlen, Fehlerfälle und Abweichungen vom Normalverhalten in getrennten Anforderungen zu beschreiben. Bei der Aufteilung der Lagerprozesse in Teilprozesse wurde im Workshop der folgende Arbeitsablauf beschrieben:

1. Die Basis der Lagerprozesse wird anhand der technischen Spezifikation und der sonstigen Projektierungsdokumente erarbeitet.
2. Die Grobprozesse werden in geringem Detaillierungsgrad graphisch abgebildet.
3. Aus den Grobprozessen werden Teilprozesse abgleitet (die nächste Kapitelebene der Spezifikation).
4. Die Teilprozesse werden in weitere Teilprozesse aufgeteilt und damit die nächsttieferen Kapitelebenen gebildet.
   Dieser Schritt wird so oft wiederholt, bis der vom jeweiligen technischen Projektleiter festgelegte Detaillierungsgrad erreicht ist.
5. Die daraus entstehende Kapitelstruktur wird anschließend mit Anforderungen befüllt. Falls nötig, wird die Struktur im Laufe der Spezifikation angepasst.

Außerdem wird empfohlen, den Anforderungen anhand eines Attributs einen Spezifikationslevel bzw. einen Detaillierungsgrad zuzuordnen. Diese einfache Maßnahme erleichtert das Lesen der Spezifikation und die Arbeit der nachfolgenden Entwicklungs- und Testaktivitäten.

9.4 Nicht-funktionale Anforderungen

Der letzte Teilbereich des Themenblocks „Anforderungen formulieren“ befasste sich mit nicht-funktionalen Anforderungen. Da nicht-funktionalen Anforderungen bei der KNAPP Systemintegration bisher eher wenig Beachtung zu Teil wurde bzw. diese eher unbewusst in die Spezifikationen aufgenommen wurden, erfolgte im Theorieteil des Workshops eine ausführlichere Behandlung dieses Themas. Am Beginn wurde erläutert, was unter dem Begriff der „nicht-funktionalen Anforderungen“ verstanden wird, anschließend wurden zwei Schemata (Volere und ISO/IEC-25000-Standard) vorgestellt, die als Hilfsmittel zur Auffindung von nicht-funktionalen Anforderungen dienen können. Danach wurde auf Methoden zur Erhebung und Spezifikation von nicht-funktionalen Anforderungen eingegangen, wobei den Teilnehmern des Workshops anhand der „Planguage“ eine Alternative zur quantifizierten Beschreibung von nicht-funktionalen Anforderungen vorgestellt wurde.

9.4.1 Diskussion und Verbesserungspotentiale

Die an den Theorieteil anknüpfende Diskussion zeigte relativ schnell, dass nicht-funktionale Anforderungen zukünftig stärker berücksichtigt werden sollen. Einige Teilnehmer führten an, dass nicht-funktionale Aspekte wie beispielsweise Leistungsanforderungen bei einigen Projekten erst sehr spät vom Kunden definiert wurden und es aufgrund ihrer Auswirkung auf bereits entwickelte Software zu zusätzlichen Ressourcenaufwänden und Verzögerungen beim Projektabschluss kam. Dennoch waren die Workshop-Teilnehmer der Meinung, dass es durch die verstärkte Beachtung von nicht-funktionalen Anforderungen nicht zu unnötigen oder unnötig hohen Anforderungen des Kunden kommen soll. Aufgrund dessen wurde die Idee, die verschiedenen Kategorien von nicht-funktionalen Anforderungen in die firmeninterne Standard-Kapitelstruktur der Spezifikation aufzunehmen, wieder verworfen. Am Ende der Diskussion kam die Gruppe jedoch zu dem Entschluss, dass es trotzdem von Vorteil wäre, das Bewusstsein der technischen Projektleiter hinsichtlich nicht-funktionaler Anforderungen zu schärfen.

9.4.2 Verbesserungsmaßnahmen

Es wurde vereinbart, das in Kapitel 1.2.3 der vorliegenden Arbeit vorgestellte, Volere-Schema den technischen Projektleitern als Ideenquelle zur Verfügung zu stellen. Anhand der verschiedenen, im Schema angeführten, Kategorien von nicht-

funktionalen Anforderungen können die technischen Projektleiter dann ergründen, ob bzw. welche nicht-funktionalen Anforderungen im jeweiligen Projekt vom Kunden gefordert werden könnten. Zusätzlich sollen den technischen Projektleitern Informationen und Tipps bezüglich der Erhebung und der Spezifikation dieser Anforderungsart zur Verfügung gestellt werden.

