TECHNISCHE UNIVERSITÄT ILMENAU
Fakultät für Informatik und Automatisierung
Institut für Theoretische und Technische Informatik
Fachgebiet Prozessinformatik

Diplomarbeit zur Erlangung des akademischen Grades
Diplom-Wirtschaftsinformatiker (Dipl.-Wirt.-Inf.)

Konzeption eines Prozessmodells zum Requirements Engineering für individuelle Softwarelösungen

von cand.-Wirt.-Inf. Thomas Havemeister
27376 WI M98

Ilmenau, den 13. Oktober 2003

Betreuende Hochschullehrer:
Univ.-Prof. Dr.-Ing. habil. Ilka Philippow, TU Ilmenau
Dipl.-Inf. Detlef Streitferdt, TU Ilmenau

Betrieblicher Betreuer:
Dr.-Ing. Ingo Schrewe, GFT Systems GmbH

Havemeister, Thomas:
Konzeption eines Prozessmodells zum Requirements Engineering für individuelle Softwarelösungen /
Technische Universität Ilmenau, Fakultät für Informatik und Automatisierung, Diplomarbeit, 2003

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AAO	Auszahlungsanordnung (ELFI)
AD	Aktivitätendiagramm
AG	Auftraggeber
AN	Auftragnehmer
Anf.-Ing.	Anforderungsingenieur
AS	Anforderungsschablone
BUC	Business Use Case
bzgl.	bezüglich
CASE	Computer Aided Software Engineering
CCP	Change Control Process
CM	Change Management
CR	Change Request
DFD	Datenflussdiagramm
DV	Datenverarbeitung
EAI	Enterprise Application Integration
ELFI	Elektronische Formular Integration
EPK	Ereignisprozesskette
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IT	Information Technology
JDBC	JavaDatabase Connectivity
KM	Konfigurationsmanagement
OMG	Object Management Group
OO	Objektorientierung
PL	Projektleiter
PM	Projektmanagement
QS	Qualitätssicherung
R/B	Standardrechnung/-bescheid (ELFI)
RE	Requirements Engineering
REP	Requirements Engineering Process
REP-M	Requirements Engineering Process Model
RUP	Rational Unified Process
RZMB	Rückzahlung Mehrbeträge (ELFI)
SE	Systemerstellung
SK	Steuerungskommitee

SOA ..... Sollabgang (ELFI)
SW ..... Software
SWEBOK ..... Software Engineering Body of Knowledge
UAO ..... Umbuchungsanordnung (ELFI)
UML ..... Unified Modeling Language
VGM ..... Vorgehensmodell
XML ..... extensible markup language
XP ..... eXtreme Programming

1. Einleitung

„Oh, wir haben angenommen, Sie wußten darüber Bescheid. Wir haben das immer so gemacht.“[1]

1.1. Motivation

Nach dem Internetboom Ende der neunziger Jahre befindet sich die IT-Branche in einer Phase der Rückbesinnung. Um langfristig wettbewerbsfähig zu sein, sind stärker als je zuvor die Grundwerte, wie überzeugende Qualität und hohe Kundenzufriedenheit zentraler Ansatzpunkt einer strategischen Neuausrichtung. Für Unternehmen stellt sich die Frage, wie man bei stetig zunehmender Komplexität der Produkte die Entwicklungszeiten verkürzen kann, wenn zugleich die versprochenen Qualitätsansprüche oft nur mit verstärktem Mitteleinsatz zu erreichen sind. Jahre nach dem Boom wird deutlich, dass in der Softwareentwicklung nicht nur Fragen der Auswahl moderner Modellierungswerkzeuge, Entwicklungsumgebungen oder Programmiersprachen relevant sind. Dabei ist die Problematik an sich nicht neu. Als der Bedarf nach hochwertiger Software Mitte der 80'er Jahre explosionsartig anstieg, überstieg die Nachfrage das Angebot bei weitem. Es kam der Begriff „Softwarekrise“ auf. Aus Sicht der Anwender standen enorme Kosten bzgl. Realisierung, Wartung und Weiterentwicklung in einem kaum annehmbaren Verhältnis. Die Entwickler kämpften ihrerseits mit schwer kalkulierbarem Aufwand und einer mangelnden Qualität ihrer Produkte.[2] Die Softwaretechnik (engl.: software engineering), als „ingenieurmäßige Vorgehensweise bei der Systementwicklung“ sollte bei der Auflösung des Dilemmas mit „ineinander greifenden, formalen Techniken für Planung, Analyse, Entwurf und Konstruktion“ endgültig die Problematik beherrschbar gestalten.[3] Wer damals dachte, die Probleme gehören mit jener vorherrschenden „Heimwerker-Mentalität kombiniert mit einem naiven Verständnis für Objektorientierung“[4] der Vergangenheit an, wurde eines Besseren belehrt. Stattdessen brachte die gewaltige Euphoriewelle der ersten aufkommenden CASE-Werkzeuge eher Ernüchterung statt der erhofften Qualitätsschübe.[5] Grund dafür war, neben dem fehlenden methodischen Weitblick und dem Know-how über eine integrierte Vorgehensweise, vor allem

[1] Vgl. [GW93] S.69, Abbildung 6-3
[2] Vgl. [Web92] für einen historischen Überblick und eine Diskussion über die Auswirkungen auf den heutigen Stand der Softwaretechnik.
[3] Vgl. [SH01] S.216 für die zitierten Auszüge.
[4] Vgl. [Oes98] S.23,24
[5] Vgl. [V+02] S.8

die andauernden Streitigkeiten zwischen den Entwicklern und die pure Vielfalt der einzelnen Ansätze.[6] Erst mit der Unified Modeling Language (UML)[7] der Object Management Group (OMG) steht ein universeller Standard zur Verfügung, der zumindest für eine einheitliche Beschreibungs- und Modellierungssprache sorgt. Dennoch führen fehlende Entwicklungsstrategien und Richtlinien weiterhin zu „einstürzenden Neubauten“ wie Oestereich bemerkt. Er betont die Tatsache, dass Methoden wie Fachkonzepte, Dokumentationen, Quellcoderichtlinien, Versionsverwaltung usw. keinen nebensächlichen Selbstzweck erfüllen.[8] So kann man zum Schluss gelangen, dass die genannten Probleme nur mit einem straffen Projektmanagement in Verbindung mit klaren Richtlinien zu begegnen sind.

Ende gut, alles gut? Leider ist die Schuldfrage nicht ganz so einfach auf die Seite der IT-Dienstleister abzustellen. Versteegen spricht auch von einer Wechselwirkung zwischen Kunden und Entwickler.[9] Denn das wohl schwierigste Problem dieser „Krise“ liegt unterdessen bei Fehlentwicklungen durch unverstandene Problemstellungen. Zurückzuführen ist die Tatsache hauptsächlich auf Kommunikationsprobleme, unklare Zielvorgaben und eine mangelhafte Berücksichtigung potentieller Änderungswünsche während der Entwicklung. Partsch bezeichnet diesen Umstand als einen Mangel bei der „Übersetzung [von] Kundenbedürfnissen in Systemanforderungen“.[10]

Sehr häufig führen die genannten Probleme zu einem vorzeitigen Abbruch oder - bei schmerzlich erkämpfter Auslieferung - zu dramatischen finanziellen Verlusten. Die Lösung für das umgangssprachlich proklamierte Motto *„Unsere Software soll besser werden!“* findet sich wie so oft in einer ausgeklügelten und durchdachten Strategie in der Planung und des Managements. Es ist kein Geheimnis, dass die erfolgreiche Projektabwicklung das unmittelbare Ergebnis einer sorgfältig durchgeführten Vorbereitungs- und Planungsphase ist. Dazu gehört vor der obligatorischen Aufwandsschätzung vor allem die Bestimmung von Kundenanforderungen.

Zur Anforderungsanalyse und -management liegen bereits sehr umfangreiche Publikationen vor, die nahezu alle Bereiche des so genannten Requirements Engineering (RE) sowie des Requirements Engineering Process (REP) abdecken.[11] Bei genauerer Betrachtung stößt man auf eine Vielzahl von Techniken, so etwa Kreativitätstechniken, Beobachtungstechniken, Strukturierungstechniken, Techniken zur Konfliktbewältigung, etc., die dazu dienen sollen, den Prozess der Erhebung, Dokumentation und Kundenabsprache in Teilen oder im Ganzen zu unterstützen. In der Realität

[6] Gemeint sind Methoden für die datenorientierte sowie prozessorientierte Anwendungsentwicklung wie z.B. Problem Statement Language (PSL) aus [TE77], Structured Analysis and Design Technique (SADT) aus [DC88] oder Object Modelling Technique (OMT) dem Vorfahre der UML aus [R+90], welche heute eher ein Nischendasein entwickeln oder sogar gänzlich obsolet sind. Einen guten Überblick und weitere Referenzen gibt [Sch97] S.51ff.

[7] Vgl. [OMG03]

[8] Vgl. [Oes98] S.23

[9] Vgl. [V+02] S.8

[10] Vgl. [Par91] S.19

[11] Für eine Definition und weiterführende Betrachtung dieser Begriffe siehe Abschnitt 1.3 auf S.7, 2.1 auf S.11, 2.2 auf S.14 und 4 auf S.14 dieser Arbeit.

fühlen sich jedoch die Projektleiter und Entwickler aufgrund der nahezu „erdrückenden Vielfalt“[12] von Vorgehensempfehlungen und Techniken schnell überfordert. Der größte Teil der zitierten Hilfsmittel sind zwar als wertvolle Gestaltungsmittel unverzichtbar, bieten aber isoliert betrachtet keinen effektiven Ansatz.

Kritisch ist daher die Frage zu stellen, warum es trotz der Vielzahl an verfügbaren Techniken und Werkzeugen zum Requirements Engineering in der Unternehmenspraxis dennoch massive Probleme bei der Erstellung, Pflege und Benutzung eines in sich schlüssigen sowie qualitativ hochwertigen Anforderungskataloges gibt.

Offensichtlich fehlt es an einem Gesamtkonzept, das im praktischen Einsatz stets den „roten Faden“ offen legt und gleichzeitig dem Benutzer als Orientierungshilfe dient. Noch vor wenigen Jahren wurde die Idee der integrierten Informationsverarbeitung als „durchgreifende Verbesserung“[13] in hohen Tönen gelobt. Die Integration gehört heute zum Standardrepertoire vieler Optimierungsansätze. Im Duden werden die Synonyme „Vervollständigung“ oder „Eingliederung“ genannt.[14] Im Kontext dieser Arbeit soll darunter die „Wiederherstellung einer Einheit“ bzw. die „Verknüpfung [..] zu einem einheitlichen Ganzen“[15] verstanden werden. Mit Hilfe dieser Idee gelingt es scheinbar eigenständige Disziplinen des RE inhaltlich so zu verknüpfen, dass -so die Theorie- während der Entwicklungsphase von einem „integrierten Softwarelebenszyklus“[16] gesprechen werden kann, um so das gesamte Potential der Optimierung auszuschöpfen.

Natürlich wurde dieses Thema in der Softwareindustrie längst aufgegriffen und manifestiert in Form von diversen Vorgehensmodellen.[17] Diese komplexen Werke haben das Ziel, praktisch alle erdenklichen Aspekte (Geschäftsprozessmodellierung, Pflichtenheft, Analyse & Design, Realisierung, Test, Einführung, Management, Organisationsgestaltung, ...) im Softwareprozess zu diskutieren und zu standardisieren. Kombiniert mit so genannten Best Practices[18](dt.: Erfahrungen) versprechen die Entwickler dieser Vorgehensmodelle, garantierte Praxistauglichkeit und ideale Rahmenbedingungen für den Erfolg von Projekten.

[12] Vgl. [Oes98] S.21
[13] Vgl. [SH01] S.2
[14] Vgl. [Dud01]
[15] Vgl. [M+97] S.208
[16] Vgl. [SH01] S.218 zum Begriff des Softwarelebenszyklus (Software Life Cycle)
[17] Vorgehensmodelle sind ein Schwerpunkt in dieser Arbeit und werden im Abschnitt 2.2 ab S.14ff. ausführlich diskutiert.
[18] Vgl. [Ver00] S.51

Es stellt sich die Frage, warum manche IT-Dienstleister von einer Krise sprechen, obwohl es jene allumfassenden Lösungsmittel in Form der Vorgehensmodelle gibt.[19]

Eine solche These aufzustellen ist ohne Darstellung der Randbedingungen nicht sinnvoll. Die Relevanz der Problematik ergibt sich zunehmend für kleine und mittelständische Softwarehäuser[20], welche unter dem Druck von Konkurrenten und Kunden ihre Umsätze hauptsächlich im Projektbetrieb erzielen. Flexibilität, Schnelligkeit und ein angemessenes Preis-Leistungs-Verhältnis können dabei die ausschlaggebenden Erfolgskriterien darstellen. Aber auch Großunternehmen profitieren von einem verbesserten Verständnis zum großen Thema Anforderungsmanagement und dessen Bedeutung im Softwarelebenszyklus. Die vorliegende Arbeit entstand aus der Situation der GFT Systems GmbH Ilmenau[21], einer Tochter der GFT AG[22] aus St.Georgen. Als typisches Projekthaus mit Kundenschwerpunkt auf große und mittelständische Unternehmen erstellt die GFT Systems im Hauptumsatzfeld individuelle Softwarelösungen für Geschäftsanwendungen im Bereich E-Commerce und Enterprise Application Integration (dt.: unternehmensweite Anwendungsintegration). Der zunehmende Erwartungsdruck und die aktuellen Trends in der Branche erfordern verstärkte Anstrengungen bzgl. Standardisierung und Optimierung interner Entwicklungsprozesse. Diese Ausgangssituation dient der Motivation zu dieser Arbeit.

1.2. Zielsetzung

Im Rahmen dieser Diplomarbeit soll der Vorgang der Entwicklung von Anforderungen in Softwareprojekten betrachtet werden. Ziel ist die Konzeption eines eigenen methodischen Ansatzes. Zu diesem Zweck wird der Begriff des RE detailliert erarbeitet. Als ein zentraler Bestandteil dieses mittlerweile etablierten Fachgebietes der Informatik, soll der Begriff „Anforderung“ (engl.: requirement) im Kontext von Anforderungsdokumenten (engl.: requirements specification) genauer betrachtet werden. Häufig wird dabei die Anforderungsanalyse mit der Erhebung von Kundenwünschen

[19] Stand: 01.06.2003. Zum Thema „Krise in der IT-Branche“ beruft sich der Autor auf die Meldungen diverser Fachzeitschriften, Tageszeitungen und Wirtschaftsnachrichten, die hier nicht einzeln referenziert werden sollen. Die Krise wird jedoch häufig gleichgesetzt mit einer allgemeinen Wirtschaftsschwäche und der rückläufigen Nachfrage bzw. zunehmenden Investitionszurückhaltung bei IT-Produkten. Im Rahmen dieser Arbeit wird der Standpunkt vertreten, dass die Krise nicht zuletzt zurückzuführen ist auf die enormen Entwicklungskosten, welche oftmals in keinem wirtschaftlichen Verhältnis zum angestrebten Kundennutzen stehen (Return over Investment). Ursachen für die Krise diskutiert Versteegen in [Ver00] S.1ff. und bezieht sich dabei auf Statistiken der Standish Group.

[20] Für diese Art der Unternehmung hat sich die Abkürzung „KMU“ (Klein- und Mittelständische Unternehmen) etabliert. Für eine konkrete Abgrenzung und Beschreibung siehe auch [Tho94] S.13.

[21] Stand: 01.06.2003, Derzeit sind 64 Mitarbeiter in unterschiedlichsten Projekten in und außerhalb des Standortes beschäftigt.

[22] Stand: 31.03.2003, Mitarbeiterzahl: 1029 (http://www.gft.com/com2003.html)

gleichgesetzt. Im Rahmen dieser Arbeit wird die „Analyse der Anforderungen“ als Teildisziplin des RE verstanden, mit dem Ziel, die ermittelten Anforderungen zu untersuchen und zu strukturieren.

Zur Dokumentation fachlicher Zusammenhänge hat sich im Geschäftsalltag die UML, insbesondere das UML Anwendungsfalldiagramm (engl.: use case diagram) als Standard durchsetzen können.[23] Mit Diagrammen werden die Sachverhalte übersichtlich visualisiert und in ein formales Modell abgebildet, das wiederum für alle Beteiligten (engl.: stakeholder) leichter verständlich ist.

Negative Erfahrungen aus vergangenen Projekten lassen jedoch berechtigte Kritik an der Effektivität dieser Techniken und Methoden aufkommen, die die bekannten Vorgehensmodelle anbieten. Relativ neu ist die Bedeutungsanalyse[24], deren Ansätze im Rahmen dieser Arbeit methodisch aufgegriffen werden. Das erklärte Ziel der Bedeutungsanalyse ist es, Mehrdeutigkeiten und Unvollständigkeiten natürlichsprachlicher Aussagen zu erkennen und frühzeitig zu beseitigen.[25] Diese Idee thematisieren u.a. Rupp et al. und geben einen Überblick zu sprachlichen Defekten[26] in der Anforderungsanalyse. Sie unterbreiten den Vorschlag, Defekte durch Anforderungsschablonen zu umgehen. Es ist vorstellbar, UML Anwendungsfalldiagramme durch mehrere Zwischenstufen in textuelle, leicht verständliche und trotzdem defektfreie Ausführungen zu überführen, um so einen höheren Qualitäts- und Detaillierungsgrad des Anforderungskataloges zu erhalten.

In der Literatur zum Requirements Engineering wird recht selten das Thema Scoping[27] aufgegriffen. Der Scope (dt.: Geltungsbereich) bezeichnet den für die analytische Betrachtung relevanten Ausschnitt einer bestimmten Domäne. Unter Scoping versteht man daher allgemein auch die Abgrenzung eines Gegenstandsbereiches. Dieser Vorgang hat sich unterdessen als eigenständige Disziplin im Rahmen des RE etabliert. Der Begriff „Scope Management“ ist daher gleichbedeutend mit der Planung, Kontrolle und Steuerung des relevanten Ausschnittes in einem Projekt. Ändert sich zum Beispiel der Scope im Laufe der Anforderungsanalyse[28], kann es zu gravierenden Auswirkungen für die zu ermittelnden Anforderungen und Abhängigkeiten führen. Ein geeignetes Change Management (dt.: Änderungsmanagement)[29] oder besser die exakte Bestimmung eines möglichst festen Scopes kann jedoch unnötig hohe Folgekosten verhindern.

[23] Vgl. [Sch02] S.42, [Bal01] S.40, [Oes98] S.20,203

[24] Vgl. [Sch02] S.233,234

[25] Eine sehr ausführliche Abhandlung über mehrdeutige Anforderungen gibt Gause in [GW93].

[26] Vgl. [R$^+$02] S.191ff.

[27] In den gängigen Nachschlagewerken gibt es keine direkte deutsche Übersetzung für diesen Begriff. Am besten passt die Bezeichnung „Abgrenzung“.

[28] Es gibt vielfältige Gründe und Ursachen, die eine Ausweitung oder Einschränkung des Scopes erfordern. Detailliert wird auf diese Problematik im Abschnitt 4.3.1 eingegangen.

[29] Die Begriffe Change Management und Change Request (dt.: Änderungsauftrag) werden im Abschnitt 4.5.1 ab S.115ff. ausführlich behandelt.

Welche Wissenslücken sollen nun in dieser Arbeit gefüllt werden? Wie bereits erwähnt, gibt es in der Praxis offensichtlich das Problem, angemessen auf die modernen Herausforderungen des RE reagieren zu können. Trotz der Vielzahl der verfügbaren Ansätze und professionellen Vorgehensmodelle mangelt es bei der praktischen Umsetzung an einem passenden methodischen Vorgehen. Folgende Argumente lassen sich gedanklich herleiten, warum eine spezielle Lösung für das RE konstruiert werden sollte:

- **Rahmenbedingungen.** Die Ansprüche des Kunden und die Art des Projektes (Mitarbeiter, Umfang, Technologie,...) haben unmittelbaren Einfluss auf das zugrunde liegende Vorgehen. Eine Abstraktion erzielt zwar einen höheren Abdeckungsgrad, berücksichtigt jedoch die aus Sicht der Benutzer eigenen Erfahrungen und Erkenntnisse des Projektalltages nur unzureichend.

- **Tailoring.** In der Praxis ist die vollständige Umsetzung eines Vorgehensmodells unüblich. Der Methodengehorsam, wie ihn Oestereich beschreibt[30], führt häufig zu einem ineffektiven Lehrbuchansatz ohne eigenen Anteil. Das Maßschneidern (engl.: tailoring) ist eine übliche Praxis, um komplexe Modelle den eigenen Bedürfnissen anzupassen.[31]

- **Informationsflut.** Welche Techniken sind gut oder besser? Rupp spricht von „Kristallkugeln und Kartenlegen“[32] während Schienmann nüchtern eine Bewertung von „Erhebungstechniken aus Sicht des Erhebenden“[33] vornimmt. Bei schätzungsweise 30 bis 40 Techniken kann es einem Projektleiter schwer fallen, die richtige Wahl zu treffen.

- **Implizites Wissen.** Die Erklärung einer bisher „gefühlsmäßig unorganisierten“ betrieblichen Aufgabe[34] lässt sich oft nicht ausreichend nachvollziehbar festhalten. Dennoch steckt in den Köpfen der Mitarbeiter wertvolles implizites Wissen über betriebliche Vorgänge, welches sich explizieren[35] lässt und damit auch Anderen zur Verfügung gestellt werden kann.

[30] Vgl. [Oes98] S.22
[31] Vgl. [BD95] S.49ff., [V+02] S.48ff.
[32] Vgl. [R+02] S.108ff.
[33] Vgl. [Sch97] S.117,118
[34] Zum Beispiel der Umgang mit einem Kunden oder die Selbstorganisation nach bestimmten individuellen Vorgehensmustern.
[35] Vgl. [NT97] S.68-87, Nonaka und Takeuchi brachten erstmals die Begriffe implizites (nonverbales/schwer auszudrückendes) und explizites (formalisierbares/dokumentierbares) Wissen in Zusammenhang mit dem organisationalen Wissensmanagement. In ihrem theoretischen Modell „Spirale des Wissens“ wird in vier Phasen ein iterativer Vorgang beschrieben, nach dem das Wissen der Mitarbeiter schrittweise der Organisation zugänglich gemacht werden kann („Explizieren von Wissen“).

Ziele der vorliegenden Arbeit sind:

1. Ausarbeitung eines Überblicks zum Requirements Engineering in Bezug auf dessen Verwendung in Vorgehensmodellen.(Stand der Technik)
2. Ermittlung und Diskussion der Problemfelder zum Anforderungsmanagement in kommerziellen Softwareprojekten.
3. Entwicklung eines, unter Argumentation der ermittelten Problemfelder, eigenen methodischen Vorgehens zum Requirements Engineering in Form eines Konzeptes für ein Prozessmodell und dessen Integration in das Bezugssystem des Softwarelebenszyklus.

1.3. Begriffsbestimmung

In diesem Abschnitt wird eine Definition bzw. Einordnung der verwendeten Begriffe vorgenommen. Dies dient hauptsächlich zur Erhaltung der inhaltlichen Konsistenz und Vereinbarung einer einheitlichen Terminologie zu dieser Arbeit. Einige Begriffe wurden in den ersten Abschnitten der Einleitung bereits verwendet, werden jedoch an dieser Stelle noch einmal präzisiert.

Projektbegriff

- **Projekt.**[36] Ein Projekt ist ein komplexes Vorhaben, das Beschränkungen hinsichtlich Zeit, Kosten und Ressourcen unterworfen ist. Projekte sind inhaltlich immer einmalige (unikale) Unterfangen mit einem klar definierten Ziel. Je nach Art und Umfang ist für die Durchführung eine eigene Organisationsstruktur und ein hohes Maß an Interdisziplinarität der Mitarbeiter erforderlich. Allen Projekten ist eine Untergliederung in **Phasen** gemeinsam. Diese werden auch als Hauptabschnitte oder Entwicklungsabschnitte (z.B. Analyse, Entwurf, Realisierung, Einführung usw.) bezeichnet. Projekte unterliegen einem **Projektmanagement**, das der Planung, Kontrolle und Steuerung vor, während und nach der Abwicklung dient.
- **Softwareprojekt. Software** bezeichnet ein immaterielles Produkt zum Betrieb von datenverarbeitenden Anlagen. Die Neuentwicklung, Modifizierung oder Einführung von Software wird in Softwareprojekten (IT-Projekte) durchgeführt. Im Unterschied zu anderen Projektarten sind während der Abwicklung überwiegend IT-Spezialisten beteiligt, die im Zuge ihrer Arbeit mit Fachspezialisten aus unterschiedlichsten Bereichen zusammenarbeiten müssen.

[36] Vgl. zum folgenden Abschnitt [SH01] S.219

Begriffe zur Softwareentwicklung

- **Softwaretechnik.**[37] Die Softwaretechnik befasst sich mit der zielorientierten Bereitstellung und der systematischen Verwendung von Prinzipien, Methoden und Werkzeugen für eine ingenieurmäßige Entwicklung von Softwaresystemen. Als **System** bezeichnet man in dem Zusammenhang einen in DV-Anlagen abgebildeten Ausschnitt der realen oder gedanklichen Welt. Je nach Betrachtungsweise besteht ein System aus **Subsystemen** mit spezifischen Merkmalen, die ihrerseits in verschiedenen Beziehungen zueinander stehen können.

- **Modelle.**[38] Modelle dienen dazu, eine bestimmte Informationsstruktur zu beschreiben. **Modellierung** bezeichnet den Vorgang, der zur Abbildung und Vereinfachung einer Wirklichkeit führt. Modelle sind Hilfsmittel zur Darstellung. Sie ermöglichen die verbesserte Nachvollziehbarkeit von Zusammenhängen durch ein zielgerichtetes Abbild eines Systems. Hinter jedem Modell steht somit die Zielstellung, gewisse Eigenschaften als wesentlich oder unwesentlich auszuzeichnen.

- **Prozess.**[39] Nach DIN 19233 ist ein Prozess definiert als „Vorgang zur Umformung, zum Transport oder zur Speicherung von Materie, Energie oder Information.“ Abstrakt formuliert, wird durch den Prozess ein Input transformiert und in den erwarteten Output überführt. Von einem **Geschäftsprozess** spricht man bei einer Verkettung von logisch zusammengehörigen Aktivitäten eines Unternehmens mit einem besonderen Beitrag zur Wertschöpfung und expliziter Ausrichtung an einen (internen oder externen) Kunden.

- **Vorgehensmodell (Prozessmodell).**[40] Das Vorgehensmodell (VGM) bezeichnet ein verbindliches, organisatorisches Ablaufkonzept über alle Phasen der Softwareentwicklung. Im engeren Sinne handelt es sich dabei um ein Empfehlungskonzept, das mit Hilfe von modellhaften Beschreibungen, Richtlinien, Anweisungen und Prozessen zu einem abgegrenzten Problembereich Lösungen anbietet. Die Beschreibungen basieren auf Erfahrungswerten früherer Projekte. Sie unterscheiden sich jedoch durch deren allgemeingültige Ausrichtung und dem Abstraktionsgrad. Die im Vorgehensmodell abgebildeten Phasen sind logische Einheiten, die ihrerseits über deren **Zwischenergebnisse** (sog. Produkte oder Artefakte) bewertet werden. Je nach Entwicklungsstufe sind dies Schaubilder, Fachkonzepte, Berichte, Dokumentationen, Quellcode, etc.

[37] Vgl. zum folgenden Abschnitt [Bal01] S.15-42
[38] Vgl. zum folgenden Abschnitt [Han98] S.23ff., [Win00] S.103ff.
[39] Vgl. zum folgenden Abschnitt [SH01] S.210, [Poh95] S.25
[40] Vgl. zum folgenden Abschnitt [V+02] S.29ff, [Ver00] S.21ff.,S.77, [Bal01] S.54ff., [R+02] S.51ff., [SH01] S.218ff.

Spezielle Begriffe zum Thema dieser Arbeit

Die folgenden Begriffe werden in der Literatur auf sehr vielfältige Art und Weise definiert.[41] Je nach Autor, Publikation und der verwendeten Sprache führt dies zu unterschiedlichen Detaillierungsgraden, Überschneidungen bzw. interpretativen Differenzierungen. Im weiteren Verlauf dieser Arbeit werden die Begriffe zunehmend detailliert. Sie sollen an dieser Stelle jedoch überblicksartig genannt und von einander abgegrenzt werden.

- **Anforderung.**[42] Eine Anforderung (engl.: requirement) wird von einem Auftraggeber oder einem Endbenutzer gestellt. Sie beschreibt, wie sich das zu implementierende System verhalten soll. Eine Anforderung wird auch bezeichnet als eine Bedingung oder Fähigkeit, die eine Software erfüllen oder besitzen muss, um einen Vertrag, Norm oder ein anderes formelles Dokument zu erfüllen. Unterschiede ergeben sich vor allem in Art (funktional/nichtfunktional/...), Form (mündlich/schriftlich/modellhaft/...), Aufgaben (Kommunikation/Dokumentation/...) und Abstraktionsgrad (konkret/allgemein).

- **Requirements Engineering.**[43] In Anlehnung an „Softwaretechnik“ würde eine mögliche deutsche Übersetzung „Anforderungstechnik“ lauten, deren Verwendung jedoch unüblich ist.[44] Gegenstand des RE ist es, im Rahmen eines systematischen Vorgehens, eine Spezifizierung (Erfassung, Beschreibung und Prüfung) von Anforderungen vorzunehmen. Dazu zählt die Identifikation und Dokumentation der Kundenwünsche (**Anforderungsanalyse**), die Erstellung eines Dokumentes zur Beschreibung des zu erwartenden Systemverhaltens (**Anforderungsdefinition**), sowie der Analyse und Validierung der Ergebnisse hinsichtlich Konsistenz, Vollständigkeit und Abhängigkeit. Unterstützung finden die Teilbereiche in Form von **Techniken und Gestaltungsmitteln** zum Requirements Engineering, die ihrerseits eigene Vorgehensweisen beinhalten. Die **Anforderungsdokumente**[45] sind das Ergebnis

[41] Dies bemerkt auch Schienmann in [Sch97]und weist darauf hin, dass bereits eine klare Abgrenzung der Begriffe ein Beitrag zur Verbesserung darstellt.

[42] Vgl. zum folgenden Abschnitt IEEE-610.12, [Ver00] S.85, [Poh95] S. 21, [R+02] S.13

[43] Vgl. zum folgenden Abschnitt [SS97] S.9ff., [Poh95] S.21

[44] In der deutschen Literatur wird der Begriff „Requirements Engineering“ auch deswegen gegenüber der Übersetzung bevorzugt, weil er aussagekräftiger ist und inhaltlich mehr abdeckt. Mit der deutschen Übersetzung könnte der Verdacht aufkommen, dass der Schwerpunkt ausschließlich auf der zu verwendenden Technik liegt. Das „Engineering“ umfasst im Gegensatz dazu, inhaltlich die ganze Palette der Aufgaben eines Ingenieurs wie Planung, Konstruktion und Umsetzung.

[45] Aufgrund der Vielfalt unterschiedlicher Bezeichnungen ergeben sich bei diesem Begriff die meisten Schwierigkeiten. Im Deutschen ergibt sich zum Beispiel aus dem Wort „Anforderungsspezifikation“ sowohl ein Ergebnis (das Dokument) als auch ein Vorgang (das Spezifizieren) und ist demzufolge unscharf in seiner Verwendung. In der deutschen Literatur trifft man auch oft auf die Begriffe „Lastenheft“ und „Pflichtenheft“, die in manchen englischen Publikationen dagegen generell als „requirements specification“ bezeichnet werden. Im Folgenden wird daher ausschließlich der Begriff „Anforderungsdokument“ gebraucht.

und enthalten eine systematisch aufbereitete Zusammenstellung aller Anforderungen.

- **Requirements Management.**[46] Im deutschen auch als „Anforderungsmanagement“ bezeichnet, richtet sich das RM hauptsächlich an den organisatorischen Umgang mit Anforderungen. Ziel ist die Steuerung, Kontrolle und Verwaltung aller operativen Aufgaben die im Rahmen des RE anfallen. Die dabei auftretenden Probleme sollen in Kombination mit geeigneten Führungsmitteln verhindert werden, indem ein systematischer Ansatz vorgeschrieben wird. Da Anforderungen im Laufe eines Projektes nicht stabil sind, spricht man auch von einem **Änderungsmanagement** (engl.: change management). Inhaltlich fällt die Problematik jedoch in das Aufgabenfeld des RM und wird deswegen nur selten isoliert betrachtet. Man kann auch von einem „kontinuierlichen Anforderungsmanagement“ sprechen, wie es Schienmann in seinen Veröffentlichungen verwendet.

  Im Kontext dieser Arbeit wird das RM außerdem als Oberbegriff bzw. Sammelbegriff verwendet.[47] Das Requirements Engineering ist daher als eine Untermenge des Requirements Managements anzusehen.

- **Requirements Engineering Process.**[48] Sommerville bezeichnet den REP (dt.: Anforderungsprozess) als einen strukturierten Satz von Aktivitäten zur Erlangung, Validierung und Pflege eines Anforderungskataloges.[49] Bei Prozessbetrachtungen empfiehlt es sich, den aufzubringenden Input und den zu erwartenden Output zu beschreiben. Vorbedingung zum Start ist wie bei jedem Prozess ein Auslöser (engl.: trigger event), den Pohl im REP **Vision** (engl.: system vision) nennt. Zu einer Vision, hier auch **Initialinput** genannt, führen z.B. Notlagen, politische Entscheidungen oder veränderte Rahmenbedingungen. Da am Anfang oft nur eine vage Vorstellung des angestrebten Systems existiert, schildern zunächst alle Beteiligten ihre eigenen Ideen und konstruieren anschließend gemeinsam eine möglichst konkrete Vision (Input). Nach Durchlaufen des REP, steht ein mit Hilfe von formalen Gestaltungsmitteln konsistentes und vertraglich bestätigtes Anforderungsdokument zur Weiterverarbeitung im Softwarelebenszyklus zur Verfügung (Output).

  Im Rahmen dieser Arbeit wird der REP als integrierter Bestandteil eines Vorgehensmodells verstanden. Im Unterschied zum RE beschreibt ein REP eine vollständige Ablaufbeschreibung.

[46] Vgl. zum folgenden Abschnitt [Sch97] S.16, S.31ff., [VSH01] S.14, S.67ff.

[47] Diese Meinung wird nicht von allen Autoren vertreten. Zave spricht zum Beispiel generell vom RE, während Schienmann sich in [Sch02] S.31ff. ausführlicher mit der Thematik beschäftigt und das RM als Oberbegriff heranzieht.

[48] Vgl. [SS97] S.9ff., [Poh95] S.25,26

[49] Originaler Wortlaut: „A requirements engineering process is a structured set of activities which are followed to derive, validate and maintain a systems requirements document.“

2. Stand der Technik

„Da sind vor allem die IT-Spezialisten gefordert, dazuzulernen. Sie dürfen nicht so sehr die technischen Details eines Softwarepakets herausstellen, als vielmehr den Beitrag, den es für einen bestimmten Geschäftszweck leistet.“[1]

2.1. Grundlagen des Requirements Engineering

In diesem Abschnitt wird geklärt, welche grundsätzlichen Anliegen mit dem RE verbunden sind und wie eine Einordnung in den Softwarelebenszyklus[2] vorzunehmen ist. Zur Erinnerung: Bisher wurde unterschieden zwischen der Identifikation, Analyse und Dokumentation der Kundenwünsche. Um Missverständnissen vorzubeugen, wurde im Rahmen der Einleitung eine zusätzliche Unterscheidung in Anforderungsanalyse und Anforderungsdefinition vorgenommen. Diese Trennung soll den Prozess der Anforderungsentwicklung unterstreichen, der seinesgleichen die Ergebnisse aus der Analyse überträgt und in dokumentierte bzw. verbindliche Anforderungen transformiert (Ergebnis der Anforderungsdefinition).

Es erscheint nun sinnvoll eine Einordnung in den Softwarelebenszyklus[3] vorzunehmen. In der Literatur gibt es eine Vielzahl theoretischer Modelle, die die Lebensdauer von Software phasenartig beschreiben. Sie unterscheiden sich jedoch häufig nur in ihrem Detaillierungsgrad und folgen prinzipiell einem ähnlichen Schema.

Die Abbildung 2.1[4] soll einen typischen Softwarelebenszyklus skizzieren. Im Mittelpunkt steht das, vom Wasserfallmodell her bekannte, treppenartige Muster bei dem die vier Hauptphasen Requirement Analysis, Design, Implementation, Integration in einem Top-Down Ansatz miteinander verknüpft sind und den eigentlichen Entwicklungsprozess symbolisieren. Nach Abschluss einer Phase übernimmt der Nachfolger die erarbeiteten Ergebnisse des Vorgängers als Input(Requirements Specification, Design Specification, Code). Alle Phasen werden durch eine mehr oder weniger stark ausgeprägte werkzeugbasierte Unterstützung begleitet. Dies liegt vor

[1] Vgl. [Rad03] Computer Zeitung Nr.25/03: Zitat von Peter Armstrong BMC/Houston.

[2] Synonyme sind unter anderem: Phasenmodell, Lebensdauerzyklus oder Softwareproduktlebenszyklus, Vgl. [Elz94] S.34.

[3] Vereinzelt wird auch der Softwarelebenszyklus mit Hilfe von Vorgehensmodellen erklärt (Versteegen). Es lässt sich jedoch bei Betrachtungen bzgl. des RE in Vorgehensmodellen besser argumentieren, wenn man zuvor abstrakt von einem Lebenszyklus spricht, darin eine Einordnung vornimmt und anschließend das spezielle Vorgehensmodell an der fraglichen Stelle betrachtet.

[4] In Anlehnung an [V+02] S.31 und [Elz94] S.34

allem daran, dass die Phasen sich inhaltlich durch ihre Ergebnisse unterscheiden und daher oft keine generelle Verfügbarkeit potentieller Werkzeuge sichergestellt werden kann. An dieser Stelle wird bereits die Idee der integrierten Softwareentwicklung angedeutet. Tatsächlich könnte ein vollständiger, d.h. über alle Phasen integrierter Ansatz dazu beitragen, den ganzen Vorgang zentral zu kontrollieren.

Man kann nicht davon ausgehen, dass jede Phase, selbst bei strengsten Abnahmekriterien, für sich fehlerfrei ohne jede Nacharbeit abschließbar ist. Im Modell sind daher auch Rücksprünge möglich. Streng genommen ist dadurch zwar die „Lebenslinie“ nicht mehr eindeutig, erfüllt jedoch mit diesem Kunstgriff eine wichtige Erkenntnis, die im Zusammenhang mit dem Top-Down Ansatz häufig kritisiert wurde.

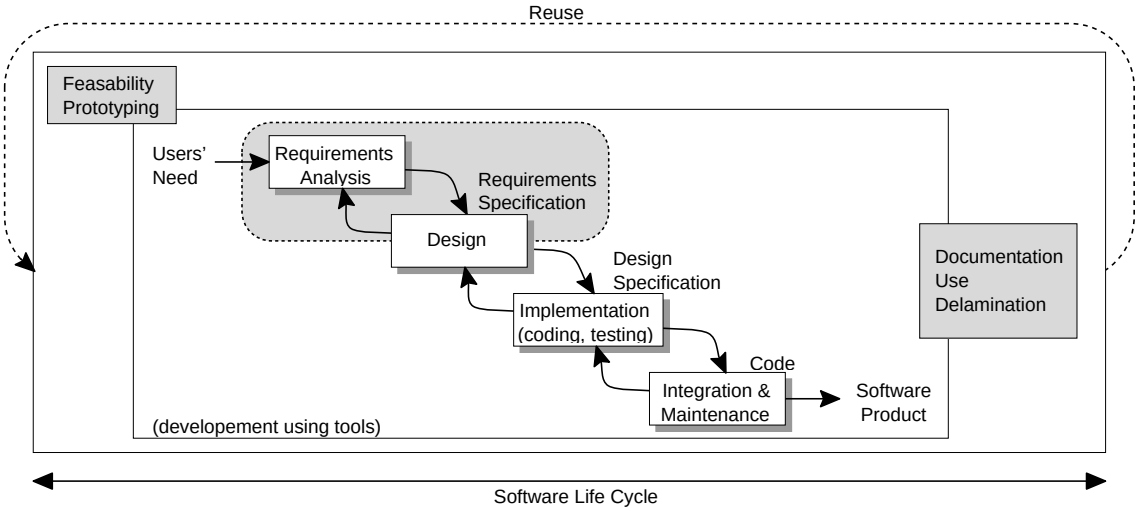

Abbildung 2.1.: *Anforderungsanalyse im Softwarelebenszyklus ([V+02],[Elz94])*

Zu Beginn des Softwarelebenszyklus wird typischerweise ein Prototyp zur Überprüfung der technischen Machbarkeit oder zu Demonstrationszwecken entwickelt. In der Regel wird dieser mit Anfang der eigentlichen Arbeiten wieder verworfen.[5] Lediglich die dabei gewonnenen Erkenntnisse werden übernommen (Transfer von Know How). Im Anschluß an den Entwicklungsprozess folgt die Dokumentation, Inbetriebnahme und später die Ablösung.

Hinweis: Es erfolgt eine Ausweitung des Begriffs „Anforderungsanalyse“. Sie bezeichnet hierbei die gesammelten Aktivitäten der nach ihr benannten Phase (Requirements Analysis) und ist nun inhaltlich nicht mehr konsistent mit der Definition aus der Einführung (Identifikation und Dokumentation) bzw. beschränkt auf

[5] Vgl. [SH01] S.223, Stahlknecht und Hasenkamp unterscheiden zwischen Wegwerfprototypen (rapid prototyping), anhand derer das endgültige System völlig neu erstellt wird, und wiederverwendbaren Prototypen, die schrittweise verbessert werden und in das Endprodukt eingehen (evolutionäres Prototyping).

die „Analyse der Anforderung“ als Teilgebiet im RE. Das Aufgabengebiet wird also erweitert durch den Einsatz von Techniken zur Anforderungsermittlung sowie Anforderungsvalidierung. Im Unterschied zum RE als Oberbegriff, bezeichnet die Anforderungsanalyse fortan eine Phase im Softwarelebenszyklus.

Elzer ergänzt in seinem Modell zum „Lebensdauerzyklus“[6] die Wiederverwendung, welche auch in der Abbildung („reuse“) berücksichtigt wurde. Dieser Sachverhalt reflektiert eine typische Eigenschaft von Software: die generative Weiterentwicklung. Der Begriff „Wiederverwendung“ ist nicht allzu technisch zu interpretieren. Gemeint ist in diesem Zusammenhang primär das iterative Durchlaufen der Phasen untereinander, sowie im Zyklus insgesamt.

Eine präzise Einordnung der Anforderungsanalyse im Softwarelebenszyklus fällt nun nicht mehr schwer. Wie in der Abbildung bereits angedeutet, findet dieser Vorgang in einer eigenen Phase statt. Alle Aktivitäten erfolgen unmittelbar im Dialog mit dem Kunden (Users' Need) und münden aufbereitet als Anforderungsspezifikation (Requirements Specification) in der Designphase. Durch die wasserfallartige Anordnung[7] der Phasen, fällt den frühen Aktivitäten eine besondere Bedeutung zu. Selbst kleine Irrtümer können zu einem „Summationseffekt [führen], bei dem sich die Fehler fortpflanzen und multiplizieren“[8]. Die Schlussfolgerung muss also lauten: je früher ein Fehler gefunden wird, um so effektiver kann dieser beseitigt werden. Im betrachteten Modell ist die frühestmögliche Fehlerquelle, die Phase der Anforderungsanalyse.

Als Teil der Softwareentwicklung obliegt der Anforderungsanalyse eine wichtige Aufgabe und erfüllt keinen reinen Selbstzweck. Allerdings bedeutet die systematische Spezifikation von Anforderungen einen hohen Aufwand in Form von Zeit und Geld.[9]

Da bei der Erstellung eines Anforderungsdokumentes kein direkter Kundennutzen entsteht, fragt der Kunde zu Recht nach der Wirtschaftlichkeit des Vorhabens.

Die Anforderungsanalyse als Phase hat jedoch das klare Ziel Kosten zu sparen. Das Anforderungsdokument kann allerdings nur dann als wirtschaftlich bezeichnet werden, wenn dem dafür zu betreibenden Aufwand entsprechende Einsparungen gegenüberstehen. Es ist unmittelbar einzusehen, dass ein Anforderungsdokument sowohl technisch als auch rechtlich unverzichtbar ist. Nur in welchem Umfang das

[6] Vgl. [Elz94] S.34

[7] Es sei darauf hingewiesen, dass hier nicht das Wasserfallmodell als Vorgehensmodell betrachtet wird, sondern dessen genereller Ansatz zur Beschreibung typischer Phasen in der Softwareentwicklung. Das Wasserfallmodell taucht in jedem Modell in irgendeiner Form auf, wie Versteegen in [V+02] S.31 zu recht behauptet.

[8] Vgl. [R+02] S.12

[9] Vgl. zum folgenden Absatz [Köp98] Abschnitt 10

Werk zu entwickeln ist, bleibt dagegen unklar. Das Ziel, trotz erhöhtem Aufwand in der Analysephase, Kosten zu senken ist realistisch, wie folgende Überlegung zeigt:

- **Anforderungsfehler.** Fehler im Anforderungsdokument erzeugen besonders hohe Kosten, da sie häufig erst sehr spät entdeckt werden. Wird der Fehler bei Abnahme oder gar bei der Produktivschaltung der Software entdeckt, kann dies im Extremfall zu einer unvorhersehbaren Kostenexplosion führen.

- **Fehlerkosten.** Wenn beispielsweise ein Anforderungsfehler vorliegt, führt dies unweigerlich zu Fehlerkosten. Dabei wird unterschieden zwischen internen und externen Fehlerkosten. Ursache interner Kosten ist der Mehraufwand für Lokalisierung, Behebung und Nacharbeit, um einen Mangel zu beseitigen. Von externen Kosten spricht man dagegen, wenn der Mangel zu Verfahrensfehlern (z.B. Schadensersatzansprüchen, Gewährleistungsrechte, etc.) führt, und finanzielle oder rechtliche Folgen mit sich bringt.

- **Fehlerverhütungskosten.** Unter Fehlerverhütungskosten werden alle Maßnahmen verstanden, die zur Qualitätssicherung direkt oder indirekt beitragen. Die Aktivitäten des RE fallen unter diese Kategorie von Kosten, mit dem Ziel Anforderungsfehlerkosten zu minimieren.

Von Wirtschaftlichkeit ist zu sprechen, wenn die Fehlerverhütungskosten kleiner sind als die zu erwartenden Fehlerkosten ohne vorbeugende Aktivitäten. Diesen Sachverhalt kann man direkt auf Umfang und Qualität des zu entwickelnden Anforderungsdokumentes anwenden, wenn man dieses als Mittel zur Fehlerverhütung versteht. Unter Argumentation des vorgestellten Modells zum Softwarelebenszyklus kann man behaupten, dass die Anforderungsfehlerkosten exponentiell mit der Verweildauer von Anforderungsfehler im System anwachsen.[10] So lautet die Empfehlung, alle möglichen Mittel einzusetzen, um eine derartige Entwicklung zu verhindern bzw. einzuschränken.