In weiterer Zukunft soll eine sich sukzessiv erweiternde Sammlung von konkreten nicht-funktionalen Anforderungen entstehen, die in den Projekten wiederverwendet werden können. Diese sollen in Anlehnung an die „Planguage“ bereits quantifizierbar attribuiert sein.

10 Anforderungen prüfen

Nachdem nun Verbesserungsmaßnahmen für die Dokumentation der Projektgrundlagen, die Anforderungsermittlung und die Formulierung von Anforderungen definiert wurden, behandelte der dritte und damit letzte Workshop, der im Rahmen des Verbesserungsprojektes abgehalten wurde, die Validierung von Anforderungen.

Wie bereits in der IST-Situation beschrieben, gab es bei der KNAPP Systemintegration bisher keinen definierten Qualitätssicherungsprozess bezüglich des Requirements-Engineerings. Aufgrund dessen wurden den Teilnehmern des Workshops zunächst die wesentlichen theoretischen Grundlagen des Qualitätsmanagements näher gebracht. Zuerst wurde der Qualitätsbegriff nach Definition der International Standard Organisation und die Qualitätsaspekte und -kriterien, die in Bezug auf das Requirements-Engineering die Qualität von Anforderungen und Spezifikationen kennzeichnen, erläutert. Anschließend wurden kurz die beiden Arten der Qualitätssicherung, die konstruktive und die analytische Qualitätssicherung beschrieben. Im darauf folgenden Themenblock des Theorieteils des Workshops wurden den Teilnehmern die folgenden Prüftechniken vorgestellt und die wesentlichen Vor- und Nachteile der jeweiligen Technik genannt:

- Stellungnahme
- Walkthrough
- Inspektion
- Testfälle
- Analysemodell
- Prototypen

Anschließend wurde den Teilnehmern noch die Möglichkeit der Kennzahlenerhebung im Zuge der Validierung von Spezifikationen aufgezeigt und anhand eines konkreten Beispiels veranschaulicht. Schließlich wurde der Qualitätssicherungsprozess im Detail beschrieben. Besonderer Fokus wurde dabei auf die verschiedenen, zu klärenden, Fragestellungen gelegt, die im Falle der Einführung eines definierten Qualitätssicherungsprozesses beantwortet werden müssen.

10.1 Diskussion und Verbesserungspotentiale

Die grundlegende Frage über die Entscheidung, ob ein definierter Qualitätssicherungsprozess für das Requirements-Engineering bei der KNAPP Systemintegration eingeführt werden soll, wurde von den Teilnehmern des Workshops sehr kontrovers diskutiert. Die Kontroverse ergab sich vor allem dadurch, dass einige Teilnehmer den Aufwand für die Einführung eines solchen Prozesses, durch die positiven Auswirkungen einer überlegten und dokumentierten Prüfung von Anforderungen auf die frühere Auffindung von Fehlern und die damit verbundenen höhere Qualität der Spezifikationen und die damit einhergehenden potentiellen Ressourceneinsparungen in den Nachfolgeprozessen (z. B. Entwicklung, Softwaretests), gerechtfertigt sahen. Andere Workshop-Teilnehmer hielten wiederum dagegen, dass der Aufwand für die Einführung eines Qualitätssicherungsprozesses den potentiellen Nutzen übersteigen würde, zumal die positiven Effekte der systematischen Prüfung der Spezifikationen nicht ohne weiteren Aufwand festgestellt werden könnten. Zudem verlangen die Kunden der KNAPP Systemintegration nicht explizit Informationen darüber, wie die Qualität der Spezifikationen geprüft und sichergestellt wird. Dennoch konnte die Mehrheit der Teilnehmer schlussendlich davon überzeugt werden, dass der Aufwand, sich näher mit dem Thema „Qualitätssicherung im Requirements-Engineering“ zu beschäftigen, durch die positiven Auswirkungen auf die Nachfolgeprozesse gerechtfertigt wäre.