Der starke Trend zur Kundenorentierung und Ausrichtung der eigenen Aktivitäten am Kundennutzen, erfordert außerdem in allen Phasen der Softwareentwicklung optimierte Ansätze hinsichtlich Aufwand, Effizienz und Geschwindigkeit. Dies gilt auch für die Anforderungsanalyse. Aus diesem Grunde werden Vorgehensmodelle und deren Vorschläge zur Prozessverbesserung auf die Ablauforganisation übertragen. Einen Überblick gibt der nächste Abschnitt.

2.2. Requirements Engineering im Rahmen von Vorgehensmodellen

In der Einleitung wurde unter einem Vorgehensmodell (VGM) ein verbindliches, organisatorisches Ablaufkonzept verstanden. Je nach Art und Ausrichtung des Vor-

[10] [Sch97] S.20

gehensmodells, werden unter Einbeziehung von festgelegten Standards (Methoden, Richtlinien, Konventionen, Checklisten, etc.) alle typischen Phasen des Softwarelebenszyklus betrachtet. VGMs bilden einen Rahmen für organisatorische Abläufe bei häufig wiederkehrenden Problemstellungen und klären die Frage, *Was*, *Wie* und *Womit* etwas zu tun ist. Es erfolgen Festlegungen über die zu durchlaufenden Aktivitäten, deren Reihenfolge und die einzubeziehenden Personen.

Versteegen gibt zu bedenken, dass vom VGM einbezogene Aktivitäten nicht unbedingt sequentiell durchzuführen sind.[11] Es kommt eher darauf an, einen „logischen Zusammenhang“ zu entwickeln. Es ist von Vorteil, in einem VGM von Disziplinen (RUP) und/oder von Submodellen (V-Modell) zu sprechen, da man mit Hilfe dieser Strukturierungsmittel komplexe Zusammenhänge effektiver beschreiben kann. In Anlehnung an Balzert könnte eine Phase im VGM wie folgt charakterisiert werden:[12]

- **Ziel.** Jede Phase hat ein definiertes Ziel, welches sich klar von den Zielen anderer Phasen unterscheidet.
- **Aktivitäten.** Durchzuführende Aktivitäten dienen der Erreichung der angestrebten Ziele. Interessant sind Überlegungen, bestimmte oder sogar alle Aktivitäten zu automatisieren, um so die Effizienz der Abläufe zu erhöhen. Wenn das VGM im Stande sein soll, flexibel auf umfangreiche Vorhaben (z.B. bei Großprojekte) reagieren zu können, stellen zeitaufwändige manuelle Aktivitäten grundsätzlich einen Engpass dar. Handelt es sich bei den Aktivitäten um eine Reihe von organisierten und steuerbaren Aufgaben, spricht man von einem Workflow[13] bzw. einem Workflow-Konzept.
- **Rollenmodell.** Typischerweise sind in einem Softwareprojekt von Anfang an eine Vielzahl von Personen beteiligt. Dabei entwickelt jeder eine ganz eigene Sicht auf die ihm übertragenen Verantwortlichkeiten. Eine Rolle definiert eine abstrakte Person durch dessen Aufgaben, Verantwortlichkeiten und Befugnisse. Rollen werden durch Personen besetzt, die den Anforderungen bzgl. Erfahrungen, Kenntnissen und Fähigkeiten genügen. Ein Rollenmodell setzt die definierten Rollen in Beziehung und dient allen Projektbeteiligten der Transparenz. Dieses Prinzip klingt auf dem ersten Blick sehr streng, ist jedoch für die Phasen im Softwareprojekt ein wertvolles organisatorisches Ordnungskonzept.
- **Zwischenergebnis und Vorlagen.** Die im Verlauf der Entwicklung erarbeiteten Zwischenergebnisse bilden zusammengenommen ein Softwareprodukt.

[11] Vgl. [Ver00] S.91

[12] Vgl. zum folgenden Absatz [Bal01] S.54,55 und [R+02] S.76, 389ff.

[13] Der Begriff Workflow wird nach [DIN96] (Geschäftsprozessmodellierung und Workflow-Management) als ein computergestützter, administrierbarer, organisierbarer und steuerbarer Prozess definiert. Da Workflows immer eine festes Ablaufgerüst unter Einbeziehung von Entscheidungskriterien darstellen, gilt dessen Verwendung in Vorgehensmodellen als nahezu ideales Gestaltungsmittel.

Ein Zwischenergebnis, im Folgenden auch als Artefakt bezeichnet[14], beschreibt „ein Teil an Information, das produziert, modifiziert oder vom Prozess genutzt [..]“[15] wird. Im engeren Sinne handelt es sich dabei um ein Dokument, ein Modell oder Modellelement. VGMs enthalten Vorlagen und Richtlinien, die Hinweise geben, wie bei der Erstellung vorzugehen ist.

- **Vorgabe von Werkzeugen und Sprachen.** Der Einsatz von Werkzeugen (engl.: tool) dient der Unterstützung und Verbesserung von Arbeitsabläufen. Im Idealfall unterstützen Werkzeuge die Arbeit über mehrere Phasen und integrieren die erarbeiteten Artefakte. In der Praxis sind Werkzeuge jedoch häufig abgegrenzt auf einen konkreten Problemfall (z.B. Anforderungserfassung, Modellierung, Konfigurationsmanagement, etc.).

Vorgehensmodelle enthalten Empfehlungen, die den gesamten Softwarelebenszyklus betreffen und betrachten dabei nahezu alle Themen rund um die Softwareentwicklung. Als die wichtigsten dieser Submodelle/Disziplinen sind folgende zu nennen:

- **Analyse, Design und (Geschäftsprozess-)Modellierung.**
- **Anforderungsmanagement.**
- **Projektmanagement.**
- **Konfigurationsmanagement.**
- **Systementwicklung.**
- **Qualitätssicherung.**
- **Dokumentation.**

Vorgehensmodelle unterscheiden sich bzgl. des prinzipiellen Verständnisses zum Softwarelebenszyklus und den damit verbundenen Managementaufgaben nur minimal. Unterschiede ergeben sich vor allem im Anwendungsbereich (Domäne), bei der Einbeziehung von Werkzeugen (von Werkzeug-Neutralität bis hin zur strikten Kopplung), im Abstraktionsgrad (von allgemeingültigen Aussagen bis hin zu konkreten Vorgaben) sowie in der internen Konzeption (Submodelle, Module, Ebenen, Begriffe, etc.).

[14] Der Begriff „Artefakt“ stammt in diesem Zusammenhang originär aus dem RUP und hat sich im Sprachgebrauch zum Requirements Engineering weitestgehend etablieren können. In der deutschen Sprache wird der Begriff häufig als „von Menschenhand geformter vorgeschichtlicher Gegenstand“ ([Dud01]) beschrieben. Da diese Definition nicht unmittelbar vereinbar ist mit der Zielstellung eines Artefaktes im RUP, sollte man das englische Original „artifact“ besser als „Gegenstand“ oder „Produkt“ interpretieren ([NMR03]).

[15] Entnommen aus [Ver00] S.77 / Zitat einer deutschen Übersetzung

Es stellt sich nun die Frage, welche Teilbereiche durch Vorgehensmodelle theoretisch abgedeckt werden müssen, um eine ausreichende Unterstützung in Bezug auf die Anforderungsanalyse zu gewährleisten. Anders formuliert: Welchen Beitrag leisten die Vorgehensmodelle zum Requirements Engineering?[16]

Um diese Frage beantworten zu können, ist es notwendig die anfallenden Aktivitäten zu benennen und abzugrenzen, da nur so der Beitrag eines konkreten Vorgehensmodells bewertet werden kann. Einen Ansatz bietet SCHIENMANN mit seinen „Hauptaufgaben im Anforderungsmanagement". Aus der Überlegung heraus, welche grundsätzliche Einteilung der Aktivitäten sinnvoll ist, entwickelte SCHIENMANN die Bereiche „Steuerung und Verwaltung" (Querschnittsaufgaben) bzw. „Entwicklung und Durchführung" (Schritte). Abbildung 2.2 skizziert den Ansatz und bringt die Aktivitäten in einen Zusammenhang.[17] SCHIENMANN benutzt in seiner Darstellung ein von dieser Arbeit abweichendes Begriffsverständnis. Dies ist ein typisches Problem bei der Betrachtung von Literatur zum Thema RE. So hat in der Abbildung z.B. der Begriff „Anforderungsanalyse" eine leicht abweichende Bedeutung (Teilaktivität), wie er in Abschnitt 2.1 vorgestellt wurde. Daher empfiehlt sich bei der Interpretation der Darstellung, die Begriffe nicht auf strenge Konsistenz zum Rest der Arbeit zu prüfen, sondern sie in den Kontext entsprechend einzuordnen. Es folgt eine kurze Erläuterung der Aktivitäten die dem Bereich „Entwicklung und Durchführung" zugeordnet sind:

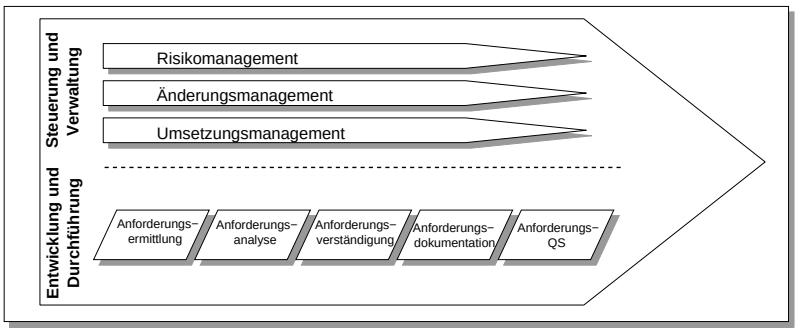

Abbildung 2.2.: *Hauptaufgaben im Anforderungsmanagement ([Sch02])*

- **Anforderungsermittlung.** Als gemeinsamer Lern- und Problemlösungsprozess werden bei der Ermittlung von Anforderungen aktiv Wünsche, Empfehlungen und Einwände entgegengenommen. Thematisiert werden auch rechtliche und organisatorische Fragestellungen. [18]

- **Anforderungsanalyse.** Nach der groben Aufnahme folgt die Analyse der Anforderungen, also eine formale Strukturierung der erfassten Daten. Dazu

[16] Vgl. zu diesem und folgenden Absatz [Ver00] S.91ff., [Sch02] S.33ff.
[17] Vgl. [Sch02] S.33, Abbildung 2.1
[18] Vgl. [Sch02] S.34ff.

gehört neben der Begriffsbestimmung vor allem die inhaltliche Konsolidierung (Bestandssicherung), Visualisierung und Abhängigkeitsbestimmung.[19]

- **Anforderungsverständigung.** Ab einem bestimmten Punkt werden Anforderungen rechtlich von den involvierten Parteien festgeschrieben. Bis die Annahme (bzw. auch Ablehnung) jeder einzelnen Anforderung in einer Releaseplanung abgeschlossen wird, kommt es im Vorfeld öfter zu konfliktreichen Spannungen. Unter Einbeziehungen von organisatorischen Techniken (Workshops, Reviews,etc.), kann diese Aktivität einfacher gestaltet werden.[20]

- **Anforderungsdokumentation.** Es folgt die obligatorische Erstellung des Anforderungsdokumentes. Dies erfolgt durch gebräuchliche Mittel der Formalisierung. Typischerweise kommen UML-Techniken (UseCase-Diagramme, Aktivitätsdiagramme, etc.) zum Einsatz sowie hierarchische Konstrukte zur Katalogisierung und Visualisierung von Abhängigkeiten.[21]

- **Anforderungsqualitätssicherung.** In der Qualitätssicherung werden durch unabhängige Stellen die ermittelten Anforderungen und zugehörige Artefakte validiert und verifiziert. Die Prüfung erfolgt hinsichtlich Struktur, Inhalt, Konsistenz und Form.[22]

Die in diesem Abschnitt vorgestellten theoretischen Überlegungen von Balzert, Rupp und Schienmann sollen als Betrachtungsschema für den nächsten Abschnitt dienen. Dabei werden Vorgehensmodelle auf ihren Beitrag hinsichtlich der Anforderungsanalyse untersucht. Es soll jedoch *nicht* der Versuch unternommen werden, das jeweilige Modell im Detail vorzustellen. Dies wäre auch im Rahmen dieser Arbeit nicht zu leisten. Das erklärte Ziel ist es, Ideen zum Requirements Engineering zu sammeln, um so eine möglichst sukzessive Einführung der dabei verwendeten Konzepte zu erlangen. Grundlage zur Auswahl der Vorgehensmodelle war eine Literaturrecherche und eine Untersuchung der ermittelten Ansätze. Um in das Untersuchungsfeld aufgenommen zu werden, wurden folgende Kriterien angesetzt:

[19] Vgl. [Sch02] S.37ff.
[20] Vgl. [Sch02] S.41
[21] Vgl. [Sch02] S.42ff.
[22] Vgl. [Sch02] S.44

- Das Vorgehensmodell soll explizit auf die Anforderungsanalyse Bezug nehmen. Grundlage ist die Begriffsdefinition aus der Einführung und eine möglichst vollständige Darstellung über den gesamten Softwarelebenszyklus. Es soll daher nur indirekt Bezug genommen werden auf die VGM-Paradigmen: evolutionäres Prototyping nach [PB93]/[Hal90], Zyklusmodell, Springbrunnenmodell (engl.: fountain model) nach [HS96], Spiralmodell nach [Boe86] und erweitertes Wasserfallmodell nach [Gil88].

- Diese Einschränkung führt dazu, dass innerhalb dieser Untersuchung Vorgehensmodelle nicht herangezogen werden sollen, die entweder Vorgänger sind oder bekannte Ansätze erweitern bzw. an bestimmten Stellen aufgreifen. Ein Beispiel dafür ist UML/USDP nach [JBR99] als Vorläufer des Rational Unified Process (RUP) oder der Business-Oriented Software Engineering Process (B-OEP) nach [Oes98], einem angepassten Abkömmling des RUP.

- Durch dieses Kriterium werden auch domänenspezifische Vorgehensmodelle wie beispielsweise ROPES oder SFB 501-Prozess nach [GKP99] (Embedded Systems) weggelassen.

- Vorgehensmodelle mit speziellen Fokus auf komponentenorientierte Entwicklungen bzw. Ansätze für Produktfamilien werden aufgrund der besonderen Vorbedingungen (geeignete Anwendungsdomäne, notwendige Geschäftsstrategie,...) nicht betrachtet. Beispiele dafür sind: Sherlock, Odyssey-DE, FAST, FODA, PuLSE oder Synthesis. Einen guten Überblick bietet [KE02].

- Das Vorgehensmodell sollte nicht ausschließlich ausgewählte Aktivitäten der Entwicklung unterstützen, sondern einen breiten Rahmen bieten. Daher wurden isolierte Methoden, formale Sprachen oder Modellierungsparadigmen nicht in die Untersuchung einbezogen. Beispiele dafür sind ERM, OMT/BOOCH Method/Objectory, SADT, SA, RSL/REVS/SREM, EPOS, GIST. Einen guten Überblick bietet [Par91].

- Ferner sollen ausschließlich Modelle betrachtet werden, die einen hohen Reifegrad erreicht haben und als Grundlage zur Entwicklung in der industriellen Praxis herangezogen werden können. Weniger bekannte Ansätze, wie Crystal Methodologies nach [CH01] oder Personal Software Process nach [Hum95] werden daher nicht betrachtet.

Im nächsten Abschnitt sollen vier spezielle Ansätze hinsichtlich der Anforderungsanalyse untersucht werden. Die theoretischen Überlegungen von Balzert, Rupp

und SCHIENMANN sollen hier als Betrachtungsschema dienen. An dieser Stelle soll eine kurze Charakterisierung inklusive einer Begründung für die Auswahl folgen:

- **V-Modell 97.**[23] Das V-Modell ist seit 1992 das offizielle Vorgehensmodell für IT-Projekte im deutschen öffentlichen Dienst und erfuhr 1997 eine Weiterentwicklung (Fortschreibung) im V-Modell 97. In Deutschland ist dieses Modell im industriellen Bereich aufgrund der freien Verfügbarkeit bestens bekannt. Da dieses Modell die breite Erfahrung im öffentlichen Dienst berücksichtigt und in der deutschen Literatur sehr häufig thematisiert wird, eignet es sich sehr gut für eine Untersuchung zum Thema dieser Arbeit.

- **Rational Unified Process (RUP).**[24] Das Prozessmodell der Rational Software Corporation besticht durch eine Vielzahl praxiserprobter Details. Im Vordergrund steht die strikte Ausrichtung auf objektorientierte Methoden, wie z.B. die effiziente und richtige Einbeziehung der UML aber auch die Integration und Unterstützung der Phasen durch passende Werkzeuge. Der RUP genießt weltweit eine konkurrenzlose Popularität und ist aufgrund der zeitgemäßen Ansätze ideal geeignet für Betrachtungen zum RE.

- **eXtreme Programming.**[25] Als viel versprechender Ansatz und Vertreter der „agilen Softwareentwicklung“ präsentiert sich das leichtgewichtige[26] Vorgehensmodell eXtreme Programming (XP). Mit XP wird der Versuch unternommen, der Lücke zwischen alten Erfahrungswerten und modernen Ansichten in der Softwareentwicklung zu begegnen. Als Ergänzung zum V-Modell und dem RUP, ihresgleichen allesamt Vertreter schwergewichtiger Vorgehensmodelle, kann man durch Betrachtung von XP und dessen zum Teil radikal neu formulierten Ansätze, den Blick auf die Anforderungsanalyse gewinnbringend erweitern.

- **Software Engineering Body of Knowledge (SWEBOK).**[27] Das „Gesammelte Wissen zur Softwaretechnik“ präsentiert sich nicht als Vorgehensmodell, sondern als gute Referenz so genannter Knowledge-Areas (dt.: Wissensgebiete), die alle wichtigen Elemente der Softwaretechnik als eine große Disziplin abdecken sollen. Die Arbeiten an SWEBOK sind noch nicht abgeschlossen. Das Dokument wird jedoch ständig durch Gutachter aus über 40 Ländern weltweit ergänzt, vervollständigt und kommentiert. Eine Betrachtung in dieser Arbeit soll einen kleinen Beitrag zu dieser Idee leisten.

[23] Vgl. Spezifikation nach: [DW99], Stand November 1999. Im Folgenden V-Modell genannt.
[24] Vgl. Spezifikation nach: [RSC02] Version 2002.05.00, Stand Juni 2002.
[25] Vgl. [Bec00]
[26] Für die Unterscheidung von schwergewichtigen und leichtgewichtigen Vorgehensmodellen siehe [Eck00]. Leichtgewichtige Modelle enthalten weniger strenge Ablaufvorgaben, wodurch Entwickler und Kunden ein größerer Spielraum für eigene Methoden gelassen wird.
[27] Vgl. Spezifikation nach: [CS01] Trial Version 1.0, Stand Mai 2001.

2.3. Ausgewählte Vorgehensmodelle im Überblick

2.3.1. V-Modell 97

Ansatz

Die im V-Modell beschriebenen Konzepte werden durch vier zentrale Submodelle beschrieben:[28]

1. Projektmanagement (PM)
2. Konfigurationsmanagement (KM)
3. Systemerstellung (SE)
4. Qualitätssicherung (QS)

Die Submodelle sind zwar eng miteinander verknüpft, thematisieren jedoch, wie bereits aus der Namensgebung ersichtlich, inhaltlich geschlossene Problemfelder der Soft- und Hardwareentwicklung. Neben der Beschreibung der einzelnen Submodelle, gibt es im V-Modell sehr komplexe Vereinbarungen zu organisatorischen Aspekten wie Bezeichnungen, Ablaufrichtlinien, Informationsflüssen, Rollenkonzepten, Schnittstellen oder Handbuchbenutzung.

In Bezug auf die Anforderungsanalyse fällt auf, dass eine direkte Zuordnung auf ein bestimmtes Submodell (z.B. Anforderungsmanagement) fehlt. Außerdem können die Submodelle PM und KM aus Betrachtung sogar gänzlich entfallen, da sie nicht direkt Thema dieser Arbeit sind. Die verbleibenden Submodelle SE und QS thematisieren jedoch die Anforderungsanalyse in Form von Hauptaktivitäten. Während das Submodell QS nur einen ergänzenden Beitrag leistet, liegt der Schwerpunkt im Submodell SE mit den folgenden Hauptaktivitäten:[29]

- SE 1: System-Anforderungsanalyse
- SE 2: Systementwurf
- SE 3: SW-Anforderungsanalyse
- SE 4: SW-Grobentwurf
- SE 5: SW-Feinentwurf

Die Aktivitäten SE2, SE4 und S5 sind kein Bestandteil der Anforderungsanalyse (Entwurf!). Das V-Modell sieht zwar eine sequentielle Bearbeitung vor, was jedoch nicht als strenge Reihenfolge verstanden werden soll. So wird argumentiert, dass

[28] Vgl. [DW99] S.8ff.
[29] Vgl. [DHM98] S.176

erst das zyklische Wiederholen der Aktivitäten Analyse und Entwurf die Grundlage bildet für die vollständige Ermittlung der Anforderungen. Die strikte Trennung zwischen „Was“ und „Wie“ sieht dieser Ansatz daher nicht vor. Abbildung 2.3 soll überblicksartig die hier angestrebte Betrachtungsweise auf das V-Modell verdeutlichen.

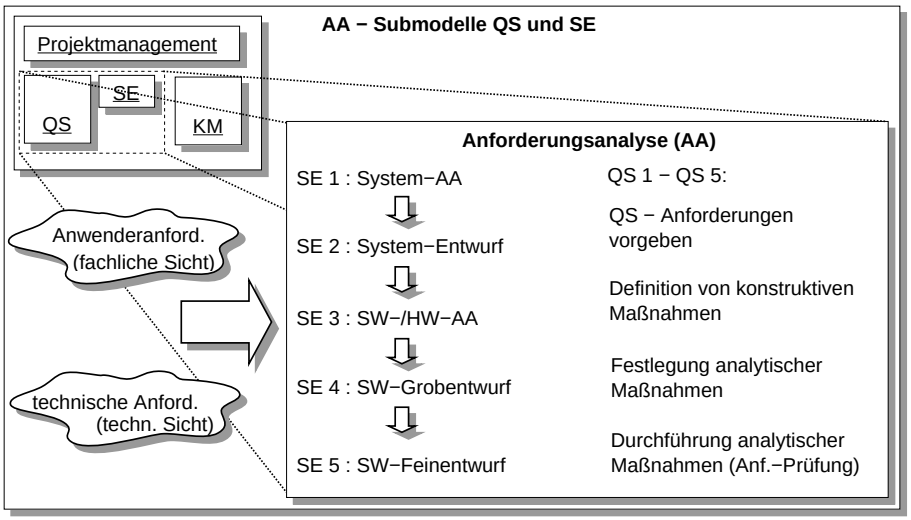

Abbildung 2.3.: *Anforderungsanalyse im V-Modell*

Zu erläutern ist die Unterteilung der Anforderungsanalyse in eine System-Sicht (SE 1) und eine Software-Sicht (SE 3).[30] Dem liegt die Idee zugrunde, dass bei der Entwicklung von komplexen Systemen zunächst eine grobe Beschreibung der Funktionalität in der Gesamtheit erarbeitet wird. Es muss also allen Beteiligten klar werden, welche fachlichen Aufgaben bzw. Funktionen durch die angestrebte IT-Lösung überhaupt abgedeckt werden sollen und welche nicht.[31] Die ab SE 3 folgenden Analysetätigkeiten spezifizieren die Anforderungen an die zu erstellenden Softwareteile.

Im Submodell SE werden insgesamt 9 Hauptaktivitäten unterschieden, wobei nur die ersten fünf Ebenen für die Anforderungsanalyse interessant sind. Im Regelungsteil des Vorgehensmodells wird jede Hauptaktivität in den Abschnitten Inhalt, Produktfluss, Abwicklung und Rollenzuordnung im Einzelnen beschrieben. In diversen Erweiterungen werden noch Ausführungen zu Methoden, Werkzeuganforderungen, externen Normen sowie Erfahrungsberichte eingebracht.[32]

Manchmal kann die Art und Ausrichtung des Projektes es jedoch erfordern, bestimmte Anpassungen im Detaillierungsgrad der (Zwischen-)Produkte vorzunehmen. Das im V-Modell als Tailoring bekannte Konzept erlaubt es, die Haupt- und

[30] Vgl. [DW99] Anhang, Teil 1: Regelungsteil
[31] Vgl. [DHM98] S.46
[32] Vgl. [KBP03], für eine Quelle mit einem sehr guten Überblick und webbasierter Aufbereitung des V-Modells.

Teilaktivitäten zu verkürzen oder zu erweitern. So sind beispielsweise einige Teilaktivitäten aus SE1, SE2 und SE4 in Großprojekten unabdingbar, während sie in kleineren Projekten lediglich zu unnötigen organisatorischen Aufwand führen.

Das Submodell QS bringt keine Aktivitäten ein, die eindeutig der Anforderungsanalyse zuzuordnen sind. Aufbau und Struktur des Submodells verfolgen die Zielstellung, den Qualitätsbegriff in der gesamten Projektabwicklung entsprechend zu berücksichtigen und die Qualität durch Tests/Nachweise sicherzustellen. Die in der Abbildung 2.3 aufgeführten Punkte beschreiben daher die Grundphilosophie zum Qualitätsmanagement, die für alle im VGM vorgesehenen Produkte gilt.

Anforderungsermittlung

Zunächst muss festgestellt werden, dass im V-Modell kein isoliert betrachtetes Prozessmodell zur Anforderungsanalyse vorgesehen ist. Außerdem wurden ursprünglich keine verbindlichen Techniken zur Ermittlung, Entwicklung oder Analyse von Anforderungen festgelegt.[33] Mit dem Entwicklungsstandard EStdIT als technische Richtlinie wurden jedoch für die Tätigkeiten und Ergebnisse in Softwareprojekten, Standardisierungen auf den Ebenen

- Vorgehensweise,
- anzuwendende Methoden und
- funktionale Anforderungen an einzusetzende Werkzeuge

erarbeitet mit dem Ziel, die Vielfalt der verfügbaren Methoden und Werkzeuge zu beschränken.[34]

Das V-Modell führt die Rolle „Systemanalytiker“ ein. Eine Person in dieser Rolle hat die Aufgabe, die externen Vorgaben des Auftraggebers (AG) entsprechend aufzubereiten, indem er zwischen

- Anwenderforderungen und
- technischen Anforderungen unterscheidet.[35]

Beide Anforderungstypen sind das Resultat bzw. die Produkte der Aktivitäten, die der Systemanalytiker durchzuführen hat. Eine grafische Ablaufbeschreibung der Systemanforderungsanalyse befindet sich im Anhang A.1.

[33] Vgl. [DHM98] S.173

[34] Vgl. [KBP03] Abschnitt GD252. Hier wird auch die ursprüngliche Konzeption des V-Modells ersichtlich, nämlich die standardisierte Entwicklung von besonders sicherheitskritischen Anwendungen wie sie beispielsweise die Bundeswehr benötigt. Da diese Arbeit lediglich auf die Konzepte diverser Vorgehensmodelle eingehen soll, erscheint es nicht hilfreich, hier die Bundesvorgaben näher zu betrachten.

[35] Vgl. [DHM98] S.178

Die Anforderungsermittlung beginnt mit einer groben Systembeschreibung, die als Eingangsinformation für die weiteren Entwicklungsschritte dient. Da diese in einem ersten Durchgang nicht vollständig erhoben werden kann, ist eine zyklische Wiederholung vorgesehen.[36] Droeschel weist in diesem Zusammenhang deutlich darauf hin, dass ein VGM keine Lösung für die Probleme bieten kann, die im Zuge des Umgangs mit Anforderungen auftreten können. Allerdings, so Droeschel, kann das V-Modell die richtigen Hinweise geben.[37] Die Kernidee besteht darin, eine strikte Trennung zwischen fachlichen und technischen Anforderungen vorzunehmen.[38]

Anforderungsanalyse

Für die Strukturierung und Abhängigkeitsbestimmung von Anforderungen ist der Einsatz von formalisierenden Techniken erforderlich. Im V-Modell sollen so genannte „Methodenzuordnungstabellen“ einen Rahmen bilden für entsprechende Methodenvorschläge.[39] Wie diese Zuordnung genau zu erfolgen hat, überlässt das VGM der Projektleitung. Dennoch werden im V-Modell überblicksartig bestimmte Ansätze erwähnt. In der Handbuchsammlung des V-Modells wird zum Beispiel der „Zusammenhang zwischen Geschäftsprozessoptimierung und dem V-Modell“[40] besonders unterstrichen. Außerdem gibt es gesonderte Regelungen zur Anwendung der UML im V-Modell.[41]

Konsequent weiterentwickelt wird jedoch die Trennung in Anwenderforderungen und technischen Anforderungen. Das V-Modell verlangt bei der Bearbeitung der Anforderungen eine erweiterte Zerlegung. Ziel dieses Ansatzes ist eine Anforderungszuordnung, wie sie in Abbildung 2.4[42] dargestellt ist. Diese dient zum einen der Ermittlung von Nachweiselementen, die später zur Abnahme der Software relevant sind, zum anderen der Transparenz der Anforderungen bzw. Nachvollziehbarkeit in der Implementierungsphase. Für die Analyse und Aufgliederung sieht das Modell bei den Anwenderforderungen folgende Kriterien vor:[43]

1. Fachliche Anforderungen
2. Qualitätsanforderungen
3. Randbedingungen
4. Organisatorische Randbedingungen

[36] Vgl. [DHM98] S.46
[37] Vgl. [DHM98] S.172
[38] Vgl. [DHM98] S.173
[39] Vgl. [DW99] S.66
[40] Vgl. [DW99] dieser Teil entfällt in der gedruckten Ausgabe, ist jedoch auf der beiliegenden CD-ROM einsehbar unter Vm-97/EStdIT/VModell/Handbuch/GPO.DOC
[41] Vgl. [DW99] S.146ff.
[42] Vgl. [DHM98] S.183
[43] Vgl. [DHM98] S.178

5. Technische Randbedingungen

Für technische Anforderungen wird keine spezielle Unterteilung empfohlen.

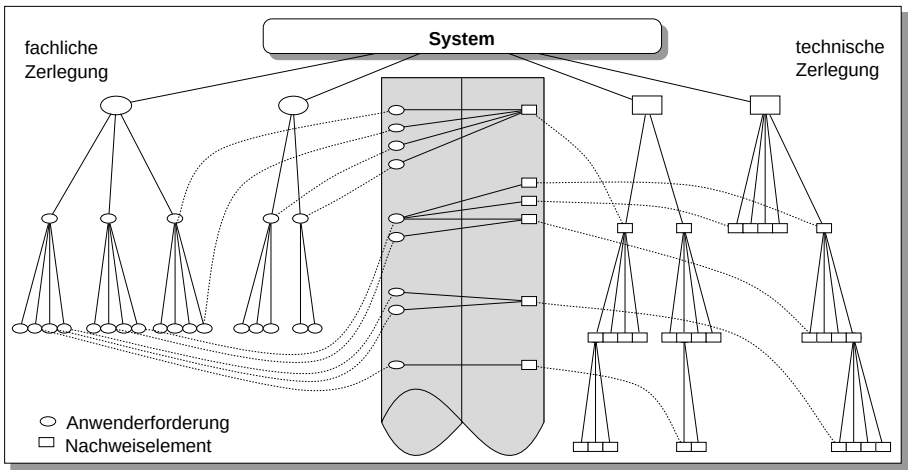

Abbildung 2.4.: *Anforderungszuordnung im V-Modell ([DHM98])*

Anforderungsverständigung

Organisatorische Empfehlungen oder methodische Techniken zur Konfliktbewältigung der involvierten Parteien sind nicht Gegenstand des V-Modells.

Anforderungsdokumentation

Für die Dokumentation der Anforderungen empfiehlt das V-Modell eine baumartige Struktur. Wie im Abschnitt zur Anforderungsanalyse schon dargelegt, zielt diese Idee auf den Einsatz von Nachweiselementen ab. Allerdings steht hier eher die Dokumentation/Nachvollziehbarkeit im Vordergrund und weniger die Analyse. Droeschel merkt außerdem an, dass die technische Architektur oft keine hierarchische Anforderungsstruktur darstellt, sondern vielmehr eine vernetzte.[44] So sollte die Anforderungsdokumentation mit Mitteln erfolgen, die die spezifischen Abhängigkeiten von technischen Anforderungen und Anwenderforderungen entsprechend berücksichtigen können. Ergänzend sei erwähnt, dass das V-Modell außerdem eine Unterteilung in funktionale und nichtfunktionale Qualitätsanforderungen vorsieht. Zur Begriffsbestimmung bedient sich das VGM der Qualitätsnorm ISO 9126 und DIN 66272.

In diesem Zusammenhang ist zu bedenken, dass das V-Modell in einer Zeit entwickelt wurde, zu der die UML noch relativ neu war. So ist es nicht verwunderlich, dass die „Elementarmethode Use Case Modellierung“ als „Ergänzungen der

[44] Vgl. [DW99] S.182,183ff.

V-Modell Richtlinien“ nur beispielhaft einbezogen wurde.[45] An dieser Stelle sei auf die Ausführungen im Abschnitt 2.5.1 verwiesen.

Anforderungsqualitätssicherung

Es gibt einen Unterschied zwischen Qualität der Anforderungen und Anforderungen an die Qualität. Beide Themen passen theoretisch unter den Titel Anforderungsqualitätssicherung. An dieser Stelle soll jedoch untersucht werden, wie im V-Modell der Qualitätsbegriff zum einen auf die Prozesse (Ermittlung, Analyse,...) und zum anderen auf die Produkte (Dokumentation, Modelle,...) übertragen wird.

Versteegen sieht in dem Submodell QS einen „kritischen Bereich“, gerade wegen der Verschiebung bzw. Neuausrichtung des Qualitätsbegriffs in den letzten Jahren.[46] Die im Submodell QS beschriebenen Aktivitäten dienen vorrangig dem Test und damit auch zum Nachweis der Erfüllung von Anforderungen. Versteegen stellt fest, dass es im V-Modell kaum Ansätze gibt, die dazu beitragen würden, schon während der Ermittlung, Analyse und Dokumentation von Anforderungen potentielle Mängel von vornherein zu verhindern.

Das Submodell QS formuliert konstruktive und präventive Maßnahmen, die unter Verwendung von analytischen Maßnahmen ergänzt werden.[47] Dies bedarf einer Erklärung. Unter konstruktiven und präventiven Maßnahmen werden im V-Modell Werkzeuge und Methoden verstanden, die zur Unterstützung und/oder Standardisierung von Entwicklungsprozessen führen. Beispiele dafür sind integrierte Entwicklungsumgebungen, Testwerkzeuge oder Dokumentvorlagen.

Trotz dieser unscharfen Formulierungen findet man im V-Modell einige Hinweise um die Qualität der Anforderungen zu sichern. Alle erstellten Produkte unterliegen einem Status-Workflow („geplant“, “vorgelegt“, etc.). Dieses sehr bürokratische Konzept kann -muss jedoch nicht- die Qualität aufgrund der formalen Prüfung und Nachvollziehbarkeit erhöhen. Es spricht auch dafür, Rollen wie „Projektleiter“, „Systemanalytiker“ und „QS-Verantwortlicher“ einzuführen, um so eine enge „Verzahnung an der Beschreibung im Projekt“[48] zu erreichen.

2.3.2. Rational Unified Process

Ansatz

Der Architekturansatz des RUP basiert auf einen konfigurierbaren Prozess. Die Idee entstand aus der Erkenntnis, dass die Softwareentwicklung ein zu komplexer Vorgang ist und Softwareprojekte stets mit unterschiedlichen Problemen konfrontiert

[45] Vgl. [DW99] S.150ff.
[46] Vgl. [V+02] S.37ff. Hier wird auch verwiesen auf neuere betriebswirtschaftliche Führungskonzepte wie z.B. das Total Quality Management (TQM) oder die Ideen des Kontinuierlichen Verbesserungsprozesses (KVP).
[47] Vgl. [VSH01] S.39
[48] Vgl. [DW99] S.68

sind. Der RUP versteht sich daher weniger als Vorgehensmodell (in seiner strengen Auslegung), sondern als Hilfsmittel in Form eines Prozessmodells[49]. Im RUP werden daher Empfehlungen, Vorlagen und Musterbeispiele kombiniert mit Werkzeugen der Softwareentwicklung, um daraus auf methodische Art und Weise[50] den typischen Alltagsproblemen zu begegnen.[51] Eine Unterteilung erfolgt im RUP durch die folgenden Disziplinen:[52]

1. Geschäftsprozessmodellierung (RUP: Business Modelling)
2. Anforderungen (RUP: Requirements)
3. Analyse und Design (RUP: Analysis and Design)
4. Realisierung (RUP: Implementation)
5. Tests (RUP: Test)
6. Verteilung (RUP: Deployment)

In Bezug auf die Anforderungsanalyse sind hierbei die Punkte 1., 2. und 3. relevant, wobei der Schwerpunkt deutlich auf der Analyse liegt. Alle Disziplinen werden unterstützt durch die organisatorisch-planenden Aktivitäten

- Konfigurations- und Änderungsmanagement
  (RUP: Configuration and Change Management) sowie
- Projektmanagement
  (RUP: Project Management).

Die genannten Disziplinen sind für sich genommen jedoch *keine* abgeschlossenen Teilaufgaben die in sequentieller Reihenfolge zu durchlaufen sind. Der iterative Ansatz im RUP sieht dafür eine grobe phasenartige Unterteilung (von der ersten Idee bis zur Abnahme) vor, um den logischen Gesamtverlauf zu skizzieren.

[49] Im Folgenden werden die RUP-spezifischen Begriffe wie z.B. Prozessmodell (Vorgehensmodell), Artefakt (Produkt), Disziplin (Hauptaktivität) usw. verwendet. Dies soll nicht als Inkonsistenz der Begrifflichkeiten interpretiert werden. Es dient hauptsächlich der Abgrenzung gegenüber dem V-Modell.

[50] Der praktische Aspekt wird betont durch das Verständnis, den RUP selbst als ein Werkzeug anzuerkennen und so auch zu benutzen. Die klar strukturierte HTML-Version des Dokumentes unterstützt den Anwender zum Beispiel durch eine Vielzahl von Grafiken, Dokumentvorschlägen, Checklisten und Hinweisen.

[51] Vgl. [Tec03]

[52] Die im Folgenden verwendeten deutschen Übersetzungen in Anlehnung an [V$^+$02] S.44ff. und [R$^+$02] S.53ff. Da der RUP nur in einer englischen Version verfügbar ist, sollen für einen besseren Vergleich an den wichtigen Stellen die Orginalbegriffe mit aufgeführt werden.

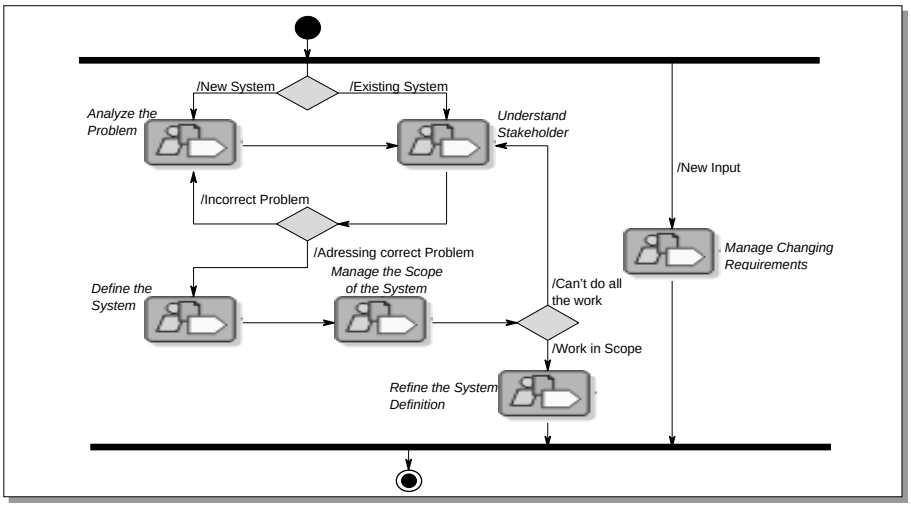

Abbildung 2.5.: *Anforderungsworkflow im RUP ([RSC02])*

Die Grafik in Abbildung 2.5[53] soll die im Folgenden genannten Begriffe in einer Übersicht darstellen. Kern der Abbildung ist der Anforderungsworkflow. Dieser stellt den zentralen Angriffspunkt für alle weiteren Überlegungen zur Anforderungsanalyse im RUP dar. Im Vergleich zu den bisher angestellten Überlegungen zur Anforderungsanalyse, erfolgt durch den RUP eine deutliche Ausweitung der Konzepte. Die Erweiterungen beziehen sich vor allem auf:

- iterative Entwicklung und Analyse von Anforderungen,
- integrierte Artefakte durch Werkzeugunterstützung,
- stark erweitertes Rollenmodell für die Analysetätigkeiten,
- erhöhte Detaillierung durch Workflows,
- Änderungsmanagement im gleichen Rang mit anderen Hauptaktivitäten sowie
- konsequenter Einsatz der UML als graphische Sprache.

Das Änderungsmanagement wird durch einen eigenen Workflow illustriert und ist im Anhang A.2 nachzuschlagen. In den weiteren Ausführungen werden die sechs Hauptaktivitäten im Anforderungsworkflow auf die von Schienmann formulierten Hauptaufgaben übertragen. Dabei wird das Rollenmodell des RUP einbezogen, das hier kurz erläutert werden soll:[54]

- **Systemanalytiker.** Leitet und koordiniert die Ermittlung der Anforderungen und die Modellierung der Anwendungsfälle, skizziert die Funktionalität des Systems und bestimmt die Grenzen.

[53] In Anlehnung an [RSC02] Overview und Disciplines→Requirements→Requirements Overview.
[54] Vgl. [V+02] S.22,23, [Ver00] S.74ff., [RSC02] Roles and Activities→Analyst Role Set

- **Anforderungsgutachter.** Plant und vollzieht formale Reviews zu den ermittelten Anforderungen und aufgestellten Anwendungsfällen.
- Der RUP beschreibt außerdem die Rollen Geschäftsdesigner, Geschäftsprozessanalytiker, Anforderungsdesigner und Geschäftsmodellgutachter. Die Einführung dieser Spezialisten ist abhängig der Projektgröße und nicht vorgeschrieben.

Anforderungsermittlung (Analyse des Problembereichs)

Versteegen beschreibt diese Aktivität auch als „High Level"-Beschreibung. Er weist auch darauf hin, dass es unmöglich ist, einen ausreichenden Systemanforderungsstatus zu bekommen, bevor die eigentlichen Entwicklungsarbeiten begonnen haben. [55] Dabei muss auch berücksichtigt werden, dass sich sowohl das Verständnis des Auftraggebers (AG) als auch das des Auftragnehmers (AN) im Laufe der Ermittlungsarbeiten ständig weiterentwickelt. Der RUP empfiehlt ohne Angabe einer speziellen Reihenfolge die folgenden Aktivitäten, die zum Teil zeitgleich aber stets unter Berücksichtigung der inhaltlichen Konsistenz durchzuführen sind:[56]

- Entwicklung und Beschreibung der Vision
- Erstellung eines Projektglossars
- Entwicklung einer Aufnahmestrategie (Anforderungs-Plan)
- Sammlung und grobe Beschreibung der Akteure und Anwendungsfälle

Der Systemanalytiker und teilweise auch der Projektmanager (als Organisator) nehmen die Anforderungen von allen extern Beteiligten (Kunde, Anwender, betroffene Abteilungen) durch Interviews, Workshops oder Umfragen auf. Nach der Erfahrung der RUP-Autoren hat es sich als hilfreich erwiesen, zur Dokumentation von Anfang an UML-Anwendungsfalldiagramme einzubinden.

Das große Ziel der Anforderungsermittlung ist die Erarbeitung der entscheidenden Informationen für die erste Konzeptionalisierungsphase (Iteration). Dabei ist die Zahl der Iterationen in dieser Disziplin maßgeblich vom Umfang des Projektes abhängig. Wie üblich wird der Systemanalytiker durch zahlreiche Vorlagen und Checklisten unterstützt. Als Vorgehensweise empfiehlt die Literatur folgende Schritte:[57]

1. Konzeption eines Fragebogens mit dem Ziel, die wesentlichen Anforderungen ohne besondere Detaillierung ermitteln zu können, d.h. eine grobe Sammlung von Akteuren und Anwendungsfällen.

[55] Vgl. [Ver00] S.92ff.
[56] Vgl. [RSC02] Disciplines→Requirements→Workflow→Analyze the Problem
[57] Vgl. zum folgenden Absatz [Ver00] S.98,99

2. Im Rahmen von Workshops oder Konzeptionsteams erfolgt anschließend die Ausarbeitung und Abgrenzung der betrieblichen Anwendungsfälle inkl. einer Beschreibung der Problemfälle.

3. Durch Einzelinterviews werden im letzten Schritt genaue Aktivitätsdiagramme erarbeitet und solange Informationen gesammelt, bis sich zur fachlichen Problematik keine weiteren Fragen mehr ergeben.

Anforderungsanalyse (Definition des Systems)

Im Vorfeld der Anforderungsanalyse muss zwischen allen Beteiligten ein Konsens über die abzudeckenden Problemfelder des Projektes bestehen. Der RUP definiert aus diesem Grunde die klare Bedingung, dass die reinen Ermittlungsarbeiten abgeschlossen sind und die fachlichen Anforderungen soweit korrekt beschrieben wurden. Ist dies nicht der Fall, so muss die Anforderungsermittlung wiederholt werden.[58]

In der Anforderungsanalyse stehen in erster Linie die bisher gesammelten Artefakte im Vordergrund. Auf der Grundlage des Modells wird folgende Unterteilung empfohlen:

- vollständige Überarbeitung und Verfeinerung des Projektglossars,
- Ergänzung der Akteure und Anwendungsfälle,
- Analyse der Anforderungen auf erweiterte Eigenschaften (Attribute) sowie
- Abhängigkeitsbestimmung von Anforderungen.

Die Arbeiten erfordern neben einer qualitätsgerechten Sorgfalt auch die intuitiven Eigenschaften des Systemanalytikers. Dieser muss in der Lage sein, die Anforderungsattribute richtig zu sammeln, zu sortieren und nach einem geeigneten Schema zu strukturieren. Erfahrungswerte haben gezeigt, dass Anforderungen nach einem Standardschema leichter zu ordnen sind. Der Systemanalytiker wird daher im RUP nach dem „FURPS+“-Prinzip unterstützt. Dabei handelt es sich um eine Ausweitung der im V-Modell vorgeschlagenen Unterteilung von Anwender- und technischen Forderungen:[59]

- Functionality - Funktionalität/Systemverhalten
- Usability - Anwenderfreundlichkeit
- Reliability - Zuverlässigkeit und Laufverhalten
- Performance - Performanz/Reaktionszeiten
- Supportability - Wartbarkeit

[58] Vgl. [RSC02] Disciplines→Requirements→Workflow→Define the System und [Ver00] S.104ff.
[59] Vgl. [RSC02] Glossar zum Begriff „FURPS+“

- Design Constraints - technische Randbedingungen

Die Unterteilung gibt nur einen groben Rahmen vor. In der RUP Dokumentation kann man zahlreiche Beispiele, Vorlagen und Checklisten finden.

Unter dem Abschnitt „Tool Mentoren“ verweist die RUP-Spezifikation auf das Werkzeug „Rational RequisitePro“, dessen Ziel es ist Anforderungen und Änderungsaufträge teambasiert zu organisieren.[60]

Anforderungsverständigung (Stakeholder verstehen)

Die Anforderungsverständigung, d.h. die Absprache zwischen externen Interessenten und dem Auftragnehmer erfolgt konzeptionell durch eine Kontrollschleife. Dies dient der Absicherung beider Seiten und wird im RUP mittels Checklisten u.a. zu folgenden Themen realisiert:[61]

- Werden die Probleme korrekt wiedergegeben?
- Adressieren die aufgenommenen Anforderungen die richtigen Problembereiche?
- Wird die Wunschliste richtig wahrgenommen?
- Wurden alle Schlüsselfunktionalitäten identifiziert?
- ...

Die Anforderungsverständigung ist eng mit der Anforderungsermittlung verknüpft. Dabei ist der Änderungsauftrag (RUP: change request) auf beiden Seiten ein integraler Bestandteil zur erfolgreichen Verständigung und wird im RUP konsequent berücksichtigt. Das betrifft alle Aspekte der organisatorischen Planung (Konzeption des Workflows), der Verwaltung (Dienstweg und Bearbeitung einer Änderung in dem Workflow) und der inhaltlichen Ausrichtung (Dokumentation, Einordnung und Nachvollziehbarkeit). Dem Anforderungsgutachter fällt die Aufgabe zu, Änderungsanträge auf ihre Abhängigkeiten zu anderen Anforderungen hin zu untersuchen. Werden Widersprüche festgestellt, kann der Gutachter einen Änderungsantrag zurückstellen. Im RUP werden sehr viele Artefakte (z.B. Glossar, Diagramme, Visionsdokumente, Anforderungsattribute, Design- und Testmodelle) definiert, die nach einer Änderung zu überarbeiten sind.

Anforderungsdokumentation (Systemdefinition verfeinern)

Die Dokumentation ist eine Kombination von diversen Vorlagen, UML-Ansätzen, Modellierungsrichtlinien sowie werkzeugspezifischen Paradigmen (hierarchisch oder vernetzt). Dabei strebt der RUP einen strukturierten Ansatz mit dem Ziel an[62]

[60] Vgl. [RSC02] Tool Mentors→Rational RequisitePro Tool Mentors
[61] Vgl. zum folgenden Absatz [Ver00] S.95,96
[62] Vgl. zum folgenden Absatz [V+02] S.23ff.

- die Anforderungen zu priorisieren, zu filtern und verfolgen zu können,
- eine Messbarkeit der Anforderungsermittlung zu erreichen,
- auf automatisierbare Dokumentationsabläufe zu setzen.

Die Dokumentationsarbeiten erfolgen im RUP streng werkzeuggestützt. Dieser Ansatz ist problematisch, wenn man den RUP werkzeugunabhängig einsetzen will. Dennoch ist die Zielsetzung dieser Idee klar: die Verbesserung der Kommunikation in und außerhalb des Projektteams durch eine integrierte Datenhaltung.[63]

Anforderungsqualitätssicherung

Im RUP gibt es kein eigenes QS-Modell oder einen QS-Spezialisten.[64] Dies ist mit dem grundsätzlichen Verständnis zum Qualitätsbegriff verbunden, wonach die Qualität nicht „im Ergebnis einfach hinzugefügt“ wird. Jede bearbeitende Stelle hat unmittelbaren Einfluss auf die Qualität der Ergebnisse. Daher unterscheidet der RUP zwischen

1. Produktqualität (Artefakte werden vorgelegt, geprüft und bestätigt) und
2. Prozessqualität (Prozesse werden geprüft, hinterfragt und bewertet).

Da Anforderungen unmittelbaren Einfluss auf die Ergebnisqualität haben, sind diese verständlicherweise besonders kritisch zu untersuchen. Bedingt durch die iterative Vorgehensweise werden falsch formulierte Anforderungen im RUP dennoch relativ früh erkannt. Natürlich bietet das iterative Vorgehen keinen Schutz vor fachlichen Missverständnissen, man kann aber durch die kürzeren Abstände der Tests das (korrigierende) Feedback vom Kunden schneller berücksichtigen.

Die zahlreichen Vorlagen sind, auch im Zusammenhang mit der Qualität ein wertvoller Beitrag. Unter Berücksichtigung des komplexen Rollenmodells wird der Qualitätsbegriff durch alle Disziplinen hinweg einheitlich thematisiert, was auch in der Form und Darstellung der Beispiele und des Modellansatzes selbst wiederzufinden ist.

2.3.3. eXtreme Programming

Ansatz

Eine stetig wachsende Gemeinde von Softwareentwickler verbindet in jüngerer Zeit mit dem Schlüsselwort „Agilität“[65] einen neuen Ansatz für die Durchführung von IT-Projekten. Sie fordern „agile Prozesse statt starre Vorgehensmodelle“[66] und argumentieren mit der langjährigen Erfahrung, wonach „Änderungen und Anpassungen

[63] Vgl. [V+02] S.27,28
[64] Vgl. [Kru01]
[65] Zum Begriff siehe auch *The AgileAlliance* unter `http://www.agilealliance.org`
[66] Vgl. [Sta02]

der Normalfall sind und nicht die Ausnahme“. Selbst sehr genau entworfene Anforderungsdokumente verändern sich im Laufe der Entwicklung. Dies wirft die Frage auf, welche Prozesse besser geeignet sind, um auf die Wünsche der Kunden schnell und effektiv reagieren zu können.

Als Synonym für die agile Softwareentwicklung steht mit eXtreme Programming (XP) eine Sammlung von Techniken zur Verfügung, die auf den folgenden zwei Grundprinzipien aufbaut:[67]

1. **Develop for today.** Strikte Konzentration auf aktuelle Problemstellungen ohne Rücksicht auf evtl. später vorzunehmende Erweiterungen. Dieses Prinzip soll unnötigen Aufwand für nicht zu realisierende zukünftige Erweiterungen vermeiden.

2. **Simple Design.** Der XP-Erfinder Beck erinnert daran, die Lösungen so einfach wie möglich, risikoarm, flexibel, kalkulierbar aber dennoch exakt (redundanzfrei) zu gestalten. Dies geht mit der Zielstellung einher, die bereits verstandenen Anforderungen strikt und ohne ausschweifenden Diskussionen umzusetzen.

Die zentrale Idee liegt darin, dass der Auftraggeber (zukünftiger Nutzer, Kunde, etc.) erst dann eine Vorstellung von dem System hat, wenn er zum ersten Mal eine lauffähige Anwendung sieht und ihm damit eine Argumentationsgrundlage zur Verfügung steht. Es ist demzufolge sicherzustellen, dass zu jedem beliebigen Zeitpunkt eine, wenn auch nicht vollständige jedoch lauffähige Version angeboten werden kann.

Das eigentliche Vorgehen in XP unterscheidet sich grundlegend von den Ansätzen der bisher betrachteten Modelle. Während diese hauptsächlich einer phasenorientierten bzw. workfloworientierten Sichtweise nachgehen, liegt die Essenz von XP in einer überschaubaren Anzahl von Techniken mit besonderer Effizienz und bewiesener Praxistauglichkeit in Verbindung mit möglichst gelockerten Vorgaben für die eigentliche Entwicklung.[68] Nicht jede XP-Technik ist für die hier angestrebte Untersuchung eine Betrachtung wert.[69] Bevor jedoch genauer darauf eingegangen wird, erscheint es sinnvoll, zunächst ein weiteres Konzept von XP grafisch aufbereitet vorzustellen. In Abbildung 2.6 wird der Teil in XP abgebildet, der am ehesten der klassischen Analysephase entsprechen würde. Der grau hinterlegte Teil ist der für die Anforderungsanalyse ausschlaggebende Bereich.

[67] Vgl. [Eck00] S.2

[68] Die Ausführungen folgen dem Wortlaut der einschlägigen Literatur zum eXtreme Programming wie in [Bec00], [LRW02] oder auszugsweise in [Eck00] beschrieben. Bestimmte euphorische Anmerkungen über die Innovation des Ansatzes finden sich also auch in dieser Beschreibung wieder, obwohl sie in einer wissenschaftlichen Darstellung natürlich durch Objektivität begründet sein sollten.

[69] Vgl. [LRW02] Kapitel 2 für eine ausführliche Darstellung und Bewertung der XP-Techniken.

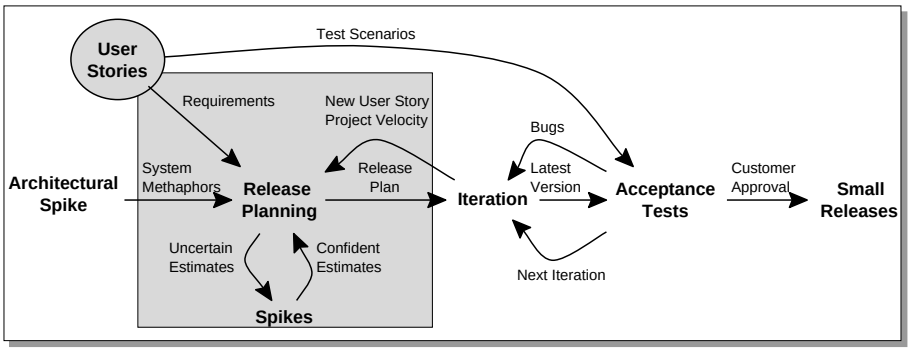

Abbildung 2.6.: *Spike Solutions bei eXtreme Programming ([Bec00])*

Unter Spike Solutions versteht Beck sehr einfache Programme, um potentielle Problemfelder schnell und direkt zu bestimmen. Die Idee liegt darin, schon während der Projektvorbereitungen die Kundenwünsche durch Beispiele zu verifizieren und gleichzeitig das Risiko zu minimieren. Die mit Hilfe von grob formulierten Anforderungen zu entwickelnden „Spikes“, werden dem Kunden vorgestellt und dabei gleichzeitig ungewisse Vermutungen (engl: uncertain estimates) durch begründete Annahmen (engl.: confident estimates) ersetzt. Missfällt dem Kunden die dabei entstandene Lösung, werden die Einwände direkt berücksichtigt. Diese Idee entspricht der direkten Verknüpfung von Analyse und Design und widerspricht der Grundüberlegung, zunächst ein oder mehrere Anforderungsdokument(e) zu spezifizieren und anschließend den technischen Entwurf vorzunehmen. Zumindest theoretisch ist dieser Sachverhalt in XP anders organisiert. Beck betont ausdrücklich die zentrale Stellung der Programmierung und formuliert daher die bereits erwähnten XP-Techniken.[70] Diese stehen im Vorgehensmodell in einem engen Beziehungsgeflecht zueinander, bilden jedoch kein eigenes Phasenmodell ab. Daher werden XP-Projekte doch ganz herkömmlich iterativ und inkrementell (stückweise) durchgeführt.[71] Der wesentliche Unterschied für die Anforderungsanalyse ergibt sich aus dem Beitrag, der folgenden kurz zu beschreibenden XP-Techniken, die sich über den gesamten Projektverlauf erstrecken:

- **Kunde vor Ort.** XP-Projekte sind stets explizit an die fachlichen Vorgaben der Kunden gebunden. Wann immer möglich wird der Auftraggeber angehalten, direkt und vor Ort dem Entwickler seine Anforderungen verständlich zu machen, um damit auf Fachkonzepte oder formale Spezifikation sogar gänzlich verzichten zu können.

- **Planungsspiel.** Wie schon aus der Abbildung 2.6 ersichtlich, stehen in XP zügig auszuliefernde inkrementelle Programmversionen im Vordergrund. Den Umfang des jeweils nächsten Inkrements vereinbaren Entwickler und Kunde

[70] Diese Tatsache wird auch schon in der Namensgebung des Ansatzes angedeutet. Für die Bedeutung „eXtreme“ gibt Beck eine ausführliche Darstellung in [Bec00] S.XV
[71] Vgl. [LRW02] S.13

zusammen unter Beachtung von Abhängigkeiten, Prioritäten und dem zu erwartenden Aufwand.

- **Metapher.** Die Kernideen des Systems werden durch so genannte Metaphern beschrieben, die auf möglichst einfache Art und Weise dokumentieren, wie die grundsätzlichen funktionalen Zusammenhänge aussehen. Im engeren Sinne handelt es sich um „metapherartig"-beschriebene Anleitungen, mit denen sowohl der Kunde, als auch die beteiligten Entwickler ihre Vorstellungen besser ausdrücken können.

- **Kurze Releasezyklen.** Wie bereits erwähnt, wird durch sehr kurze Abstände zwischen einzelnen Versionen versucht, die Umsetzungen der Änderungswünsche direkt als Feedback in die Entwicklung wieder einzutakten. Davon profitiert zum einen der Kunde, indem er deutlich schneller seine Anforderungen im Programm vorfindet, zum anderen wird für die Entwickler das Risiko gemindert, eine potentiell falsch verstandene Anforderung mit unnötig hohem Aufwand weiter zu implementieren.

Das XP-Rollenmodell ist im Vergleich zum RUP oder dem V-Modell deutlich einfacher gehalten und konzentriert sich im Wesentlichen auf die drei wichtigsten Beteiligten im Softwareprozess: Kunde, Programmierer und Tester. Das Aufgabenfeld dieser Rollen ist selbsterklärend.

Darüber hinaus führt Beck die Rollen[72] Verfolger (eng.: Tracker), XP-Trainer (engl.: XP Coach), Berater (engl.: Consultant) und Big Boss ein, die aber bzgl. der Anforderungsanalyse kaum einen Beitrag leisten. An dieser Stelle sei erwähnt, dass in XP keine eigene Phase für die Anforderungsanalyse vorgesehen ist. Während eine Phase sich vor allem durch einen definierten Anfangs- und Endzeitpunkt abgrenzt und im Resultat durch dessen Ergebnisse bewertet wird, versteht man in XP die Anforderungsanalyse als eine mehrfach durchzuführende Aktivität.[73]

Anforderungsermittlung

Die Anforderungsermittlung wird im Referenzwerk von Beck hauptsächlich durch das „Planungsspiel" thematisiert. Dabei werden die Spielregeln recht ausführlich dargelegt und Beispiele gegeben. Wie bereits erwähnt, handelt es sich dabei um eine geregelte Sitzung, bei der Kunde und Entwickler die Anforderungen für die nächste Auslieferung mit Hilfe von Story-Cards formulieren. Diese dienen der Erhebung, Protokollierung, Herleitung und Diskussion von Anforderungen. An dieser Stelle soll jedoch auf die Originalquelle verwiesen werden, die eine genaue Anleitung zum Planungsspiel enthält.[74]

[72] Vgl. [Bec00] S.139
[73] Vgl. [LRW02] S.152
[74] Vgl. [Bec00] S.86ff.

Am Planungsspiel lässt sich bereits erkennen, dass bei XP der aktive Bezug zum zukünftigen Anwender im Mittelpunkt steht. Im Wesentlichen basiert die Idee darauf, die Kommunikation so in den Vordergrund zu stellen, dass eine offene und direkte Konversation „totgeschwiegene Probleme“ weitestgehend verhindert. Daher empfiehlt Lippert auch den Einsatz von offenen Interviews in Verbindung mit Arbeitsplatzbeoachtungen. Der zukünftige Anwender wird dazu angehalten, möglichst ungezwungen seine Problemstellung zu artikulieren. Dabei wird er in den meisten Fällen die Probleme ansprechen, die ihm am dringlichsten erscheinen. Natürlich erfordert diese Technik ein hohes Maß an Menschenkenntnis und findet schnell seine Grenzen bei unerfahrenen Mitarbeitern. Die Praxis hat aber gezeigt, dass bei einem hohen Vertrauensgrad zwischen Entwickler und Anwender, die Anforderungen so vermittelt werden können, dass auf beiden Seiten ein Vorteil entsteht.

Letztendlich sind bei der Ermittlung von Anforderungen die besonderen Rahmenbedingungen der agilen Softwareentwicklung mit einzubeziehen. Gerade deswegen fehlt es dem XP-Ansatz an einer workflowartigen Ablaufbeschreibung, wie es zum Beispiel im RUP existiert. Die Anforderungsermittlung mit XP ist ein intuitiver Prozess, der ganz bewusst das menschliche (Un-)Vermögen einbezieht.

Anforderungsanalyse

Für die Analyse der ermittelten Anforderungen sind keine gesonderten Ausführungen zu finden. Obwohl die Analyse fester Bestandteil bei der Diskussion mit Story-Cards ist, liegt keine eigene und in sich geschlossene Aktivitätsbeschreibung vor.

Anforderungsverständigung

Der gemeinsame Lernprozess (Teamgedanke) hat in der XP-Philosophie einen hohen Stellenwert. Die Entwickler, Auftraggeber und Anwender[75] werden dazu aufgefordert, sich gegenseitig mit der Fachlichkeit der beteiligten Personen vertraut zu machen. Deutlich wird dies, bei der Äußerung von Änderungswünschen. In XP wird davon ausgegangen, dass die vom Kunden kommenden Änderungswünsche auch in späteren Durchläufen verwaltbar sind. Diese Vorgehensweise ist vertretbar, solange es sich nicht um gravierende Architekturänderungen handelt und die Auswirkungen mit dem Kunden diskutiert werden können.[76]

Die XP-Verfechter gehen auch davon aus, dass moderne Werkzeuge[77] inzwischen soweit entwickelt sind, dass Änderungen, auch in späten Phasen, keine wesentlich höheren Kosten verursachen. Dies kann natürlich nur funktionieren, wenn sowohl

[75] Ein Kritikpunkt zum XP ist die Tatsache, dass Beck keine Rollenunterscheidung zwischen Auftraggeber und Anwender vornimmt. Dies ist jedoch in der betrieblichen Praxis durchaus üblich. Daher sollte diese Tatsache in IT-Projekten zur Kenntnis genommen und entsprechend berücksichtigt werden.

[76] Vgl. [Bec00] S.21ff.

[77] Hier sind z.B. Refactoring-Werkzeuge zu nennen, die es ermöglichen sollen, große Teile von Programmcode auch bei größeren Änderungsabsichten konsistent zu ändern.

Kunde als auch Entwickler sich über die Auswirkungen (Zeit und Kosten) im Klaren sind.

Anforderungsdokumentation

Bis auf die erwähnten Story-Cards finden sich in XP keine Vorlagen für Niederschriften wie Protokolle, Szenarios oder Glossare.

Anforderungsqualitätssicherung

Beck definiert Zeit, Kosten, Umfang und Qualität als die vier wichtigsten Kontrollvariablen in Softwareprojekten. Dies allein unterscheidet sich nicht von anderen Ansätzen. Der indirekt Qualitätsbegriff von XP ist einzigartig. Die relevanten XP-Techniken (Kunde vor Ort, Planungsspiel, Metapher, kurze Releasezyklen, etc.) leisten vor allem dadurch einen qualitätssichernden Beitrag, dass sie den Auftraggeber/Anwender und seine Wünsche (Anforderungen) direkt in den Entwicklungsprozess einbeziehen. Das Kunden-Feedback ist damit das wichtigste Steuerinstrument für die (Anforderungs-)Qualität in XP-Projekten.

2.3.4. SWEBOK

Ansatz

Wie bereits vorweggenommen, handelt es sich bei Software Engineering Body of Knowledge (SWEBOK) nicht um ein Vorgehensmodell im Sinne der aus Abschnitt 2.2 eingeführten Definition. Warum soll es dennoch hier betrachtet werden?

Das erklärte Ziel der Autoren von SWEBOK ist es, die Softwaretechnik als ingenieurmäßige Disziplin weiter zu professionalisieren. Im Unterschied zu bekannten anderen umfangreichen Standardwerken (z.B. Balzert, Sommerville, McConnel,...), versucht SWEBOK dies nicht durch eine lose Sammlung von „softwareintensiven“ Themen zu erreichen. Es wird versucht, die bekannten Teilbereiche der Softwaretechnik durch so genannte Wissensbereiche (engl.: knowledge areas) zu gliedern. Dabei handelt es sich nicht um eine simple Neuanordnung der Themen, sondern um den Versuch, ein einheitliches und umfassendes Gesamtwerk zur Softwaretechnik zu erarbeiten, das mit Hilfe zahlreicher Referenzen als Ausgangspunkt für vertiefende Betrachtungen dienen kann. Nüchtern betrachtet entspricht dies dem typischen Vorgehen zur Ausarbeitung neuer Ansätze und soll daher im Rahmen dieser Arbeit als Argumentationsgrundlage für die ab Kapitel 3 angestrebten eigenen Ansätze dienen. Die Idee von SWEBOK passt damit unmittelbar zu der Zielstellung dieser Arbeit und soll aus diesem Grunde mit in die Überlegungen einbezogen werden.[78]

[78] Die folgenden Auszüge beziehen sich auf Diskussionsbeiträge der internationalen Usenet-Newsgroup `news://comp.software-eng` und der Community für Softwaretests und Qualitätssicherung, erreichbar unter `http://www.qaforums.com`, bei denen der Autor dieser Arbeit

In dem knapp 300 Seiten umfassenden Dokument wird viel „kosmisches Wissen“[79] einbezogen, das sonst nur in Form von Büchern, Lektüren, Anleitungen oder Aufsätzen ungenutzt bleibt. Die konzeptionelle Ausrichtung in Wissensbereiche (engl: knowledge areas) beschreibt den Charakter des Dokumentes. Nach Meinung der Autoren handelt es sich um ein Handbuch zur Benutzung dieser Wissensbereiche (engl.: the guide is guide to that knowledge). In der vorliegenden Trial Version 1.00 wird auf insgesamt 10 Themenkomplexe referenziert, die ihrerseits eine Synthese von generell akzeptierten Erkenntnissen darstellen und wann immer sinnvoll, auf vertiefende Ausführungen entsprechend verweisen.

Die Autoren von SWEBOK verfolgen den Ansatz, jeden Wissensbereich der Softwaretechnik zu untersuchen und die Kernaspekte (gewissermaßen das Fundament der Themenkomplexe) entsprechend auszuarbeiten (engl.: knowledge area breakdown). Um dies zu erreichen ist es nötig, eine möglichst breite Auswahl der bisher verfügbaren und anerkannten Ansätze aus Literatur und Wissenschaft einzubeziehen und diese so zu kombinieren, dass am Ende tatsächlich die Relevanz bestimmter Teilaufgaben einwandfrei feststeht. Diese Aussage sollte jedoch mit ausreichend kritischem Abstand betrachtet werden. Häufig fällt auch im Zusammenhang mit SWEBOK der Begriff „Framework“ (dt.: Rahmenwerk). Dazu ist anzumerken, dass dieser Begriff in der Literatur eine sehr vielschichtige Definition erfährt und je nach Kontext variiert. Hier soll es jedoch genügen, SWEBOK als Rahmenwerk für einen Gesamtüberblick über alle Themen der Softwaretechnik zu verstehen. Im Zuge der hier angestrebten Betrachtung werden die folgenden Ausarbeitungen jedoch auf den Punkt „Software Requirements“ beschränkt sein.

In SWEBOK wird das Requirements Engineering in Anlehnung an ISO/IEC 12207-1995 in folgende sechs Teilgebiete gegliedert:[80]

- Requirements Engineering Process
- Requirements Elicitation (dt.: Anforderungserhebung)
- Requirements Analysis (dt.: Anforderungsanalyse)
- Requirements Specification (dt.: Anforderungsspezifikation)
- Requirements Validation (dt.: Anforderungsvalidierung)
- Requirements Management

mitwirkte. Des Weiteren werden hier auch Meinungen zitiert von offiziellen Kritikern (engl: reviewer) des SWEBOK-Projektes, mit denen der Autor zum Teil auch in Kontakt stand.

[79] Vgl. [Pau98] S.81. Pautzke bezieht sich in seiner Theorie zum organisatorischen Lernen auf das Wissen, das dem Einzelnen/der Organisation nicht zugänglich ist und bezeichnet dieses als umgebendes „kosmisches“ Wissen.

[80] Vgl. [CS01] Chapter 2, S.15

Im Unterschied zu der besagten ISO/IEC-Norm haben die Autoren die Punkte „Requirements Validation“ und die „Requirements Management“ als separate Teilgebiete ergänzt. Wie schon aus der Namensgebung zum Teil erkennbar, decken sich die Aktivitäten größtenteils mit den in dieser Arbeit zur Hilfe genommenen Hauptaufgaben des Anforderungsmanagements nach SCHIENMANN, der seinerseits als deutscher Vertreter in SWEBOK nicht berücksichtigt wurde. Eine vollständige Übersicht inkl. weiterer Untergliederung ist im Anhang A.3 zu finden.

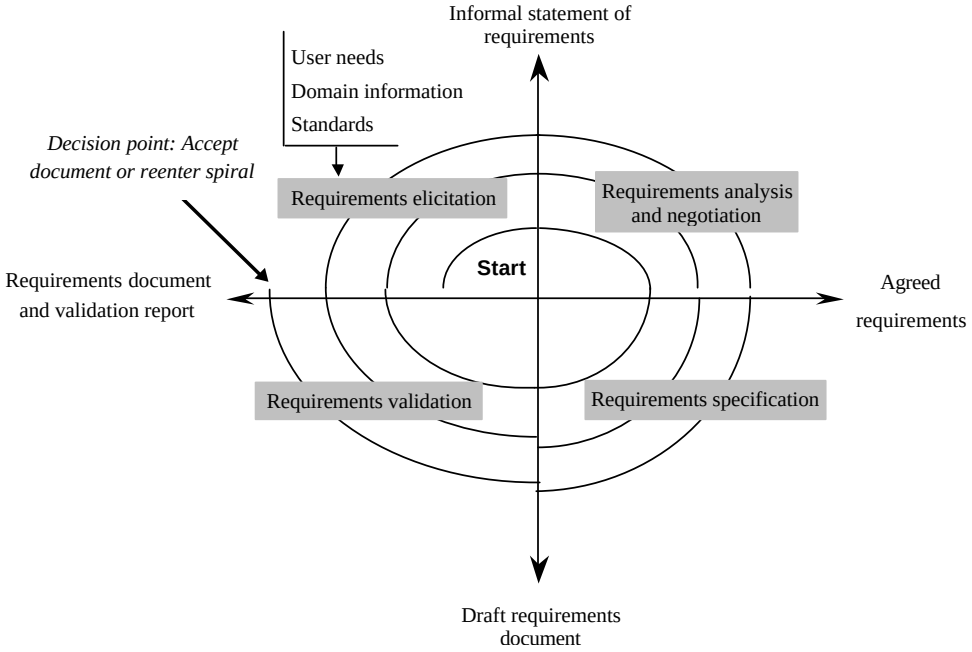

Abbildung 2.7.: *Spiralmodell des Requirements Engineering Process ([CS01])*

Trotz Gliederung in sechs Teilgebiete wird in SWEBOK explizit darauf hingewiesen, dass im Requirements Engineering *kein* wasserfallartiger Ansatz vorliegt. Um diese Tatsache deutlich zu unterstreichen, wird als erstes Teilgebiet der Requirements Engineering Process (REP) thematisiert. In Abbildung 2.7 wird ein Spiralmodell für einen iterativen Prozess dargestellt. Darin begründet sich der Grundgedanke des REP, wonach die Teilaktivitäten *Erhebung, Analyse, Spezifikation* und *Validierung* solange wiederholt werden, bis entweder das Anforderungsdokument fertiggestellt ist oder aufgrund externer Faktoren (Kundendruck, Ressourcen, Zeit, etc.) ein Abbruch erforderlich wird.[81] Die in der Abbildung benannte Abszisse bzw. Ordinate symbolisiert den „messbaren Reifegrad“ der Ergebnisse nach dem Abschluss einer Teilaktivität. Als theoretisches Modell ist diese Darstellung ein Erklärungsansatz, für die komplexen Zusammenhänge, die im REP unweigerlich entstehen.

Die im Folgenden untersuchten Teilgebiete basieren auf den Vorarbeiten bekannter Autoren. Für den Nachweis, bedient sich SWEBOK einer so genannten Refe-

[81] Vgl. [CS01] Chapter 2, S.16

renzenmatrix (engl.: Matrix of Topics). Dabei handelt es sich um eine tabellarische Gegenüberstellung RE-bezogener Inhalte und deren Vertreter.[82]

Anforderungsermittlung (Requirements Elicitation)

Zunächst ist festzustellen, dass SWEBOK kaum methodische Ansätze bietet und demzufolge auch kein eigenes Prozessmodell definiert, wie es z.B. im RUP der Fall ist. Dies äußerst sich dahingehend, dass lediglich konzeptionelle und gedankliche Anmerkungen vorzufinden sind. Im Zusammenhang mit der Anforderungserhebung[83] führt SWEBOK im REP den Anforderungsingenieur (engl.: requirements engineer) als den Hauptprotagonisten ein. Dieser hat die Aufgabe, die Anforderungen der Beteiligten zu erkennen und zu erfassen. Prinzipiell unterscheidet sich sein Aufgabengebiet nur minimal zu den Ansätzen der bisher vorgestellten Vorgehensmodelle. Dennoch finden sich einige konzeptionelle Ergänzungen, die hier kurz vorgestellt werden sollen:[84]

- **Anforderungsquellen (engl.: requirements sources).** Bei der Entwicklung eines System sind, abhängig von der Größe des Projektes, mehrere potentiell wichtige Anforderungsquellen (Ansprechpartner, Organisationseinheiten, Projektgruppe, ...) zu berücksichtigen. Dies kann sich jedoch als unerwartet schwierig erweisen, da die Quellen zunächst nicht bekannt sind. Im Rahmen einer ersten Machbarkeitsstudie, werden die höheren Projektziele definiert und aus dieser Überlegung heraus die betroffenen Quellen isoliert.

- **Erhebungstechniken (engl.: elicitation techniques).**

  Nach Bestimmung der Quellen kann der Anforderungsingenieur damit beginnen, die jeweils relevanten Anforderungen zu erheben. Im Zuge dieser Aufgabe nennt SWEBOK überblicksartig auch die dafür wichtigsten Erhebungstechniken wie Interviews, Szenarien, Prototypen, Meetings, etc. Das Dokument verweist zwar auf die Vertreter, gibt jedoch keine Vorschläge über die Einordnung und Nützlichkeit dieser Techniken. Dieses Defizit soll in dieser Arbeit mit einer verbesserten Darstellung in Abschnitt 2.4 ab S. 44 ausgeglichen werden.

Die Anforderungserhebung kann sich als schwieriger Prozess herausstellen, da in diesem Zusammenhang viel von der Kooperation der beteiligten Parteien abhängt. Häufig kommt es zu Schwierigkeiten, die typischerweise auf die Unfähigkeit oder sogar dem Unwillen der Beteiligten zurückzuführen sind.[85] Daher lautet eine wichtige

[82] Vgl. [CS01] Chapter 2, S.24ff. Einbezogen werden Standardwerke von Vertretern wie z.B. Byrne, Davis, Pfleeger, Sommerville oder Rosenberger. SWEBOK gibt zu allen Referenzen einen Überblick und eine Begründung für die Auswahl.

[83] Zugunsten der Konsistenz mit der Definition aus der Einleitung und der wörtlichen Übersetzung von „elicitation“ wird hier der Begriff „Erhebung“ statt „Ermittlung“ verwendet.

[84] Vgl. [CS01] Chapter 2, S.17,18

[85] Sicher könnte man sich fragen, wieso der Unwille des Auftraggebers eine Rolle spielt, wo er doch an der Fertigstellung ein Interesse hat. Nimmt man eine Unterscheidung zwischen Auftrageber

Empfehlung in SWEBOK, die Erhebung nicht als passiven Prozess zu verstehen, sondern diesen aktiv zu steuern. Der Anforderungsingenieur steht daher vor der herausfordernden Aufgabe, die Beteiligten „aus der Reserve zu locken“. Wie dies allerdings aussehen sollte, lässt SWEBOK offen.

Anforderungsanalyse (Requirements Analysis)

Die in SWEBOK beschriebenen Ausführungen zur Anforderungsanalyse decken sich weitgehend mit den bisher vorgestellten Ansätzen. Aufgrund der streng sachlichen Zerlegung der Thematik, ergibt sich dennoch eine gewinnbringende Ausweitung der theoretischen Teilaufgaben. Wie bereits bekannt, besteht das grundsätzliche Anliegen darin, Zusammenhänge und Konflikte zu erkennen und entsprechend zu lösen (Systemgrenzen definieren, Isolation von konträren Anforderungen, etc.). Ganz ähnlich dem V-Modell wird zwischen Systemanforderungen und Softwareanforderungen unterschieden. Dies hängt damit zusammen, dass in frühen Phasen zunächst von „High Level“-Anforderungen die Rede ist, während im späteren Verlauf und mit wachsenden Detaillierungsgrad die Anforderungen spürbar einfacher klassifizierbar werden.

Klassifikation von Anforderungen

Zur Klassifikation sollen Anforderungsklassen definiert werden, die ihrerseits mit spezifischen Anforderungsattributen ausgestattet sind. Wie üblich werden dafür Dimensionen formuliert, anhand derer man die ermittelten Anforderungen analysieren kann. Die nach SWEBOK wichtigsten Unterscheidungskriterien sind nachfolgend tabellarisch aufgelistet und werden kurz erklärt:

	Beschreibung
Funktionale / Nicht-Funktionale Anforderung	Unterscheidung zwischen zu realisierenden Fähigkeiten und einzubeziehenden Beschränkungen.
High Level Requirement	Abstrakte fachorientierte Anforderungen, ohne direkte technische Zuordnung.
Emergent Properties	Anforderungen, die nicht einer bestimmten Komponente zu zuordnen sind, aber wesentlichen Einfluss auf die Systemarchitektur haben.
Produkt- / Prozessanforderung	Produktanforderungen richten sich an den zu realisierenden Gegenstand, Prozessanforderungen hingegen an die Entwicklungsarbeiten.

und zukünftige Benutzer vor, resultiert daraus zuweilen ein Interessenkonflikt zwischen Vorgesetzten (Auftraggeber) und Untergebenen (Anwender). Bei einer unternehmensinternen Eskalation kann dieser Umstand zu unkooperativen Widerstand (sog. retardierende Kräfte (Lewin)) führen.

Anforderungspriorität	Klassifikation anhand einer festen Skala (z.B.: obligatorisch, wünschenswert, optional).
Umfang der Anforderung (Scope)	Klassifikation nach dem Einfluss der Anforderungen auf die Systemarchitektur (Global Scope) oder der begrenzten Ausrichtung auf eine eindeutige Komponente (Narrow Scope).
sonstige Klassifikationen	Hier empfiehlt SWEBOK die Ausweitung auf individuelle organisatorische Bedürfnisse.

Tabelle 2.1.: *Klassifikationsansätze für Anforderungen ([CS01])*

Anforderungsattribute sind mit allen genannten Typen zu verknüpfen. Inhaltlich handelt es sich um Metadaten (z.B. Autor, Historie, Status, ...), die der effizienten Verwaltung und Weiterbearbeitung der ermittelten Anforderungen dienen.

Konzeptmodellierung

Die Zuordnung der betrachteten Geschäftsprozesse und deren Übertrag auf konkrete Anforderungen soll durch die Konzeptmodellierung[86] erreicht bzw. verbessert werden. Dabei treffen die Autoren eine klare Aussage: SWEBOK beabsichtigt *nicht*, eine oder mehrere Methode(n) zu beschreiben und deren richtige Anwendung zu propagieren, sondern stellt Gegenstand und Absicht der Konzeptmodellierung im Zusammenhang mit dem Requirements Engineering heraus. Unter werkzeug- und modellunabhängiger Argumentation trifft SWEBOK zwei wesentliche Aussagen:

- Die Notation ist so zu wählen, dass die jeweilige Domäne am besten berücksichtigt wird (Echtzeitsystem, Workflowgestaltung, Stapelverarbeitung, etc.).
- Die verwendete Methode sollte in Absprache mit dem Kunden gewählt werden.

Architekturdesign und Anforderungszuordnung

An einem bestimmten Punkt wird sich jeder Anforderungsingenieur auf eine Systemarchitektur festlegen müssen. Dabei überschneidet sich das Aufgabengebiet zwischen Analytiker und Designer. Mit Bezug auf Sommerville betont SWEBOK eindringlich, dass es unmöglich ist, Analyse und Design vollständig zu entkoppeln. In manchen Fällen muss der Anforderungsingenieur als Systemarchitekt arbeiten, da er auch verantwortlich ist, die Komponenten und Subsysteme zu erkennen und den Anforderungsprozess entsprechend auszurichten. Mit engen Bezug an die konzeptionelle Modellierung können die ermittelten Anforderungen besser den Komponenten des Systems zugeordnet werden.

[86] Der Begriff Konzeptmodellierung wird auch im Rahmen der Metamodellierung verwendet und meint die Entwicklung von Konzepten für Modelle (Vererbung, Beziehungen, etc.). An dieser Stelle wird jedoch die konzeptionelle Modellierung von Sachverhalten beschrieben.

Anforderungsverständigung (SWEBOK: Requirements Negotiation)

Das Dokument greift erstaunlich konkret die Problematik auf, was passiert, wenn zwei oder mehrere Parteien (Stakeholder) konträre Anforderungen formulieren und diese im Rahmen der Analyse entdeckt werden.[87] Problematisch wird die Angelegenheit beispielsweise wenn:

- Anforderungen zueinander inkompatibel sind,
- Anforderungen aufkommen, die nach Ansicht einer anderen Partei falsch sind,
- individuelle Ziele nicht vereinbar sind mit dem Gesamtziel des Projektes,
- Anforderungen auf unvollständige oder inkonsistente Aussagen basieren.

Die Anforderungsverhandlung ist ein Abstimmungsprozess, bei dem die für den Projekterfolg kritischen Anforderungen isoliert und in enger Zusammenarbeit mit dem Kunden die nötigen Absprachen durchgeführt werden.

Anforderungsdokumentation (SWEBOK: Requirements Specification)

Bezüglich der Anforderungsdokumentation bringt SWEBOK nur die typische Unterscheidung zwischen Lastenheft (engl: system requirements definition) und dem Pflichtenheft (software requirements definition) mit. Für die inhaltliche Konzeption wird auf mehr oder weniger bekannte Strukturrichtlinien nach IEEE- und ISO-Dokumenten verwiesen, die im deutschen Raum jedoch nur eine geringe Bedeutung haben.

Anforderungsqualitätssicherung (SWEBOK: Requirements Validation)

Die Sicherung der Qualität wird im Sprachgebrauch von SWEBOK als Anforderungsvalidierung bezeichnet. Dabei setzen die Autoren den Schwerpunkt auf folgende Teilbereiche:[88]

- **Anforderungsreview.** Die Durchsicht (engl.: review) der Anforderungen ist das wichtigste Mittel, das im Rahmen der Qualitätssicherung eingesetzt wird. Eine Gruppe wird beauftragt Inspektionen vorzunehmen, um die produzierten Artefakte auf Fehler zu untersuchen, missverstandene Annahmen zu identifizieren und unklare Formulierungen zu verbessern. Entscheidend ist dabei die Zusammensetzung der Gruppe, bei der unbedingt die Auftraggeber berücksichtigt werden sollten.

  Obwohl qualitative Messungen von Anforderungen oft nur unscharfe Werte liefern, empfiehlt SWEBOK die Kontrolle hinsichtlich Umfang, Tiefgang, Lesbarkeit und Strukturierung der Artefakte. Insbesondere beim Einsatz von Richtlinien lassen sich leicht Qualitätskriterien aufstellen.

[87] Vgl. [CS01] Chapter 2, S.20
[88] Vgl. [CS01] Chapter 2, S.22

- **Prototyping & Akzeptanztests.** Hier werden vor allem die aus XP bekannten Ansätze genannt und auf die besondere Bedeutung des Feedbacks verwiesen.

- **Modellvalidierung.** Modelle können durch Integritätsregeln formal auf fehlende Verbindungen, unterbrochene Kommunikationspfade oder Regelverletzungen untersucht werden.

2.4. Techniken und Gestaltungsmittel im Requirements Engineering

Schon in der Einleitung wurde erwähnt, dass professionelle Anforderungsingenieure auf eine Vielzahl von Techniken zurückgreifen. Dies geschieht z.T. unbewusst (z.B. durch Steuerung in einem Kundengespräch) und manchmal explizit, weil bestimmte Erfahrungswerte vorliegen (z.B. Fragebogen statt Interviewreihe) oder Vorgehensmodelle entsprechende Hinweise liefern (z.B. Methodenzuordnungstabellen im V-Modell). In der Literatur gibt es zum Thema „Techniken im Requirements Engineering“ eine Vielzahl von Beiträgen, die im Wesentlichen eine Bewertung zum Ziel haben oder den Vergleich anstreben. Wann immer eine Betrachtung zum RE vorliegt, erscheint es jedoch sinnvoll dem Leser einen groben Überblick zu den besagten Techniken und Gestaltungsmittel anzubieten, damit dieser im Zweifel auf die zusammengestellten Referenzen zurückgreifen kann. Im Folgenden soll eine knappe Klassifikation und Einordnung der bekanntesten Ansätze vorgestellt werden.

Hinweis: Bei den Betrachtungen über Vorgehensmodelle wurde eine Erweiterung des Begriffs „Anforderungsanalyse“ eingeführt. Die Vereinbarung bezog sich darauf, die gesammelten Aktivitäten in einem Softwareprojekt gegenüber dem Begriff „Requirements Engineering“ abzugrenzen und der nach ihr benannten Phase im Softwarelebenszyklus gerecht zu werden (Requirements Analysis). Dies hat sich im Zusammenhang mit Vorgehensmodellen als praktisch erwiesen, ist aber für die nun folgenden Betrachtungen nicht mehr notwendig. Deswegen wird die Anforderungsanalyse als Teilgebiet wieder dem Requirements Engineering untergeordnet.

In diesem Abschnitt wird unter einer Technik ein erprobtes Hilfsmittel verstanden, das den Anforderungsingenieur bei der Durchführung seiner Aufgaben unterstützen soll.[89] Der Spezialist greift zu diesem Zweck auf möglichst weit verbreitete Gestaltungsmittel zurück, die ihm helfen sollen, die Ergebnisse entsprechend zu dokumentieren und mit dem Kunden zu diskutieren. Charakteristisch ist der starke Einfluss der Technik auf die Effizienz der analytischen Arbeiten. Dies liegt vor allem daran, dass der RE-Prozess erheblich vom Risikofaktor „Mensch“ abhängt und die Techniken diese Tatsache (mehr oder weniger) berücksichtigen.

[89] Vgl. [Sch02] S.187

Eingeschränkt wird die Betrachtung durch den ausschließlichen Bezug auf Techniken, die für die Anforderungsentwicklung individueller Softwarelösungen nützlich sind.[90] Die Klassifizierung erfolgt nach den Hauptaufgaben des Requirements Engineering und ist eine Kombination der Ausführungen von Rupp und Schienmann.[91]

Anforderungsermittlung

Die wohl umfangreichste Sammlung von Techniken ist der Anforderungsermittlung zu zuordnen. Die in Tabelle 2.2 eingruppierten Techniken geben einen groben Überblick der Ansätze, die sich in der Praxis bereits etablieren konnten.[92]

Gruppen- und Kreativitätstechniken

Brainstorming	Ideensammlung und Aufdeckung unterbewusster/impliziter Anforderungen durch Gruppen.
Mind Mapping	Baumartige Verknüpfung von Begriffen zur Darstellung von Assoziationen und Erarbeitung einer Wissenslandkarte.
Anforderungs-workshops	Maximal zweitägiges Zusammentreffen der Beteiligten, um grundlegende Anforderungen in abstrakter Form für die Weiterbearbeitung zu dokumentieren.
Snowcards	Gruppenbasierte Ermittlung von Anforderungen durch sukzessive Ergänzung im Team.

Beobachtungstechniken und Tätigkeitsanalysen

Feldbeobachtung	Arbeitsabläufe werden direkt oder indirekt von unabhängigen Mitarbeitern verfolgt und analysiert.
Apprenticing (dt.: Ausbilden)	Der Analytiker erlernt einen Sachverhalt durch praktische Tätigkeiten und erhält dadurch ein erlebtes Bild über die Vorgänge.

Dokumentenanalyse

Literaturstudium	Anforderungen werden abgeleitet durch die intensive Recherche von Fachliteratur.
Protokollanalyse	Ermittlung der Anforderungen aus Mitschriften und Veranstaltungsprotokollen.

Befragungstechniken

Fragebogen	Adressierung großer Gruppen durch Standardfragen.

[90] Es entfallen Ansätze wie z.B. Quality Function Deployment (Produktentwicklung) oder Feature Modelling (Systemfamilien).

[91] Vgl. [R+02] S.109ff., [Sch02] S.195ff.

[92] Die genannten Techniken sind z.T. nicht spezifisch für das Requirements Engineering. Häufig handelt es sich um bewährte Methoden, die in vielen unterschiedlichen Bereichen zur Problemlösung eingesetzt werden können.

Interview [GW93]	Anforderungen werden mündlich erarbeitet und Probleme direkt im Gespräch geklärt.
Selbstaufschreibung	Eine Beschreibung wird durch denjenigen erstellt, dessen Arbeitsablauf die Vorlage bildet.
On-Site-Customer (dt.: Kunde-vor-Ort)	Der Kunde stellt einen Berater ab, der vor Ort bei den Entwicklern Fragen beantworten kann oder mitarbeitet.

Vergangenheitsorientierte Techniken

Systemarchäologie	Ableitung von Anforderungen mittels Untersuchung eines abzulösenden Systems.
Reuse und Muster	Anforderungen aus ähnlichen Projekten ableiten und Wiederverwendung von Anforderungen.

Tabelle 2.2.: *Techniken zur Anforderungsermittlung ([R+02],[Sch02])*

Anforderungsanalyse

Die Techniken zur Anforderungsanalyse zielen stark darauf ab, die inhaltliche Konsistenz zu erreichen und den Umgang mit den ermittelten Anforderungen für die späteren Entwurfs- und Realisierungsarbeiten zu erleichtern. Inhaltlich lassen sich zwei Gruppen identifizieren: Strukturierungstechniken und Techniken zur Bedeutungsanalyse.

Strukturierungstechniken

Anwendungsfälle	UML-Gestaltungsmittel, das zur Beschreibung von fachlichen Vorgängen dient und in seiner speziellen grafischen Darstellungsform zur Analyse herangezogen werden kann (siehe auch Abschnitt 2.5.1).
CRC-Karten [Sch02, BS95]	Strukturierung des Problemraumes durch eine Menge von Anwendungsfällen mit spielerischer Erhebung und Vervollständigung von Anforderungen.
Problem Frames	Technik zur Strukturierung und Spezifikation von Kundenproblemen in Teilprobleme, ohne dabei die spätere technische Umsetzung zu berücksichtigen.
Entscheidungstabellen [Str77, Par91]	Bei komplexen Entscheidungsfolgen kann man mit Hilfe einer tabellarischen Übersicht mehrere abhängige Bedingungen kompakt definieren.

Techniken zur Bedeutungsanalyse

Prüflisten und Glossare	Fachbegriffe werden in speziellen Datenbanken gesammelt, definiert und aufbereitet.
Domänenanalyse	Besonderheiten der betrachteten Domäne werden eingehend untersucht und in Verbindung mit den Anforderungen überprüft.

sprachliche Bedeutungsanalyse	Sammlung, Klärung, Definition und Standardisierung von natürlichsprachlichen Anforderungen. Die Aussagensammlung wird durch ein teilformales Schema auf Sprachdefekte hin untersucht und normiert.

Tabelle 2.3.: *Techniken zur Anforderungsanalyse ([R+02],[Sch02])*

Anforderungsverständigung

Im Vordergrund stehen Konflikte und deren Bewältigung durch die Kooperation der streitenden Parteien. Häufig können die Probleme schnell gelöst werden, indem sie zur Sprache gebracht und in Gruppen diskutiert werden.