10.2 Verbesserungsmaßnahmen

Es wurde beschlossen dem Thema der Qualitätssicherung Zeit und Ressourcen im Rahmen eines eigenständigen Projektes zu widmen, da der Aufwand für die Einführung einer umfassenden und vor allem erfolgsversprechenden Qualitätssicherung den Rahmen des in der vorliegenden Arbeit beschriebenen Verbesserungsprozesses übersteigen würde. Es konnte jedoch bei den Teilnehmern ein grundlegendes Verständnis für die Wichtigkeit und den Nutzen eines strukturierten und standardisierten Vorgehens bei der Validierung von Anforderungen geschaffen werden. In weiteren Workshops sollen daher die, im Theorieteil des Workshops erläuterten, Fragestellungen beantwortet und damit die Grundlagen eines Qualitätssicherungsprozesses festgelegt werden.

Teil III Schlussbetrachtung

Nachdem nun im zweiten Teil der vorliegenden Arbeit die konkrete und praktische Anwendung der in Teil I ausgeführten theoretischen Grundlagen im Rahmen eines Verbesserungsprojektes beschrieben wurden, erfolgt im nachfolgenden dritten Teil der Arbeit eine abschließende Betrachtung des Themas „Requirements-Engineering für Intralogistik Software Projekte“.

11 Conclusio

Das grundlegende Ziel der KNAPP Systemintegration, ist es, das Logistiksystem des Anlagenbetreibers so zu gestalten, dass dieser seine Intralogistikprozesse möglichst effektiv und effizient abwickeln kann und dadurch die stetig steigenden Erwartungen seiner Kunden an die Qualität, die Geschwindigkeit, die Variantenvielfalt und die Kosten der Logistikleistung zu erfüllen. Insbesondere die Intralogistiksoftware steht dabei vor der Herausforderung, immer komplexere Prozesse abbilden und unterstützen zu müssen. Der Planung der Systemstruktur der Intralogistikanlage kommt damit besondere Bedeutung zu, da nicht nur die aktuellen, sondern auch die langfristigen Leistungsanforderungen des Kunden an die Lagerprozesse berücksichtigt werden müssen. Die während der Planungsphase, im Rahmen des Requirements-Engineerings, erhobenen und spezifizierten Anforderungen des Kunden an die zukünftige Intralogistik-Software, bilden die Basis des gesamten weiteren Projektverlaufes. Sie wirken sich unmittelbar auf den Entwicklungsprozess aus und sind die Referenz für die nachfolgenden Software-Tests. Während des gesamten Projektes dienen die Anforderungen als Kommunikations-, Diskussions- und Argumentationsgrundlage. Ihre Aufgabe ist es, das gemeinsame Verständnis und Wissen der Teammitglieder widerzuspiegeln und zu explizieren. Die Anforderungsspezifikation, ist ein zentraler Vertragsbestandteil und die Qualität der Anforderungen kann sich somit wesentlich auf den Projekterfolg, sowohl in fachlicher als auch in finanzieller Hinsicht auswirken. Das Requirements-Engineering als systematischer und disziplinierter Ansatz zur Erhebung, Spezifikation und Prüfung von Anforderungen nimmt damit eine entscheidende Rolle ein.

Das Requirements-Engineering kann jedoch weder als einfaches „Einsammeln“ und Niederschreiben der Anforderungen, noch als wortgetreues Protokollieren der Aussagen der Stakeholder verstanden werden. Viel mehr ist das Requirements-Engineering ein Prozess, der aus zahlreichen Einzelaktivitäten besteht und vom Software-Engineer neben dem fachlichen und methodischen Wissen vor allem analytisches Denken, Empathie, Kommunikationsfähigkeit, Konfliktlösungsfähigkeit und Überzeugungsfähigkeit abverlangt. Das Requirements-Engineering gilt somit als herausfordernder, kritischer, fehleranfälliger und kommunikations-intensiver Prozess, denn häufig ist den Stakeholdern zu Beginn des Projektes noch nicht bewusst, was sie von einem neuen System erwarten. Es ist jedoch die Aufgabe des Software-Engineers die

Stakeholder dabei zu unterstützen, sich über die Rahmenbedingungen und Anforderungen an das zukünftige System klar zu werden. Aufgrund dieser Tatsachen wurden im Laufe der Zeit zahlreiche Methoden und Techniken entwickelt, die den Software-Engineer bei der Erhebung, Spezifikation und Prüfung von Anforderungen unterstützen sollen. Sie alle dienen dem Ziel, mit möglichst geringem Aufwand und angepasst an die jeweiligen Projektrahmenbedingungen ein System zu spezifizieren, das den Stakeholdern möglichst viel Gewinn bringt.