Konfliktbewältigung

Mediation (dt.: Vermittlung)	Eine spezielle Form der Konfliktregelung, bei der die Konfliktparteien mit Unterstützung von qualifizierten Vermittlern (Mediatoren) eine Regelung erarbeiten.
Workshops mit Moderation	Zielgerichtete und moderierte Konfliktbewältigung im Rahmen von z.T. mehrtägigen Workshops.
Priorisierung	Anforderungen werden nach ihrer Wichtigkeit sortiert und Konflikte im Zuge der Neubewertung gelöst.

Tabelle 2.4.: *Techniken zur Anforderungsverständigung ([R+02],[Sch02])*

Anforderungsdokumentation

Zur Dokumentation gibt es eine Vielzahl von Ansätzen mit sehr unterschiedlichen Schwerpunkten. Typischerweise werden Anforderungen durch formale/teilformale Gestaltungsmittel wie z.B. die UML, Ereignisprozessketten (EPK), Datenflussdiagramme (DFD), Entity-Relationship Modelle (ERM), Struktogramme, Entscheidungstabellen u.ä. dokumentiert. Die Wahl der Darstellungsform ist abhängig von der Domäne und wird je nach Bedarf auch kombiniert. Ein typisches, teilformales Mittel ist beispielsweise der klassische Anforderungskatalog. Häufig beinhaltet dieser, eine nach fachlichen Kriterien strukturierte Auflistung von textuellen Anforderungen, die in einfachen Textdokumenten oder Tabellen zusammengefasst sind.

Da im Rahmen dieser Arbeit eine vollständige Übersicht nicht nötig ist, wird an dieser Stelle auf die zahlreichen Standardwerke zur Modellierung betrieblicher Softwarelösungen verwiesen.[93]

[93] Siehe [Sch97], [Par91], [Bal01], [Coc01], [Oes98], [Sch01] um nur wenige zu nennen.

Anforderungsqualitätssicherung

Besonders bei der Qualitätssicherung wird klar, dass die hier vorgenommene Unterteilung nicht immer trennscharf ist und auch nicht sein kann.[94] Einige Techniken sind genauso nützlich im Rahmen der Anforderungsermittlung, -analyse oder -dokumentation. Typisch für die Techniken zur Qualitätssicherung ist jedoch die Ausrichtung auf das Kunden-Feedback, wie in Tabelle 2.5 ersichtlich wird.

Feedback und Review

Simulationen, Szenarien, Prototypen [Bec00]	Beispielhafte Demonstration von Dynamik, Zeitverhalten und Oberflächengestaltung.
Anforderungsreview (Inspektion) [Bec00]	Durch Zustimmung/Ablehnung vom Kunden, die Anforderungen auf Korrektheit und Verständlichkeit prüfen.

Tabelle 2.5.: *Techniken zur Anforderungsqualitätssicherung ([R+02],[Sch02])*

2.5. Ausgewählte Techniken im Überblick

In der Einführung wurde bereits erwähnt, dass im Hinblick auf die Zielsetzung dieser Arbeit zwei Techniken einen besonderen Stellenwert einnehmen. Dies ist zum einen die Modellierung von Anwendungsfällen und zum anderen die Bedeutungsanalyse durch Anforderungsschablonen. Zu diesen Themen ist bereits eine vielfältige Auswahl von Quellen verfügbar. Hier soll lediglich eine Zusammenfassung erarbeitet werden, die die inhaltlichen Kernideen der beiden Techniken darstellt. Daher wird an geeigneter Stelle auf die vertiefende Literatur verwiesen.

2.5.1. Modellierung von Anwendungsfällen

Seit dem „Siegeszug“ der UML ist die englische Bezeichnung „use case“ nahezu jedem Softwareentwickler ein geläufiger Begriff. Wie kaum eine andere Technik wird die Modellierung von Anwendungsfällen jedoch häufig missverstanden und zum Teil sogar falsch praktiziert. Symptomatisch für die Fehlinterpretation ist die unscharfe Unterscheidung zwischen Anwendungsfall und Anwendungsfalldiagramm. Daher soll zunächst die konzeptionelle Zielsetzung von UML-unabhängigen Anwendungsfällen dargestellt werden.

Wenn zum Beispiel der Bedarf eines Unternehmens darin besteht, geschäftliche Vorgänge oder ganze Geschäftsprozesse durch Software abzubilden, untersucht der IT-Spezialist den relevanten Teil, indem er die typischen Aktivitäten eines Mitarbeiters betrachtet. Ihm wird auffallen, dass der Mitarbeiter bestimmte Schritte

[94] Schienmann greift dieses Thema in [Sch02] S.202 auf und bewertet die Nützlichkeit der Techniken für jede Hauptaktivität mit der Skala: gut geeignet/geeignet/nicht geeignet.

wiederholt und dass man seine Aufgaben grob klassifizieren kann. Oestereich findet dafür eine passende Definition: „Ein Anwendungsfall beschreibt eine Menge von Aktivitäten, die aus der Sicht der Endanwender zu einem wahrnehmbaren Ergebnis führen“.[95] Wozu braucht man aber eine Übersicht dieser Aktivitäten?

Der Use Case-Erfinder Jacobsen versteht die Arbeit mit Anwendungsfällen als ein Instrumentarium für die Definition von Softwareanforderungen.[96] Die konzeptionelle Idee liegt darin, durch Beschreibung des Geschäftsumfeldes die Anforderungen an das zu entwickelnde System abzuleiten, da anzunehmen ist, dass die Anwendungsfälle, die vom Benutzer erwarteten Leistungen bereits beinhalten.[97] Die mit der Erstellung von Anwendungsfällen verfolgten Ziele lassen sich wie folgt zusammenfassen:[98]

- Verbesserung des Problemverständnisses
- Entwicklung einer Basis für die Kommunikation mit dem Endanwender
- Sicherstellung der Nützlichkeit aufgenommener Anforderungen
- Erstellung eines organisatorischen Rahmenwerkes
- Identifikation von Systemgrenzen

Betrachtet man die Ausführungen in der Literatur wird man feststellen, dass Anwendungsfälle in recht vielfältigen Bereichen zum Einsatz kommen. So beschreibt Cockburn außerdem Möglichkeiten zur [99]

- Beschreibung von Geschäftsprozessen oder
- Dokumentation von Systementwürfen.

Das Ergebnis der Überlegungen bleibt aber in allen Fällen gleich: man erarbeitet eine Menge von Anwendungsfallbeschreibungen, die eine Gesamtsicht auf das zu entwickelnde System darstellen und Ausgangspunkt für spätere Entwurfsentscheidungen sind. Holub gibt zu bedenken, dass zu den analytischen Arbeiten auch die Aufbereitung aus unterschiedlichen Blickwinkeln zu erfolgen hat. So besteht die Aufgabe der *Anwendungsfallanalyse* darin, die dynamischen Aspekte der fachlichen Seite zu bestimmen.[100] Dabei erfolgt die eigentliche Beschreibung meist in Textform und wird je nach Komplexität des Systems durch Kriterien gegliedert. Hierfür gibt

[95] Vgl. [Oes98] S.207
[96] Vgl. [JCJO92]
[97] Vgl. [Sch02] S.223
[98] Vgl. [Hol01, Oes98, Sch02, OMG03]
[99] Vgl. zum folgenden Absatz [Coc01] S.7
[100] Vgl. [Hol01]

es zahlreiche Vorschläge[101], die mehr oder weniger Checklisten bzw. Formulare mit Leerfeldern darstellen.

Die eigentlichen Informationen werden im Anwendungsfall hinterlegt, können aber neben textuellen Beschreibungen auch durch Ablaufdiagramme, Zustandsdiagramme, Petri-Netze oder in Form von Pseudo-Code beschrieben werden. Jeder Anwendungsfall hat im fachlichen Kontext einen Auslöser, der auch als Aktor bezeichnet wird. Dabei handelt es sich meistens um Mitarbeiter oder Kunden, die ein spezielles Anliegen verfolgen.[102] Durch Akteure kann der IT-Spezialist die Interaktionsbeschreibungen zwischen den Beteiligten und dem System dokumentieren und erhält wertvolle Informationen über die Systemgrenzen. Dadurch entsteht ein recht umfangreiches Bild, das dem IT-Spezialisten die Erstellung des *Anwendungsfallmodells* erlaubt. Dieses dient zum einen als Diskussionsgrundlage im Gespräch mit dem Kunden (Feedback) und ist zum anderen ein Strukturierungsmittel für das Gesamtsystem.

Als grafisches Mittel für die Anwendungsfallmodellierung hat sich das *UML-Anwendungsfalldiagramm* (engl: use case diagram) etablieren können. Abbildung 2.8 zeigt das Standardbeispiel aus der UML-Spezifikation.[103]

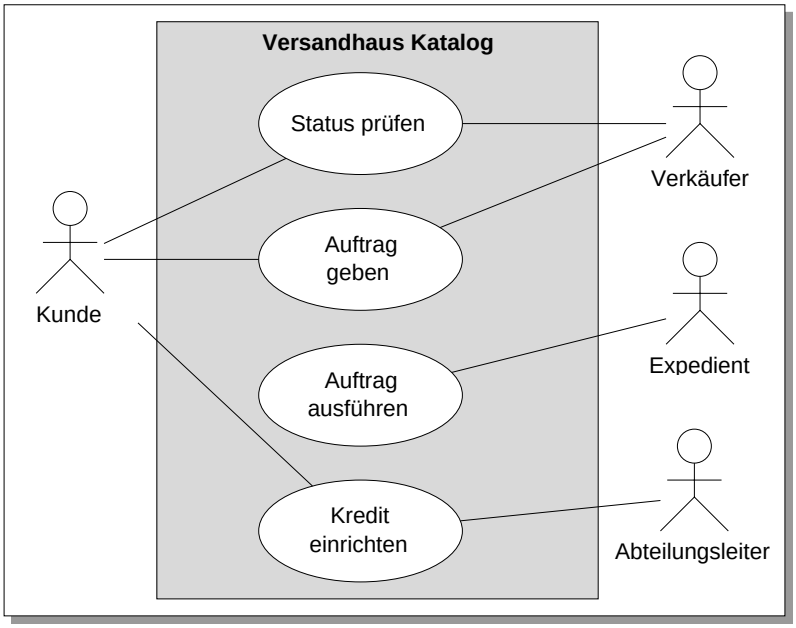

Abbildung 2.8.: *UML-Anwendungsfalldiagramm ([OMG03])*

[101] Beispielsweise formuliert ObjectiF mit dem actiF-Prozessmodell ([MT00]) ein sehr umfangreiches deutsches Beschreibungsschema für Anwendungsfälle und verbindet dieses mit einer speziellen Modellierungssoftware.

[102] Als Aktor kann auch eine Software auftreten, die durch einen bestimmten automatischen Auslöser (eng.: trigger event) eine Aktivität anstößt.

[103] [OMG03] Abschnitt 3.54.3

In dem Diagramm werden die Anwendungsfälle durch grafische Symbole mit ihren Akteuren in Beziehung gesetzt. All dies geschieht im Kontext des gleichen geschäftlichen Problemfeldes („Versandhaus Katalog“). Umgeben von einer (optionalen) Systemgrenze, beschreibt das Diagramm die Anforderungen für das externe Verhalten des Gesamtsystems. Zum Beispiel muss das System in der Lage sein, Kundenaufträge anzunehmen („Auftrag geben“), die Zahlungsmoral des Kunden zu prüfen („Status prüfen“) oder Kredite zu vergeben („Kredit einrichten“).

Da die eigentlichen Informationen eines Anwendungsfalles in textueller Form hinterlegt sind, stößt die grafische Notation schnell an ihre Grenzen. Tatsächlich ist der Aussagewert von UML-Diagrammen eher gering und hat lediglich den Vorteil, dass die Anwendungsfälle etwas kompakter dargestellt werden. Theoretisch könnte man auch eine weitere funktionale Zerlegung, Ablaufbeschreibung oder Modularisierung durch die grafischen Mittel des UML-Diagrammes durchführen. Dafür erweitert die OMG-Spezifikation die Diagramme durch die Beziehungskonzepte Einbeziehung (engl: include), Erweiterung (engl: extend) und Generalisierung (engl: generalization). Die Beziehungen sind zwischen Anwendungsfällen (engl: use case relationships) und Aktoren erlaubt (engl.: actor relationships).[104]

Bei Nutzung dieser Mittel besteht die Gefahr, die ursprüngliche Intension des Anwendungsfallmodells zu unterlaufen. Der Hauptvorteil des Anwendungsfallmodells besteht darin, komplexe Systeme durch wohldefinierte, logisch zusammenhängende Dienste, die streng zu den Erwartungen des Endbenutzers passen, zu strukturieren. Eine funktionale Dekomposition (Zerlegung) würde den Vorteil dieser ersten „High-Level“-Strukturierung von Anforderungen wieder verspielen.[105]

Fazit: Das Anwendungsfalldiagramm setzt die Menge von identifizierten Anwendungsfällen in Beziehung zueinander, soll aber nicht dazu benutzt werden, eine genaue funktionale Zerlegung des Systems zu erarbeiten. Dabei ist festzustellen, dass hierbei keine speziell objektorientierten Methoden zum Einsatz kommen, obwohl das Diagramm Bestandteil der UML ist. Die Darstellung dient dem Austausch zwischen den Beteiligten, da eine Interpretation auch ohne speziellem Training möglich ist. Als „High-Level“-Modell fällt den Anwendungsfällen die Aufgabe zu, die Kundenanforderungen und deren Nützlichkeit im System hervorzuheben.[106] Cockburn verbindet in seinem „Speichenmodell“ (engl.: hub-and-spoke model) die unterschiedlichsten Aspekte der Anforderungsdokumentation, in dessen Mittelpunkt ein gemeinsamer Nenner auftritt: der Anwendungsfall als logische Ausgangsbasis für die Definition von Anforderungen.[107]

[104] [OMG03] Abschnitt 3.57.1

[105] Vgl. [Sch02] S.232, [Oes98] S.215

[106] Die Autoren Constantine und Lockwood thematisieren in [CL99] den Begriff „Essentieller Anwendungsfall“. Damit wird eine zusätzliche Entkopplung zwischen potentiellen Lösungsansätzen und der fachlichen Essenz angestrebt, um so möglichst unbelastet die Anforderungsanalyse zu betreiben.

[107] Vgl. [Coc01] S.15

2.5.2. Bedeutungsanalyse durch Anforderungsschablonen

Um Auftraggeber und Anwender stärker in die Analysetätigkeiten einzubinden, kann man natürlichsprachliche Methoden zur Ermittlung und Qualitätssicherung von Anforderungen einsetzen.[108] Dabei handelt es sich nicht nur um einen weiteren Vorschlag zur Formalisierung von Kundenwünschen. Im Unterschied zu abstrakten Darstellungsversuchen, basiert der Ansatz auf der natürlichen Sprache, um Anforderungen zu verfassen und abzuleiten. Auf der Grundlage von qualitativ hochwertigen, eindeutigen, vollständigen, testbaren, umsetzbaren sowie juristisch verbindlichen Anforderungen in Form von einfachen Sätzen und Wortgruppen kann man auch eine hohe Akzeptanz bei der Verständigung mit dem Kunden erwarten. Die Autoren Rupp et al. beschreiben einen Weg, der es ermöglicht, mit „erstaunlich einfachen Mitteln, Anforderungen in hoher Qualität zu verfassen“. Dabei übertragen sie die praktischen Erkenntnisse der Philosophie, Psychologie und Linguistik auf das Gebiet der Softwaretechnik.

Die theoretischen Vorüberlegungen zur Analyse und Bedeutung der Sprache basieren auf dem Konzept der neuro-linguistischen Programmierung (NLP). Darunter verstehen Sprachwissenschaftler eine Sammlung von Verfahrensweisen die zur Verbesserung der Kommunikation zwischen einer Person und seinen Mitmenschen beitragen können.[109] Dabei untersuchten die Wissenschaftler das Verhalten von „erfolgreichen Sprachkünstlern“ und erstellten schließlich ein Modell menschlicher Kommunikation. Durch Mittel der Abstraktion gelangt man zu einem Metamodell der Sprache, das Grundlage für eine theoretische Analyse von Wörtern, Wortgruppen und Sätzen ist. Bei einer konsequenten Ausweitung der Idee und in Kombination mit NLP kann man ein Schema entwickeln, das Rückschlüsse auf die Qualität einer Aussage ermöglicht.

Die praktische Relevanz für das Requirements Engineering ergibt sich aus der Überlegung heraus, wie im Zuge der Ermittlung von Anforderungen eine Dokumentation in natürlichsprachlicher Form erfolgen kann.

Exkurs: Sprachdefekte

Das wissenschaftliche Gebiet der Linguistik dürfte aufgrund der außergewöhnlichen Komplexität und Vielschichtigkeit der natürlichen Sprache für Fachfremde eine große Herausforderung darstellen. Vereinfacht ausgedrückt sind Sprachdefekte lediglich Wissensverluste, die bei dem Transformationsprozess zwischen Wahrnehmung (persönliches Wissen) und der Wissensdarstellung (sprachlicher

[108] Vgl. zu dem folgenden Absatz [RD01], [RD00], [R+02] S.229ff.

[109] Das Modell des NLP wurde seit Mitte der 70er Jahre in den USA von Bandler und Grinder entwickelt. Die Eigenschaften einer kongruenten (stimmigen) Persönlichkeit haben großen Einfluss auf die zwischenmenschliche Kommunikation. Unter Argumentation von „Grundelementen einer erfolgreichen Kommunikation“ versprechen die Theoretiker eine erfolgreiche Neuausrichtung (Programmierung) der eigenen Fähigkeiten. Quelle: `http://www.nlp.at`

Ausdruck des Wissens) auftreten. Beim Gebrauch der Sprache werden bewusst oder unbewusst Informationen transportiert, weggelassen, verallgemeinert oder verzerrt. Zum besseren Verständnis folgt ein beispielhafter Auszug potentieller Sprachdefekte, die im Zusammenhang mit der Anforderungsanalyse häufig in der Praxis vorkommen.

- **Tilgung.** Die Komplexität wird durch selektive Aufmerksamkeit und zu Gunsten der Übersicht auf ein geringes Maß reduziert.

  Unvollständige Prozesswörter:

  „`Das Kennwort soll im Webfrontend eingeben werden.`“

  Passivsatz, bei dem nicht klar ist, WER etwas durchführen darf.

  Unvollständige Vergleiche:

  „`leicht erfassen`“ oder „`schnell auflisten`“

  Defektbehaftet sind Aussagen und Steigerungen ohne Angabe von Metriken oder Bezug. Wie ist „leicht“ oder „schnell“ im Unterschied zu „schwer“ oder „langsam“ definiert?

  Implizite Annahmen:

  „`Nach Ablauf des Quartals soll das System alle Formulare für das Archiv ausdrucken.`“

  Selbstverständliche Aussagen und Sachverhalte werden durch Wörter kodiert und nicht extra erfasst. Implizite Annahmen im Beispiel: Das Geschäftsjahr ist in Quartalen unterteilt. Es gibt ein Report zum Ende des Jahres. Wie viele Geschäftstage umfasst ein Quartal? Durch welche Daten wird ein Formular beschrieben?

- **Generalisierung.** Gesammelte Erfahrungen werden durch Abstraktion auf andere Sachverhalte übertragen, was jedoch bei einer zu starken Verallgemeinerung zu Fehlern führen kann.

  Unvollständige Bedingungen:

  „`Es soll die Möglichkeit geben, die Felder der Stammdaten zu editieren, wenn die Anmeldung erfolgreich war.`“

  Was ist, wenn die Anmeldung fehlgeschlagen ist?

  Universalquantoren (unscharfe Quantifizierung):

  „`Einmal im Monat sollen die erfassten Formulardaten automatisch gesichert werden.`“

  Welche Daten sind gemeint? Alle Formulardaten? Oder nur die Neuen, seit der letzten Sicherung?

- **Verzerrung.** Von Verzerrung spricht man, wenn Menschen falsche Modelle konstruieren bzw. wenn Menschen die Realität falsch wahrnehmen.

  Nominalisierung:

  „`Nach Erfassung der Formulardaten werden diese gespeichert und zur Durchsicht vorgelegt.`“

  Komplexe Prozesse sollten nicht durch Ereignisse vereinfacht werden: Wer ist berechtigt für eine Durchsicht? Was muss bei einer Durchsicht erledigt werden?

  Funktionsverbgefüge:

  „`Das System muss die Lohnberechnungen zum letzten Werktag des Monats innerhalb einer Stunde zu Ende bringen.`“

  Inhaltsarme Verben müssen präzisiert werden. Was heißt „zu Ende bringen“?

Berücksichtigt man die (offensichtlich zahlreichen) Schwächen der Sprache von Anfang an, besteht die Möglichkeit potentielle Fehler bewusst zu vermeiden. Einen konkreten Lösungsvorschlag für die Problematik bieten Rupp et al. mit Anforderungsschablonen. Eigentlich ist deren Verwendung durchaus logisch, da Pläne, Masken und Muster in praktisch allen kreativen Bereichen eingesetzt werden, jedoch ist festzustellen, dass bisher kaum Ansätze zum Requirements Engineering vorlagen.

Anforderungschablonen erzwingen über alle ermittelten Anforderungen eine ähnliche Struktur mit dem Ziel, typische Formulierungsfehler von Anfang an zu verhindern. In Abbildung 2.9[110] wird der syntaktische Kern einer Anforderung dargestellt, der zunächst aus den zwei Blöcken „Rechtliche Verbindlichkeit“ und „Systemaktivität“ besteht.

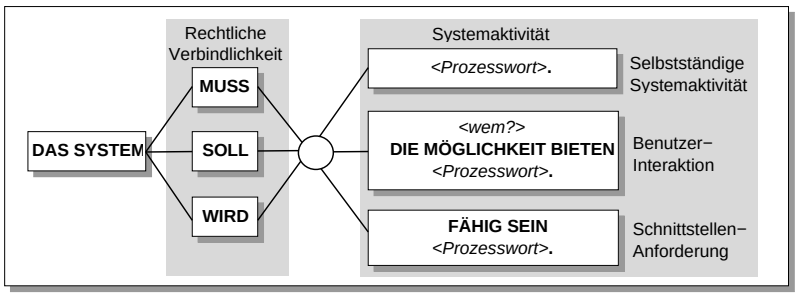

Abbildung 2.9.: *Grundform der Anforderungsschablone ([R+02])*

Aus der Abbildung 2.9 wird ersichtlich, dass durch die Unterscheidung der Systemaktivität in drei Teilbereiche die meisten funktionalen Anforderung an das System

[110] In Anlehnung an [R+02] S.236.

abzubilden sind.[111] Die Konstruktion der Anforderung erfolgt mit Hilfe der Schablone nach wenigen klaren Regeln, indem einfach der gewählte Pfad einen vollständigen Satz ergibt. In fünf Schritten wird dafür die Grundform sukzessive durch Vor- und Nachbedingungen erweitert und hier durch ein Beispiel beschrieben:

1. **Identifizieren des Prozesses.**

   Die geforderte Funktionalität der Anforderung wird durch das *Prozesswort* definiert und ist ein Verb, das den gesamten grammatikalischen Bau des Satzes beeinflusst. Der Prozess charakterisiert den funktionalen Aspekt einer Anforderung.

   - „`hochfahren`“
   - „`speichern`“
   - „`drucken`“

2. **Festlegen der rechtlichen Verbindlichkeit.**

   - muss - Die Anforderung ist *rechtlich bindend.*
   - soll - Die Anforderung ist *dringend empfohlen.*
   - wird - Die Anforderung ist *zukünftig bindend.*

   - „`muss hochfahren`“
   - „`soll speichern`“
   - „`wird drucken`“

3. **Charakterisieren der Aktivität im System.**

   - Das System *führt* den Prozess *selbstständig* durch. (Abb. 2.9: Selbstständige Systemaktivität)
   - Das System *stellt* dem Nutzer die Prozessfunktionalität *zur Verfügung.* (Abb. 2.9: Benutzerinteraktion)
   - Das System führt den Prozess *in Abhängigkeit eines Dritten* aus, ist passiv und wartet auf einen externen Auslöser. (Abb. 2.9: Schnittstellenanforderung)

   - „`Das System` ***`muss hochfahren.`*** “
   - „`Das System` ***`soll dem Benutzer`*** `die Möglichkeit bieten,` ***`zu speichern.`***“
   - „`Das System` ***`wird`*** `fähig sein,` ***`zu drucken.`***“

[111] Vgl. [R+02] S.232 für eine vollständige Herleitung.

4. **Feinschliff für den Prozess.**

   Bisher wurde lediglich die Grundform einer Anforderung konstruiert. Anforderungen werden neben dem Prozesswort, der rechtlichen Verbindlichkeit und der Systemaktivität häufig durch *Objekte und Ergänzungen des Objektes* beschrieben. Es fehlt bisher die nähere Bestimmung des Prozesswortes.

   - „Das System *muss selbstständig hochfahren.*“
   - „Das System *soll dem Benutzer* die Möglichkeit, bieten *im XML-Format zu speichern.*“
   - „Das System *wird* fähig sein, *in einem Netzwerk zu drucken.*“

5. **Formulieren von logischen und zeitlichen Bedingungen.**

   Bei technischen Systemen müssen alle logischen Vor- und Nachbedingungen, die häufig auch zeitlich variieren, berücksichtigt werden. Diese als *Randbedingungen* bezeichneten Aspekte werden an den Anfang einer Anforderung gesetzt.

   - „*Nach einem Stromausfall muss* das System *selbstständig hochfahren.*“
   - „*Bei einem Export soll* das System *dem Benutzer* die Möglichkeit bieten, *im XML-Format zu speichern.*“
   - „*Wenn kein lokaler Drucker zur Verfügung steht, wird* das System fähig sein *in einem Netzwerk zu drucken.*“

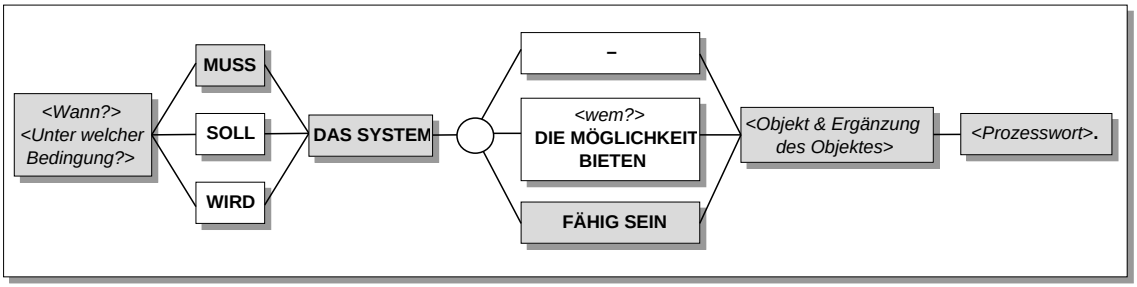

Abbildung 2.10.: *Vollständige Anforderungsschablone ([R+02])*

Abbildung 2.10[112] zeigt die vollständig konstruierte Anforderungsschablone, die im Vergleich zur Abbildung 2.9 durch die Schritte vier („Objekt & Ergänzung des Objektes“) und fünf („Unter welcher Bedingung?“) erweitert wurde. Das Umrücken

[112] Vgl. [R+02] S.238

der syntaktischen Reihenfolge der Satzobjekte (Bedingungen stehen am Anfang des Satzes bzw. Prozesswort im Infinitiv mit ≪zu≫ (pronominales Objekt)), ist begründet durch die besonderen Eigenschaften der deutschen Sprache.

Es steht außer Frage, dass durch eine einheitliche syntaktische Struktur ein qualitativer Vorsprung entsteht. Dennoch befindet man sich erst „auf halber Strecke zu einer guten Anforderung“, wie es Rupp et al. beschreiben. Es sind Optimierungen notwendig, die sich auf den semantischen Teil der Anforderung beziehen. Diese Überlegung wird in Abbildung 2.11[113] dargestellt.

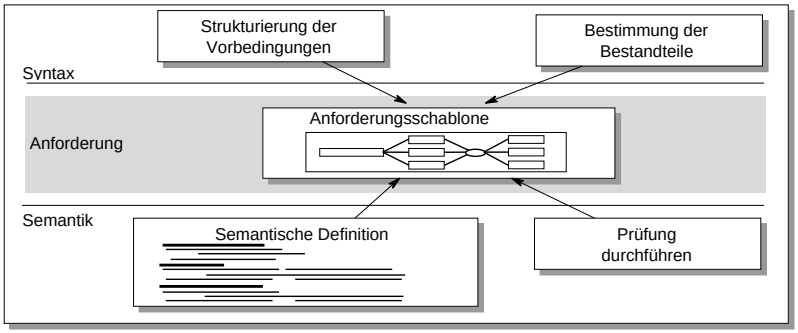

Abbildung 2.11.: *Zusammengesetzte Anforderung ([R+02])*

Eine „gute Anforderung“ besteht demnach aus dem korrekt konstruierten formalen Teil (Syntax) und dem inhaltlichen Gegenpart (Semantik). Die Anforderung wird in ihrer Grundform immer durch die gleiche Schablone entwickelt. Wie bereits erwähnt, wird der syntaktische Teil einer Anforderung lediglich aus den ermittelten Schablonenbestandteilen (Prozesswort, Objektergänzung, etc.) nach Konstruktionsprinzipien „zusammengesteckt“. Dem gegenüber steht die semantische Logik, die prinzipiell nicht in der Schablone hinterlegt werden kann. Das ist jedoch kein Grund, den Teil nur mit Hilfe der „analytischen Genialität“ des Anforderungsingenieurs zu erarbeiten. Der Ansatz bringt daher die folgenden zwei Empfehlungen ein:

1. **Prüfung der Anforderung:** Durchführung einer erweiterten sprachlichen Analyse

2. **Semantische Definition:** Definition der semantischen Anforderungsbestandteile

Die sprachliche Analyse kann durch wissenschaftliche Mittel der Linguistik erzielt werden. Wie schon im Exkurs durch Beispiele angedeutet, basiert dieser Ansatz auf den Erkenntnissen der NLP. Rupp et al. haben dafür einen Algorithmus entwickelt, der es erlaubt, systematisch verschiedene Regeln durchzuprüfen. Da dieser Ansatz zum einen recht komplex ist und zum anderen die Praxistauglichkeit noch nicht

[113] In Anlehnung an [R+02] S.242.

hinreichend bewiesen wurde, soll an dieser Stelle auf die Originalquelle verwiesen werden.[114]

Dagegen ist die semantische Definition mit recht einfachen Mitteln umzusetzen. Das Grundproblem der Begriffs- und Sprachvielfalt kann zwar mit der Schablone eingeschränkt werden, jedoch ist das Problem der Mehrdeutigkeiten einzelner Begriffe damit noch nicht endgültig geklärt. Daher wird ein Glossar aufgesetzt, das die Verwendung und die inhaltliche Auslegung der Schablonenbestandteile eindeutig festlegt.

- „*Nach einem* [*Stromausfall*] *muss* das System [*selbstständig hochfahren*].“
- „*Bei einem* [*Export*] *soll* das System *dem* [*Benutzer*] die Möglichkeit bieten, *im* [*XML-Format*] *zu* [*speichern*].“
- „*Wenn kein* [*lokaler Drucker*] *zur Verfügung steht, wird* das System fähig sein *in einem* [*Netzwerk*] *zu* [*drucken*].“

Im Beispiel werden die wichtigen Satzobjekte identifiziert und können anschließend in einem Glossar bzw. in tabellarischer Form weiterverarbeitet werden. Die inhaltliche Beschreibung kann umgangssprachlich erfolgen, muss aber den Begriff angemessen und vollständig erklären können. Bei konsequenter Einhaltung der Vorgehensweise und der daraus resultierenden Konsistenz der Begriffe kann der Wortschatz und die Sprachvielfalt auf ein akzeptables Maß reduziert werden.

Fazit: Die Konstruktion natürlichsprachlicher Anforderungen dürfte ein wertvoller Beitrag für ein professionelles Requirements Engineering sein. Mit dem sehr einfachen und schnell erlernbaren Ansatz steht ein Mittel zur Verfügung, um den Problemen des Alltags auf effiziente Weise zu begegnen. Dennoch steht fest, dass hier kein „Wundermittel für das Requirements Engineering“ entdeckt wurde. Das kreative Geschick bleibt weiterhin eine Eigenschaft der Spezialisten, die während ihrer Arbeit durch Techniken lediglich unterstützt werden. Die Anforderungsschablone ist darüber hinaus stark werkzeugabhängig und lässt sich nur unzureichend auf manuellem Wege benutzen. Selbst mit einem passenden Werkzeug können die Satzfragmente „verbogen“ werden. Die Anforderungsschablone sorgt zwar stets für einen vernünftigen Rahmen, schränkt aber keinesfalls die Phantasie von Sprachkünstlern ein. So werden Fehlinterpretationen zwar deutlich eingeschränkt, aber gänzlich ausschließen kann man sie nie.

[114] Vgl. [R+02] S.223 / Algorithmus zum SOPHIST REgelwerk.

2.6. Zusammenfassung und Einordnung in die Zielsetzung dieser Arbeit

In diesem Kapitel wurde ein breites Spektrum aus Theorie und Praxis der Softwaretechnik in Bezug auf das Requirements Engineering aufbereitet. Basierend auf einer Literaturrecherche wurde in Hinblick auf die folgenden Kapitel ein Überblick erarbeitet, der einen sicheren Umgang mit der Problemstellung und den damit verbundenen Begrifflichkeiten erlaubt.

Ziel der bisherigen Ausführungen war es, theoretische Überlegungen zum RE-Begriff vorzunehmen. Bereits in der Einleitung wurden Definitionen eingeführt, die jedoch nur oberflächlich die Thematik aufgreifen konnten. Daher wurde die Trennung zwischen der Anforderungsanalyse und der Anforderungsentwicklung eingeführt und anhand eines generischen Softwarelebenszykluses eine erste Einordnung des RE vorgenommen. Da aus theoretischen und praktischen Gründen zunächst eine phasenartige Betrachtung angebracht war, konnte das RE auf seine Bedeutung für den gesamten Entwicklungsprozess untersucht werden. Eine Erkenntnis aus dieser Herangehensweise ist, dass die Analysephase die erste mögliche und auch potentiell folgenreichste Fehlerquelle (Summationseffekt) in einem IT-Projekt ist. Unter Argumentation von Qualität und Fehlerverhütungskosten wurde die Wichtigkeit des Anforderungsdokumentes verdeutlicht.

Für die Ideensammlung zum RE war es interessant, wie die Anforderungsanalyse, hier als Phase betrachtet, in unterschiedlichen Vorgehensmodellen berücksichtigt wird. Nach eingehender Literaturrecherche wurden drei Vorgehensmodelle nach den von Schienmann formulierten Hauptaufgaben im Anforderungsmanagement untersucht.

Die Untersuchung des V-Modells ist verbunden mit komplexen Submodellen und bürokratischen Formalismen. Da es kein eigenes Submodell zum Anforderungsmanagement gibt und auch sonst stark auf abstrahierte Konzepte eingegangen wird, ist der praktische Erkenntnisgewinn eher gering. Das Tailoring-Konzept soll den Umfang des Vorgehensmodells relativieren. Trotzdem ist man im Projektverlauf eher dazu geneigt, bei der Auswahl der erforderlichen Aktivitäten/Produkte einen zu hohen Dokumentationsaufwand zu betreiben. Aufgrund der hohen Relevanz für deutsche IT-Dienstleister ist eine Untersuchung des V-Modells hinsichtlich der Ideen des RE trotzdem durchaus angebracht. Die Trennung in eine fachliche und technische Sicht, sowie die Einführung von Nachweiselementen, ist eine theoretisch stichhaltige Erklärung für die Verknüpfung von Kundenanforderungen mit technischen Anforderungen. Das Nachweiselement bildet dabei die Grundlage für Testfälle.

Die Vorgehensweise erinnert insgesamt jedoch stark an wasserfallartige Modelle. Selbst wenn man das V-Modell aus seiner Zeit heraus beurteilt, bietet der Ansatz einen konservativ anmutenden Beitrag zum RE.

Den Rational Unified Process könnte man als „objektorientierte Methode zur Anforderungsentwicklung“ bezeichnen. Im Unterschied zum V-Modell nimmt das Requirements Engineering (RUP: Anforderungsmanagement) den gleichen Rang wie die anderen thematisierten Bereiche ein. Dies betont die Wichtigkeit der Problematik für das Vorgehensmodell. Inhaltlich erfolgt eine Ausweitung des bisherigen Verständnisses zum Begriff. Der RUP beschreibt sehr konkrete Workflows, bringt sie in einen praktischen Zusammenhang durch Vorlagen und Beispiele und führt ein komplexes Rollenmodell für die Entwicklung von Anforderungen ein. Dabei wird auch das Konzept der Anforderungsverständigung deutlich erweitert. Mittels Checklisten und Change Requests wird konzeptionell eine Kontrollschleife über alle Aktivitäten gelegt und der Feedback-Gedanke auf das Requirements Engineering übertragen. Der RUP setzt auf den Anwendungsfall als gedanklichen Ausgangspunkt für Folgephasen wie Entwurf, Test und Dokumentation. Unter praktischen Gesichtspunkten ebenfalls positiv zu bewerten, ist die Möglichkeit, den RUP selbst als ein Produkt anzusehen, das im Projektalltag per Web-Frontend eingesetzt werden kann und Unterstützung bietet. Die Konfiguration erfolgt durch das Werkzeug „RUP Builder“ und erleichtert so die Benutzung, der ohnehin bereits übersichtlichen Spezifikation.

Bewertet man den RUP auf dessen Nützlichkeit für die Entwicklung individueller Softwarelösungen kann Kritik angebracht werden. Der RUP liegt nur in einer englischen Fassung vor, wodurch Anpassungskosten für deutsche IT-Unternehmen entstehen dürften. Außerdem ist das Vorgehensmodell mit seinen über vier Phasen parallel ablaufenden Workflows zu komplex. Die Zuordnung der Analyse-Aktivitäten ist auch bei eingehendem Studium der Dokumente nicht immer nachvollziehbar. Des Weiteren ist der Ansatz „Best Practices“ zu kritisieren. Hier verlässt sich das Vorgehensmodell zu sehr auf das Meinungsbild Dritter. Das Problem besteht darin, dass Empfehlungen häufig erst dann als wertvoll befunden werden, wenn die Organisation selber aus erlebten Fehlern gelernt hat. Natürlich kann sich ein Unternehmen nicht erlauben zunächst „Entgleisungen“ in Kauf zu nehmen. Es ist jedoch dringend erforderlich, Empfehlungen zunächst kritisch zu prüfen, bevor sie als „Best Practices“ blind übernommen werden.

Ein weiterer nicht unerheblicher Kritikpunkt bezieht sich auf die Werkzeugunterstützung. Eine herstellerunabhängige Betrachtung ist als Vorteil zu sehen, da nicht zuletzt die Beschaffung und der breite Einsatz der Rational-Software für kleine und mittelständische Unternehmen eine enorme Kostenbelastung darstellt.

Das eXtreme Programming geht einen anderen Weg als die vorherigen Ansätze. Obwohl der RUP bereits die Qualitätssicherung durch Feedback thematisiert, steckt im Kern von XP eine deutliche Ausweitung bzgl. der Anforderungsqualitätssicherung. Die Ausführungen von Beck stellen die Validierung von Anforderungen - möglichst durch Prototypen - in den Vordergrund, um sicherzugehen, dass die Kundenwünsche richtig erkannt wurden. Dies ist kein Alleinstellungsmerkmal von XP, lediglich die an den Tag gelegte Radikalität des agilen Ansatzes unterscheidet sich von den anderen Vorgehensmodellen. Nur wenige XP-Techniken sind für das RE relevant. Der wohl wichtigste Beitrag der XP-Techniken liegt in der Akzeptanz, den

Prozess der Anforderungsentwicklung, der mehrheitlich durch den Menschen bestimmt wird, auch in Beziehung mít den Besonderheiten des menschlichen Handelns zu setzen (z.B.: Story Cards, Pair Programming). XP ist im engeren Sinne dennoch nur ein Beispiel für die kombinierte Anwendungen von Techniken. Fundamental neue Aussagen zum RE sucht man vergeblich.

SWEBOK als theoretischer Gesamtüberblick zur Softwaretechnik, deckt sich an vielen Stellen mit dem von Schienmann eingebrachten Schema. Die konzeptionelle Ausrichtung auf Wissensbereiche und die Gliederung des RE in sechs Teilgebiete könnte man als strukturierte Zusammenfassung der Hauptaktivitäten und der Erkenntnisse aus den Vorgehensmodellen beschreiben. Der wohl wichtigste theoretische Beitrag dürfte jedoch das Spiralmodell zum RE sein. Damit wird erstmals ein vollständiger Erklärungsversuch für die komplexen Zusammenhänge der RE-Aktivitäten eingebracht. Obwohl die einzelnen Schritte sehr theorielastig sind, kann das Modell für die Beschreibung praktischer Zusammenhänge herangezogen werden. Nur indirekt sind die drei Vorgehensmodelle auf die Bedeutung der Anforderungsquellen eingegangen. In SWEBOK wird diese Problematik als weitere Aktivität für das Requirements Engineering („Bestimmung der Anforderungsquellen“) ergänzt und die Anforderungsverständigung als Abstimmungsprozess entsprechend hervorgehoben.

Das noch im Review-Prozess befindliche SWEBOK wird häufig zu recht kritisiert. Abgesehen von den sprachlichen Schwierigkeiten, die sich aus den Beiträgen der zahlreichen internationalen Autoren ergeben, kann die Interpretation der Konzepte einige Schwierigkeiten bei der praktischen Auslegung hervorrufen. Schließlich handelt es sich lediglich um ein Bündel von gemeinhin akzeptierten Erkenntnissen. Sehr zu kritisieren sind allerdings die zahlreichen strengen Aussagen, die im Kontext von SWEBOK als unbestritten gehandelt werden. Dies kann kreative bzw. neuartige Ansätze sogar verhindern. Die XP-Idee wird im SWEBOK weder im Zusammenhang mit RE, noch in den anderen Kapiteln thematisiert und scheint unvereinbar mit den aufgestellten Wissensgebieten zur Softwaretechnik zu sein. Trotz aller Kritik handelt es sich bei SWEBOK um einen fundierten akademischen Ansatz, der auf Grundlage vielfach berücksichtigter Expertenbeiträge für weitere Überlegungen gut geeignet ist.

Es wurde ersichtlich, dass die Anforderungsanalyse als eine Phase zum Anfang eines Projektes in den meisten Vorgehensmodellen zwar angestrebt wird, dieses Vorhaben jedoch aus den unterschiedlichsten Gründen nicht ausreichen beschrieben wird. Berücksichtigt man die iterative Natur der Tätigkeiten, fällt das Vorhaben leichter, wenn man die Ergebnisse schrittweise verbessert (phasenübergreifendes Vorgehen). Unterstützung findet man dabei durch den Einsatz von Techniken und Gestaltungsmittel, die im letzten Abschnitt des Kapitels vorgestellt wurden. Ziel war es, eine Gliederung der häufig nur unbewusst wahrgenommen Techniken zu entwickeln. Dabei stand nicht im Vordergrund möglichst alle Techniken zu nennen, sondern ein Gefühl zu vermitteln, wie die Hauptaktivitäten durch den bewussten Einsatz von Techniken unterstützt werden können.

Mit Blick auf die Zielstellung der Arbeit wurden zwei ausgewählte Techniken überblicksartig vorgestellt. Die Modellierung mit Anwendungsfällen ist ein oft zitierter Ansatz, der gemeinhin als Ausgangspunkt für die Anforderungsanalyse gilt. Problematisch ist allerdings die ungenaue UML-Dokumentation der Anwendungsfalldiagramme, die in der Praxis oft unzweckmäßig gedeutet wird. Zum Beispiel wird durch die zwanghafte Abbildung von Regeln, Ablaufbeschreibungen oder Vererbungsbeziehungen mit dem ohnehin begrenzte UseCase-Modellierungparadigma nur ein höchst fragwürdiger Beitrag zur Anforderungsanalyse erreicht. Die Fehlinterpretation stößt auch auf das Unverständnis beim Kunden, dem die Zielsetzung einfacher Diagramme mit den hochwissenschaftlichen „Strichmännchen"-Symbolen nicht ersichtlich wird. Daher wurde auch bewusst auf die Beschreibung höherer UML-Beziehungskonzepte wie ≪include≫ oder ≪extend≫ verzichtet, da höchstens zu unnötiger Verwirrung des Kunden beitragen.[115] Anlass zur Kritik geben auch die textuell überladenen Anwendungsfallbeschreibungen. Mit Hilfe von Gliederungsvorlagen wird zumindest die Komplexität kontrollierbar, wie es in dem RUP-Beispiel ersichtlich wurde.

Konzentriert man sich jedoch auf die ursprünglichen Intention der Anwendungsfälle, kann man von einem nützlichen Mittel zur Entwicklung von High-Level Anforderungen sprechen.

Wenn Anwendungsfälle auf hoher fachlicher Ebene eingesetzt werden, fragt man zu Recht nach einem Mittel, mit dem man auch Low-Level Anforderungen definieren kann. Ausgearbeitet und beispielhaft beschrieben wurde deswegen die Bedeutungsanalyse mit Anforderungsschablonen der Sophist Group, deren Namensgebung sich aus den philosophischen Vorüberlegungen der Sophisten aus dem antiken Griechenland ergibt. Obwohl natürlichsprachliche Ansätze immer die Gefahr von Mehrdeutigkeiten enthalten, steht ein Ansatz zur Verfügung, wie zukünftig Sprachdefekte verhindert werden können. Was bisher mit formalen und grafischen Methoden lediglich umgangen wurde, wird hier im Kern aufgegriffen (Sprachdefekte analysieren) und aufbereitet (Sprachdefekte vermeiden).

[115] Vgl. Oestereich in seinem Beitrag „Pathologie der Anwendungsfälle" aus [R+02] S.212.

3. Analyse kommerzieller Softwareprojekte

„Der Mensch ist der entscheidende Faktor in der Softwareentwicklung, dabei aber gleichzeitig die unberechenbarste Variable - unberechenbar, weil er sein Denken und Tun, sein Fühlen und Handeln mit in die Arbeit einbringt.“[1]

3.1. Erhebung von Problemfeldern

Es gibt viele Studien zum Thema „Erfolgsfaktoren von IT-Projekten“ die Aussagen treffen über die häufigsten Mängel gescheiterter Projekte. Die bekannteste Studie dürfte der 1994 veröffentlichte „CHAOS Report“ bzw. dessen Nachfolger „CHAOS Chronicles“ der Standish Group sein.[2] Den Ausführungen zu Folge fallen ca. 40% der Fehlerursachen (unvollständige Anforderungen, mangelnde Nutzerbeteiligung, unrealistische Erwartungen, etc.) unmittelbar mit dem Anforderungsmanagement zusammen.[3] Durch Interpretation der Ergebnisse verschiedener Studien namhafter Organisationen wie European Software Process Improvement Training Initiative (ESPITI), OASIG, Cap Gemini und Silicon, kommt man bei IT-Projekten je nach Land und Domäne auf eine alarmierend hohe Misserfolgsquote die zwischen 50% und 80% liegt.[4]

Die meisten der zugänglichen Studien befassen sich mit allen Aspekten des Projektmanagements und demzufolge nicht ausschließlich mit den besonderen Herausforderungen des Anforderungsmanagements. Dabei mangelt es an gesicherten Erkenntnissen, wie der Kunde als wichtigster Partner im Verständigungsprozess seine Sicht auf das Anforderungsmanagement und den Anforderungsprozess entwickelt. Für einen eigenen Prozess zum Requirements Engineering ist es von hoher Wichtigkeit, die Problemfelder aus der Praxis möglichst umfassend einzubeziehen.

Gegenstand der Betrachtungen

Es soll das Anforderungsmanagement und der Anforderungsprozess betrachtet werden. Zur Begründung kann angeführt werden, dass typische Problemfelder nicht

[1] Vgl. [LR02] OBJEKTspektrum Nr.5/10: Zitat aus einem Artikel von Helga Lüpschen und Ulrich Reukauf.
[2] Vgl. [Gro03] CHAOS Chronicle v3.0/2002
[3] Entnommen aus [Sch02] S.14
[4] Entnommen aus [SMF03]

speziell auf Teilaktivitäten des RE zurückzuführen sind, sondern insbesondere im organisatorischen Umgang mit Anforderungen auftreten. Die nachfolgenden Untersuchungen beziehen sich ausschließlich auf IT-Projekte für kundenspezifisch anzufertigende, individuelle Softwarelösungen mit den folgenden Eigenschaften:

- Die Projekte unterscheiden sich inhaltlich deutlich von früher durchgeführten Projekten mit anderen Kunden.
- Eine Wiederverwendung ist häufig nur durch das Erfahrungswissen der Teammitglieder möglich.
- Die Auftraggeber bringen sehr unterschiedliche Vorleistungen ein (z.B.: Ideenskizzen, Grobkonzepte, Feinkonzepte, Lastenhefte, etc.).
- Die Anforderungen des Kunden sind Gegenstand der Vertragsvereinbarung und damit rechtlich bindend.
- Der Umfang des Projekts erstreckt sich über zahlreiche Anforderungen (>100) mit hoher Komplexität in einem längeren Zeitraum.

Das Ziel der Untersuchung soll sein, einen stärkeren Bezug zwischen Theorie und Praxis herzustellen. Dafür sollen zwei gesonderte Teilbereiche herangezogen werden:

1. **Kundensicht.** Es wird die Wahrnehmung der Auftraggeber (Kunden) auf das Anforderungsmanagement untersucht und hinterfragt, wie typischerweise die Vorbereitung, Umsetzung und Nachbereitung bei IT-Projekten erfolgt. Die Untersuchung ist so zu gestalten, dass auftretende Probleme im Anforderungsprozess erkennbar, sowie nachvollziehbar sind.
2. **Expertensicht.** Aus diversen Gründen unterscheidet sich die Sicht der Experten auf das Anforderungsmanagement gegenüber der Kundensicht erheblich. Die Untersuchung bezieht sich auf regelmäßig wiederkehrende Schwierigkeiten, mit denen der Auftragnehmer in erfolgskritischen Situationen konfrontiert wird.

Die Vorgehensweise für die Untersuchung der Teilbereiche soll im Folgenden erklärt werden.

3.1.1. Kundensicht

Versucht man die Sicht von Kunden auf IT-Projekte zu hinterfragen, sollten zunächst einige prinzipiellen Sachzwänge vergegenwärtigt werden:

- die Kunden fühlen sich oft von der Entwicklungsabteilung unverstanden
- die Weiterverarbeitung der Anforderungen durch den Auftragnehmer ist meist unbekannt

- die Kunden stehen vor einer starken zeitlichen Verzögerungen (die Zeit zwischen Projektstart und Auslieferung)
- die ausgelieferte Anwendung entspricht nach der langen Entwicklungsphase häufig nicht oder nur teilweise den Erwartungen

Es muss ein geeigneter Weg gefunden werden, um die auftretenden Schwierigkeiten des Kunden und die daraus abzuleitenden Konsequenzen für den Auftragnehmer ermitteln zu können. Die Untersuchung erfolgt in Zusammenarbeit mit Kunden der GFT Systems GmbH. Als Erhebungstechnik soll ein speziell entwickelter Online-Fragebogen verwendet werden, der nach Absprache mit den Kunden per Internet auszufüllen ist. Als Begründung, warum die Erhebung durch einen Online-Fragebogen erfolgen soll, werden folgende Punkte angeführt:

- Der Umgang mit dem Kunden erfordert ein besonderes Feingefühl, da gute Kundenbeziehungen einen wesentlichen Einfluss auf den Geschäftserfolg haben. Durch ein professionell gestaltetes Fragebogensystem wird das Image der Firma nicht unnötig belastet, sondern eher gestärkt.
- Die Kunden sind in verschiedenen Regionen ansässig und auf anderem Weg schlecht erreichbar. Außerdem sind sie dem Tagesgeschäft verpflichtet, weswegen die Erhebung ohne großen Zeitaufwand erfolgen muss.

Konzeption des Fragebogens

Der Fragebogen wurde in Anlehnung an ein Projekt des Fraunhofer IESE entwickelt, dessen ursprüngliche Intension eine Analyse zum Stand des Anforderungsprozesses bei IT-Dienstleistern war.[5] Im Fragebogen werden vier Kategorien unterschieden:

- **Allgemeines.** Der Kunde wird zu seiner grundsätzlichen Haltung gegenüber IT-Projekten befragt und kann eine Aussage treffen, ob ihm das prinzipielle Anliegen des RE bekannt ist.
- **Anforderungsprozess.**

  Hier soll der Kunde durch Beantwortung mehrerer Fragen, Aussagen zum Anforderungsprozess treffen. Aus den Ergebnissen lässt sich ableiten, ob und wie dem Kunden die Problematik des Anforderungsmanagements bekannt ist. Durch gezielte Fragen zur Ermittlung und Dokumentation von Anforderungen kann außerdem eine Aussage getroffen werden, inwiefern Kunden zu Teilaufgaben des Requirements Engineering unterstützend mitwirken.
- **Techniken zum RE.** Der Kunde soll eine subjektive Bewertung zu ausgewählten Techniken der Anforderungsermittlung vornehmen. Die Fragen orientieren sich an der in Abschnitt 2.4 aufgestellten Klassifikation. Es sollen nur

[5] Vgl. [IES03]

Techniken bewertet werden, die im Zuge der Anforderungsermittlung für den Kunden relevant sind.

- **Softwareprojekte.** Im letzten Punkt wird nach Kritikpunkten eines typischen IT-Projektes gefragt, um festzustellen, welche Schwierigkeiten der Kunde äußert, die unmittelbar auf das Anforderungsmanagement zurückzuführen sind.

Wir anhand der Kategorien ersichtlich wird, handelt es sich bei den Fragen nicht nur um eine strenge Erfassung von Problemen im Anforderungsmanagement. Vielmehr soll ein Gesamtbild der Kundensicht bestimmt werden, um nach Interpretation zukünftig ein besseres Verständnis für das Anliegen der Kunden entwickeln zu können. Die Druckversion des Online-Fragebogens ist im Anhang A.5 zu finden.

3.1.2. Expertensicht

Die Expertensicht wird im Wesentlichen durch zwei Gruppen bestimmt, die sich unmittelbar an den Bedürfnissen des Kunden orientieren müssen. Es ergibt sich folgendes Bild:

Entwickler - Experten der Softwareentwicklung

- Die Entwickler arbeiten mit veralteten Anforderungsdokumenten mit z.T. widersprüchlichen Anforderungen, die ständig wechselnden Randbedingungen und häufigen Änderungen unterworfen sind.
- Änderungsaufträge des Kunden werden ohne Rücksichtnahme auf bisher produzierte Ergebnisse, den Entwicklern „aufgezwungen“.
- In der Bemühung möglichst viele Informationen zu sammeln, wird eine Großteil der Anforderungen unbearbeitet den Entwicklern übergeben.

Anforderungsingenieure - Experten des Requirements Engineering

- Bei Kundengesprächen entsteht häufig das Knowledge Engineering Paradoxon.[6] Dies ist ein Phänomen, bei dem der Befragte Schwierigkeiten hat, sein Wissen verständlich zu artikulieren.
- Die Bewältigung der Komplexität und Informationsvielfalt einer Anwendungsdomäne wird durch unzureichende Vorleistungen (unklare Beispiele, inkonsistente Aussagen, etc.) erschwert.
- Die Verflechtung zwischen Analyse und Design bewegen den Kunden zu einem unberechenbaren Richtungswechsel seiner Ansichten, auf die ein Anforderungsingenieur entsprechend reagieren muss (Verschiebung des Scopes).

[6] Vgl. [Sch02] S.197

Die Arbeit der Experten ist durch ein hohes Maß an Erfahrungen geprägt. Die Untersuchung soll daher mit den Mitarbeitern der GFT Systems GmbH erfolgen[7], die in zahlreichen Projekten mit Großkunden, aber auch mit klein- und mittelständischen Unternehmen die Vielschichtigkeit des Anforderungsmanagements im Projektalltag erfahren konnten. Als Erhebungstechnik soll eine Kombination aus Interviews, Brainstorming-Workshops und Diskussionsrunden zur Anwendung kommen. Begründet wird die Erhebungstechnik durch folgende Punkte:

- In einem Interview können Bemerkungen der Mitarbeiter in die Auswertung einfließen, die in einem formalen Fragbogen nicht erfassbar sind.
- In den Brainstorming- und Diskussionsrunden werden besonders wichtige Aspekte erkennbar, wobei auch die gegenseitige Beeinflussung ein gewollter Nebeneffekt für die Aufdeckung weiterer Details ist.

Wie auch bei der Kundensicht, soll hier nicht ausschließlich ein Schwerpunkt auf der Ermittlung potentieller Probleme liegen, sondern ein praxisnaher Bezug zur Thematik hergestellt werden.

3.2. Auswertung und Interpretation der Ergebnisse

3.2.1. Kundensicht

Insgesamt wurden 20 namhafte Großkunden der GFT Systems GmbH wie z.B. Deutsche Post/DHL, Thyssen Krupp, T-Online, AWD, Stadt Leipzig und weitere durch ein persönliches E-Mail Anschreiben kontaktiert und um Unterstützung gebeten. Unter Bewahrung der Anonymität der Befragten, soll im Folgenden eine Interpretation der Ergebnisse vorgestellt werden. Hierbei ist anzumerken, dass bewusst auf exakte prozentuale Angaben verzichtet wurde, da bei der vorliegenden Grundgesamtheit nicht von einer repräsentativen Statistik auszugehen ist. Bei der Auswertung soll vor allem das prinzipielle Meinungsbild der GFT-Kunden in der Rolle als Auftraggeber[8] reflektiert werden.

Allgemeines

- Nach der Frage, welcher Schwerpunkt bei Softwareprojekten priorisiert wird, gab die Mehrheit (ca. 40%) den Zielfaktor Qualität an. Etwa gleich verteilt (jeweils ca. 25%) waren die Zielfaktoren Zeit (pünktliche Auslieferung) und Kosten (Einhaltung des Budgets). Die Unabänderlichkeit des Funktionsumfanges wurde von keinem Befragten priorisiert.

[7] Befragt werden Führungsangestellte und Mitarbeiter mit Schwerpunkt auf Projektleitung, Anforderungsmanagement und Softwareentwicklung.

[8] Im Folgenden werden zum Zweck der Lesbarkeit die Begriffe Befragter, Auftraggeber und Kunde synonym behandelt.

- Drei der Befragten gaben an, dass ihnen die RE-Problematik völlig neu ist. Der überwiegenden Mehrheit war das prinzipielle Anliegen zwar bekannt, jedoch sind keine eigenen Ansätze, die den Umgang mit Anforderung verbessern würden, verfügbar. Lediglich zwei Kunden gab an, dass ihnen die Inhalte des RE vertraut sind und sie bereits eigene Ansätze erarbeitet haben.

Anforderungsprozess

- Die klare Mehrheit von über 80% der Befragten verstand den Anforderungsprozess als eine kontinuierliche Aktivität, die mit Beginn des Projektes startet und mit Auslieferung endet.

- Erstaunlich war die verhaltene Einstellung gegenüber dem Änderungsmanagement. Während ein kleiner Teil den Anforderungsprozess immerhin als eine feste Phase mit anschließendem Änderungsmanagement begriff (ca. 20%), konnten sich die Auftraggeber nicht damit anfreunden, ihre abweichenden Vereinbarungen in zukünftige Versionen (Nachfolgeprojekte) zu verlagern. Auch nicht mit dem Zugeständnis, kleine Änderungen durch Change Requests sofort zu berücksichtigen.

- Die Mehrheit (ca. 80%) plant den Anforderungsprozess explizit und sieht die Aufgabe als eigenständigen Teil im Rahmen der Projektbearbeitung.

- Diejenigen, die einen solchen Anforderungsprozess einräumten, wurden befragt, ob in ihrem Unternehmen Richtlinien, Ablaufbeschreibungen und Leitfäden existieren, die den Prozess bereits dokumentieren. Hier konnte die Mehrheit (ca. 60%) bestätigen, dass derartige Dokumente mit dem Ziel der Standardisierung bereits vorliegen. Die Frage, ob der dokumentierte Anforderungsprozess auch wirklich mit dem gelebten Prozess übereinstimmt, verneinten hingegen *alle* Befragten.

- Ein gemischtes Bild ergab die Frage nach einer verantwortlichen Person, die ausschließlich für die Anforderungsentwicklung zuständig ist. Die knappe Mehrheit (ca. 60%) der Befragten gab an, dass sie *keinen* IT-Spezialisten, Experten oder Anforderungsingenieur festlegen, der für die Formulierung von Anforderungen verantwortlich ist.

- Nach der Frage, welche Artefakte die Auftraggeber selbständig vorbereiten, wurde am häufigsten das Lastenheft dicht gefolgt vom Visionsdokument genannt. Geschäftsprozessmodelle, Fachglossare, Grob- und Feinkonzepte würden hingegen nur bei Bedarf angefertigt. Von allen Befragten wurde es als eher untypisch bezeichnet, dass der Auftragnehmer gleichzeitig als Berater fungiert und selbstständig Visions- und Anforderungsdokumente anfertigt.

Techniken zum RE

Die Befragten wurden aufgefordert, mit Hilfe einer Notenskala von 1 (sehr hilfreich) bis 5 (wenig hilfreich) die Nützlichkeit von ausgewählten Techniken zu bewerten. War die Technik völlig unbekannt, konnte dies mit der Note 6 signalisiert werden.

- Kreativitätstechniken sind den Befragten sehr vertraut. Das Brainstorming (ϕ 1,6) wird dem etwas komplexeren Mind Mapping (ϕ 2,2) vorgezogen.
- Beobachtungstechniken sind den Befragten zwar bekannt, deren Nützlichkeit wird jedoch durch die Mehrheit in Frage gestellt wird. Die Feldbeobachtung (ϕ 3) wird dem nahezu unbekannten Apprenticing (ϕ 4,6) vorgezogen.
- Der Bekanntheitsgrad der Befragungstechniken ist erwartungsgemäß hoch. Das Interview (ϕ 1,6) wird als am hilfsreichsten eingestuft, dicht gefolgt von Selbstaufschreibung (ϕ 2,2) und On-Site-Customer (ϕ 3,3).
- Gegenüber den vergangenheitsorientierten Techniken wie Systemarchäologie (ϕ 2,9) und Reuse (ϕ 3,0) sind die Befragten überwiegend neutral eingestellt.
- Die Nützlichkeit von Feedbacktechniken steht hingegen außer Frage. Sowohl Prototypen (ϕ 2,0) als auch Anforderungsreview (ϕ 2,0) wurden von fast allen Befragten als (sehr) hilfreich eingestuft.
- Die Auswertung der sonstigen RE-spezifischen Techniken ergab ein recht gemischtes Bild. Lediglich drei Techniken waren den Befragten überwiegend bekannt. Allen voran wurde die Modellierung mit Anwendungsfällen (ϕ 1,8) und die Anforderungspriorisierung (ϕ 1,8) als sehr hilfreich bestätigt. Die Essenzbildung (ϕ 2,8) wurde gerade noch als nützlich eingestuft. Weitgehend unbekannt sind die in der RE-Literatur viel zitierten Ansätze Problem Frames (ϕ 3,0), CRC-Karten (ϕ 3,8), Snowcards (ϕ 4,0) und NLP/Bedeutungsanalyse (ϕ 4,8).

Softwareprojekte

- Wie nicht anders zu erwarten, sehen auch die Auftraggeber die größten Probleme bei unklar formulierten Zielstellungen und kämpfen häufig mit den Folgen, die aufgrund fachlicher Missverständnisse auftreten.
- Die Auftraggeber (ca. 45%) bemängeln, dass die wirtschaftliche Zielsetzung und die damit verbundenen fachlichen Bedürfnisse nur unzureichend in die Anforderungen einbezogen werden. Dies äußert sich dadurch, dass nach Auslieferung der Produkte mehr technische Funktionalität umgesetzt wurde, als der Auftraggeber fachlich tatsächlich benötigt.
- Der fachliche Umfang (Scope) der Projekte wird durch die Auftragnehmer jedoch weitgehend richtig interpretiert. Keiner der Befragten äußerte Probleme zu unbrauchbaren oder gänzlich fehlenden Funktionalitäten.

3.2.2. Expertensicht

Für die Untersuchung der Expertensicht sind insgesamt 15 Mitarbeiter der GFT Systems GmbH befragt worden. Der fachlicher Schwerpunkt der Spezialisten liegt hauptsächlich im Requirements Engineering bzw. in der Projektleitung. Ergänzend wurde außerdem das Meinungsbild von Entwicklern und Testern mit einbezogen. Neben persönlichen Interviews wurde im Rahmen eines virtuellen Brainstormings via E-Mail dazu aufgefordert, typische Schwierigkeiten im täglichen Umgang mit Kundenanforderungen aufzuzählen und diese zu diskutieren. Die Erhebung ergab im Resultat sehr unterschiedlich geartete Beiträge, die zur Interpretation geordnet und klassifiziert werden. Es soll jedoch *absichtlich* keine Klassifikation nach bekannten theoretischen Konzepten (z.B.: Hauptaufgaben des Anforderungsmanagement nach Schienmann oder Spiralmodell des REP aus SWEBOK.) erfolgen, da der Praxisbezug im Vordergrund stehen soll. Daher erfolgt die Unterteilung durch eine Abstraktion der am häufigsten genannten Punkte. Es ergibt sich folgendes Schema:

- Politik und Aufwandsproblematik
- Organisatorische Aspekte
- Informationsdichte der Anforderungsquellen
- Dokumentation und Umgang mit Anforderungen
- Weiterverarbeitung der Ergebnisse

Organisatorische Aspekte

- Der IT-Dienstleister hat oft keinen Einblick in die Projektplanung und Ressourcenauslastung des Kunden. Dies erschwert die Abstimmung arbeitsteiliger Prozesse und die Koordination von Teilaufgaben. Infolge dessen entstehen unfreiwillige Arbeitspausen und Leerzeiten trotz hohem Zeitdruck.
- Eine unorganisierte Kundenbeziehung führt zwangsläufig zu zwischenmenschlichen Problemen. Dies ist insbesondere dann problematisch, wenn mehrere Beteiligte (Anforderungsquellen) Einfluss nehmen und somit Verwirrung und Widersprüche provozieren.
- Widersprüchliche Anforderungen wirken sich besonders negativ auf die Entwicklungsarbeiten aus, da sie nur durch aufwändige manuelle Prozesse und Rücksprachen zu klären sind.
- Wenn ein Anforderungsingenieur ausfällt oder das Projekt verlässt, gibt es oft kurzfristig keinen adäquaten Ersatz. Dadurch wird das gesamte Projekt gefährdet. Das kann nur verhindert werden, wenn das Wissen über die Anwendungsdomäne ausreichend in Artefakte und Dokumente schriftlich fixiert wird.

Politik und Aufwandsproblematik

- Ausgangspunkt für eine Vielzahl von Schwierigkeiten ist die Tatsache, dass der Kunde unklare Vorstellungen über das Endprodukt hat. In Kombination mit den dabei typischen Anforderungsänderungen durch das Change Managements stößt jedoch die Vergütung auf Unverständnis.

- Es mangelt dem Kunden an der Einsicht, den Aufwand zur Installation eines funktionierenden Anforderungsprozesses entsprechend zu vergüten. Infolge dessen verzichtet der IT-Dienstleister auf einen für den Kunden zugeschnittenen Anforderungsprozess und verfolgt eine einseitige Strategie zum Anforderungsmanagement.

- Die fehlende Kenntnis einiger Kunden über spezifische Besonderheiten von IT-Projekten, führt zu einer Fehlinterpretation des Begriffs „Risikozuschlag“. Dem Kunden fehlt das Verständnis, dass jeder Richtungswechsel einen enormen Änderungsaufwand mit sich bringt und selbst durch Risikosätze nicht vollständig kalkulierbar ist.

- Im Verlauf des Analyseprozesses ergeben sich durch Prototypen oder Zwischenergebnisse neue Möglichkeiten, die dem Kunden einen zusätzlichen Wert verschaffen könnten. Es läßt sich feststellen, dass Kunden deshalb vorsätzlich mehr Funktionalität erwirken, als vorher vereinbart war. Bei Festpreis-Angeboten sollte unbedingt am vertraglichen Gegenstand gearbeitet und potentielle Verbesserungen in Nachfolgeprojekte verlegt werden.

- Der Kunde formt eine bestimmte Erwartungshaltung, die nach der Projektabwicklung möglichst erfüllt werden soll. Häufig kommt es jedoch dazu, dass der IT-Dienstleister (auch mit höherem Aufwand) aus Eigeninitiative ein besseres Ergebnis produziert, das der Kunde so aber nicht erwartet hat. Der Kunde freut sich zwar darüber, dies kann jedoch nicht die Enttäuschung über verfehlte Erwartungen kompensieren (Qualitätsdilemma).

Informationsdichte der Anforderungsquellen

- Gerade in der Anfangsphase ergibt sich aus der enormen Anzahl von Artefakten, Spezifikationen, Diagrammen, Präsentationen und Beispielen ein Komplexitätsgrad, der selbst erfahrene Anforderungsingenieure überlastet. Aufgrund der Formatvielfalt werden die vorgegeben Ausarbeitungen des Kunden zunächst zeitraubend aufgenommen, um sie anschließend unter hohem Aufwand in eine projektspezifische Struktur zu überführen.

- Sehr anspruchsvolle Kunden (z.B.: Öffentlicher Dienst, Großkunden) stellen die Forderung, die Dokumentation und Historie der formulierten Anforderungen exakt nachzuweisen. Bei einer peniblen Nachverfolgung der Zustände steigt

der Verwaltungsaufwand allerdings enorm an. Aufgrund der Masse von Informationen, bindet dieser Prozess einen Großteil der Ressourcen allein für das Management dieser „Metadaten“.

- Die Verarbeitung von Massenanforderungen (hier Projekte mit mehr als 1500 Anforderungen) ist mit manuellen Mitteln nicht mehr effektiv zu handhaben. Insbesondere die Ermittlung und Auflösung von Abhängigkeiten kann durch herkömmliche Mittel, wie z.B. einer manuell gepflegten MS Excel-Tabelle, nicht fehlerfrei funktionieren.

Dokumentation und Umgang mit Anforderungen

- Kunden, denen das Anforderungsmanagement nicht vertraut ist, sehen keine Notwendigkeit darin, die Anforderungen exakt aufzunehmen und zu dokumentieren. Die Anforderungsdokumentation stößt dann auf Unverständnis und belastet aufgrund der offenbar fehlenden Sinnhaftigkeit des Unterfangens die Teamarbeit.

- Die Bestimmung von Anforderungen ist bei jedem Kunden unterschiedlich. Fast immer müssen die Anforderungen projektspezifisch angepasst werden.[9] Es stellt die Frage, bis zu welchem Detaillierungsgrad die Anforderungen dokumentiert werden müssen (Wo ist die Grenze?).

- Die Ergebnisse aus der Anwendungsfallanalyse werden ungenügend mit anderen Ergebnissen der Analyse integriert. Es ist beispielsweise unklar, welche Stellung ein Anforderungskatalog einnimmt und wie die unterschiedlichen Dokumente und Formate mit diesem zu verbinden sind. Dazu kommt das Problem, dass nicht alle Anforderungen des Kunden in einem einzigen Dokument zu hinterlegen sind.

- Problematisch im Anforderungsmanagement ist stets die redundante Datenhaltung, d.h. das Pflichtenheft (das vom Kunden geforderte Dokument) wird parallel zum eigenen Anforderungskatalog gepflegt. Eine Prüfung des Anforderungskataloges durch den Kunden unterbleibt meist völlig.

- Durch den zwangsläufigen Informationsverlust zwischen Lastenheft und Pflichtenheft, gibt es häufig Fehlermeldungen vom Kunden, die sich als Change Request bzw. Fehlinterpretation erweisen und unnötig Recherchezeit kosten.

Weiterverarbeitung der Ergebnisse

- Mit Beginn der Design- und Realisierungsarbeiten werden zahlreiche Nachbesserungen an den Anforderungsdokumenten vorgenommen. Häufig wird hier allerdings die Nachführung der vorgenommenen Änderungen vernachlässigt, was zu Inkonsistenzen gegenüber den Originaldokumenten führt.

[9] Vgl. „Informationsdichte der Anforderungen“

- Der Übergang der Anforderungen in Testfälle ist oft unvollständig. Obwohl Anforderungen unabhängig von Tests betrachtet werden können, gilt dies nicht im Umkehrschluss (Test ohne Anforderung?). Daher müssen Anforderungen so konstruiert werden, dass sie von vornherein testbar sind. Häufig wird dieser Sachverhalt vom Kunden jedoch nicht erkannt, wodurch ein sehr hoher Nachbesserungsaufwand entsteht.

- Die im weiteren Verlauf unterschiedliche Interpretation der Anforderungen durch die beteiligten Designer, Entwickler und Tester führt dazu, dass bestimmte (fachlogische) Fehler erst später entdeckt werden können. Das Gleiche gilt für Change Requests, die zwar vom zuständigen Bearbeiter akzeptiert wurden, aber erst vom Entwickler oder Tester auf Konsistenz geprüft werden können.

3.3. Ein Beispiel aus der Praxis

In diesem Abschnitt wird ein in der Vergangenheit durchgeführtes Projekt der GFT Systems GmbH exemplarisch, ein Beispiel für die Entwicklung einer individuellen Softwarelösung vorgestellt. Ziel ist es, die Komplexität des Requirements Engineerings, anhand einiger praxisnaher Problemstellungen zu beschreiben. Die dabei vorgestellte fachliche Domäne soll außerdem verwendet werden, um die theoretischen Konzepte in Kapitel 4 durch Modelle anwendungsnah zu veranschaulichen.

3.3.1. Vision und Scope

Vor einigen Jahren wurde für die Stadt Leipzig eine externe Organisationsuntersuchung zum Haushalts-, Kassen- und Belegwesen durchgeführt. Ziel dieser Untersuchung war die Aufdeckung von Potentialen für eine effizientere Gestaltung von Abläufen in der Verwaltung (Geschäftsprozessoptimierung). Als Ergebnis wurde die Einführung eines neuen IT-Verfahrens empfohlen, um die genannten Komplexe zukünftig besser zu bearbeiten. Das zu diesem Zeitpunkt eingesetzte Großrechnerverfahren (HKR-Verfahren) war bereits seit 1992 im Einsatz. Eine Programmerweiterung wurde zwar erwogen, erwies sich aber aus den gestellten fachlichen Anforderungen des Auftraggebers und aufgrund des beträchtlichen Alters des Systems als unzweckmäßig. Gesucht war deshalb ein Verfahren, mit dem die zu den Komplexen gehörenden Verwaltungsabläufe abgebildet werden können.

3.3.2. Aufgabenstellung

Für die Aufgabenstellung waren insbesondere folgende Teile relevant:

- **Kassenverfahren (Rechnungserstellung und Anordnungswesen).** Die Buchführung über sämtliche Einnahmen und Ausgaben der Stadtverwaltung

erfolgt über Sachkonten. Dies ist im Einnahme- und Ausgabereich durch öffentlich-rechtliche Vorgaben beschrieben. Sämtliche Aktivitäten werden über den Ordnungsbegriff „Kassenzeichen“ identifiziert und nach Überprüfung der Ämter im Großrechner gespeichert.

- **Haushaltsüberwachung.** Der Haushaltsplan bildet die Grundlage für die Wirtschaftsvorhaben der Gemeinde. Nach dem Haushaltsgrundsatz der sachlichen Bindung ist der Haushaltsplan für die Haushaltsführung verbindlich. Das bedeutet, dass die Verwaltung bei der Ausführung des Haushaltsplanes in der Weise gebunden ist, dass sie Ausgaben nur für den vorgesehenen Zweck und bis zur Höhe der veranschlagten Haushaltsmittel leisten darf.

- **Formular- und Erfassungsmaskengestaltung.** Im Verfahren sollen die bei der Stadtfinanzkasse Leipzig vorhandenen Formulare vollständig umgesetzt werden.

- **Prozessdesign.** Die Formulare der Stadt sollen für eine effizientere Bearbeitung optimiert werden. Bei den Komplexen Rechnungserstellung und Anordnungswesen liegen jedoch unterschiedliche Abläufe der Verwaltungsprozesse vor. In Abhängigkeit von bestimmten Feldinhalten und diversen Plausibilitätsprüfungen muss daher eine automatische Verzweigung in Unterformulare (Unterprozesse) erfolgen, wobei die Verbindung zum Hauptformular zu erhalten ist.

- **Ausdruck.** Die produzierten Formulare sollten mit dem bisherigen Erscheinungsbild identisch sein. Aufgrund der öffentlich-rechtlichen Nachweispflicht von Dokumenten gibt es die Unterscheidung und geplante Steuerung von Ausdrucksrechten bzgl. Originaldokumenten, Duplikaten und Kopien, die in einem Druckmanagement zu berücksichtigen sind.

- **Anbindung an das bisherige Fachverfahren.** Eine Kernaufgabe ist die Einbindung der großrechnergestützten Transaktionssteuerung und die Übergabe der erfassten Formulardaten.

Das bisherige Großrechnerverfahren gestattete den Zugriff lediglich über bereits veraltete Terminal-Systeme. Es ergaben sich darüber hinaus folgende Nachteile:

- die Terminals arbeiteten sehr langsam und benutzerunfreundlich,

- es konnte lediglich eine Erfassung der Daten vorgenommen werden (die schriftlichen Formulare wurden vorher ausgefüllt und kontrolliert),

- Aufgrund der fehlenden Plausibilitätsprüfung war das System sehr fehleranfällig,

- insgesamt ergab sich eine schwierige und unflexible Anbindung an heute übliche Arbeitplatzrechner.

Das Optimierungspotential bestand darin, durch ein neues webbasiertes Client-System zukünftig Mehrfacherfassungen, Fehlereingaben und Medienbrüche zu verhindern. Die Kostenersparnis ergab sich aus der Aufwandsreduzierung bei Fehlern und Informationsrecherchen.

3.3.3. Formular-Managementsystem für die Stadtfinanzkasse Leipzig

Ziel des Projektes war es, ein DV-gestütztes Prozess- und Formular-Managementverfahren für die dezentrale Erfassung und Bearbeitung von Rechnungen, Bescheiden und Anordnungen zu entwickeln. Das Verfahren sollte einen browserbasierten Einsatz auf den Arbeitsplatzrechnern der Fachämter ermöglichen. Im Mittelpunkt stand die Anbindung an das bestehende Großrechnerverfahren und die Ablösung der veralteten Terminals.

Das als *Elektronische Formular Integration* (ELFI) bezeichnete System soll Datenlieferant für das (bereits existierende) Haushaltsüberwachungssystem und das Kassenverfahren (ebenso Bestandteil des Großrechnerverfahrens) der Stadt Leipzig sein. ELFI ist eine Fachanwendung und kommuniziert im internen Netzwerk mit diversen anderen Systemen. Eine Übersicht ist in Abbildung 3.1 dargestellt.

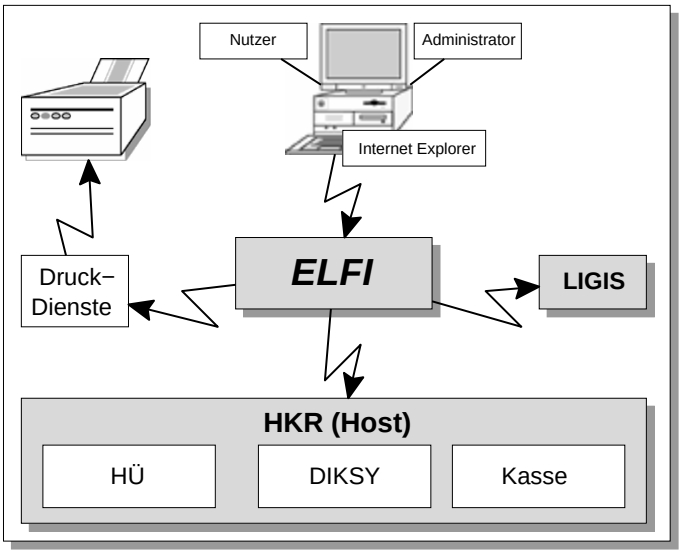

Abbildung 3.1.: *Produktumgebung Elektronische Formularintegration (ELFI)*

Die Nutzer und Administratoren des Systems ELFI kommunizieren über einen **Web-Browser** (Internet Explorer) auf dem Arbeitsplatzrechner mit dem System. ELFI ist eine interne Fachanwendung, so dass ausschließlich Nutzer des internen Netzes der Stadt Leipzig Zugriff haben. Der Zugriff ist nur registrierten Nutzern gestattet, die sich mit Namen und Passwort gegenüber ELFI authenitifiziert haben.

Das **HKR-Verfahren**[10], das auf dem Host betrieben wird, besteht aus mehreren Teilverfahren: HÜ-Verfahren (Haushaltsüberwachung), DIKSY (Dialogorientiertes integriertes Kostenrechnungssystem) und Kassenverfahren. Diese Verfahren sind zentrale, geschäftskritische Anwendungen der Stadt Leipzig. Das System ELFI ist ein wesentlicher Datenlieferant für HÜ- und Kassenverfahren, da es die elektronische Erfassung und Übermittlung von Daten unterstützt.

Das **HÜ-Verfahren** dient der Haushaltsüberwachung der Stadt Leipzig. Dort wird sichergestellt, dass nur finanzielle Mittel, die tatsächlich zur Verfügung stehen, verwendet werden. Bei ausgabeseitigen Anordnungen können durch entsprechende Auftragsbuchungen Mittel reserviert werden. Die dafür erforderlichen Daten werden von ELFI bei der Erstellung von Anordnungen an das HÜ-Verfahren geliefert.

DIKSY ist ein System zur Kostenrechnung bzw. zur Verwaltung von Betriebskonten und Kostenstellen der Stadt Leipzig. Es baut auf dem HÜ-Verfahren auf. Berührungspunkt mit ELFI sind Zahlungsanordnungen, für die es, in Abhängigkeit von der Haushaltsstelle, speziell vorbereitete Datensätze gibt.

Das **Kassenverfahren** (Kasse) dient der Buchführung über sämtliche Einnahmen und Ausgaben der Stadtverwaltung. Zur Verarbeitung einnahmeseitiger Formulare in ELFI gehören im Allgemeinen entsprechende Buchungen im Kassenverfahren. Außerdem werden Datenbestände aus dem Kassenverfahren, wie z.B. die zentrale Adressdatei und die Bankleitzahl-Datei, zur Unterstützung der Datenerfassung in ELFI genutzt.

Das **LIGIS** (Liegenschaftsinformationssystem) ist ein System zur Verwaltung von Daten, Verträgen, Abrechnungen etc. für die Liegenschaften der Stadt Leipzig. Berührungspunkt mit ELFI sind Zahlungsanordnungen, die üblicherweise mit ELFI erfasst werden, für die es jedoch spezielle Ausprägungen unter Beachtung von Liegenschaftsdaten gibt.

Neben der elektronischen Erfassung, Verarbeitung und Übermittlung von Daten ist das Drucken von Belegen eine wesentliche Funktion des Systems. Dabei ist es erforderlich, dass das System weitgehend Kontrolle darüber hat, ob und wie ein Beleg gedruckt wurde. Deswegen läuft der Druck nicht über den Web-Browser des Nutzers, sondern über das System. Dazu werden die vom Netzwerkbetreiber zur Verfügung gestellten **Druck-Dienste** genutzt. ELFI spricht über die lokal (auf dem ELFI-Anwendungsserver) installierten Druckertreiber einen Druck-Server an, der die Druckaufträge von ELFI entgegennimmt.

Vom AG wurden für die Realisierung folgende fünf Formulare vorgesehen:

- Auszahlungsanordnung (AAO)

[10]Die fachliche Abkürzung HKR steht für Haushalts-, Kassen- und Rechnungswesen.

- Umbuchungsanordnung (UAO)
- Standardrechnung/-bescheid (R/B)
- Sollabgang (SOA)
- Rückzahlung Mehrbeträge (RZMB) inkl. Nachweisführung und Freigabe

Das AAO-Formular unterscheidet außerdem bis zu fünf weitere Formularvarianten (z.B.: Überweisung, Ratenzahlung, Verrechnung, etc.) und maximal drei Unterformulare (größere Eingabemasken). Bei genauerer Betrachtung nimmt die dafür nötige Datenbeschreibung (Feldbeschreibungen, Plausibilitäten, Fehlerkorrekturen, etc.) in Umfang und Komplexität schnell zu. Abbildung 3.2 zeigt eine der Formularmasken.

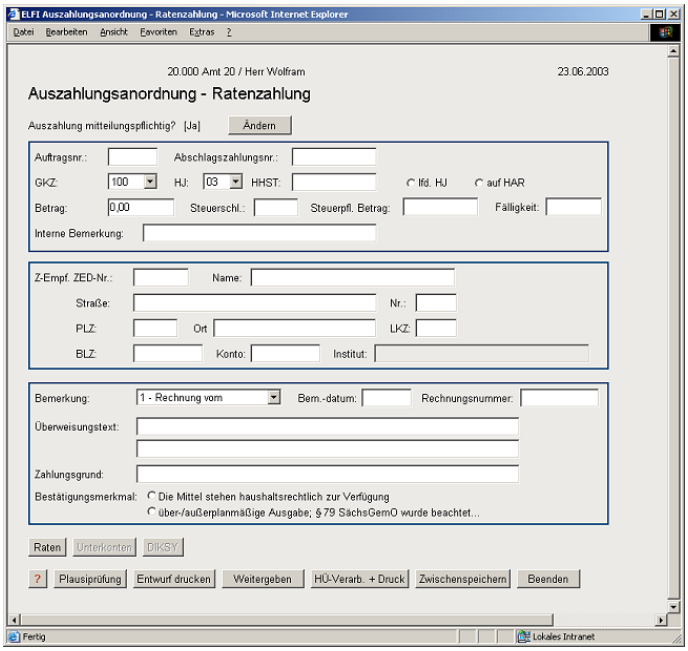

Abbildung 3.2.: *Beispiel-Formular AAO-Ratenzahlung (ELFI)*

Theoretisch wäre mit der Abbildung der Masken die fachliche Seite bereits beschrieben. Jedoch ergibt sich aus der DV-gestützten Vorgehensweise der Bedarf nach einer ebenso DV-gestützten Recherche und Administration des Systems bzgl. Nutzerrechten, Gruppenverwaltung, Druckerkonfiguration und Systemparameter. Die „virtuelle Formularverwaltung“ erfordert die Unterscheidung zwischen einem leeren Formulartypen (Klasse des Formulars) und dem konkreten Fall (Instanz eines ausgefüllten Formulars). Ein Fall ist identifizierbar durch einen Barcode inkl. typisch-bürokratischer Fallzustände (neu, zwischengespeichert, gedruckt, zurückgewiesen, etc.).

3.3.4. Projektreview zum Requirements Engineering

An dieser Stelle soll eine Aufbereitung der gesammelten Erfahrungswerte in Form eines Projektreviews erfolgen. Im Zuge der Recherchearbeiten wurden Interviews mit der Projektleitung, sowie den Anforderungsingenieuren durchgeführt. Die Befragten äußerten rollenspezifisch ihre Meinung in Form von positiven und negativen Erfahrungsberichten. Durch die Erhebung soll die bisher lediglich allgemein formulierte Expertensicht durch real existierende Probleme zum RE beispielhaft geschildert werden. Die Gliederung folgt dem Versuch, die genannten Punkte logisch aufzubereiten.

Anforderungsmanagement im Projekt

- Im Projektplan wurde eine separate Pflichtenheft-Phase inkl. Workshops und Reviews mit dem Kunden berücksichtigt. Als jedoch klar wurde, dass der dafür nötige Aufwand deutlich höher ist als vorher geschätzt, überlagerten sich einige der Review-Termine mit dringenden Analyse-/Modellierungstätigkeiten und fielen aus. Somit verlief die Pflichtenheft-Phase von Anfang an nicht planmäßig.

- Die Reviews waren immer nur für einzelne Formulare vorgesehen, jedoch nie für das System als Ganzes. Die Navigation der Anwendungsoberfläche wurde daher nicht ausreichend spezifiziert und erst durch die Intuition der Entwickler konkret. Dieser Umstand war begründet, durch das zum Teil unkalkulierbare technische Risiko der Anwendung. Die Projektleitung wollte keine Anforderungen vertraglich vereinbaren, deren Umsetzung zu dem Zeitpunkt nicht garantiert werden konnte.

- Aus Zeitgründen konnten nicht alle Anforderungen (z.B. viele technische und nicht-funktionale) erfasst und schriftlich fixiert werden. Insgesamt wurden 1500 einzelne Anforderungen berücksichtigt, während geschätzt ca. 2000 erforderlich gewesen wären.

Anforderungsverständigung und Kundenverhalten

- Die dokumentierten Abläufe wurden in Form von Präsentationen validiert. Dabei wurden die bisher aufgenommenen Anforderungen vorwiegend durch UML-Modelle (Aktivitätendiagramme) und anhand von frühen Maskenentwürfen vorgetragen. Seitens des AGs ergaben sich neue Diskussionen (fachliche Wissenslücken/Unstimmigkeiten), da mehr Teilnehmer als bei der Aufnahme der Anforderungen anwesend war.

  Dies führte zu einer Verzögerung und zusätzlichem Aufwand in der Anforderungsanalyse. Es war notwendig bereits erfasste Anforderungen zu löschen und dafür neue Anforderungen aufzunehmen. Auch musste die Konsistenz dieser Anforderungen mit den Vorhandenen nochmals überprüft werden. Zum Beispiel war das Feld „Mehrwertsteuersatz“ vorher nur ein Anzeigefeld, dessen

Wert vom Host geliefert wurde. Nach dem Review sollte es jedoch ein Eingabefeld sein. Daraus ergab sich die Notwendigkeit einen zusätzlichen Gültigkeitstest durchzuführen, einen neuen Übergabeparameter an den Host zu definieren und somit auch Änderungen an der Schnittstellendefinition vorzunehmen.

- Ein positiver Nebeneffekt war die Tatsache, dass der Projektleiter (PL) des AG gegenüber seinen Anwendern selbst auf Basis der bereits ermittelten Anforderungen agierte und disziplinierte. Sonderwünsche der Anwender fielen aufgrund fehlender Anforderungen weg.

- Grundsätzlich erwies sich die straffe Projektleitung seitens des AG als vorteilhaft. Einige Abteilungen stellten für die fachliche Wissensvermittlung Mitarbeiter frei, die zügig Unklarheiten durch Zuarbeiten beseitigen konnten. Dabei wurden Verantwortliche bestimmt und Termine zur Übergabe der Ergebnisse festgelegt. Zum Beispiel übernahm der Fachbereich bereitwillig die Erstellung eines Glossars, wodurch zügig Informationen gesammelt werden konnten.

- Bei abteilungsübergreifenden Aufgaben (z.B. DV-Abteilung, Host-Anbindung) kam es hingegen öfter zu Schwierigkeiten. Zum einen wegen der fehlenden Querschnittskompetenzen des PL (AG) und zum anderen wegen der mangelnden Erfahrung der Facharbeiter mit IT-Projekten. Selbst kleine Änderungswünsche führten zu zeitintensivem Abstimmungsbedarf.

Umgang mit den Anforderungen und deren fachliche Besonderheiten

- Der AG entwickelte ein abweichendes Begriffsverständnis bzgl. den in der Ausschreibung geforderten fünf Formularen. Der AG unterschied im „Formular AAO“ vier Varianten (statt einer) wodurch insgesamt vier Masken und vier Drucklayouts hervorgingen. Ein ähnliches Problem ergab sich aus anderen Formulartypen, bei denen bis zu drei Unterformulare angeknüpft waren. Die Erfassung wurde deswegen zwar nicht schwerer, der Aufwand multiplizierte sich jedoch um ein Vielfaches der ursprünglichen Berechnungen.

- Die Formulierungen des AG enthielten häufig mehr Anforderungen, als dem AN ersichtlich wurde. Die Anforderung „Ausdruck der Formulare wie im Layoutentwurf“ erwies sich als mehrfach unterschätzte Problematik, da die Fallzustände (Entwurf, Original, Kopie, usw.) vom AG nicht explizit unterschieden wurden. Die Phase der Anforderungsanalyse hätte sich damit deutlich verlängert, da die Masken (Eingabedaten, Reihenfolge, Formate) und die unterschiedlichen Druckarten (Daten, Formate, Anordnung) erfasst werden mussten.

  Die zusätzlichen Anforderungen wurden unter hohem Zeitdruck in den Anforderungskatalog eingearbeitet. Dennoch konnten nicht alle Problemfälle eindeutig definiert werden. Zum Beispiel entstand die Frage, ob eine leere Tabelle (als

Bestandteil des Formulars) ausgedruckt oder absichtlich weggelassen werden sollte.

Gesammelte Erfahrungswerte

- Dem RE muss in künftigen Projekten ein wesentlich höherer Stellenwert eingeräumt werden. Die Erfahrungen zeigen, dass mindestens der gleiche Aufwand einberechnet werden muss, wie zur technischen Umsetzung benötigt wird.
- Das RE darf nicht nur das Erfassen und Verwalten umfassen, sondern muss dringend auch die Struktur der Anforderungen abbilden. Nur so werden Abhängigkeiten und Querverbindungen transparent und können bei Änderungen/neuen Anforderungen berücksichtigt werden.
- Ab dem Umfang von ca. 500 Anforderungen ist die Verwaltung nur noch mit einem passenden Werkzeug effektiv. RE und Änderungsmanagement sollten dabei nicht getrennt werden, sondern im RE-Prozess eine Einheit bilden.
- Erst nach der Aufnahme der Geschäftsprozesse und ihrer Dokumentation, konnten verwertbare Maskenentwürfe für das Gesamtsystem ELFI entstehen. Daher sollte am Anfang der Analysetätigkeiten diese Tätigkeit priorisiert werden.

3.4. Zusammenfassung und präzisierte Problemstellung

In diesem Kapitel wurden viele praktische Erfahrungswerte zu kommerziellen Softwareprojekten genannt und diskutiert. Die von namhaften Organisationen aufgestellten Studien über IT-Projekte sind als Warnsignale zu interpretieren, denn nur so wird der bittere Ernst der Problematik wahrgenommen und endlich im Tagesgeschäft der Unternehmen berücksichtigt. Dabei ist es ein erster und zugleich wichtiger Schritt, die Probleme der Organisation und der Kunden zu erkennen und diese zu thematisieren. Die Erhebung hat gezeigt, wo die Schwierigkeiten auftreten und welche Effekte damit verbunden sind.