Um zur Erfüllung dieses Ziels beizutragen, erläutert die vorliegende Masterarbeit im ersten Abschnitt zahlreiche Techniken für die Beschreibung der Projektgrundlagen, die natürlichsprachliche und grafische Spezifikation und die Prüfung von Anforderungen und Anforderungsdokumenten. Es wird auf die Charakteristika der jeweiligen Technik eingegangen und praxisorientiert beschrieben, wie die jeweilige Technik konkret in einem Projekt angewendet werden kann. Insbesondere bei den Techniken zur Spezifikation und zur Validierung von Anforderungen hat sich jedoch gezeigt, dass es keineswegs eine einzelne Methode gibt, die für sämtliche Intralogistik-Projekte der KNAPP Systemintegration die richtige ist, vielmehr ist es eine Kombination von Techniken, die abhängig von den Projektrahmenbedingungen, den beteiligten Personen und den fachlichen bzw. inhaltlichen Gegebenheiten geeignet sein können. Daher wurden in diesen beiden Hauptkapiteln Empfehlungsmatrizen angeführt, die die Eignung der zuvor beschriebenen Techniken in Abhängigkeit von den jeweiligen Projektrahmenbedingungen bewerten.

Die im ersten Teil der vorliegenden Arbeit beschriebenen theoretischen Grundlagen wurden anschließend im Rahmen eines Verbesserungsprojektes in der Abteilung „Software Engineering“ bei der KNAPP Systemintegration auf ihre konkrete Anwendbarkeit geprüft. Ziel dieses Verbesserungsprojektes war es, den Requirements-Engineering-Prozess in Zusammenarbeit mit einem abteilungsübergreifenden Projektteam kritisch zu untersuchen, Verbesserungspotentiale zu identifizieren und mithilfe des erlangten theoretischen Wissens, konkrete Verbesserungsmaßnahmen abzuleiten.

Wie im zweiten Teil der vorliegenden Arbeit beschrieben, wurden die jeweiligen Themenbereiche in den Workshops, die im Zuge des Verbesserungsprojektes abgehalten wurden, von der Autorin der vorliegenden Arbeit zunächst auf Basis der durchgeführten Literaturrecherche theoretisch vorgestellt, anschließend wurden in der Gruppe Verbesserungspotentiale und –maßnahmen diskutiert. Da grundlegende Einigkeit

darüber herrschte, dass es Verbesserungspotential im Requirements-Engineering bei der KNAPP Systemintegration gibt, wurde der theoretische Input offen angenommen und die Arbeit mit dem Projektteam gestaltete sich sehr konstruktiv.

Sehr kontrovers und kritisch wurden dabei vor allem jene Themen besprochen, die bisher keine konkrete Anwendung im Requirements-Engineering bei der KNAPP Systemintegration fanden und daher, im Falle ihrer Einführung, größere Veränderungen für die Arbeit der Software-Engineers bedeuten würden. Als Beispiel sei an dieser Stelle die Diskussion über die Einführung eines standardisieren Qualitätssicherungsprozesses für die Prüfung der Anforderungen und Anforderungsdokumente genannt. Diesbezüglich wurden von einigen Projektmitgliedern Bedenken geäußert, wonach der Aufwand für die Etablierung eines derartigen Prozesses dessen Nutzen übersteigen würde. Wie auch bei anderen Themen zeigte sich hier, dass die Konzepte des Requirements-Engineerings den Projektteilnehmern zwar durchaus aus der Theorie bekannt waren, ihre praktische Anwendbarkeit und Sinnhaftigkeit aber immer wieder kritisch betrachtet wurde. Diesbezüglich müsste die Idee, sich an Best-Practice Beispielen aus dem Requirements-Engineering zu orientieren und diese in den Intralogistik-Kontext überzuführen, in Zukunft noch stärker in die Praxis einfließen und eine qualitätsorientierte Denkweise bei den Mitarbeitern weiter gefördert werden.