Viel konkreter kann man jedoch mit einem Beispiel-Projekt den Sachverhalt argumentieren. ELFI wurde als Beispielprojekt gewählt, weil es ein geradezu typisches Problem spezieller Softwarelösungen reflektiert. Denn trotz erfahrenem Projektmanagement mit einem bereits modernen Verständnis zum Requirements Engineering, bleibt stets das Risiko von Mehraufwand infolge missverstandener Kundenwünsche. Eine mögliche Erklärung für dieses Phänomen wird ersichtlich, wenn man bedenkt, dass bei den Entwicklungsarbeiten stets ein Interessenkonflikt aufgrund unterschiedlicher Präferenzen der Parteien vorliegt:

- Der AG ist bemüht die Budgeteinhaltung durchzusetzen und wegen der Kostenkontrolle möglichst Nachbesserungen unvergütet zu erwirken, wenn nötig durch das Gewährleistungsrecht.

- Der AN nimmt genau die entgegengesetzte Position ein und versucht möglichst alle erbrachten Leistungen abzurechnen und nicht vereinbarte Änderungen nur gegen Vergütung einzubringen.

In Abschnitt 2.2 wurde das Vorgehensmodell als organisatorisches Ablaufkonzept beschrieben, das den Beteiligten bei häufig wiederkehrenden Problemstellungen Unterstützung bieten soll. Die Untersuchungen im Abschnitt 2.3 haben gezeigt, dass die ausgewählten Vorgehensmodelle[11] zwar den prinzipiellen Verlauf aufgreifen, jedoch nur indirekt auf die Hauptaufgaben des Requirements Engineering Bezug nehmen. Es ist einzusehen, dass Vorgehensmodelle und deren Beschreibungsmittel einen generischen Anspruch entwickeln und nicht alle Bereiche im Detail abdecken können bzw. sollen. Besonders deutlich wird dies, durch den „Siegeszug“ der agilen Vorgehensmodelle, bei denen eine Deregulierung und kontinuierliche Anpassung auf die jeweilige Projektsituation im Vordergrund steht. Alles frei nach dem Motto: *„Starre Vorgehensmodelle sind out!“*.[12]

Woran „scheitern“ nun bekannte Vorgehensmodelle wie RUP, V-Modell oder XP in Bezug auf das RE? Welche Aspekte werden ungenügend oder überhaupt nicht berücksichtigt?

- Kein Vorgehensmodell konnte einen prozessartigen Ablauf von der Erhebung bis zur Validierung beschreiben. Der „rote Faden“ ist daher nicht eindeutig abzulesen. XP definiert lediglich ein loses Bündel von Techniken. Das V-Modell kann zwar in den Hauptaktivitäten „System-Anforderungsanalyse“ und „SW-/HW-Anforderungsanalyse“ (Submodell SE) ein Ablaufschema bieten, welches aber aufgrund des hohen Abstraktionsgrades (z.B.: entkoppelte Methodenzuordnungstabellen) und der allgemeinen Ausrichtung auf Hardware- *und* Softwareentwicklung nicht aussagekräftig genug ist. Die im RUP dokumentierte Anforderungsentwicklung ist fast das extreme Gegenteil. Dem Leser erschließt sich aufgrund der Vielzahl definierter Rollen (>30), Artefakte (>50), englischsprachiger Vorlagen (>50), Richtlinien (>10), Werkzeuge (>20), Workflows (>12) und parallele Phasen nur mit großem Aufwand ein umsetzbares Gesamtkonzept (Prozessmodell).

- Kein Vorgehensmodell hat für die Anforderungsentwicklung Teilaufgaben formuliert, die sich mit den theoretischen Ansätzen aus der Literatur (z.B. SWEBOK) decken. Die Bezeichnung der Teilaktivitäten erfolgt je nach Ansatz völlig

[11] Hier werden die VGMs durch die aufgestellten Auswahlkriterien und in Bezug auf die Entwicklung kundenindividueller Softwarelösungen auch als repräsentative Menge sonstiger Ansätze begriffen.

[12] Vgl. [R+02] S.57

unterschiedlich (z.B. V-Modell „System fachlich strukturieren“ vs. RUP „Manage the System Scope“) und führt zu einem unscharfen Begriffsverständnis. Eine logische Herleitung der Begriffe trägt jedoch zur Transparenz und damit zu Verständlichkeit bei. (Theoretische Herleitung)

- Kein Vorgehensmodell trifft spezifische Aussagen im Umgang mit textbasierten Anforderungen. Allgemein wird davon ausgegangen, dass fachliche Anforderungen in einer grafischen Zusammenfassung und durch Prosatexte beschrieben werden. Linguistische Probleme / Sprachdefekte die anschließend bei Interpretation der Dokumente auftreten können, werden nicht thematisiert.

- Die Wiederverwendung von Anforderungen ist im Gegensatz zur Wiederverwendung von Komponenten oder Entwurfsmustern wenig ausgeprägt. Trotz Ausrichtung des RUP auf die UML, wird kein schlüssiges Konzept zur mehrfachen Verwendung von (Sammel-)Anforderungen vorgestellt. Dies liegt vor allem daran, dass die UML hauptsächlich ein Mittel zum objektorientierten Entwurf ist und weniger für die Analyse.

- Die Verknüpfung der Methoden und Techniken wird nicht ausreichend beschrieben. Es fehlt an einer dokumentierten Übersetzung von High-Level Anforderungen (Anwendungsfälle) zu strukturierten, programmierbaren und testbaren Anforderungen (Anforderungssätze). Das V-Modell oder der RUP thematisieren zwar die Problematik, ein vollständig dokumentiertes Überführungsschema stellen sie jedoch nicht vor. Hier ist wieder die Kreativität bzw. die Intuition der Anforderungsexperten gefragt.

- Um grundlegende Verbesserungen zu erreichen, genügt es nicht, Workflows aus dem V-Modell oder RUP zu adaptieren oder bestimmte Vorlagen lose einzubinden. Es fehlt ein Konzept das generisch die Vielfalt der individuellen Kundenwünsche abdecken kann und praktische Hinweise bietet, wie man Zwischenergebnisse logisch in den gewünschten Zusammenhang überführt. (Dokumentationskette)

- Eine Verbesserung interner Arbeitsabläufe („hausgemachte Prozesse“) wird zwar prinzipiell durch die beschriebenen Vorgehensmodelle angestrebt, eine explizite Ausrichtung auf die RE-Aktivitäten ist jedoch nicht vorgesehen. Einzelne isolierte Maßnahmen (z.B. Änderungsanträge) oder die Einführung von softwaregestützten RE-Werkzeugen müssen in das organisatorische Umfeld integriert werden. Ein Prozess soll nicht von Werkzeugen abhängig gemacht werden (z.B. Rational Software - RUP), sondern sollte an den Stellen unterstützt werden, wo eine Automatisierung sinnvoll ist.

4. Konzeption eines Prozessmodells für das Requirements Engineering

„In der Softwareentwicklung gibt es - anders als bei herkömmlichen Ingenieurswissenschaften - zwischen dem Modell und dem Produkt keinen Domänenwechsel. Das Modell einer Softwareapplikation ist wie die daraus resultierende Applikation digital.“[1]

4.1. Überblick

Ziel dieser Arbeit ist die Vorstellung eines methodischen Vorgehens zum RE, in Form eines konzeptionellen Prozessmodells. Zum besseren Verständnis dieses Kapitels folgt eine Wiederholung bzw. Ergänzung der Definitionen:

- **Requirements Engineering Process (REP).** Der Anforderungsprozess als Satz von Aktivitäten dient dem sukzessiven Aufbau von Wissen über eine Domäne und der strukturierten Aufbereitung der gesammelten Erkenntnisse zum Zweck der Zielerfüllung.
- **Requirements Engineering Process Model (REP-M).** Unter dem RE-Prozessmodell wird eine abstrakte Beschreibung zum REP verstanden. Das REP-M gibt Regeln und Richtlinien vor, die einem REP zu Grunde liegen.
- **Konzeptionelles RE-Prozessmodell.** Im Unterschied zum REP-M beschreibt ein konzeptionelles Prozessmodell die im Modell hinterlegte Aufbau-, Ablauf- und Sachlogik des angestrebten methodischen Vorgehens durch entwurfsartige Muster (Konzepte).

Im Verlauf des Kapitels wird kein weiteres Vorgehensmodell beschrieben, sondern dem Motto *„Konfektionieren Sie Ihre eigene Methode!“* von Oestereich nachgegangen.[2] Dies soll nicht durch Präzisierung eines Workflowsmodells geschehen, sondern durch Kombination von Hilfsmitteln und Denkanregungen die eigentliche Intention des RE in den Mittelpunkt stellen. Zur besseren Lesbarkeit wird im Folgenden der Begriff „konzeptionelles RE-Prozessmodell“ synonym mit dem Begriff „Prozessmodell“ benutzt.

[1] Vgl. [Gra03] Computer Zeitung Nr.22/03: Zitat von Wolfgang Bertol IBM/Rational.
[2] Vgl. [Oes98] S.22

4.2. Prozessmodell

4.2.1. Anforderungen an das Modell und dessen Ergebnisse

Aufbauend auf den Ergebnissen dieser Arbeit, sollen an dieser Stelle grundsätzliche Anforderungen an ein eigenes Prozessmodell aufgestellt werden.

Anforderungen an den Prozess

Bei der Betrachtung von Prozessmodellen ist es besonders wichtig, stets in Erinnerung zu halten, dass bei den zu definierenden Prozessen über menschliches Handeln nachgedacht wird. Der direkte Praxisbezug spielt dabei eine wichtige Rolle. Starre Vorgaben sind unrealistisch und sogar gefährlich, denn sie verhindern spontane Anpassungen im Projektverlauf.

Eine Anforderung an den Prozess muss lauten: Mit Hilfe einer fundierten theoretischen Vorüberlegung ist der potentielle Erfahrungsmangel auszugleichen, um so Planlosigkeit zu verhindern.

Dafür ist eine Didaktik erforderlich, die nicht zu abstrakt gehalten ist und an bereits gesammelte Erfahrungswerte anknüpfen kann. Ausschließlich an Beispielen kann jedoch die Methodik nicht vollständig vorgestellt werden, wodurch ein Mindestmaß an Abstraktion erforderlich wird. Jedoch hat es bewährt, den „roten Faden“ anhand eines durchgehenden Beispiels zu dokumentieren, so dass die Strukturierung verdeutlicht und als Ideenlieferant dienen kann.

Da an Vorgehensmodelle prinzipiell die gleichen Anforderungen gestellt werden, sollen die von Balzert formulierten Ausführungen interpretiert und auf das Prozessmodell übertragen werden:[3]

- **Umfang und Mächtigkeit.** *„Lean is better!“* - Das im Kapitel zum V-Modell vorgestellte Tailoring-Konzept soll hier nicht verwendet werden. Daher werden nur die Phasen, Artefakte und Rollen definiert, die unbedingt erforderlich sind.
- **Erweiterbarkeit.** Um das Prozessmodell an zukünftige Anforderungen anpassen zu können, sollte es in Bezug auf Darstellungsmittel, Regeln und Richtlinien erweiterbar sein.
- **Erlernbarkeit und Verständlichkeit.** Konzepte sollten im richtigen Verhältnis zu Effizienz und Einarbeitungsaufwand stehen.
- **Flexibilität.** Hinsichtlich des verwendeten Vorgehensmodells und in Bezug auf die Werkzeugunterstützung des Projekts darf das Prozessmodell keine Festlegungen treffen, die im Konflikt mit den Vorgaben stehen.

[3] Vgl. [Bal01] S.55ff.

Wie bereits in dem Leipziger Projekt ersichtlich wurde, können größere Vorhaben schnell bis zu mehreren Tausend Anforderungen umfassen. Ein zu stark manuell-fixiertes Prozessmodell kann bei der „Massenverarbeitung“ behindernd wirken, wenn nicht sogar bestimmte Aufgaben gänzlich unmöglich machen. Nach Möglichkeit sollte jeder Prozess auf seine Automatisierbarkeit hin untersucht werden, um so die gewünschte Flexibilität auch bei stark unterschiedlichen Rahmenbedingungen beizubehalten. Nur durch automatisierte Vorgänge[4] können Prozesse nachhaltig verbessert werden.

Anforderungen an die Ergebnisse

Eine Lösung für inhaltlich-fachliche Probleme, die im Verlauf des RE auftreten, kann und soll ein Prozessmodell nicht liefern. Bewährte Ansätze können jedoch wichtige Impulse geben. Dabei stehen Fragen des AN im Mittelpunkt wie:

- *„Wo stehen wir gerade und wo wollen wir eigentlich hin?“*
- *„Verfolgen wir noch die ursprünglichen Ziele?“*
- *„Sprechen wir mit den richtigen Leuten und verstehen sie uns?“*
- ...

Bezüglich der Ergebnisse steht die Frage nach dem *„Was?“* und nicht dem *„Wie?“* im Vordergrund. Diese Regel sollte nach Möglichkeit eingehalten werden. Praktisch ist dies allerdings unmöglich, denn Inhalt und Struktur der Anforderungen enthalten teilweise Aussagen und Vorüberlegungen über die technische Realisierung. Siedersleben et al. formulieren Ergebnisse, die ein Prozessmodell prinzipiell umfassen sollte:[5]

- Funktionenmodell, Anwendungsfälle, Abläufe,
- Bedienoberflächengestaltung, Fensterfolgen,
- Benutzerschnittstellen, sowie
- Schnittstellen zu Nachbarsystemen.

Im Ergebnis muss ersichtlich werden, wie die Dokumente sich entwickeln und in welchem Zusammenhang sie stehen.

[4] Insbesondere lassen sich Verwaltungsvorgänge wie Konsistenzchecks, Versionierung, Metaisierung, etc. automatisieren.
[5] Vgl. [S+03] S.24

4.2.2. Vorbetrachtung und Einordnung in den Projektverlauf

In Bezug auf die Zielstellung soll das Prozessmodell in einem speziellen Umfeld einsetzbar und integrierbar sein. Die Funktionsbedingungen, sowie die theoretische Platzierung im übergeordneten Ablaufschema des Projektes wird nachfolgend beschrieben.

Anwendbarkeit

Ziel des Prozessmodells ist die Entwicklung von Anforderungen für individuell zu fertigende Geschäftsanwendungen, deren Ziel es ist, spezifische Geschäftsprozesse zu optimieren, indem Teilprozesse oder vollständige Geschäftsvorfälle durch integrierte IT-Systeme abgebildet werden.

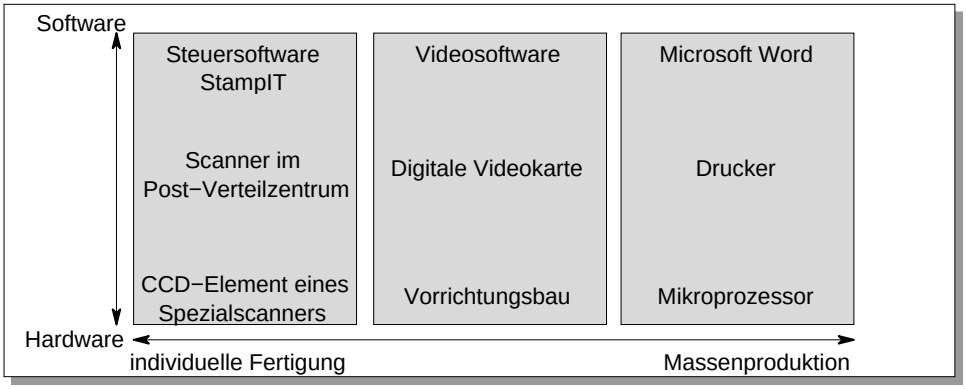

Abbildung 4.1.: *Typen der Entwicklung*

Die individuelle Fertigung von Software, in der linken Spalte der Abbildung 4.1 beispielhaft durch „Steuersoftware StampIT“ dargestellt, erfordert im Gegensatz zur Massenproduktion stets projektbasierte Speziallösungen. Zur Begründung werden folgende Punkte angeführt:

- Art der Projekte ist sehr kundenspezifisch
- starker Bezug zur Ablauf- und Aufbauorganisation des AG
- keine Verfügbarkeit passender Standardprodukte
- hoher Beratungs- und Anpassungsaufwand
- hoher Innovationsgrad der angestrebten Lösung

Für die Anwendbarkeit des Prozessmodells wird eine Vision, d.h. ein Zukunftsentwurf des Auftraggebers[6], als Vorbedingung festgelegt. Außerdem verlangt das Prozessmodell ein Minimum an organisatorischen Vorkehrungen, wie z.B. eine Projektleitung sowohl auf Seite des AN als auch auf Seite des AG.

[6]Im Folgenden werden die Begriffe AG (Auftraggeber) und Kunde synonym verwendet.

Einordnung in das Projektverlaufsschema

In der Praxis und in der Literatur[7] findet man häufig die wichtige Unterscheidung zwischen einer Pre-Sales Phase (Ausschreibung, Angebotserstellung, Voruntersuchung, etc.) und der eigentlichen Projektarbeit. In Abbildung 4.2 ist ein typisches Ablaufschema dargestellt.

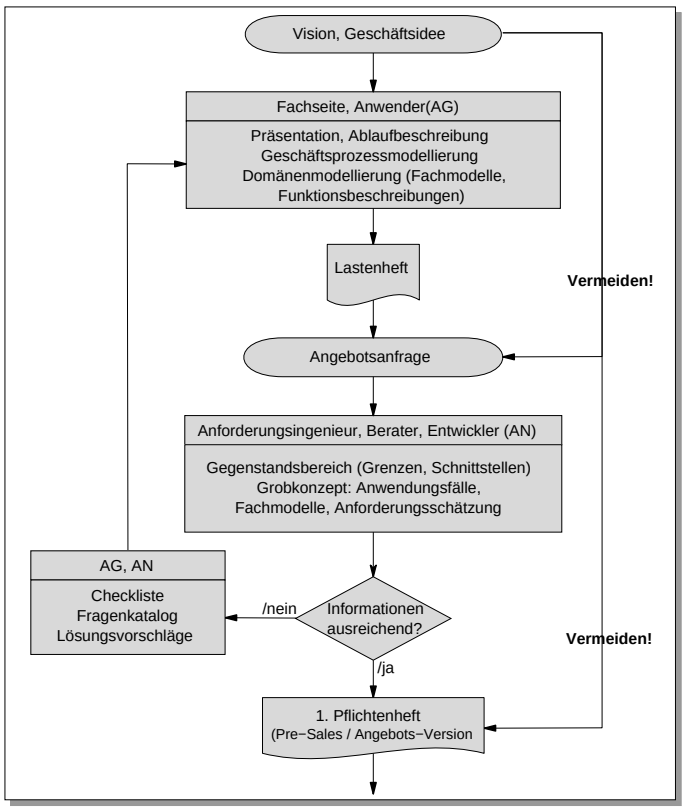

Abbildung 4.2.: *REP in der Pre-Sales Phase*

Als Konsequenz aus dieser Unterscheidung ergibt sich ein etwas differenziertes Bild in der Pre-Sales Phase. Dies hat folgende Gründe:

- Die Unterscheidung berücksichtigt die übliche Praxis von Ausschreibungen zur Vergabe von IT-Projekten, insbesondere bei öffentlichen Auftraggebern.
- Unter hohem Zeitdruck und lediglich mit Hilfe der durch den AG übergebenen Materialen soll eine Vorstudie erstellt und eine Aufwandsschätzung vorgelegt werden.
- Die interdisziplinäre „Estimation Force“ (dt.: Schätzgruppe) unterliegt dem Zwang, einerseits den Auftrag zu erhalten und zum anderen schnell und effek-

[7] Vgl. [Oes98] S.75

tiv die Anforderungen zu identifizieren, damit in der Aufwandsschätzung und Angebotserstellung keine falschen Annahmen einbezogen werden.[8]

Prinzipiell sind Konzept und Vorgehen in der Pre-Sales Phase als eine beschleunigte Version des eigentlichen REP zu verstehen. Da das Thema dieser Arbeit nicht auf die Angebotserstellung abzielt, soll der Schwerpunkt des Prozessmodells an dieser Stelle bei dem REP von vertraglich zugesicherten Projekten liegen.

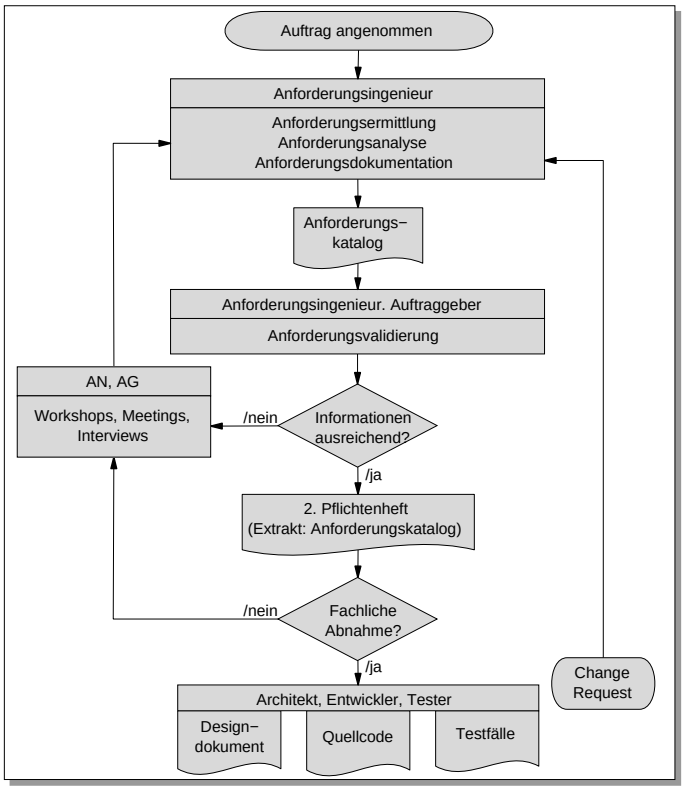

Abbildung 4.3.: *REP nach Angebotsannahme*

In Abbildung 4.3 wird die Positionierung des REP im Gesamtgefüge eines generischen Projektverlaufs skizziert. An dieser Stelle liegt jedoch noch keine Ablaufbeschreibung des REP vor. Die Darstellung ist lediglich Ausgangspunkt der folgenden Betrachtungen. Die vollständige REP-Ablaufbeschreibung wird nach Vorstellung der Teilaufgaben in Abschnitt 4.3.6 vorgestellt.

[8]Theoretisch sollte die Schätzgruppe eine wirtschaftlich unabhängigen Standpunkt einnehmen, um zu realistischen Ergebnissen zu gelangen. Dies ist jedoch in der Praxis selten vorzufinden.

4.2.3. Inhalt und Konstruktion des Prozessmodells

Grundsätzliches Verständnisprinzip

Das Prozessmodell verfolgt einen iterativen Ansatz. Damit kommt es zwangsläufig zu Überschneidungen zwischen den RE-Aktivitäten und der Designphase. Es ist beabsichtigt, die Ergebnisse der Analyse frühzeitig in das Design zu überführen. Mit jeder Iteration der Analyse geht damit eine Iteration im Design einher (Erkenntniszuwachs). Damit ergänzen sich Analyse und Design gegenseitig. Eine gut konzipierte Organisation und eine klare Vorstellung von dem zugrunde liegenden Prozess kann dazu beitragen, unnötige Überschneidungen und Wechselwirkungen (Tunnelblick durch Designzwänge) zu verhindern.

Die Entwicklung der Anforderungen und der Beweis ihrer Erfüllung (Testfälle) haben keinen direkten Bezug zur Objektorientierung. Moderne Softwaresysteme jedoch unterliegen dem OO-Paradigma. Daher kann es im Interesse der Softwareentwickler von Vorteil sein, wenn die Kundenwünsche gewissermaßen durch eine „objektorientierte Brille“ analysiert werden.

Die logische Gliederung des Prozessmodells basiert auf den Erkenntnissen aus Kapitel 2. Die Ableitung folgt der theoretischen Vorbetrachtung von Schienmann in Kombination mit den vier Aktivitäten aus dem SWEBOK-Spiralmodell. Die daraus abgeleiteten Teilaktivitäten werden hier als *Tasks* (dt.: Maßnahme) bezeichnet. Die Namensgebung resultiert aus der Verknüpfung der beiden Ansätze:

1. **Scoping.**
2. **Ermittlung und Ableitung.**[9]
3. **Dokumentation.**[10]
4. **Analyse.**[11]
5. **Validierung.**[12]

Der Abarbeitung der Tasks wird als *Anforderungsentwicklung* bezeichnet und beschreibt genau einen vollständigen Durchlauf. Das Prozessmodell definiert durch die Tasks unterschiedliche Einstiegspunkte, die sequentiell eine logische Abfolge ergeben. Abbildung 4.4 gibt einen Überblick zwischen den Tasks der Anforderungsentwicklung und den im Prozess zu berücksichtigenden *Aufgaben des Anforderungsmanagements*. Die Vorgehensweise in den einzelnen Punkten soll, mit Schwerpunkt auf der Anforderungsentwicklung, zum eigentlichen Gegenstand des Prozessmodells werden.

[9] Schienmann: Anforderungsermittlung; SWEBOK: Requirements Elicitation
[10] Schienmann: Anforderungsdokumentation; SWEBOK: Requirements Specification
[11] Schienmann: Anforderungsanalyse; SWEBOK: Requirements Analysis and negotiation
[12] Schienmann: Anforderungsqualitätssicherung; SWEBOK: Requirements Validation

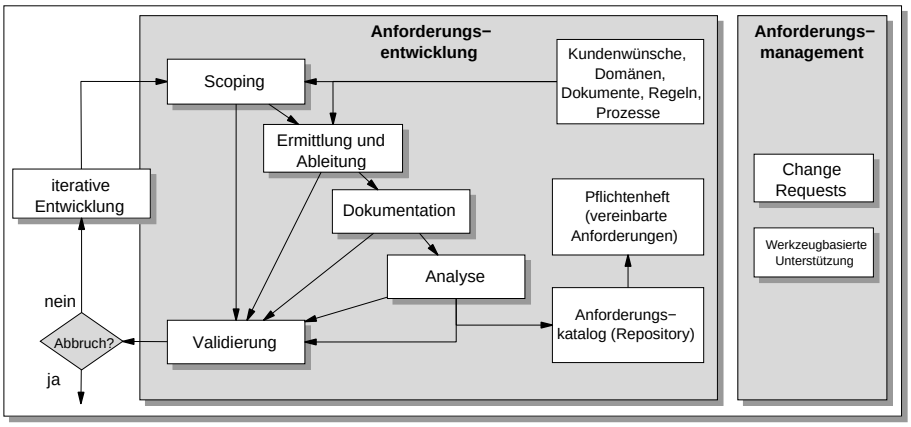

Abbildung 4.4.: *Konstruktion des Prozessmodells*

Hinweis: Der REP ist kein phasenartiger Prozess. Beispielsweise ist die Trennung zwischen Dokumentation und Analyse nur von theoretischer Natur. Man kann nichts dokumentieren, was man nicht vorher analysiert hat. Dies gilt auch im Umkehrschluss. Eine gewisse sequentielle Strukturierung ist für die Beschreibung jedoch erforderlich.

Für die grundlegende Organisation der Anforderungen werden, wie auch im Rational Unified Process, Anwendungsfälle eingesetzt. Obwohl Anwendungsfälle in Kombination mit den Gliederungsvorlagen eine Möglichkeit bieten, umfangreiche Anforderungen zu konstruieren, soll dieser Ansatz lediglich als Ausgangspunkt dienen. Gemäß dem Speichenmodell (engl.: hub-and-spoke model) nach COCKBURN soll mit dem Anwendungsfall eine Verlinkung der gesamten Anforderungs- und Dokumentationsstruktur angestrebt werden. Die hier beschriebenen Konzepte und Beispiele basieren größtenteils auf grafischen Mittel der UML. Grundsätzlich ist das Prozessmodell jedoch nicht abhängig von einem spezifischen Darstellungsmittel. Es kann stets nach Bedarf gewählt werden. In bestimmten Domänen sind beispielsweise Petri-Netze statt der UML-Anwendungsfall- oder Aktivitätsdiagramme aussagekräftiger (Austauschbarkeit der Darstellungsmittel). Dies ist jedoch immer zusammen mit dem jeweiligen Kunden zu klären.[13]

Erweitertes Verständnisprinzip

In Kapitel 2.3 vorgestellten Ansätzen ist nicht völlig klar, welches Begriffsverständnis zu Artefakten wie Anforderungsdokument, Pflichtenheft, Anforderungsspezifikation, etc. vorlag. Daher werden hier folgende Festlegungen getroffen:

- Im Projekt gibt es genau einen *Anforderungskatalog*, der auch als *Requirements Model* bezeichnet wird. Es handelt sich dabei um ein Single-Source Repository.

[13]Nach einer Empfehlung aus Abschitt 2.3.4 (SWEBOK/Konzeptmodellierung) dieser Arbeit.

Dieses zielt darauf ab, sämtliche domänenspezifische Informationen zu strukturieren, zu verknüpfen und zu dokumentieren. Der Anforderungskatalog ist Wissenslieferant für Anforderungsingenieur, Entwickler und Tester. Änderungen und Abfragen erfolgen ausschließlich durch die interne Steuerungslogik des Anforderungskataloges.

- Es gibt beliebig viele *Anforderungsdokumente* verschiedener *Anforderungsquellen*[14](AG oder AN), die verknüpft oder integriert sind mit dem Anforderungskatalog. Beispiele sind Grob-/Feinkonzepte, Protokolle aus Workshops/Interviews, E-Mails, Maskenentwürfe und Diagramme.

- Das für den Kunden zu erstellende *Pflichtenheft* dient zumeist der vertraglichen Absicherung. Es ist ein *historischer Auszug (Extrakt)* vom letzen aktuellen Stand des Anforderungskataloges (siehe auch Abbildung 4.3).

Eine *Anforderung* ist das kleinste zu behandelnde Element und leitet sich aus Anforderungsdokumenten ab. Die Klassifikation aus SWEBOK (Produkt- und Prozessanforderung) erfüllt aus Sicht des Anforderungsingenieurs nur indirekt die notwendige Trennschärfe der Anforderungsarten. Daher sollen in Kombination mit den Ansätzen aus dem V-Modell, RUP (FURPS+) und weiterführender Literatur[15] folgende Festlegungen getroffen werden:

1. **Required Features.** Repräsentieren und detaillieren Anwendungsfälle des Systems (funktionale Anforderungen) und schließen sowohl Geschäfts- als auch Anwenderanforderungen ein.

2. **Required Constraints.** Beschreiben alle Randbedingungen (nichtfunktionale Anforderungen), die die Freiheitsgrade der Required Features bei der Umsetzung einschränken.

   a) Technisch (Infrastruktur, Technologie, Plattform)
   b) Benutzerschnittstelle (GUI, Oberflächenlogik)
   c) Externe Schnittstellen (Fremdsysteme)
   d) Dienstqualität (definiert nach DIN die Aspekte Benutzbarkeit, Zuverlässigkeit, Änderbarkeit, Effizienz, Übertragbarkeit)
   e) Durchführung und Entwicklung (Vorgehensmodell, Standards, Tools, Reviews)
   f) Rechtlich (Vertragsstrafen, CR-Organisation, Eskalationspfade)
   g) Sonstige Lieferbestandteile (Release Notizen, CD-Gestaltung, ...)

Lösungsvorschläge werden im Prozessmodell durch *Konzepte* beschrieben. Ein Konzept beinhaltet die Beschreibung eines methodischen Prinzips, das den Kern der Vorüberlegungen darstellt. Im Rahmen der Task-Beschreibung werden Konzepte benannt und durch ein Beispiel erläutert.

[14] Vgl. Kapitel 2.3.4 - SWEBOK: Anforderungsquellen
[15] Vgl. [Hru00], [R+02] S.170,266, [Sch02] S.50

Vereinbarungen

Für ein Mindestmaß an organisatorischer Verbindlichkeit werden im Prozessmodell folgende Vereinbarungen getroffen:

- **Rollenmodell.** Es wird *kein eigenständiges Rollenmodell* eingeführt. Es erfolgt lediglich eine Unterscheidung zwischen *Anforderungsquelle* (AG, Nutzer, Stakeholder) und *Anforderungsingenieur* (Hauptverantwortlicher im REP). Begründet ist dies in der Tatsache, dass normalerweise die Vorgaben des verwendeten Vorgehensmodells einzuhalten sind.
- **Templates.** Es werden in diesem konzeptionellen Prozessmodell *keine Protokoll- und Dokumentationsvorlagen* konstruiert. Für die spätere Prozessimplementierung sind jedoch verbindliche Templates zu vereinbaren oder zu adaptieren (z.B. RUP).
- **Richtlinien.** Die Dokumentation natürlichsprachlicher Anforderungen erfolgt durch Einsatz von Anforderungsschablonen.

4.3. Entwicklung von Anforderungen

Die Anforderungsentwicklung ist ein kreativer Prozess, bei dem zu Beginn des Projektes der Aufwand am höchsten ist und nach Fertigstellung des Pflichtenheftes gemäß dem theoretischen Modell des RUP der Aufwand stetig abnimmt. Wie schon mehrfach betont, handelt es sich also nicht um einen einmaligen Prozess, sondern um einen stetig fortlaufenden Parallelprozess.

Aus Sicht des Projektmanagements ergibt sich der Bedarf, einen bestimmten Zeitraum zur iterativen Ausführung der Tasks festzulegen. Dieser Vorgang wird hier als *Kernphase* bezeichnet. Daraus ergibt sich zwangsläufig die Frage, welcher Umfang einzuplanen ist. Unter Berücksichtigung der Erfahrungen aus dem Leipziger Projekt und der Ergebnisse der Interview-Reihe, lassen sich folgende Empfehlungen formulieren:

- Der zeitliche Umfang der Kernphase sollte mindestens dem Umfang entsprechen, der auch für die Realisierung vorgesehen ist.
- Stehen Vorleistungen des Kunden (Visionsdokumente, Grobkonzepte, Feinkonzepte, etc.) zur Verfügung, sollte der Aufwand der Kernphase keinesfalls zugunsten der Realisierung reduziert werden. Grund dafür ist, dass Umfang und Qualität der Dokumente (zu groß, zu ungenau, etc.) nur selten die Anforderungsentwicklung erleichtern.
- Ist trotzdem eine Verschiebung des Zeitplans notwendig, so ist unbedingt darauf zu achten, dass keine Reviews (Feedback-Gespräche) ausfallen, denn fehlerhafte Annahmen und eigene Interpretationsversuche können zu schweren Folgefehlern führen.

Die Anforderungsentwicklung ist in der Kernphase ein aufwändiger Prozess, bei dem sich die Frage stellt, wie man in einer Problemdomäne die Übersicht behalten soll und die Komplexität auf ein umgängliches Maß reduzieren kann. Ein möglicher Lösungsansatz ist die Zerlegung in Teilprobleme und deren schrittweise Analyse. Das so genannte „*Teile und Herrsche*“-Prinzip sieht vor, eine Aufgabe so lange zu zerlegen, bis dafür mit elementaren Mitteln eine einfache Lösung konstruiert werden kann. Die Summe aller gelösten Teilprobleme ergibt anschließend die Gesamtlösung. Diese Idee soll als Lösungsansatz übernommen werden.

4.3.1. Scoping

Vorbetrachtung und Ziel

Die erste Aufgabe der Anforderungsentwicklung besteht darin festzustellen, wo der Anwendungsbereich liegt und wie der Systemumfang (engl.: scope of the system) vom Kunden angedacht ist. Die exakte Bestimmung des *Scopes* (dt.: Geltungsbereich) folgt der Grundüberlegung, dass in einem komplexen Problembereich nicht alle Vorhaben durch ein einzelnes Projekt zu realisieren sind. Das so genannte *„Can't do all the work“*-Phänomen ist während der Projektarbeit ein deutlicher Hinweis, dass der bisher ermittelte Scope verkleinert bzw. überarbeitet werden muss.

Aus einer gewissen Konsequenz heraus ergibt sich die Notwendigkeit, dass geäußerte Kundenanforderungen stets auf ihre Relevanz bzgl. des Problembereichs geprüft werden müssen. Daher sollte im Dialog mit dem Kunden deutlich gemacht werden, dass Anforderungen, die außerhalb des Scopes liegen (engl.: out of scope), nicht berücksichtigt bzw. zurückgestellt werden. Scoping ist die Tätigkeit, bei der die Vision abgegrenzt und der Gegenstandbereich vereinbart wird. Zur Bestimmung des Scopes sollte zunächst zwischen den zwei folgenden Ausgangssituationen unterschieden werden:

1. **Scoping bei fester Vision.**
   - Kunde hat Scope und Vision bereits für sich festgelegt
   - benötigt keine weitere Beratung
   - Visionsdokument/Konzept liegt vor (lediglich Analyse durch AN)
2. **Scoping bei ungenauer Vision.**
   - Kunde hat nur unklare Vorstellungen und noch keinen festen Scope
   - nimmt bewusst eine Beratungs-Dienstleistung in Anspruch
   - Unterstützung bei der Entwicklung eines Visionsdokumentes

Im zweiten Fall erfolgt die Unterstützung entweder durch den AN selbst oder durch ein unabhängiges Beratungsunternehmen. Beide Situationen haben jedoch keinen Einfluss auf die prinzipielle Herangehensweise des Anforderungsingenieurs.

Konzept

Ausgangspunkt für die Überlegungen bildet die Betrachtung der Geschäftsprozesse, bei denen sich der Kunde durch eine automatisierte bzw. IT-gestützte Abbildung einen Vorteil verspricht.[16] Die Herausforderung besteht nun darin, die Geschäftsprozessdefinition in eine Anwendungsdefinition zu überführen. Dieser Ansatz wird in Abbildung 4.5 dargestellt.

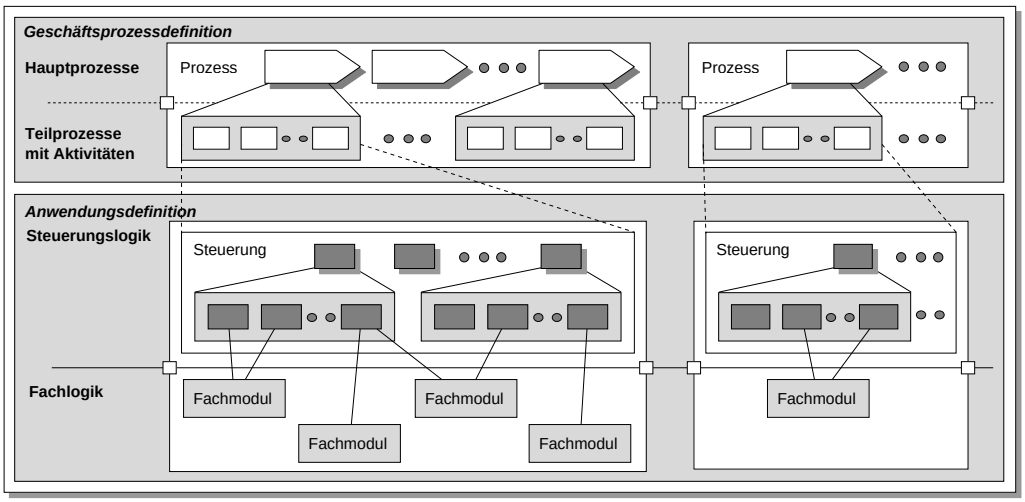

Abbildung 4.5.: *Scoping durch Geschäftsprozesse*

Dem Anforderungsingenieur obliegt nun die analytische Aufgabe festzustellen, welche *Fachmodule* sich aus der fachlichen Steuerungslogik ergeben. Zu diesem Zweck parametrisiert er die Fachmodule (Dokumentation des Scopes) nach folgendem Schema:

- **Priorität.** Wie wichtig ist dem Kunden die Umsetzung des Fachmoduls?
- **Dringlichkeit.** Wie schnell braucht der Kunde die Umsetzung?
- **Iteration.** In welcher Version soll es berücksichtigt werden?
- **Verbindlichkeit.** Muss, soll oder wird das Fachmodul berücksichtigt?[17]

Wie später gezeigt wird, kann der Scope auf jede Kundenanforderung parametrisiert werden. Jede Anforderung kann einzeln untersucht und für potentielle Folgeprojekte dokumentiert werden. Aus den ermittelten Parametern lässt sich anschließend eine Gewichtung vornehmen, die als Entscheidungskriterium für die Bestimmung des Scopes herangezogen werden kann.

[16] Die Ermittlung und Anpassung der Geschäftsprozesse wird im Rahmen des Business Process Reengineering (Hammer) angestrebt. Diese Aufgabe wird hier als Vorleistung erwartet, da es sich dabei um eine sehr komplexe Problematik handelt, die indirekt in das RE fällt.

[17] Zur Definition siehe Kapitel 2.5.2 dieser Arbeit.

Ergebnisse

Durch die grobe Zuordnung und Isolation der Teilprozesse auf Fachmodule sollen folgende Ergebnisse erzielt werden:

- die Vision ist vollständig und lückenlos erarbeitet,
- die Anforderungsquellen (AG, Prozessverantwortliche) wurden identifiziert und dokumentiert,
- der Systemumfang und Systemgrenzen sind durch die Fachmodule verbindlich vereinbart.

Mit Hilfe der Fachmodule ergibt sich eine frühe Auswahl potentieller Anwendungsfälle. Des Weiteren kann damit begonnen werden, die ersten Fachbegriffe grob zu erfassen und diese in einem Glossar zu beschreiben.

Empfehlungen und Beispiel

- Es ist ein typischer Fall, dass die Kunden ihre eigenen Prozesse nicht genau kennen. Hier ist es besonders wichtig, darauf reagieren zu können und den Kunden bei seiner „Selbstfindung“ aktiv zu unterstützen.
- Weniger Scope (Umfang) enthält die Chance zu höherer Qualität, so lange die Geschäftsprobleme des Kunden im Kern gelöst werden.[18]
- Eine allumfassende Lösung sollte nicht in der ersten Begegnung angestrebt werden. Die Erkenntnisse durch XP zeigen, dass eine schrittweise Einführung wesentlich schneller und unkomplizierter zu Erfolgen führt.[19]

Im Beispielprojekt ELFI wurden die Geschäftsprozesse zum Kassen- und Belegwesen untersucht und die Teilprozesse Kassenverfahren (Rechnungs- und Bescheiderstellung) und Haushaltsüberwachung (Anordnungswesen) für eine erweiterte Betrachtung isoliert. Darüber hinaus sollte die organisatorische Kompetenzverteilung (Berechtigungen, Administration, Prüfung, etc.) in Form einer elektronischen Verfahrensadministration abgebildet werden. Die in dem Teilprozess definierte Verwaltung (Erfassung, Recherche und Versand) sollte nun nicht mehr durch handschriftliche Formulare und manuelle Schritte durchgeführt werden, sondern durch ein elektronisches Formularsystem mit Druckfunktion. Im Gespräch mit dem Kunden konnten folgende grundsätzliche Fachmodule identifiziert werden:

- **Formularbearbeitung.** - Erfassung/Recherche von Fällen
- **Belegvorbereitung.** - Freigabe oder Zurückstellung durch Vorgesetzte

[18] Vgl. [Bec00] S.16 und Kapitel 2.3.3 - XP
[19] Vgl. [LRW02] S.46

- **Mahnlistenbearbeitung.** - Mahnlisten anfordern
- **Administration.** - Formulargruppen, Nutzergruppen, Berechtigungen

Im Gespräch war außerdem die Thematik „Elektronische Unterschrift“, auf die jedoch im endgültigen Leistungsumfang verzichtet wurde (out of scope). Der AG hat sich jedoch vorbehalten, künftig derartige Entwicklungsschritte vorzunehmen („wird fähig sein...“).

4.3.2. Ermittlung und Ableitung

Vorbetrachtung und Ziel

Für die Ermittlung und Ableitung muss als Vorbedingung eine Fallunterscheidung erfolgen. Abhängig von Art und Größe des Projektes sowie von den Vorleistungen des Kunden lassen sich zwei Erscheinungsformen feststellen:

1. **Transformation.**
   - Ermittlung über Vorleistung (Lastenheft, Konzepte, Unterlagen)
   - Kontaktperson ist durch AG definiert
   - intensive Analysephase der Dokumente durch AN
2. **Erhebung.**
   - Kontaktpersonen bestimmen (über Scoping)
   - intensive Interview- und Befragungsphase durch AN
   - Ermittlung bei AG vor Ort

In beiden Fällen ist es das Ziel, Anforderungen zu ermitteln bzw. aus den übergebenen Dokumenten abzuleiten und in das Format des Anforderungskataloges zu überführen. Hierbei ist es besonders wichtig, diesen Prozess als einen sich ständig wiederholenden Vorgang zu verstehen. Der dabei auftretende Effekt wird hier *Knowledge Engineering* genannt und bezeichnet den konsequenten Verständnisgewinn über eine Problemstellung.

Beide Fälle werden, trotz abweichender Ausgangssituation, in diesem Prozessmodell durch das gleiche Konzept beschrieben. Unterschiede ergeben sich lediglich bei der zu verwendenden Erhebungstechnik[20](Dokumentenanalyse vs. Interview) und dem Analysemodell[21](Checkliste für Dokumente vs. Fragenliste im Interview). Die Ergebnisse in Form von Anforderungen müssen jedoch in beiden Fällen identisch sein.

[20]Vgl. Kapitel 2.4 dieser Arbeit

[21]Das Analysemodell beschreibt ein Vorgehensschema für die Erfassung in spezifischen Situationen wie Interviews, Meetings, etc.

Konzept

Ausgangspunkt für die Bestimmung von Anforderungen ist die Betrachtung der bereits ermittelten Fachmodule, die ihrerseits abgeleitet sind aus den relevanten Geschäftsprozessen. Hierbei erfolgt eine Übertragung der Fachmodule auf möglichst aussagekräftige Anwendungsfälle, die in einem Anwendungsfallmodell durch Akteure in Beziehung gesetzt werden. Die Grobstrukturierung wird in Abbildung 4.6 abstrakt dargestellt.

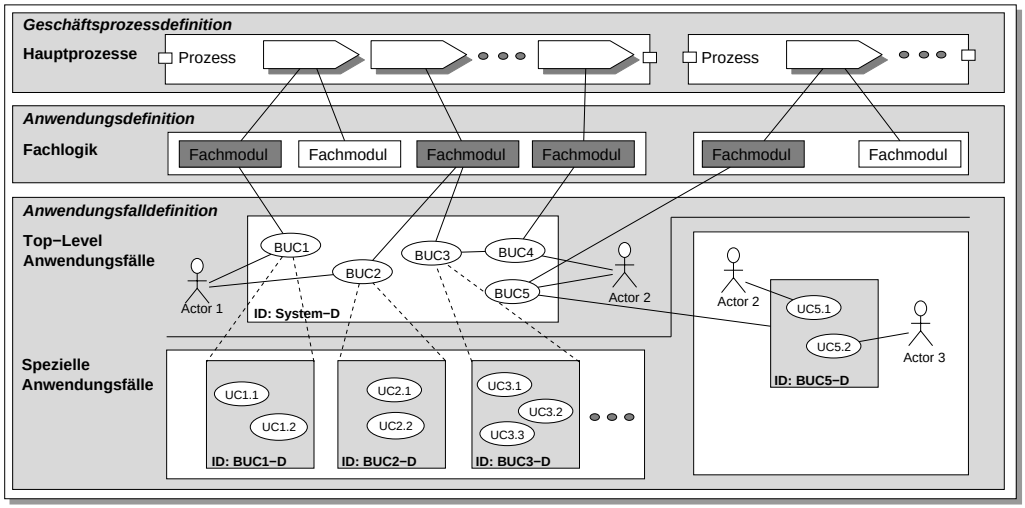

Abbildung 4.6.: *Anwendungsfälle durch Fachmodule*

Ganz wesentlich ist hierbei die Ermittlung von *Business Use Cases* (BUC, dt.: Geschäftsvorfälle). Im Unterschied zu den „Speziellen Anwendungsfällen“ (use cases) beschreiben die „Top Level Anwendungsfälle“ ausschließlich abgeleitete Anforderungen mit direktem Bezug auf ein Fachmodul. BUCs geben dadurch sehr konkrete Hinweise auf die Modularisierung/Komponentenbildung der späteren Anwendung und bilden den definierten Ausgangspunkt der nun folgenden sukzessiven Problemzerlegung nach dem „Teile und Herrsche“-Prinzip.[22]

Die BUCs werden durch *spezielle Anwendungsfälle* granularisiert. Dadurch erhält man einen weiteren Erkenntnisgewinn auf hoher fachlicher Ebene. Es ist jedoch fraglich, bis zu welchem Detaillierungsgrad Anwendungsfälle einzusetzen sind. Als Orientierungshilfe sollte der Anforderungsingenieur stets hinterfragen, ob er sich noch auf hoher fachlicher Ebene befindet. Ist dies nicht mehr der Fall, z.B. bei Modellierung einer Produkteigenschaft („Im System anmelden“ (Login-Dialog)) oder der Abbildung von Ablaufregeln, sollte auf die nächste Beschreibungsebene, wie in Abbildung 4.7 dargestellt, gewechselt werden.

[22]Die Unterscheidung zwischen BUC und UC folgt der Überlegung, Einstiegselemente gesondert zu kennzeichnen.

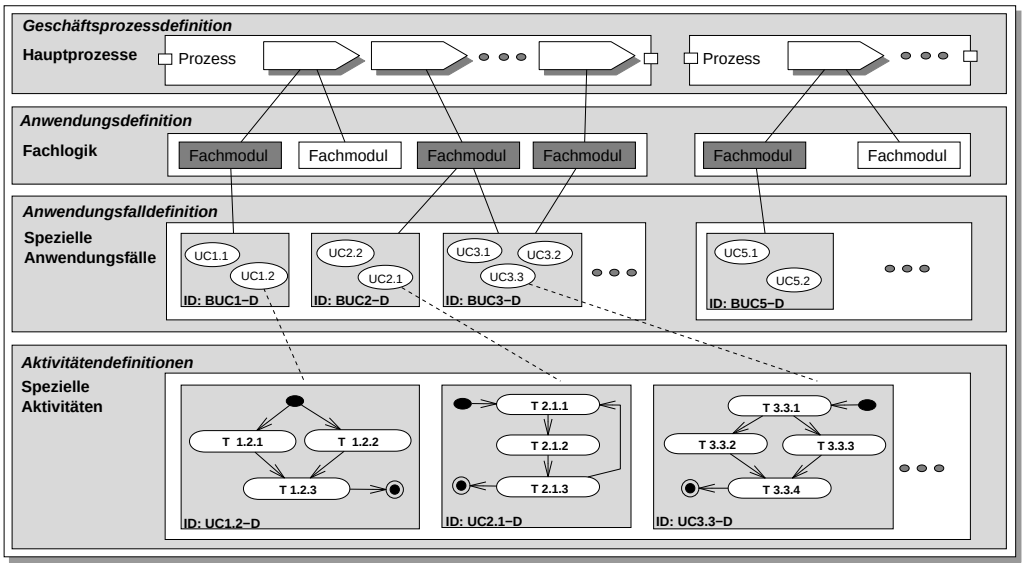

Abbildung 4.7.: *Aktivitätenmodellierung durch Anwendungsfälle*

Da Anwendungsfälle nur mit Kompromissen eine Ablauflogik modellieren können, soll dies durch die Verknüpfung von UML-Aktivitätendiagrammen (AD) erfolgen. Dabei werden die bereits ermittelten funktionalen Anforderungen in ihrer logischen Zusammengehörigkeit stückweise konkretisiert. Nach diesem Schema sollte jeder BUC konsequent analysiert und abgeleitet werden. Die Ermittlung ist beendet, wenn vom Kunden keine neuen Anwendungsfälle gemeldet werden und alle angesprochenen Punkte nur noch von niedriger Priorität sind bzw. außerhalb des Scopes liegen.

Ergebnisse

Nach der Zuordnung und Auswahl der Fachmodule sollten im Rahmen der Ermittlung/Ableitung abhängig vom Ausgangsmaterial folgende Ergebnisse erreicht werden:

- BUCs und Akteure wurden aus den Fachmodulen identifiziert und Anwendungsmodule sind erkennbar
- die Anwendungsfälle wurden granularisiert und modelliert
- die Ablaufbeschreibung aller Anwendungsfälle wird durch ADs beschrieben

Mit Hilfe der streng auf Geschäftsprozesse orientierten Analyse ergibt sich ein umfassendes Bild der fachlichen Anforderungen und eine deutliche Zunahme der Fachbegriffe im Glossar.

Empfehlungen und Beispiel

- Ohne systematischen Ansatz wird der Zusammenhang zwischen Geschäftsvorfall und den Required Features unscharf oder widersprüchlich. Daher muss konsequent über alle identifizierten Anwendungsfälle die Ermittlung nach einem einheitlichen Schema erfolgen.

- Anwendungsfälle eignen sich sehr gut, um Required Features zu ermitteln. Für Required Constraints sind sie dagegen *nicht* geeignet. Außerdem sollte im Diagramm auf die Modellierung mit höheren Beziehungskonzepten der UML verzichtet werden, da sie unnötig die Lesbarkeit für den Kunden erschweren.

- Bei der Analyse sollte dem Anforderungsingenieur stets die Trennung zwischen Produkt und Prozess bewusst sein. Produkteigenschaften sind Required Constraints und haben keine direkte Verbindung zu fachlichen Belangen.[23]

- Die Ermittlung ist eine zeitaufwändige Aufgabe, die viel Geduld und Zusammenarbeit mit dem Kunden erfordert. In der Praxis kommt es häufig vor, dass unklare Anforderungen vorliegen und geklärt werden müssen. Damit der Anforderungsingenieur konzentriert und kontinuierlich arbeiten kann, sollte eine Kontaktperson beim Kunden benannt werden, die für Nachfragen und Zuarbeiten zuständig ist. Dies könnte man auch als Variante einer intensiven Interview-Phase bezeichnen.

Im Beispielprojekt ELFI konnten nach dem hier beschriebenen Konzept, exemplarisch die Grafiken in Abbildung 4.8 und Abbildung 4.9 entwickelt werden. Es wird der Anwendungsfall „AAO vorbereiten“ gewählt und beispielhaft der Prozess zum Formular „AAO-Ratenzahlung“ vorgestellt.[24]

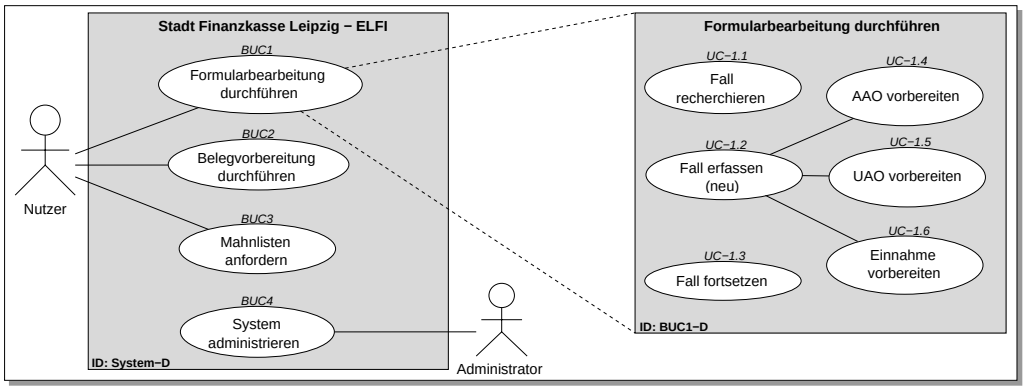

Abbildung 4.8.: *Beispiel BUC und UC in ELFI*

[23] Vgl. [GW93] S.65 - „Kontextfreie Produktfragen“
[24] Vgl. Kapitel 3.3.3 für ein besseres Verständnis.

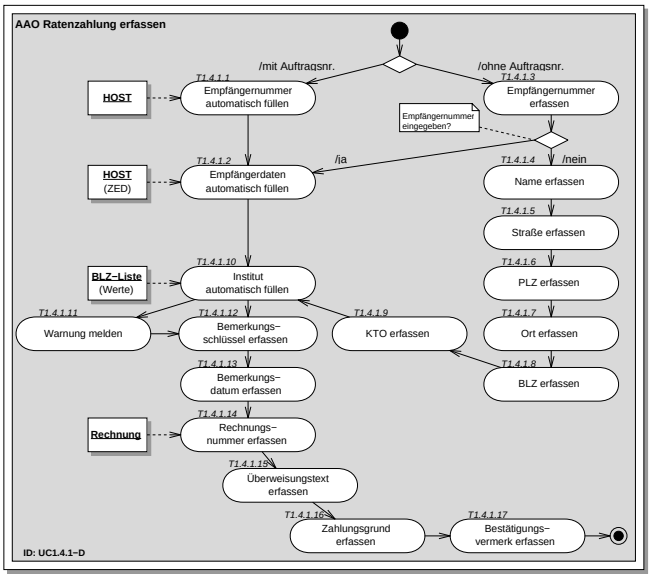

Abbildung 4.9.: *Aktivitätendiagramm in AAO - Ratenzahlung*

4.3.3. Dokumentation

Vorbetrachtung und Ziel

„Die Erfassung von Anforderungen in einem einzigen Pflichtenheft-Dokument ist ein Garant für das Scheitern des Projektes, sofern es sich nicht um ein Kleinstprojekt handelt.“[25] Zu dieser Ansicht von Versteegen kann man gelangen, wenn man das Pflichtenheft nicht als einzige Arbeitsgrundlage versteht und gleichzeitig auf vernetzte Dokumentationsstrukturen setzt. Bei der praktischen Umsetzung sind folgende Eigenschaften zu berücksichtigen:

- Es müssen sehr unterschiedliche Dokumente (Texte, Diagramme, Handbücher, Präsentationen, Skizzen, etc.) und Medien (E-Mails, Protokolle, Telefongespräche, etc.) einbezogen werden.
- Den einzelnen Anforderungsdokumenten fehlt eine Struktur, die den große Zusammenhang dargestellt kann und die Nachvollziehbarkeit sicherstellt.

Die Herausforderung besteht darin, nicht nur die vom Kunden formulierten Anforderungen zu dokumentieren, sondern auch die eigenen Erkenntnisse aufzubereiten. Dies kann durch textuelle Beschreibungen, Notizen oder Fachbeispiele erfolgen, aber auch durch formale Mittel wie Modell- oder Layout-Richtlinien, Strukturvorgaben, etc. unterstützt werden.

[25] Vgl. [VSH01] S.27 (Zitat)

Konzept: Spezielle Anforderungsdokumente

Die hier beschriebene Anforderungsentwicklung basiert auf der Idee, eine konsequente Ableitung und Transformation der Teilergebnisse (Geschäftsprozess, Fachmodul, BUC, UC, AD) vorzunehmen. Ein wesentlicher Bestandteil des Prozessmodells bildet dabei die *Dokumentationskette*. Die in Abbildung 4.11 dargestellten Anforderungsdokumente (Diagramme, textuelle Beschreibungen, Maskenentwürfe, etc.) werden direkt mit anderen Elementen verknüpft. Die Anforderungsdokumentation bezieht sich daher sowohl auf Inhalt als auch auf die Nachvollziehbarkeit (Traceability) der erfassten Anforderungsdokumente.

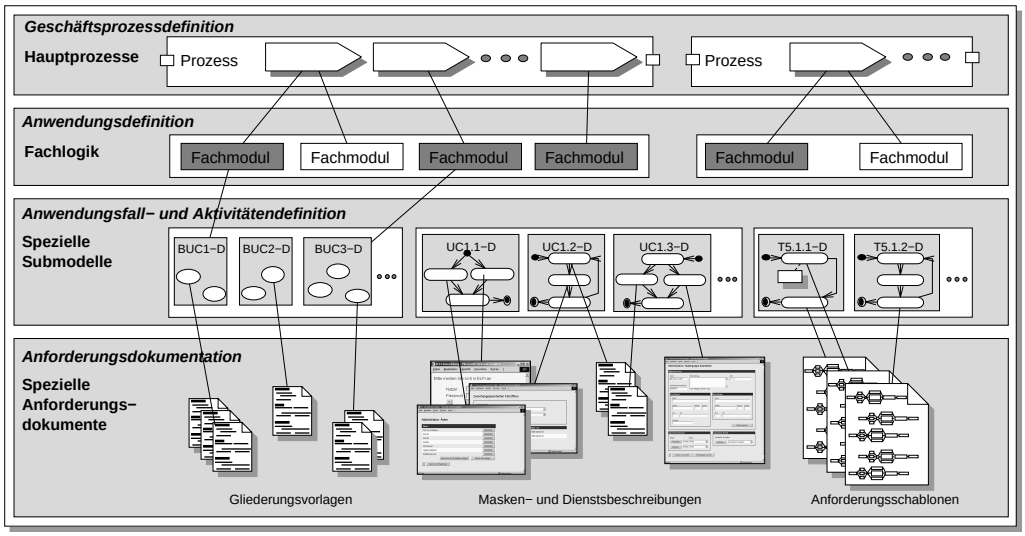

Abbildung 4.10.: *Beziehungskonzept von Anforderungsdokumenten*

Die unterschiedliche Ausrichtung der Modelle erfordert angepasste Mittel der Dokumentation. Auf sehr hoher Fachebene (Fachmodul, BUC) werden hauptsächlich Grundlagen und Verweise dokumentiert, während bei Submodellen mit zunehmender Präzision (UC, AD, Sub-AD) exakte und eindeutige Aussagen im Vordergrund stehen. Für BUCs und UCs eignen sich besonders Gliederungsvorlagen [26] für eine textuelle Kurzbeschreibung der Diagramme. Eine Vorlage sollte mindestens folgende Punkte enthalten:

- **Kurzbeschreibung.** Verbale Beschreibung für einen zusammenfassenden Überblick der Teilergebnisse. „Dieser Use Case dient der/dem ...“
- **Vorbedingungen.** Verbale Beschreibung der notwendigen Voraussetzungen für die Aktivierung des BUC/UC. Sind diese Vorbedingungen nicht erfüllt, ist der Anwendungsfall nicht ausführbar.
- **Basisablauf.** Es folgt eine stichpunktartige Ablaufbeschreibung der Aktivitäten, die in den ADs hinterlegt sind.

[26] Vgl. Kapitel 2.5.1

- **Ergebnisse und Nachbedingungen.** Hier soll der fachliche Sinn des Anwendungsfalls dokumentiert werden und eine Einordnung in das Geschäftsmodell erfolgen.
- **Alternativabläufe.** Es werden alternative Abläufe im Falle von Ausnahmebedingungen beschrieben.

Gliederungsvorlagen sind nicht beschränkt auf BUCs und UCs, sondern prinzipiell einsetzbar bei allen grafischen Beschreibungsmitteln. Bei sehr fein gegliederten Modellen wie ADs und Sub-ADs sollten jedoch zunehmend konkrete Beschreibungen und Entwurfsvorlagen mit den Modellelementen verknüpft werden:

- **Masken- und Dienstbeschreibungen.** Explorative Prototypen[27] und spezielle Funktionsbeschreibungen werden mit den Elementen verknüpft.
- **Anforderungsschablonen.** Es erfolgt eine exakte Definition und Verknüpfung von programmierbaren *Anforderungen* an fachliche Modellelemente. Die entwickelten Anforderungssätze werden direkt den Aktivitäten zugeordnet und beschreiben auf höchster Präzisionsebene das Verhalten des Systems. [28]

Konzept: Nomenklatur für eine Dokumentationskette

In den bisher vorgestellten Beispieldiagrammen, wurde im Sinne der Dokumentationskette ein ID-Konzept[29] (Traceability-Ansatz) verwendet, welches im Folgenden abstrakt beschrieben wird:

- Das modellierte System aggregiert aus den Fachmodulen Business Use Cases (Komponenten); jedes BUC bekommt eine Nummer: `BUCx: <Name des BUC>`
- In jedem BUC gibt es Anwendungsfälle; jeder Anwendungsfall bekommt eine Nummer: `UCx.y: <Name des UC>`.
- Zu einem UC kann es ein AD geben; jede Transition(Aktivität) bekommt eine Nummer: `Tx.y.z: <Name der Transition>`
- In einem BUC kann es wiederum BUCs geben; die Nomenklatur ist analog zum ersten Punkt: `BUC x` enthält `BUC x.a`; `BUC x.a` enthält `n` UCs: `UCx.a.y1..-UCx.a.yn`. Dies ist übertragbar auf Anwendungsfälle und Transitionen.
- Die (Sub-)Diagramme werden nach gleichem Schema benannt und durch das Suffix „-D“ unterschieden.

  Abstrakt: `X-D(X1(X1-D(..))..Xn(Xn-D(..)))`, wobei `X` $\in$ {`BUC,UC,T`}.[30]

  Beispiel: `BUC-D(BUC1(BUC1-D(..))..BUC25(BUC25-D(UC25.1(..)..(..)))`[31]

[27] Vgl. [Oes98] S.146, Anforderungsanalyse durch entwurfsartige Bildschirmmasken
[28] Vgl. Kapitel 2.5.2
[29] Eine ID identifiziert ein Element eindeutig und fungiert als Schlüssel für Zugriff und Verwaltung.
[30] Hinweis: T-D (Transitionsdiagramm) ist ein AD (Aktivitätsdiagramm).
[31] Vgl. Abbildungen 4.6, 4.7, 4.8, 4.9, 4.11 für weitere Beispiele.

Konzept: abstrakte Anforderungen

SWEBOK definiert High Level Requirements als fachorientierte Anforderungen ohne Nachweiselement im Endergebnis. Im Prozessmodell wird dafür die *abstrakte Anforderung* eingeführt. Diese beschreibt alle *Sammelanforderungen* (z.B. Modelle, Teilanforderungen). Abstrakte Anforderungen sind explizit zu dokumentieren, da sie nicht direkt programmierbar bzw. testbar sind.

Konzept: Metadaten von Anforderungen dokumentieren

In Anlehnung an SWEBOK sollten folgende Anforderungsattribute zu jeder (abstrakten) Anforderung und jedem Anforderungsdokument hinterlegt werden:

- Änderungshistorie/Nachweisbarkeit (Autor, Rolle, Grund, etc.)
- Scope-Dokumentation (Priorität, Dringlichkeit, Iteration, Verbindlichkeit)
- Anforderungsquellen (Ansprechpartner, E-Mail, Quelldokument, etc.)
- offene/nicht verstandene Punkte (Unklarheiten)

Ergebnisse

Nach Dokumentation der Elemente sollten folgende Ergebnisse erreicht werden:

- inhaltliche Aufbau- bzw. Ablaufbeschreibung von UCs und ADs,
- konsistente Dokumentationskette,
- Sammlung von natürlichsprachlichen Anforderungen (durch Anforderungsschablonen) sowie
- durch Metadaten angereicherte Dokumente und Anforderungen.

Empfehlungen und Beispiel

- Bei entsprechender Gründlichkeit werden sehr viele Anforderungen aufgestellt, die sich nur schwer verwalten lassen. Daher empfiehlt es sich, das ID-Konzept bis auf das kleinste Element (Anforderungssatz) anzuwenden, damit ein Ordnungsprinzip verfügbar ist.
- Der Anforderungskatalog ist das dokumentierte und explizierte Gegenbild vom Wissensstand des Projektteams. Die Entwickler sollen nicht auf Basis der undokumentierten/impliziten Annahmen des Anforderungsingenieurs programmieren, sondern mit Hilfe der Fakten aus dem Anforderungskatalog arbeiten können.
- Abbildung 4.11 demonstriert auszugsweise die Dokumentation für den ELFI Anwendungfall „AAO Ratenzahlung erfassen“.

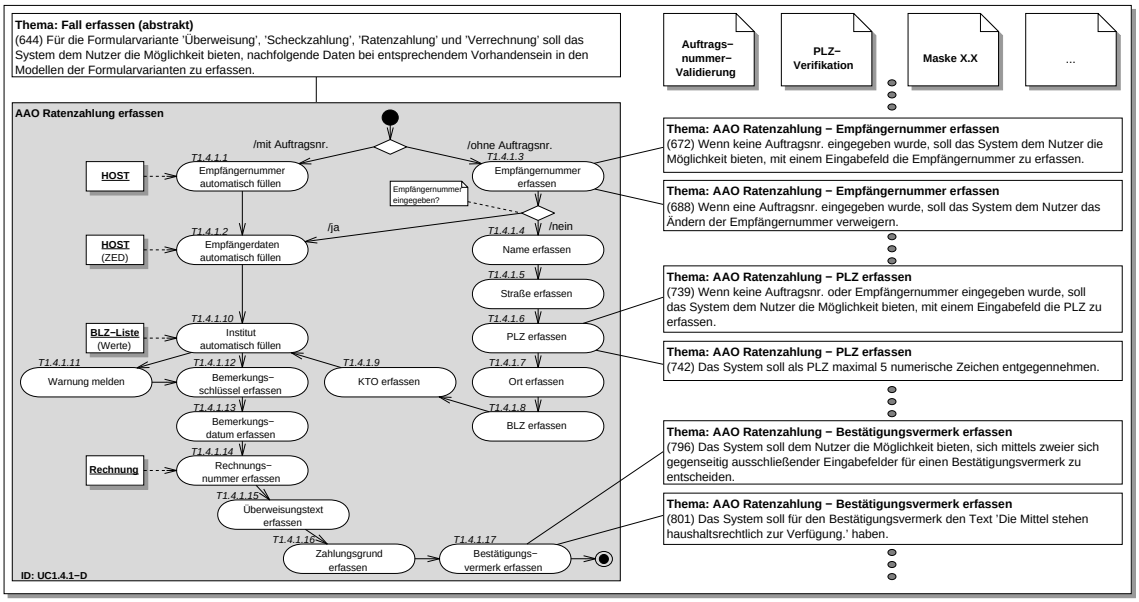

Abbildung 4.11.: *Dokumentationskonzepte am Beispiel*

4.3.4. Analyse

Vorbetrachtung und Ziel

Der Aufbau von speziellen Branchenwissen erfordert im Rahmen des Knowledge Engineering eine intensive Bearbeitungsphase der gesammelten Anforderungsdokumente. Ein Schwerpunkt liegt in der Organisation (Kontrolle und Übersicht) der Anforderungen. Ein transparentes Beziehungsgeflecht[32] zwischen Anforderungen kann daher das gesamte Domänenverständnis bestimmen. Analysieren heißt auch hinterfragen, denn eine frühzeitige Abstraktion der Anforderungen hilft, ein erweitertes Konzeptverständnis über die Problemstellung des Kunden aufzubauen. Im Verlauf dieser Arbeiten wird der Anforderungsingenieur feststellen, dass zwischen den Anforderungen zahlreiche Interdependenzen vorliegen, die schwer zu dokumentieren und zu verwalten sind. Die Analysetätigkeit bezieht sich daher auf die folgenden Punkte:

- strukturelles Gefüge der Anforderungen
- Definition von Required Constraints
- Vernetzung und Integration der Required Constraints
- vorbereitende Analyse in Hinblick auf Änderungswünsche und deren Auswirkungen

[32] Vgl. [Sch02] S.162

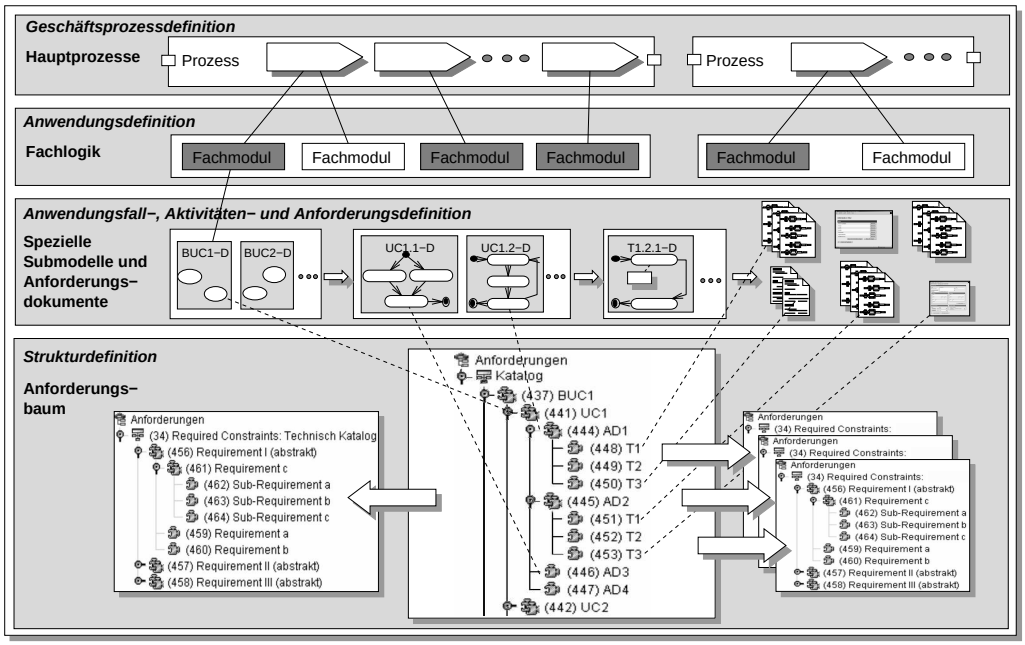

Abbildung 4.12.: *Struktur- und Analysekonzept durch Anforderungsbäume*

Konzept: Anforderungsbaum

Das Beziehungsgeflecht im Anforderungskatalog besteht zwischen den Teilergebnissen der Anforderungsentwicklung. Zu analysieren sind daher Abhängigkeiten zwischen Geschäftsprozessen, Fachmodulen, Anwendungsfällen, Aktivitäten (Transitionen), funktionalen Anforderungen (Required Features) und nicht-funktionalen Anforderungen (Required Constraints). Es bietet sich an, das strukturelle Gefüge in einen *Anforderungsbaum* zu übertragen. Mit Hilfe eines hierarchischen Datenstrukturtyps lässt sich das bisher erarbeitete Beziehungsgeflecht der Anforderungen 1:1 abbilden.

Ausgangspunkt für die baumartige Strukturierung bilden die Required Features. In Abbildung 4.12 wird abstrakt beschrieben, wie eine Konstruktion des Anforderungsbaumes zu realisieren ist.[33].

Konzept: Required Constraints

Die erste Konstruktion des Anforderungsbaumes, umfasst lediglich Required Features. Die geforderten Nebenbedingungen[34] (engl.: Required Constraints) blieben zu nächst unberücksichtigt, da ein strukturelles Gefüge für eine effiziente Zuordnung

[33]Im Anforderungsbaum werden alle Elemente aus Diagrammen einbezogen. Ein BUC-D enthält beispielsweise eine Anzahl von BUCs, die wiederum UCs enthalten usw. Ein Diagramm wird damit implizit im strukturellen Gefüge des Baumes abgebildet. Da ein Diagramm jedoch einen hohen Verständnisbeitrag bietet, wird durch das Metadaten-Prinzip die grafische Repräsentation hinterlegt. (beschrieben in Abschnitt 4.3.3)

[34]Vgl. [R+02] S.266

fehlte. Es folgt eine Tiefenanalyse der funktionalen Anforderungen, um zu klären, welche Required Constraints der Kunde wünscht. Sollte der Kunde bereits entsprechende Anforderungen formuliert haben, erfolgt eine Zuordnung.

Required Constraints können unabhängig von ihrem Typ, nach dem Prinzip von Required Features dokumentiert (Anforderungsschablone) und strukturiert (Anforderungsbaum) werden. Wie in Abbildung 4.12 ersichtlich, ergeben sich aus den Typen (Technisch, GUI, Schnittstelle, etc.)[35] mehrere Teilbäume.

Konzept: Anforderungsbeziehungen

Für die Verknüpfung[36] von Required Features und Required Constraints werden in diesem Prozessmodell zwei Gruppen vorgeschlagen:

1. Beziehung von Anforderungen gleichen Typs
    - hierarchische Ist-Teil-Von-Beziehung
    - gerichtete Ist-Abhängig-Von-Beziehung
2. Beziehung von Anforderungen unterschiedlichen Typs
    - gerichtete Ist-Abhängig-Von-Beziehung

Beziehungskonstrukte von Anforderungen gleichen Typs (erster Fall) eignen sich für die Abbildung von inhaltlichen Abhängigkeiten, die implizit durch die Baumstruktur entstehen (hierarchisch) oder explizit modelliert werden (gerichtet). Die Verknüpfung von Anforderungen unterschiedlichen Typs ist hierarchisch nicht möglich, weil der Anforderungsbaum nur Elemente des gleichen Typs enthält. Daher ist lediglich die gerichtete Beziehung von Elementen möglich, die eine „spinnennetzartige Verwebung“ der Teilbäume entstehen lässt.

In der Praxis kommt es zu einem weiteren Beziehungstyp: der Wiederverwendung von Anforderungen. Dies wird hier als Erweiterung behandelt, da der Ansatz sich auf den hierarchischen Beziehungstyp reduzieren lässt. Eine Wiederverwendung betrifft die mehrfache Platzierung identischer „Äste“ in der Anforderungshierarchie. Eine Änderung oder der Wegfall dieser Anforderungen würde eine Kettenreaktion auslösen, da jede betroffene Stelle zu überarbeiten ist. Ein Lösungsvorschlag ist die Erstellung einer konkreten Anforderungshierarchie und der Einsatz von Link-Elementen, d.h. die Verwendung von abstrakten Anforderungen in Form von Zeigern.

Hinweis: Zirkuläre Abhängigkeiten sind zu vermeiden, da im Prozessmodell die widerspruchsfreie Beschreibung der Anforderung angestrebt wird.

[35] Siehe „Erweitertes Verständnisprinzip“ aus Abschnitt 4.2.3.
[36] Im Folgenden werden die Begriffe Verknüpfung, Beziehung und Vernetzung synonym verwendet.

Ergebnisse

Nach Analyse der Elemente sollten folgende Ergebnisse erreicht werden:

- Anforderungsbaum der Required Features,
- Teilbäume der Required Constraints,
- modellierte Beziehungen zwischen Anforderungen gleichen und unterschiedlichen Typs,
- identifizierte Zusammenhänge zwischen Anforderungen sowie
- erweiterte Dokumentationskette durch die neuen Beziehungskonstrukte.

Empfehlungen und Beispiel

- Im Analyseprozess ergibt sich der Vorteil, dass sich alle Beteiligten intensiv mit der Problemstellung auseinandersetzen müssen. Zusammenhänge aber auch Unklarheiten, werden dadurch erst erkennbar.
- Required Constraints, die das gesamte Gesamtsystem betreffen, sollten mit Elementen auf möglichst hoher fachlicher Ebene (BUCs, UCs) verknüpft werden.
- Sind Anforderungen nicht erfüllbar, weil sie entweder technisch unmöglich, in der Umsetzung zu teuer oder sich aufgrund konträrer Vorgaben gegenseitig ausschließen, sollten diese als *Konfliktanforderungen* gekennzeichnet und zur Klärung vorgemerkt werden.[37]
- Anhand der Abhängigkeiten einer Anforderung (gerichtet/referenziert) lässt sich die Kopplung ableiten. Änderungen an stark gekoppelten Anforderungen sind besonders kritisch zu prüfen.

Bei dem Projekt ELFI konnten die Required Features und Required Constraints in die hierarchische Struktur des Anforderungsbaumes nach dem hier beschriebenen Schema übertragen werden. Im Folgenden werden die Beziehungstypen am Beispiel dargestellt:

- **Ist-Teil-Von-Beziehung.** „*Formularbearbeitung durchführen - Fall erfassen - AAO vorbereiten - Ratenzahlung - PLZ erfassen - Format PLZ - ...*“
- **Ist-Abhängig-Von-Beziehung (gleicher Typ).** „*Das System soll als PLZ maximal 5 numerische Zeichen entgegennehmen.*“ verknüpft mit:

  „*Das System soll die Möglichkeit bieten, die PLZ auf ihre Plausibilität zu prüfen.*“

[37] In Anlehnung an [CS01] S.12

- **Ist-Abhängig-Von-Beziehung (unterschiedlicher Typ).** „*Das System muss die Möglichkeit bieten, neue Fälle durch Masken zu erfassen.*“ (Required Feature) verknüpft mit:

  „*In allen Masken muss das System die Richtlinien des Styleguide einhalten.*“ (Required Constraint: GUI)

- **Wiederverwendung.** Mehrfaches Vorkommen von Formatvorgaben für Kennzahlfelder (*Empfängernummer, Auftragsnummer*, etc.) in den Masken. AAO Ratenzahlung, AAO Überweisung und AAO Scheckzahlung: „*Das System soll die Empfängernummer mit exakten Längen im Format n(5)/n(1) anzeigen.*“

- **Konfliktanforderung.** Kundenanforderungen in Konflikt mit dem Styleguide (alle numerischen Angaben rechtsbündig): „*Das System soll in der Maske AAO Ratenzahlung den Betrag linksbündig formatieren.*“

4.3.5. Validierung

Vorbetrachtung und Ziel

Die Validierung von Anforderungen hat das Ziel, den Anforderungskatalog zu untersuchen, um so sicherzustellen, dass das beschriebene System die ursprüngliche Intention des AG adäquat wiedergibt. Im Folgenden wird zwischen zwei Ausprägungen differenziert:[38]

- **Verifikation.** Ziel ist die Überprüfung der Artefakte und Anforderungen nach formell festgelegten Richtlinien.

- **Validierung.** Das Ziel dieser Aufgabe ist die Bestätigung, dass die im Anforderungskatalog dokumentierten Anforderungen den Kundenwünschen entsprechen.

Das Gegenprüfen kann nur mit und durch den Kunden erfolgen. Durch die Validierung wird die vom Anforderungsingenieur konstruierte Interpretation mit den Wünschen des AG konfrontiert. Die Validierung ist eine äußerst wichtige Aufgabe, da nur geprüfte Anforderungen eine (rechts-)verbindliche Bedeutung im weiteren Verlauf des Projektes haben.

Konzept: Anforderungsverständigung vorbereiten

In den meisten Fällen werden die ermittelten Anforderungen in Zusammenarbeit mit dem Kunden validiert. Der Vorgang kann permanent erfolgen, bedarf jedoch einiger organisatorischer Vorbereitungen und der prinzipiellen Bereitschaft des AG. Folgende Punkte sollten im Zuge der Anforderungsverständigung vorbereitet werden:

[38] Vgl. [FFF03]

- Beschreibung des REP und der Konzepte (Anforderungskatalog, Anforderungsbaum, Dokumentationskette, etc.)
- Workshop „Einführung in UML“ (UseCase- und Aktivitätendiagramme)
- Workshop „Sprachdefekte“

Der AN formuliert vorbereitend *Akzeptanzkriterien* für Anforderungen, um eine kritische Prüfung seiner Ergebnisse durchführen zu können. Die Akzeptanzkriterien sind abhängig von der Projektgröße, den Vorgaben des Kunden und der Qualitätssicherung des verwendeten Vorgehensmodells.

Konzept: Fragenkatalog/Checklisten

Mittels eines systematischen Vorgehens können Fehler schneller freigelegt werden. Ein Lösungsvorschlag ist der Einsatz von vorgefertigten Checklisten[39], um irrtümliche Annahmen, unklare Begriffe und Abweichungen zu identifizieren.[40] Die Checkliste sollte wenigstens folgende Punkte berücksichtigen:

- **Verifikation.**
  - Konsistenz und Verständlichkeit der Artefakte
  - Vollständigkeit der Metadaten
  - Formalisierungsgrad der Anforderungsdokumente (Integritätstests)
- **Validierung.**
  - Umfang der Anforderungen (Anforderung fehlt? Anforderung überflüssig?)
  - fachliche Vollständigkeit und Korrektheit der Anforderungen
  - Programmier- und Testbarkeit der Anforderungen

Die eigentliche Validierung erfolgt durch Inspektionen bzw. Reviews und sollte durch Akzeptanztests (Prototypen, Maskenentwürfe) unterstützt werden. Die Anforderungen an ein System werden nicht nur einmal, sondern zu verschiedenen Zeitpunkten des REP validiert. Die Validierung kann nach Bearbeitung einzelner Arbeitspakete (z.B. Fachmodul, BUC, etc.) oder zu festgelegten Zeitpunkten erfolgen.

[39] Vgl. [Sch02] S.177, Schienmann konstruiert eine umfassende Checkliste der Eigenschaften von Anforderungen(korrekt, vollständig, konsistent, etc.) und Anforderungsdokumenten(strukturiert, aktuell, relevant, etc.).

[40] Entnommen aus [R+02] S. 225, Frühauf und Sandmayr

Ergebnisse

- Die formale Darstellung und Struktur der Anforderungen ist konsistent.
- Die kritischen Teile, die das gewünschte Systemverhalten abbilden, wurden fehlerfrei identifiziert und im Anforderungskatalog dokumentiert.
- Unvollständige und defektbehaftete Anforderungen wurden aufgedeckt und in Zusammenarbeit mit dem AG neu formuliert.

Empfehlung und Beispiel

- Im Rahmen der Anforderungsverständigung muss unbedingt sichergestellt werden, dass der AG in der Lage ist, die Ergebnisse (UML Diagramme, Anforderungsschablonen) richtig zu interpretieren.
- Durch Präsentation von „lebenden Prototypen“ wird das beste Feedback erzielt. Dabei werden Unklarheiten des AG („sieht und bewertet“) und des AN („erstellt und hinterfragt“) schnell erkannt.
- Es sollten mit dem Kunden mindestens drei Reviewrunden geplant werden. Erst wenn der AG etwas sieht, weiß er, was er *nicht* will. (Ausschlussprinzip: Vorstellen - Ändern - Zustimmen)

Die Arbeit mit Checklisten in Kombination mit Erfahrungswerten früherer Projekte, soll den Anforderungsingenieur darin unterstützen, die vielfältigen „Warnsignale“ gezielt zu erkennen. Im Projekt ELFI konnten beispielhaft folgende Erfahrungswerte berücksichtigt werden:

- „*Für das Web-Frontend soll das System Netscape-Browser ab V4.7x unterstützen.*“ Erklärung: Pauschale Vorgaben („*ab*“) und ungenaue Versionsangaben („*4.7x*“) multiplizieren den Testaufwand und erfüllen nur selten einen funktionalen Mehrwert.
- „*Dem Anwender soll das System die Möglichkeit bieten, bei Formulardruck zwischen mehreren Druckern zu wählen.*“ Erklärung: Das Kriterium „Testbarkeit“ ist nicht erfüllt, da kein spezieller Druckertyp („mehrere Drucker“) angegeben wurde.

4.3.6. Task Workflow

In den Abbildungen 4.13 und 4.14 wird unter Verwendung der hier eingeführten Begriffe, der vollständige Workflow der Tasks zusammenfassend aufbereitet. Die Abbildungen sind so zu interpretieren, dass trotz der dabei verwendeten sequentiellen Ablaufdokumentation ein mehrfach zu wiederholendes Durchlaufen der Tasks erforderlich ist. Im Workflow erfolgt nach jeder Task eine Validierung. Dies soll die permanente Bestätigung und die Einbeziehung von Feedback durch den AG symbolisieren (gemäß Abbildung 4.4).

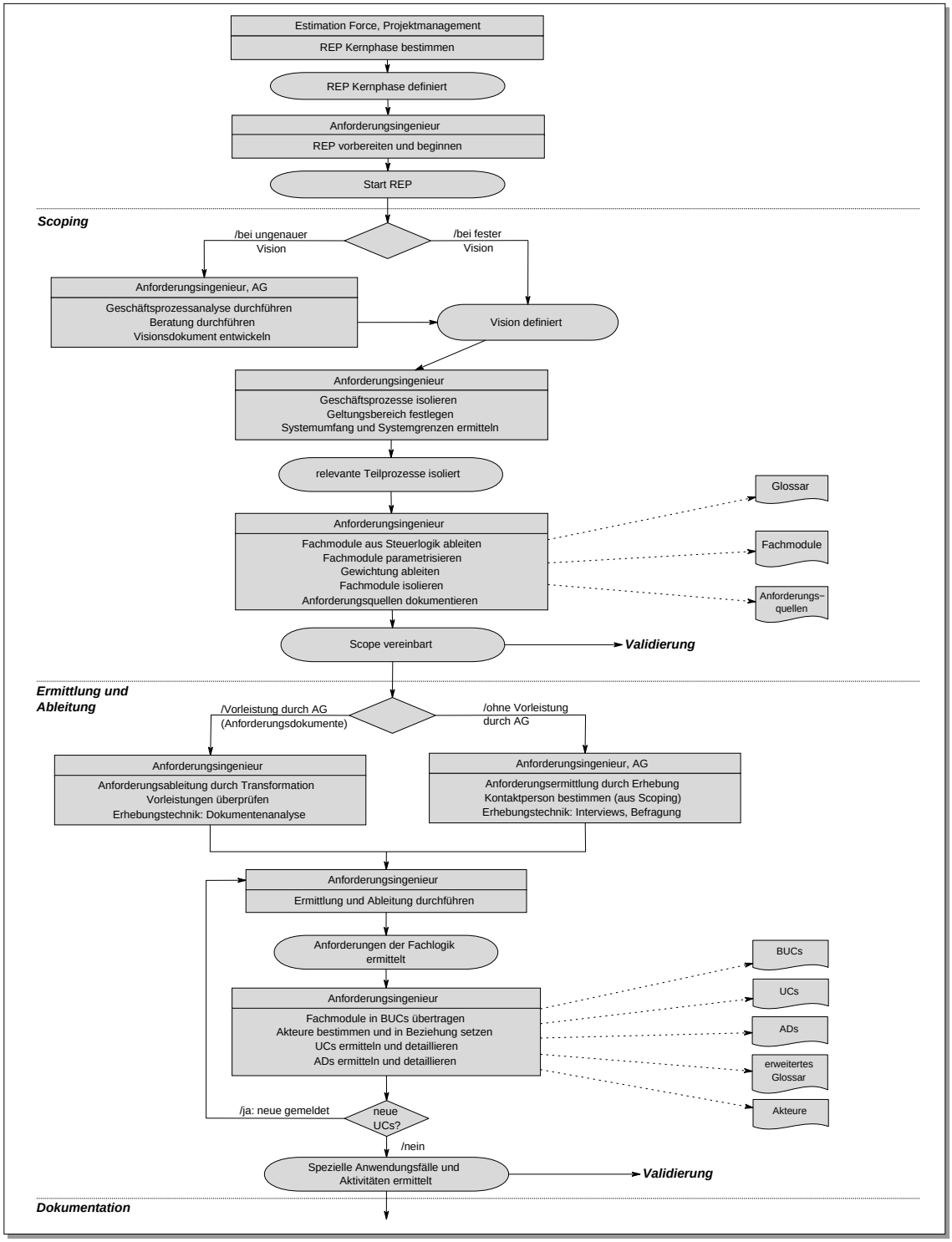

Abbildung 4.13.: *Ablaufbeschreibung des REP 1/2*

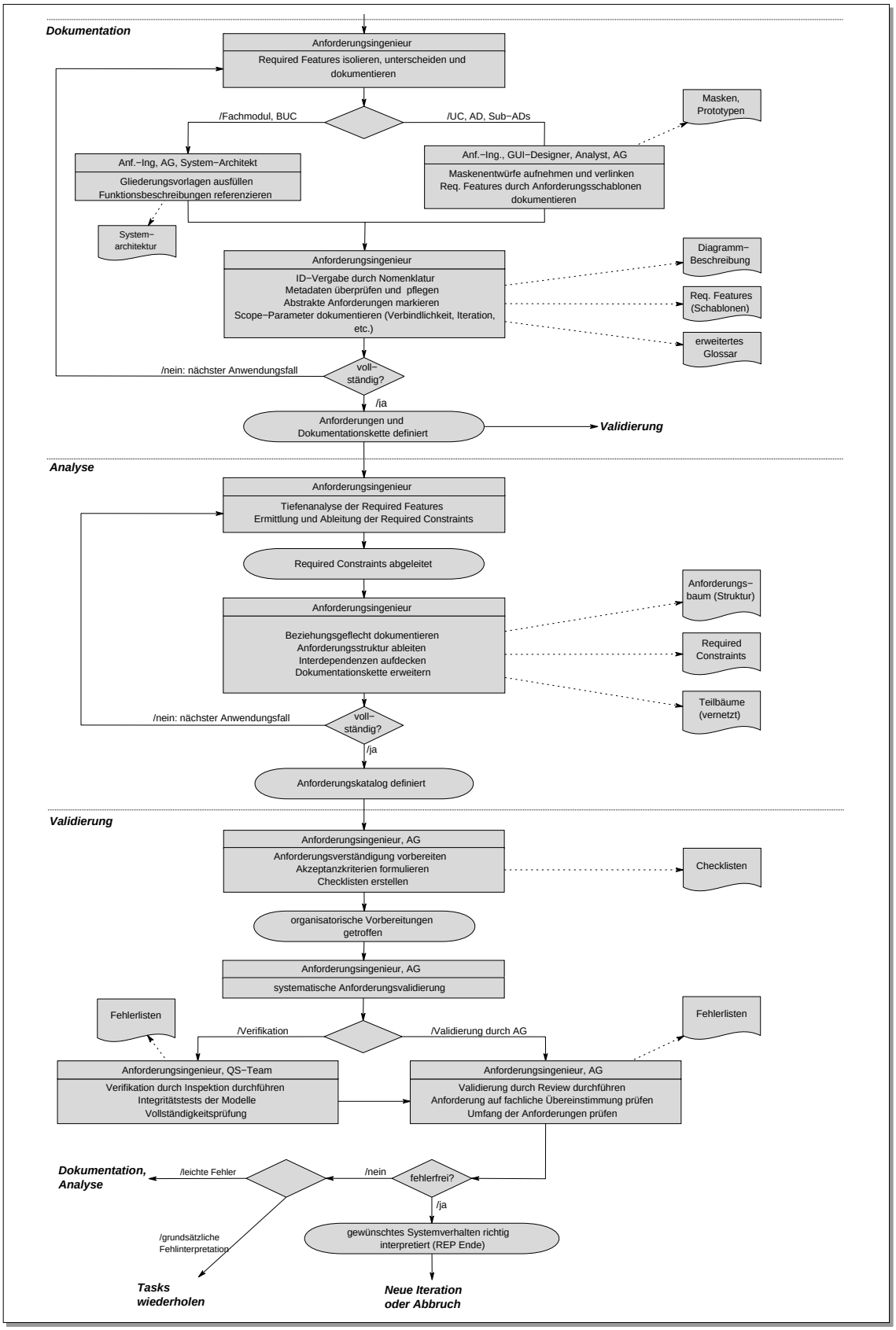

Abbildung 4.14.: *Ablaufbeschreibung des REP 2/2*

4.4. Iterative Entwicklung

Auf die Iteration als ein Konzept der Fehlerreduktion, wurde in dem Prozessmodell mehrfach eingegangen. Die schleifenartige Wiederholung der Tasks dient dem Ziel, den Informationsverlust, der bei der Modellbildung unweigerlich auftritt, durch einen stetigen Erkenntniszuwachs über die Anwendungsdomäne auszugleichen (Knowledge Engineering). Mit Hilfe einer Iteration verringert sich schrittweise der Abstand zwischen Soll- (Kundenwunsch) und Ist-Zustand (Anforderungskatalog).