Auffallend war außerdem die teils vorherrschende Funktionsorientierung. Diese ist aufgrund der strikten Trennung in Vorverkaufs- und Feinplanungsphase besonders in der Zusammenarbeit mit dem „Project Engineering“ klar erkennbar und erschwert einen fließenden Übergang des Projektes von der Grob- in die Feinplanung. Wie in Kapitel 8.5 im Detail beschrieben, führt dies zu einem Projektwissensverlust und ungenutzten Verbesserungspotentialen auf beiden Seiten. Der bei einigen Projekten bereits umgesetzte fließende Übergang von der Grob- in die Feinplanung wurde sowohl vom Project Engineering, als auch von den technischen Projektleitern sehr positiv bewertet und sollte daher, im Sinne der ganzheitlichen Kunden- und Prozessorientierung weiter forciert werden.

Durch das im Zuge der vorliegenden Arbeit durchgeführte Verbesserungsprojekt, konnte ein wichtiger Schritt, in Richtung der Nutzung der Optimierungspotentiale in der Planung von Intralogistikanlagen gesetzt werden, indem die universell einsetzbaren Methoden des klassischen Requirements-Engineerings, explizit für die Intralogistikprojekte der KNAPP Systemintegration, abgeleitet wurden. Es konnten zahlreiche konkrete Verbesserungsmaßnahmen festgelegt werden, die die Qualität der funktio-

nalen Spezifikation verbessern und damit die Projektabwicklung in Zukunft erleichtern sollen. Sie bilden die Basis für weitere Optimierungen im Requirements-Engineering und sollten nach ihrer Einführung gelebt und langfristig weiterentwickelt werden. Neben den positiven Auswirkungen auf die Projektarbeit sollen die Verbesserungsmaßnahmen auch gezielt dazu beitragen, den im Zuge des forcierten Wachstumsprozesses der KNAPP Systemintegration in den kommenden Jahren neu beschäftigten Mitarbeitern den Einstieg in die Projekte und die tägliche Arbeit des Software-Engineers zu erleichtern.

12 Ausblick

Die im Rahmen der vorliegenden Masterarbeit definierten Verbesserungsmaßnahmen sollen, im Laufe der Abwicklung der kommenden Intralogistikprojekte der KNAPP Systemintegration, von den technischen Projektleitern erstmalig angewendet werden. Um den nachhaltigen Erfolg der eingesetzten Maßnahmen zu sichern, müssen die technischen Projektleiter in Hinblick auf die Anwendung der Methoden vorbereitet werden, entsprechende Schulungsmaßnahmen befinden sich bereits in Planung. Im Zuge der Weiterentwicklung der Optimierung des Requirements-Engineerings, sollen die Verbesserungsmaßnahmen laufend evaluiert und auf Basis der gesammelten praktischen Erfahrungen, angepasst werden.

Aktuell wird bereits an der Planung des in Kapitel 10.2 beschriebenen Qualitätssicherungsprozesses gearbeitet. Zudem wird die Verwaltung der spezifizierten Anforderungen hinsichtlich Verbesserungspotentiale und der Ableitung weiterer Verbesserungsmaßnahmen untersucht.

Literaturverzeichnis

Alexander, I. F., & Stevens, R. (2002). *Writing Better Requirements.* Edinburgh: Pearson Verlag.

Alexander, I., & Beus-Dukic, L. (2009). *Discovering Requirements.* Chichester: John Wiley & Sons.

Alper, M. (1994). *Professionelle Softwaretests.* Wiesbaden: Vieweg & Sohn Verlag.

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

Bandler, R., & Grinder, J. (1975). *The Structure of Magic I. A Book about Language and Therapy.* Palo Alto, California: Science and Behavior Books Inc.

Bijan, Y., Yu, J., Stracener, J., & Woods, T. (2013). Systems Requirement Enigneering - State of the Methology. *Systems Engineering, 16*(3), S. 267-276.

Bono, E. d. (1986). *Six Thinking Hats.* Viking: Juan Granica S.A.

Bourne, L. (2009). *Stakeholder Relationship Management: A Maturity Model for Organisational Implementation.* Farnham: Gower Verlag.

Brüggemann, H., & Bremer, P. (2012). *Grundlagen Qualitätsmanagement.* Wiesbaden: Springer Vieweg Verlag.

Charette, R. N. (1990). *Applications Strategies for Risk Analysis.* New York: McGraw-Hill.

Chomsky, N. (1965). *Aspects of the Theory of Syntax.* Cambridge, Massachusetts: The M.I.T. Press.

DeMarco, T. (1979). *Structured Analysis and System Specification.* Englewood Cliffs, New Jersey: Yourdon Press.

DeMarco, T., Hruschka, P., Lister, T., McMenamin, S., Robertson, J., & Robertson, S. (2007). *Adrenalin Junkies & Formular Zombies: Typisches Verhalten in Projekten.* München: Hanser Verlag.

Deming, W. E. (2000). *Out of the Crisis.* Cambridge, MA: MIT Press.

Denger, C., & Olsson, T. (2005). Quality Assurance in Requirements Engineering. (A. Aurum, & C. Wohlin, Hrsg.) *Engineering and Managing Software Requirments*, S. 163-185.

Dörr, J. (2011). Elicitation of a Complete Set of Non-Functional Requirements. *PhD Theses in Experimental Software Engineering*. Stuttgart: Fraunhofer Verlag.

Ebert, C. (2005). *Systematisches Requirements Management.* Heidelberg: dpunkt.verlag.

Emmanuel, T. (Juni 2010). Planguage - Spezifikation nichtfunktionaler Anforderungen. *Informatik-Spektrum, 33*(3), S. 292-295.

Fagan, M. E. (1976). Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal, 15*(3), 182-211.

Finkelstein, A., & Dowell, J. (1996). A Comedy of Errors: the London Ambulance Service case study. *Proceedings of the 8th International Workshop on Software Specification and Design* (S. 2-4). Los Alamitos, CA: IEEE Computer Society Press.

Freund, J., & Rücker, B. (2014). *Praxishandbuch BPMN 2.0.* München: Hanser-Verlag.

Gilb, T. (2005). *Competitive Engineering.* Amsterdam: Elsevier Butterworth, Heinemann.

Goeken, M. (2006). *Entwicklung von Data-Warehouse-Systemen.* Wiesbaden: Deutscher Universitäts-Verlag.

Gottesdiener, E. (2002). *Requirements by Collaboration.* Boston: Pearson Education.

Graham, D. (2002). Requirements and Testing: Seven Missing-Link Myths. *IEEE Software, 19*(5), 15-17.

Grojer, S. (2014). Anforderungserhebungstechniken für Intralogistik Software Projekte. Bachelorarbeit. Leoben: Lehrstuhl für Industrielogistik.

Gudehus, T. (2010). *Logistik: Grundlagen, Strategien, Anwendungen.* Heidelberg: Springer Verlag.

Herrmann, A., & Knauss, E. (2013). *Requirements Engineering und Projektmanagement.* (R. Weißbach, Hrsg.) Berlin Heidelberg: Springer Verlag.

Hompel, M. t., & Schmidt, T. (2008). *Warehouse Management.* Berlin Heidelberg: Springer Verlag.

Hruschka, P. (2011). Effektives Requirements Engineering. Aachen - London - New York: The Atlantic Systems Guild.

Hruschka, P. (2014). *Business Analysis und Requirements Engineering: Produkte und Prozesse nachhaltig verbessern.* München: Carl Hanser Verlag.

Hull, E., Jackson, K., & Dick, J. (2011). *Requirements Engineering.* London: Springer Verlag.

Institute of Electric and Electronic Engineers. (1990). *IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990).* New York: IEEE.

Institute of Electric and Electronic Engineers. (1998). *IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998).* New York: IEEE Computer Society.

Institute of Electric and Electronic Engineers. (2011). *IEEE Systems and Software Engineering - Life Cycle Process - Requirements Engineering (Std 29148-2011).* New York: IEEE.

ISO/IEC 25000 Norm. (2005). Software Engineering - Software product Quality Requirements and Evaluation. International Organization for Standardization.

ISO/IEC 9000 Norm. (2005). Quality Management Systems. International Organization for Standardization.

ISO/IEC 9126. (2001). Software Product Quality. International Organization for Standardization.

Jacobson, I., Christerson, M., & Jonsson, P. (1992). *Object-Oriented Software Engineering. A Use Case Driven Approach.* Amsterdam: Addison-Wesley Longman.

Kamiske, G., & Brauer, J.-P. (2002). *ABC des Qualitätsmanagements.* München: Hanser Verlag.

KNAPP Systemintegration GmbH (Hrsg.). (2015). Geschäfts- und Nachhaltigkeitsbericht 2014/15.

KNAPP Systemintegration GmbH. (2016). Functional Specification KiSoft WCS.

Lamsweerde, A. v. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications.* Chichester: John Wiley & Sons Verlag.

Lawerence, B., Wiegers, K., & Ebert, C. (2001). The Top Risks of Requirements Engineering. *Software, 18*(4), 62-63.

Lawrence, B., Wiegers, K., & Ebert, C. (2001). The top risk of requirements engineering. (IEEE, Hrsg.) *Software, 18*(4), S. 62-63.

Leffingwell, D. (2011). *Agile Software Requirements.* Boston: Pearson Education .

Mullery, G. (1979). CORE - A Method for Controlled Requirements Specification. *4th International Conference on Software Engineering*, (S. 126-135). München.

Niebisch, T. (2013). *Anforderungsmanagement in sieben Tagen.* Berlin Heidelberg: Springer Verlag.

Nuseibeh, B., Kramer, J., & Finkelstein, A. (1994). A Framework for Expressing the Relationship between Multiple Views in Requirements Specification. *IEEE Transactions on Software Engineering, 2*(10), S. 760-773.

Partsch, H. (2010). *Requirments-Engineering systematisch: Modellbildung für softwaregestützte Systeme.* Berlin Heidelberg: Springer Verlag.

Pohl, K. (2007). *Requirements Engineering.* Heidelberg: dpunkt.verlag .

Pohl, K., & Rupp, C. (2011). *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements .* Heidelberg: dpunkt.verlag.

Reynolds, T. J., & Gutman, J. (Feb/Mar 1988). Laddering Theory. Method, Analysis, and Interpretation. *Journal of Advertising Research, 28*(1), S. 11-31.

Robertson, J., & Robertson, S. (2012). *Mastering the Requirements Process.* Upper Saddle River NJ: Addison-Wesley.

Rolland, C., & Salinesi, C. (2005). Modeling Goals and Reasoning with Them. In A. Aurum, & C. Wohlin, *Engineering and Managing Software Requirements* (S. 189-217). Berlin Heidelberg: Springer Verlag.

Rumbaugh, J., Jacobson, I., & Booch, G. (2005). *The Unified Modeling Language Reference Manual.* Boston, Massachusetts: Addison-Wesley.

Rupp, C., & die SOPHISTen. (2013). *Systemanalyse kompakt.* Berlin Heidelberg: Springer Verlag.

Rupp, C., & die SOPHISTen. (2014). *Requirementsengineering und -management.* München: Hanser Verlag.

Rupp, C., Queins, S., & die SOPHISTen. (2012). *UML 2 glasklar: Praxiswissen für die UML-Modellierung.* München: Carl Hanser Verlag.

Savolainen, J. V. (2002). An Integrated Model for Requirements Structuring and Architecture Design. *Proceedings of the Seventh Australian Workshop on Requirements Engineering.* Melbourne: Cybulski, J. L. et al.

Sharma, S., & Pandey, S. K. (August 2013). Revisting Requirements Elicitation Techniques. *International Journal of Computer Applications, 75*(12), S. 35-39.

Simmons, E. (2001). Quantifiying Quality Requirements Using Planguage. *Intel Corporation Quality Week* (S. 1-8). Hillsboro: Intel Corporation.

Sommerville, I. (2012). *Software Engineering.* München: Pearson Verlag.

Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering: A Good Practice Guide.* Chichester: John Wiley & Sons.

Sommerville, I., Sawyer, P., & Viller, S. (1998). Viewpoints For Requirements Elicitation: A Practical Approach. *Proceedings of the IEEE International Confrerence on Requirements Engineering*, (S. 74-81). Colorado Springs.

Srinivasan, B. (24. 03 2008). *What is Stakeholder Analysis? - Part 3*. Abgerufen am 28. 10 2014 von https://leadershipchamps.wordpress.com/2008/03/24/what-is-stakeholder-analysis-part-3/

White, S. A. (2004). Introduction to BPMN. *BPTrends*, 1-11.

Wiegers, K. E. (2006). *More About Software Requirements: Thorny Issues and Practical Advice.* Redmont, Washington: Microsoft Press.

Wiegers, K., & Beatty, J. (2013). *Software Requirements.* Redmond: Microsoft Press.

Zowghi, D., & Coulin, C. (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In A. Aurum, & C. Wohlin (Hrsg.), *Engineering and Managing Software Requirements* (S. 19-46). Berlin, Heidelberg, New York: Springer Verlag.