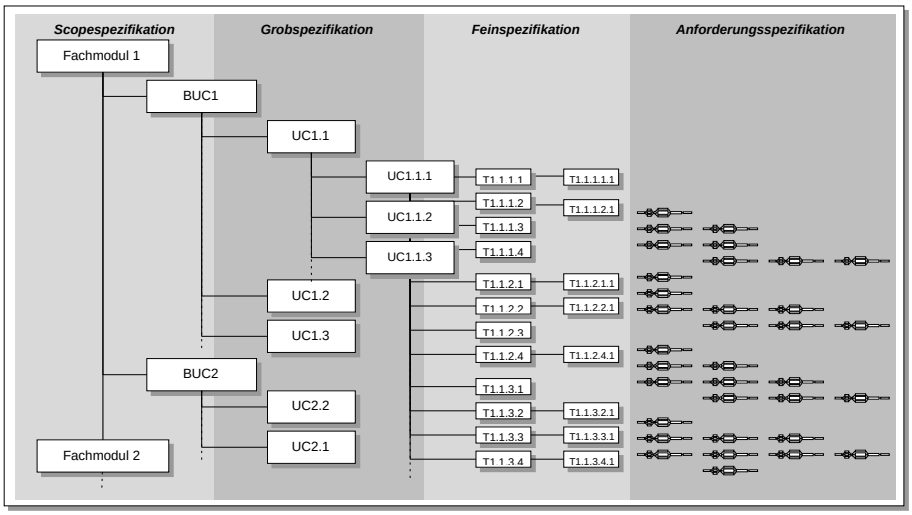

Abbildung 4.15.: *Spezifikation durch Wiederholung*

Das Projektmanagement kann bei der Planung und Überwachung der Kernphase durch die Definition von Meilensteinen unterstützt werden. Die Abbildung 4.15 beschreibt eine Möglichkeit, wie man aus dem Anforderungsbaum vier Meilensteine in Form von Spezifikationen (Scope-, Grob-, Fein- und Anforderungsspezifikation) ableiten kann. Mit jeder Stufe nimmt der Detaillierungsgrad des Anforderungskataloges zu. Die Abbildung muss jedoch kritisch bewertet werden, da eine Iteration sowohl in horizontaler Richtung (Granularisierung), als auch in vertikaler Richtung (Ausweitung/Verbreiterung) von Bedeutung ist.

Abbruch der Anforderungsentwicklung

Nach einer Iteration wird anhand von Abbruchkriterien geprüft, ob die Kernphase beendet werden kann/muss oder ein neuer Durchlauf angebracht ist.[41] Die Iteration hat jedoch in der Praxis nur eine geringe Bedeutung und wird nur selten nach dem theoretischen Vorbild umgesetzt. Daher sollen folgende Punkte als Richtlinie dienen und besonders kritisch hinterfragt werden:

[41] Vgl. [CS01] S.16, requirements engineering process termination

- Eine Wiederholung ist so lange richtig, bis der AG sich durch die Interpretation des AN bestätigt fühlt und weder externe Faktoren (Kundendruck) noch interne Faktoren (Ressourcenmangel durch Zeit und Kosten) einen frühzeitigen Abbruch fordern.
- Die Fehlerfreiheit ist ein unwahrscheinlicher Zustand. Deswegen sollten sich AG und AN auf ein „akzeptables Risiko“ einigen.
- Es sollte kritisch hinterfragt werden, ob sich aus den erfassten Erkenntnissen und Anforderungen die nächsten Schritte im Projekt ableiten lassen.

Die Anforderungen werden in Form eines Pflichtenheftes manifestiert und zum Vertragsgegenstand erklärt. Die Vereinbarung beendet die Kernphase des REP und es erfolgt der Übergang zum Change Request Management (CR-M). Da Kunden häufig zögerlich auf einen verbindlichen Vertrag reagieren, empfiehlt es sich, dem Kunden zu erklären, dass nur so ein effektives CR-M zu etablieren ist.

4.5. Management von Anforderungen

Das Anforderungsmanagement umfasst alle Aktivitäten, die die Richtigkeit, Gültigkeit und Konsistenz des Anforderungskataloges (inkl. der verbundenen Dokumente) im weiteren Projektverlauf sicherstellen.[42] Schienmann und Wiegers beschreiben folgende *Aufgaben des Anforderungsmanagement*:[43]

- Verhinderung eines unkontrollierten Projektwachstums (Ressourcensteuerung)
- Bearbeitung von Änderungsanträgen
- Abhängigkeitsbestimmung zwischen Anforderungsdokumenten und den Ergebnissen aus Entwurf und Implementierung
- Versionskontrolle der Anforderungen
- Statusverfolgung der Anforderungen

Gemäß der Abbildung 4.4 (Konstruktion des Prozessmodells) erfolgt die Einordnung der Aufgaben als ein nebenläufiger Vorgang, der im Unterschied zur REP-Kernphase als Querschnittsprozess fungiert. Hier soll die Problematik in den zwei Abschnitten „Change Requests“ und „Werkzeugbasierte Unterstützung“ aufgegriffen werden.

[42] Vgl. [Wie03]
[43] Vgl. [Sch02] S.21, Vgl. [Wie03]

4.5.1. Change Requests

Vorbetrachtung und Ziel

Nur auf Basis eines vorher vereinbarten Vertragsgegenstandes (Pflichtenheft), ist ein sinnvoller Änderungsprozess und die Verwaltung (Annahme, Ablehnung) von Änderungsaufträgen[44] (engl.: change request (CR)) zu realisieren.[45] Die Änderungen stehen in keinem zeitlichen Zusammenhang mit den Tasks der REP-Kernphase. Da Änderungsaufträge durch den Kunden im gesamten Projektverlauf möglich sind, ist eine parallele Bearbeitung erforderlich.

Die Behandlung von Änderungen ist Gegenstand vieler Vorgehensmodelle und wird in den etablierten Ansätzen bereits erschöpfend thematisiert. Einen sehr umfassenden Beitrag bieten die in Abschnitt 2.3 vorgestellten Vorgehensmodelle RUP und V-Modell 97. Die Änderungsmanagement-Beschreibungen, auf die hier lediglich verwiesen werden soll, umfassen neben Ablaufbeschreibungen auch Vorlagen, organisatorische Empfehlungskonzepte und umfangreiche Rollenmodelle.[46] Unabhängig von den speziellen VGM-Ansätzen ergeben sich bei der Bearbeitung folgende Punkte:

- Eintaktung der Änderungen in den Anforderungskatalog
- Untersuchung der potentiellen Abhängigkeiten
- Restrukturierung der betroffenen Modelle unter Wahrung der Integrität
- Dokumentation der Änderung

Konzept: Change Control Process

Die Möglichkeit den Projektbeteiligten Änderungen am vereinbarten Vertragsgegenstand und damit auch am Projektergebnis vornehmen zu lassen, wird durch die Einführung eines organisatorischen Ablaufkonzeptes, dem *Change Control Process* (CCP), realisiert. Zur Konstruktion werden folgende Vereinbarungen getroffen:

- Änderungswünsche werden in Form von Anträgen eingereicht.
- Alle Anträge müssen ein formelles Verfahren durchlaufen.
- Vor der offiziellen Antragsbestätigung dürfen keine Änderungswünsche übernommen werden.

[44] In der Literatur werden auch die Synonyme Änderungsanforderung oder Änderungswunsch verwendet.
[45] Vgl. [Wie03], [Ver00] S.100ff.
[46] Vgl. [RSC02] Disciplines→Configuration & Change Management→Concepts→Change Request Management, Vgl. [DW99] KM 3 „Änderungsmanagement“ (Konfigurationssteuerung)

- Über Anträge entscheidet ausschließlich das PM oder ein vor Projektbeginn zu bestimmendes *Steuerungskomitee* (SK).

Die Autoren Kotonya und Sommerville nennen drei Schritte, um Änderungswünsche durch Anträge in gültige Anforderungen zu überführen:[47]

1. **Problemanalyse.** Welche Probleme liegen dem Änderungswunsch zu Grunde?
2. **Änderungsanalyse und Kostenabschätzung.** Welche Subsysteme sind betroffen und welche Kosten würden durch die Änderungen entstehen?
3. **Durchführung der Änderung.** Akzeptiert der Kunde das Angebot, wird die Änderung zur Planung, Umsetzung und Kontrolle übergeben.

Die konzeptionelle Ablaufbeschreibung eines eigenen CCP wird in Abbildung 4.16 dargestellt. Die Unterteilung basiert auf dem hier eingeführten Begriffsverständnis und enthält eine erweiterte Darstellung der Prozessaktivitäten. Der Schritt „Durchführung und Änderung“ implementiert die im Prozessmodell beschriebene Anforderungsentwicklung. So wird ein einheitlicher RE-Ansatz auch bei Umsetzung eines Change Requests eingehalten. Der CCP ist abgeschlossen, wenn der Änderungswunsch entweder abgelehnt, in ein Folgeprojekt verschoben oder im aktuellen Durchlauf umgesetzt und konsistent in allen Artefakten nachgezogen wurde.

Empfehlungen

- Nach einer Studie des Fraunhofer-Institutes für experimentelles Software- Engineering steht als Erfolgsfaktor nach wie vor die Stabilität der Anforderungen an erster Stelle.[48]
- Veraltete oder inkonsistente Dokumente sind von zweifelhaftem Wert. Die konsequente Nachbesserung erfordert jedoch eine Konfigurationsverwaltung mit Änderungshistorie.
- Die wichtigste Aufgabe ist die fachgerechte Beurteilung der Auswirkungen einer Änderung. Größere Umstrukturierungsvorhaben (z.B. Verschiebung des Scopes, Richtungswechsel) fallen nicht in das Änderungsmanagement und deuten auf Analysefehler in der Kernphase hin.
- Fazit: Anforderungen bleiben nicht konstant, eine angemessene Reaktion und die störungsfreie Übernahme in den fortgeschrittenen Projektverlauf ist unumgänglich.

[47] Vgl. [KS98] S.124
[48] Vgl. [S+03] S.21

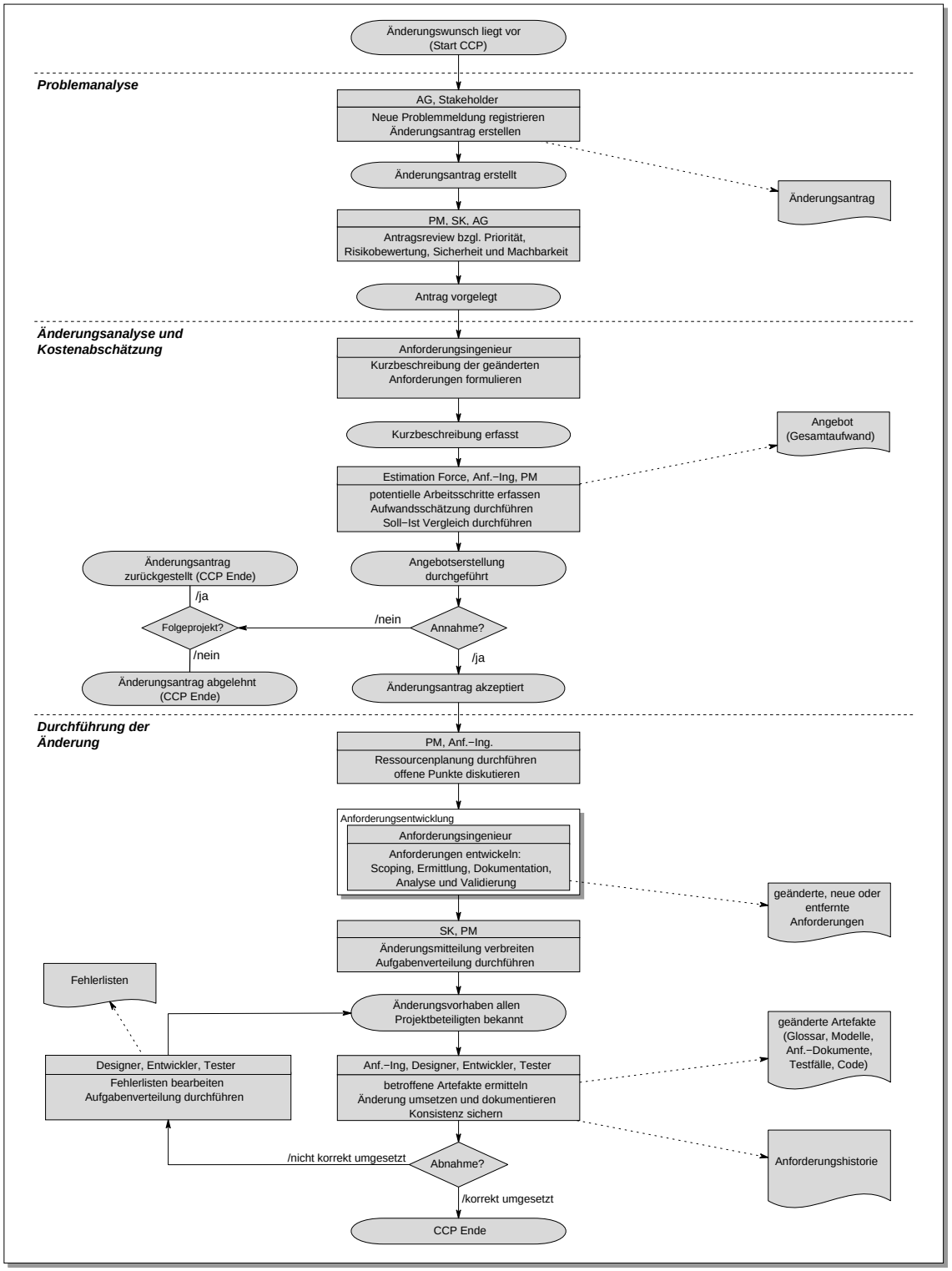

Abbildung 4.16.: *Ablaufbeschreibung des CCP*

4.5.2. Werkzeugbasierte Unterstützung

Vorbetrachtung und Ziel

Der zunehmende Wettbewerb in der IT-Branche hat dazu geführt, dass Produkte sich mehr durch eine spezielle Kundenorientierung, einen hohen Innovationsgrad und der Schnelligkeit bei der Einführung auszeichnen. Die Komplexität der Produkte hat sich außerdem stark erhöht, da die Kundenerwartungen durch die rasante Entwicklung der verfügbaren Technologien stetig mitwachsen. Es entsteht für die Anbieter der Bedarf, die eigenen Entwicklungsprozesse konkurrenzfähig umzugestalten.

Die werkzeugbasierte Unterstützung ist in den Bereichen Entwurf, Realisierung, Test und Projektmanagement längst eine Selbstverständlichkeit. Bislang kommen jedoch spezielle RE-Werkzeuge nur sporadisch zum Einsatz. Dies liegt insbesondere an dem fehlenden Prozessverständnis für das Requirements Engineering und der schwierigen Integration der Ergebnisse. Zwar können die Aufgaben des REP (z.B. Pflege der Dokumentationskette, Namensgebung, etc.) über einen sehr hohen manuellen Aufwand realisiert werden, ein passendes Werkzeug kann jedoch da ansetzen, wo der REP die Automatisierung zulässt:[49]

- Verwaltung und Organisation von Massendaten
- Dokumentation der Zusammenhänge zwischen Anforderungen und Anforderungsdokumenten
- schnelle Erstellung von Dokumenten/Berichten für die Beteiligten
- Generierung, Verifizierung und Export von Anforderungen
- konsistente Integration von Anforderungsänderungen
- Unterstützung kooperativer Entwicklung (Teamarbeit) und Versionierung

Die softwaregestützte Abbildung kann die Effizienz der Arbeiten deutlich erhöhen oder überhaupt erst ermöglichen. Zum Beispiel basiert der in diesem Prozessmodell beschriebene Anforderungskatalog auf der Annahme, dass ein strukturierter Speicher zur Verfügung steht. Das als Repository bezeichnete Medium bietet einen Mechanismus für die Definition, Speicherung und Verwaltung von Kundenwünschen sowie den Zugriff darauf.[50] Dieser Ansatz ist bestens geeignet für die Umsetzung in einem RE-Werkzeug.

ProjectMentor

Im Rahmen der „GFT Softwarefabrik“ wird die Verbesserung der Qualität interner Entwicklungsprozesse besonders im Zusammenhang mit einem professionellen

[49] Vgl. [Ver00] S.239 ff.,[Sch02] S.279 ff.
[50] Vgl. [McC93] S.161, Definition Repository

Anforderungsmanagement angestrebt. Der „ProjectMentor“ ist eine in der Entwicklung befindliche RE-Speziallösung der GFT Systems GmbH mit Schwerpunkt auf Strukturierung und Verwaltung einer großen Zahl von Anforderungen. In Abbildung 4.17 ist die Oberfläche des Prototypen zu sehen, die im linken Ausschnitt den ProjectMentor-Anforderungskatalog darstellt. Dieser konnte nach Implementierung der wichtigsten Funktion durch den ProjectMentor selbst modelliert werden. Das dafür nötige Repository wurde durch ein relationales Datenmodell realisiert. Mit Hilfe einer JDBC-Anbindung und eines Java-Clients kann der Anforderungsingenieur systematisch Anforderungen erfassen und dokumentieren.

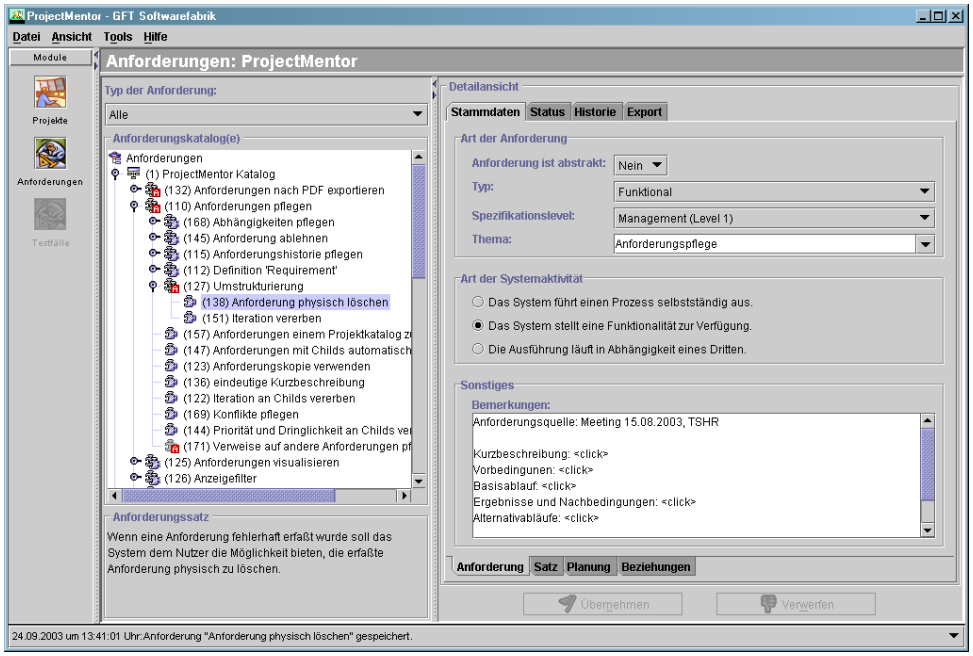

Abbildung 4.17.: *Anforderungskatalog im ProjectMentor*

Der ProjectMentor bildet Teile des hier beschriebenen Prozessmodells ab. Die bereits im Prototypen implementierten Konzepte werden in Tabelle 4.1 zusammengefasst:

Konzepte

Scoping	- Scope-Parametrisierung durch Attribute - Iterationsplanung/Anforderungsrückstellung
Ermittlung & Ableitung	- verteilte/standortunabhängige Aufnahme - Anforderungsmuster/Templates
Analyse	- frei editierbare Baumstruktur - Req. Features und Req. Constraints - Beziehungsmodellierung/Vernetzung

Dokumentation	- abstrakte Sammelanforderungen - natürlichsprachliche Anforderungen durch Schablonen - Masken- und Dienstbeschreibungen durch Links - Metadatenverwaltung und Änderungshistorie
Validierung	- Exportfunktion/selektive Abfrage

Tabelle 4.1.: *Funktionsumfang im ProjectMentor*

Um eine weitere Qualitätssteigerung zu erreichen, wäre es für zukünftige Versionen denkbar, die in Tabelle 4.2 genannten Punkte softwaretechnisch abzubilden.

Konzepte

Allgemein	- einheitliches Verzeichnis von Anforderungsquellen - Glossar-Generator - Testfall-Generator - Pflichtenheft-Generator - CR-Unterstützung (Abbildung des CCP)
Anforderungs-entwicklung	- Wiederverwendung durch Soft-Links - Gliederungsvorlagen für High-Level Anforderungen - semantische Analyse der Anforderungssätze - externe Schnittstellen zu Case-Tools - Thin Client für Kunden (Lesezugriff zur Validierung)

Tabelle 4.2.: *Denkbarer Funktionsumfang zukünftiger Versionen*

Empfehlungen

- Die Funktionalität eines Werkzeuges darf nicht überbewertet werden, denn nicht alle Aufgaben des REP lassen sich abbilden. Entscheidend ist die richtige Unterstützung an den automatisierbaren Stellen im Prozess.
- Eine spezielle Softwarelösung zu entwickeln ist ein aufwändiges, sowie risikoreiches Vorhaben, garantiert jedoch die optimale Berücksichtigung eigener Prozesskonzepte.
- Fällt die Entscheidung auf Standard-Werkzeuge, sollte vor Auswahl eine ausführliche Evaluierungsphase erfolgen, um sicherzustellen, dass eine optimale Prozessunterstützung gewährleistet wird.[51].

[51] Gute Ansätze und Kriterien für eine vergleichende Bewertung bietet z.B. Schienmann in [Sch02] S.287

4.6. Prozessimplementierung

Die Implementierung des Prozessmodells in die bestehende Ablauforganisation des IT-Dienstleisters[52] kann auch eine erfahrene Geschäftsführung vor große Herausforderungen stellen. Auf strategischer Ebene bedeutet die „Anpassung und Optimierung der RE-Aktivitäten“[53] zunächst die unternehmensweite Einführung und Etablierung des REP-Gedankens. Entscheidend ist neben dem festen Willen zur Veränderung, die uneingeschränkte Unterstützung durch die Unternehmensführung und die Bereitschaft, die vorliegende Ist-Situation kritisch zu bewerten.[54]

Der organisationale Wandel geht jedoch stets einher mit psychologischen Hindernissen, wodurch die Prozessimplementierung zusätzlich erschwert wird. Zu diesem Phänomen gibt es bereits verschiedene theoretische Ansätze (z.B. Lewin's Model of Change), deren gemeinsamer Nenner im Gutenberg'schen Sinne der Produktionsfaktor „Mensch“ ist. Die breite Akzeptanz durch die Mitarbeiter ist daher eine wichtige Voraussetzung, um Erneuerungen erfolgreich und ohne Reibungsverluste etablieren zu können. Die dafür nötige Einsicht muss jedoch mit klaren Argumenten erarbeitet werden.

Durch autoritäre Handlungsanweisungen und starre Vorgaben sind Änderungen im Entwicklungsprozess auf Dauer nur mit Widerstand der Betroffenen durchzusetzen. Echte Fortschritte werden erreicht, wenn auf die Erfahrungswerte der Organisation Bezug genommen wird und die Mitarbeiter selbst einen Beitrag zu dem Verbesserungsvorhaben leisten. Daher sollte zu Beginn der Prozessimplementierung die theoretische Vorüberlegung und die Nützlichkeit des REP im Mittelpunkt stehen. Es ist hilfreich, einen Einführungsplan zu erarbeiten und diesen den Mitarbeitern vorzustellen. Nach Präsentation der Konzepte werden die Ergebnisse anschließend in Projekthandbüchern, Anleitungen (Tutorials) oder Leitfäden manifestiert. Da mit diesen Mitteln noch keine verbindlichen Vorgaben existieren, sollte nach der allgemeinen Akzeptanz der dahinterstehenden Idee damit angefangen werden, die Änderungen durch Festlegung konkreter Handlungsanweisungen durchzusetzen. Dies kann in Form von Standardwerkzeugen, Vorlagen oder Ablaufvorgaben erfolgen. Für die dauerhafte Etablierung ist es unumgänglich, in der Einführungsphase Kontrollen (z.B. Stichproben, Dokumentenreviews, etc.) durchzuführen. Damit soll ein Unterlaufen der neu eingeführten Vorschriften verhindert werden.

[52] Im folgenden auch allgemein als Organisation bezeichnet.
[53] Vgl. [Sch02] S.47
[54] Die objektive Selbstkritik ist nahezu unmöglich. Es empfiehlt sich, unabhängige Gutachter zu beauftragen, die eine unparteiische Analyse durchführen können.

4.7. Zusammenfassung und Bewertung der Ergebnisse

Inspiriert vom „Object Engineering“-Ansatz nach Rupp et al. handelt es sich bei dem hier beschriebenen konzeptionellen Prozessmodell um eine „hausgemachte Interpretation“ von etablierten Ansätzen des Requirements Engineering in Kombination mit den Erfahrungswerten der befragten Experten. Der REP soll jedoch nicht Vorgehensmodelle wie V-Modell, RUP oder XP ersetzen, sondern lediglich die im Abschnitt 3.4 formulierten Defizite ausgleichen. Die Tabelle 4.3 zeigt eine Gegenüberstellung der wichtigsten Expertenprobleme aus Kapitel 3.2 (Auswertung und Interpretation von Kunden - und Expertensicht) und des Erfüllungsgrades im Prozessmodell:

	Grad	Bemerkungen/Verweis
Organisatorische Aspekte		
Anforderungsprozess	$\oplus$	konzeptionelle Konstruktion eines Requirements Engineering Process
Ressourcenauslastung	$\ominus$	-
Anforderungsquellen	$\oplus$	Scoping, Dokumentationskette
Stellung des Anforderungsingenieurs.	$\odot$	Der Ausfall des Anf.-Ing. kann nur durch die konsequente Dokumentation der Erkenntnisse ausgeglichen werden.
Politik und Aufwandsproblematik		
Risikoverteilung	$\oplus$	Durch Pflichtenheft und CCP wird das Risiko auf beide Parteien verteilt.
Management von Nachfolgeprojekten	$\oplus$	Scope-Parameter und CCP sehen die Dokumentation von Anforderungen potentieller Nachfolgeprojekte vor.
Steuerung der Erwartungshaltung	$\ominus$	-
Informationsdichte der Quellen		
Formatvielfalt	$\oplus$	Transformation des Materials in Low-Level Anforderungen nach dem „Teile und Herrsche“-Prinzip
Dokumentationsverwaltung	$\oplus$	Änderungshistorie, Metadaten, Dokumentationskette
Massenanforderungen	$\oplus$	Wiederverwendung durch Links, Beziehungsmodellierung, Anforderungsbaum

Umgang mit Anforderungen		
Anforderungsverständigung	$\oplus$	Kunden-Vorbereitung (Validierung)
redundante Datenhaltung	$\oplus$	Anforderungskatalog als Repository, Pflichtenheft als historischer Auszug
kundenspezifische Besonderheiten	$\oplus$	durch Transformationskonzept abgedeckt
Fehlinterpretation durch Kunden	$\ominus$	-
Weiterverarbeitung der Ergebnisse		
Änderungsnachführung	$\oplus$	durch CCP geregelt
Fehlererkennung bei Realisierung	$\ominus$	-
Testfallgenerierung	$\odot$	Übertragung einer atomaren Anforderung in Testfallschablone möglich
evolutionäre Sackgasse	$\odot$	Entwicklungszweige und Roll-Backs im Anforderungskatalog
Legende: $\oplus$ = Konzept; $\odot$ = thematisiert; $\ominus$ = unberücksichtigt		

Tabelle 4.3.: *Identifizierte Probleme und deren Erfüllungsgrad*

Was ist neu ?

- Konsistentes Begriffsverständnis zum RE, abgeleitet aus theoretischen Ansätzen der Literatur.
- Übertrag der RE-Ideen in ein modernes betriebswirtschaftliches Prozessverständnis mit dem Ziel einer Beschleunigung und Verbesserung der ersten Projektphase hinsichtlich Zeit, Kosten und Qualität („Softwarefabrik“).
- Vorstellung einer detaillierten konzeptionellen REP-Ablaufbeschreibung und deren grafische Aufbereitung unter Berücksichtigung des typischen Projektverlaufsschemas in der individuellen Fertigung von Softwarelösungen.
- Kombination und Erweiterung bewährter Dokumentationstypen der UML mit dem Konzept der Anforderungsschablonen, um Defekte in natürlichsprachlichen Anforderungen präventiv vorzubeugen.
- Vorstellung eines Konzeptes zur Strukturierung und Integration sämtlicher Anforderungsdokumente und Anforderungsarten/Sätze in Form einer hierarchischen und vernetzten Baumstruktur (Anforderungsbaum).
- Durchgängige Beschreibung des „roten Fadens“ anhand eines Beispielprojektes und der Empfehlungen aus den Ergebnissen der Expertenbefragung.
- Entwicklung eines eigenen konzeptionellen Change Control Process durch Kombination der Ansätze aus der Literatur, V-Modell, RUP und XP.

- Vorstellung eines Prototypen zur werkzeugbasierten Unterstützung für den hier formulierten REP sowie die Ableitung künftig zu implementierender Funktionalitäten.

Bewertung

Zum gegenwärtigen Zeitpunkt lassen sich erste Erkenntnisse zur Nützlichkeit des Prozessmodells ableiten. Ein Großteil der hier vorgestellten Konzepte basiert auf den bewährten UML-Techniken, deren praktischer Wert durch die hohe Akzeptanz in der IT-Branche bewiesen sein dürfte.

Unabhängig von den bereits etablierten Ansätze muss die Schablonentechnik der Sophist Group ihre Praxistauglichkeit zunächst noch beweisen. Hier können die Erfahrungen aus dem ELFI-Projekt einbezogen werden. Mit Hilfe eines frühen Prototypen des ProjectMentors wurden teilweise die Kundenanforderungen in Form von Anforderungsschablonen hinterlegt und dem Auftraggeber zur Validierung vorgelegt. Obwohl zum damaligen Zeitpunkt noch kein dokumentierter REP vorlag, ähnelte das Vorgehen dem hier beschriebenen Ablaufschema. Nach Aussagen der Projektleitung wurde die Vorgehensweise von Kunden und Entwicklern sehr gut angenommen. Trotz einiger „normaler“ Probleme im Projektverlauf (Vgl. Kapitel 3.3.4) konnte die Geschäftsführung das Projekt als Erfolg werten, was u.a. auf die anfangs unkonventionell anmutende Anforderungsentwicklung zurückzuführen war.

An dem Prozessmodell kann auch Kritik geübt werden:

- Der Ansatz ist stark werkzeugabhängig, eine manuelle Ausführung ist z.B. in Bezug auf die Konstruktion von Anforderungsschablonen mit sehr hohem Aufwand verbunden.
- Das Pflichtenheft, als historischer Auszug des Anforderungskataloges erfordert eine mächtige Exportfunktion (Generator), die beispielsweise im ProjectMentor noch nicht realisiert wurde. Als Workaround bietet es sich an, das Pflichtenheft auf herkömmliche (manuelle) Art und Weise zu verfassen und mit entsprechenden Dokumentverweisen zu versehen.
- Die Vernetzung und ID-Vergabe der Anforderungen ist ebenfalls nur durch Werkzeuge zu erreichen. Eine manuelle Bearbeitung dürfte nur mit Kompromissen machbar sein.

Es sei erwähnt, dass im Prozessmodell prinzipiell die Werkzeugunabhängigkeit angestrebt wird. Dies bedeutet jedoch nicht, dass man auf den Einsatz von Werkzeugen verzichten kann, es ist nur ein Hinweis darauf, dass der Prozess nicht von einem konkreten Programm oder Hersteller abhängig sein darf. Analog zur Erstellung von Quellcode mit Hilfe einer integrierten Entwicklungsumgebung (IDE), sollte die Anforderungsentwicklung ebenso selbstverständlich durch geeignete Werkzeuge unterstützt werden.

5. Fazit und Ausblick

Für IT-Dienstleister steht heute fest, dass gerade im Bereich der internen Entwicklungsprozesse ein enormer Verbesserungsbedarf besteht. Während materielle Produkte in der industriellen Fertigung vor allem durch hohe Qualität und kostengünstige Herstellungsprozesse ihren Absatz finden, wird in der IT-Branche noch viel zu oft versucht, die Softwareentwicklung mit einfältigen Mitteln zu organisieren. Einen ersten Beitrag zur Verbesserung dieser Situation kann die Einführung von klar strukturierten Prozessen leisten.

Das Requirements Engineering dient der Dokumentation von Anforderungen zur bedarfsgerechten Entwicklung qualitativ hochwertiger Softwareprodukte. Die Reduzierung der Problematik auf das reine Dokumentieren von Kundenwünschen hat sich im Zuge der „Professionalisierung der Softwaretechnik“[1] als unzureichend erwiesen. Fest steht, dass die Anforderungsentwicklung einen ähnlich komplizierten Vorgang darstellt wie Entwurf oder Realisierung. Unbestritten ist jedoch auch, dass gerade dieser Vorgang folgenschwere Auswirkungen auf alle Aktivitäten im Softwarelebenszyklus hat. Ohne Anforderungen gäbe es keine Aufwandsschätzung, keinen Entwurf, keine Programmierung und keinen Test.

Die Vorstellung, mit Hilfe eines Prozessmodells sei es nun möglich, auch mit unerfahrenen Entwicklern erfolgreich Projekte durchzuführen, ist so verbreitet, dass sie in kaum einer Diskussion fehlt. Es darf aber nicht übersehen werden, dass gute Architekten und Entwickler viele Jahre benötigen, um Erfahrungen zu sammeln. Ein definierter Prozess ersetzt weder Ausbildung, noch die berufliche Praxis. Auch ein Werkzeug wird nicht in der Lage sein, einen komplexen Vorgang vollständig abzubilden. Um trotzdem eine Verbesserung zu erhalten, sollte es zumindest möglich sein, die Prozessidee, d.h. das Konzept eines Prozesses, auf den überwiegend durch Menschen bestimmten Vorgang der Anforderungsentwicklung zu übertragen. Eine solche Idee wurde durch das konzeptionelle Prozessmodell zum Requirements Engineering in dieser Arbeit entwickelt und vorgestellt.

Mit Blick auf die wachsende Komplexität von IT-Lösungen stellt sich zu Recht die Frage, wie man zukünftig den Kundenanforderungen gerecht werden soll. Hier wird die These vertreten, dass schon bald das heute so verbreitete Pflichtenheft als Vertragsgegenstand abgelöst wird. Das Pflichtenheft stellt in der Mehrzahl der Fälle das einzige Austauschmedium zwischen Auftraggeber und -nehmer. Die Arbeit mit einem statischen Dokument ist jedoch, insbesondere bei häufigen Änderungen,

[1] Vgl. [CS01]

zeitaufwändig und fehleranfällig. Viel besser ist ein Vertragsgegenstand wie der hier beschriebene Anforderungskatalog mitsamt allen Grafiken, Metadaten und Anforderungssätzen. Die dabei entstehenden technischen und organisatorischen Schwierigkeiten (Zugriffsproblematik, Sichtbarkeit, Gewährleistungen, etc.) sind lösbar und mögliches Thema zukünftiger Arbeiten. Weiterhin sollte in Zukunft eine stärkere Fokussierung des Requirements Engineering auf das Projektmanagement erfolgen. Dies betrifft neben der Planung, Steuerung und Kontrolle von Ressourcen der Anforderungsentwicklung, auch die Definition und Einführung von Metriken, wie zum Beispiel Anforderungsumfang oder Änderungshäufigkeit. Auch phasenübergreifend sind noch einige Verbesserungen denkbar. So ist zum Beispiel die Verbindung zwischen Anforderungskatalog und Testfalldatenbank Voraussetzung für eine (teil-) automatisierte Qualitätssicherung.

Die Entwicklung von Anforderungen wird auch weiterhin eine analytische Denkleistung des Anforderungsingenieurs bleiben. Das Requirements Engineering ist ein Beitrag zur Verbesserung dieses kreativen Potentials und erlaubt es, Erfolg zu steuern und nicht länger dem Zufall zu überlassen.

Literatur

[Bal01] Balzert, Helmut: *Lehrbuch der Software-Technik*. 2. Auflage. Heidelberg, Berlin : Spektrum Akademischer Verlag, 2001 (Bd.1 Software-Entwicklung)

[BD95] Bröhl, Adolf-Peter ; Dröschel, Wolfgang: *Das V-Modell*. 2. Auflage. München, Wien : R. Oldenburg Verlag, 1995

[Bec00] Beck, Kent: *Extreme Programming explained*. Reading, MA : Addison-Wesley, 2000

[Boe86] Boehm, Barry W.: *A Spiral Model of Software Development and Enhancement*. USA : Software Engineering Notes, 1986

[BS95] Bellin, D. ; Simone, S.S.: *The CRC Card Book*. Reading, MA : Addison-Wesley, 1995

[CH01] Cockburn, Alistair ; Highsmith, Jim: *Cystal Methodologies*. Website. August 2001. – `http://www.crystalmethodologies.org/`

[CL99] Constantine, Larry ; Lockwood, Lucy: *Software for Use*. Reading, MA : Addison-Wesley, 1999

[Coc01] Cockburn, Alistair: *Writing Effective Use Cases*. 3rd printing. New York, London : Addison-Wesley, 2001 (The Crystal Collection for Software Professionals)

[CS01] CS, IEEE Computer Society: *Guide to the Software Engineering Body of Knowledge - Trial Version 1.00*. Webseite. May 2001. – `http://www.swebok.org/`

[DC88] David A. Marca ; Clement L. McGowan: *SADT: Structured Analysis and Design Techniques*. New York : McGraw-Hill Book Company, 1988

[DHM98] Dröschel, Wolfgang ; Heuser, Walter ; Midderhoff, Rainer: *Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97*. München, Wien : R. Oldenburg Verlag, 1998

[DIN96] DIN, Deutsches Institut für Normung e.V.: *Geschäftsprozessmodellierung und Workflow-Management Forschungs- und Entwicklungsbedarf im Rahmen der Entwicklungsbegleitenden Normung (EBN). DIN-Fachbericht 50.* Berlin : Beuth Verlag, 1996

[Dud01] Dudenredaktion: *Duden - Die deutsche Rechtschreibung.* 22. erweiterte Auflage. Mannheim, Leipzig, Wien, Zürich : Dudenverlag, 2001 (Duden Band 1)

[DW99] Dröschel, Wolfgang ; Wiemers, Manuela: *Das V-Modell 97.* München, Wien : R. Oldenburg Verlag, 1999

[Eck00] Eckstein, Jutta: XP - eXtreme Programming. Ein leichtgewichtiger Software-Entwicklungsprozess. In: *basicpro* 35 (2000), S. 6–1. – `http://jeckstein.com/`

[Elz94] Elzer, Hubert F.: *Management von Softwareprojekten: eine Einführung für Studenten und Praktiker.* Braunschweig, Wiesbaden : Friedr. Vieweg & Sohn Verlagsgesellschaft, 1994 (Wirtschaftsinformatik/DV-Praxis)

[FFF03] FFF, Fraunhofer Gesellschaft zur Förderung der angewandten Forschung e.V.: *Virtuelles Software Engineering Kompetenzzentrum.* Webseite. 2003. – `http://www.visek.de`

[Gil88] Gilb, Tom: *Principles of Software Engineering Management.* Reading, MA : Addison-Wesley, 1988

[GKP99] Gotzhein, R. ; Kronenburg, M. ; Peper, C.: *Pattern-Based Requirements Capture Applied: The SFB 501 Case Study.* Website. August 1999. – `http://rn.informatik.uni-kl.de/publications/details/GoKrPe99/index_eng.html`

[Gra03] Graser, Franz: Programmierer werden zu Softwaremodellierern. In: *Computer Zeitung* Nr. 22 (26. Mai 2003), S. 9

[Gro03] Group, Standish: *The Scope of Software Development Project Failures. CHAOS-Report.* West Yarmouth, MA : The Standish Group, 2003

[GW93] Gause, D. C. ; Weinberg, G.M.: *Software Requirements: Anforderungen erkennen, verstehen und erfüllen.* München, Wien : Carl Hanser Verlag, 1993

[Hal90] Hallmann, Matthias: *Prototyping komplexer Softwaresysteme.* Stuttgart : Teubner, 1990

[Han98] Hansen, Hans Robert: *Wirtschaftsinformatik 1: Grundlagen betrieblicher Informationsverarbeitung.* Stuttgart : Uni-TB, 1998

[Hol01] Holub, Allen: OO design process: use cases, an introduction. In: *IBM developerWorks* (2001). – `http://www.ibm.com/developerworks/`

[Hru00] Hruschka, Peter: *Process for System Architecture and Requirements Engineering.* New York : Dorset House, 2000

[HS96] Henderson-Sellers, Brian: *Object-Oriented Metrics - Measures of Complexity.* Englewood Cliffs, NJ : Prentice-Hall, 1996

[Hum95] Humphrey, W. S.: *A Discipline for Software Engineering.* 1st Edition. Reading, MA : Addison-Wesley, 1995

[IES03] IESE, Fraunhofer Institut Experimentelles Software Engineering: *Fragebogen zum Stand des Software-Anforderungsprozesses in Ihrem Unternehmen.* Webseite. 2003. – `http://www.iese.fhg.de/re-kit/Fragebogen`

[JBR99] Jacobson, Ivar ; Booch, Grady ; Rumbaugh, James: *Unified Software Development Process.* 1st Edition. Reading, MA : Addison-Wesley, 1999

[JCJO92] Jacobson, Ivar ; Christerson, Mahnus ; Jonsson, Patrik ; Overgaard, Gunnar: *Object-Oriented Software Engineering.* Reading, MA : Addison-Wesley, 1992

[KBP03] Krieg-Brückner, Bernd ; Peleska, Jan: *GDPA - Graphical Development Process Assistant.* Bremen : Informatik der Universität Bremen, 2003. – `http://www.informatik.uni-bremen.de/uniform/gdpa/`

[KE02] Kuloor, Chethana ; Eberlein, Armin: Requirements Engineering for Software Product Lines. In: *15th International Conference on Software & Systems Engineering and their Applications (ICSSEA), Paris, France*, 2002. – `http://www.enel.ucalgary.ca/People/eberlein/publications/index.html`

[Köp98] Köppen, Eckhart: *Veranstaltungsreihe - Modellierung 1.* Wirtschaftsinformatik und Softwaretechnik, Universität Essen, 1998. – `http://nestroy.wi-inf.uni-essen.de/Lv/mod1/folien/10.html`

[Kru01] Kruchten, Phillipe: What Is the Rational Unified Process? In: *The Rational Edge e-Zine*, 2001. – `http://www.therationaledge.com/content/jan_01/f_rup_pk.html`

[KS98] Kotonya, Gerald ; Sommerville, Ian: *Requirements Engineering. Processes and Techniques.* New Jersey : John Wiley & Sons, 1998

[LR02] Lüpschen, Helga ; Reukauf, Ulrich: Der Mensch als Erfolgsfaktor. In: *OBJEKTspektrum* Nr. 5 (Oktober 2002), S. 35–39

[LRW02] Lippert, Martin ; Roock, Stefan ; Wolf, Henning: *Software entwickeln mit eXtreme Programming.* 1. Auflage. Heidelberg : dpunkt.verlag, 2002

[M+97] Mertens, Peter et al.: *Integrierte Informationsverarbeitung.* 3.Auflage. Berlin, Heidelberg, New York : Gabler Verlag, 1997 (Lexikon der Wirtschaftsinformatik)

[McC93] McClure, Carma: *Software Automatisierung. Reengineering, Repository, Wiederverwendbarkeit.* München, London : Carl Hanser Verlag, 1993

[MT00] MT, microTOOL GmbH: *actiF - Das Prozessmodell für die objektorientierte Entwicklung mit objectiF - Version 2.0.* Webseite. 2000. – `http://www.microtool.de/objectiF`

[NMR03] Nutzinger, Verena ; Moorwood, Katrina ; Redfern, Eva: *English - German dictionary (LEO Online Service).* München : Informatik der Technischen Universität München, 2003. – `http://dict.leo.org`

[NT97] Nonaka, Ikujiro ; Takeuchi, Hirotaka: *Die Organisation des Wissens: Wie japanische Unternehmen eine brachliegende Ressource nutzbar machen.* Frankfurt, New York : Campus Verlag, 1997

[Oes98] Oestereich, Bernd: *Objektorientierte Softwareentwicklung - Analyse und Design mit der UML.* 4., aktualisierte Auflage. München, Wien : Oldenburg Verlag, 1998

[OMG03] OMG, Object Management Group: *Unified Modeling Language (UML) Version 1.5, Framingham.* Webseite. March 2003. – `http://www.omg.org/uml`

[Par91] Partsch, Helmut: *Requirements Engineering.* Band 5.5. München, Wien : R.Oldenburg Verlag, 1991 (Handbuch der Informatik)

[Pau98] Pautzke, Gunnar: *Die Evolution der organisatorischen Wissensbasis. Bausteine zu einer Theorie des organisatorischen Lernens.* Münchener Schriften zur angewandten Führungslehre Nr. 58. München : Kirsch, 1998

[PB93] Pomberger, Gustav ; Blaschek, Günther: *Grundlagen des Software Engineering - Prototyping und objektorientierte Software-Entwicklung.* München, Wien : Carl Hanser Verlag, 1993

[Poh95] Pohl, Klaus: *A Process Centered Requirements Engineering Environment*, Technische Hochschule Aachen, Diss., 1995

[R+90] Rambaugh, James et al.: *Object-Oriented Modeling and Design.* 1st edition. New Jersey : Prentice Hall, 1990

[R+02] Rupp, Chris et al.: *Requirements-Engineering und -Management.* 2., überarbeitete Auflage. München, Wien : Carl Hanser Verlag, 2002

[Rad03] Rademacher, Rochus: IT-Profis reden nur von Technik. In: *Computer Zeitung* Nr. 25 (16. Juni 2003), S. 12

[RD00] Rupp, Chris ; Dallner, Jürgen: Requirements-Engineering – der Einsatz einer natürlichsprachlichen Methode bei der Ermittlung und Qualitätsprüfung von Anforderungen. In: *OBJEKTspektrum* Nr. 2 (Januar 2000)

[RD01] Rupp, Chris ; Dallner, Jürgen: Mustergültige Anforderungen. In: *OBJEKTspektrum* Nr. 3 (März 2001)

[RSC02] RSC, Rational Software Corporation: *Rational Unified Process - Version 2002.05.00.* Hypertext-Version. June 2002. – `http://www.rational.com/rup`

[S+03] Siedersleben, Johannes et al.: *Softwaretechnik: Praxiswissen für Software-Ingenieure.* 2., überarbeitete und aktualisierte Auflage. München, Wien : Carl Hanser Verlag, 2003

[Sch97] Schienmann, Bruno: *Objektorientierter Fachentwurf.* Band 20. Stuttgart, Leipzig : B.G. Teubner Verlagsgesellschaft, 1997 (TEUBNER-TEXTE zur Informatik)

[Sch01] Scheer, August-Wilhelm: *ARIS, Modellierungsmethoden, Metamodelle, Anwendungen.* 4. Auflage. Berlin, Heidelberg : Springer Verlag, 2001

[Sch02] Schienmann, Bruno: *Kontinuierliches Anforderungsmanagement (Prozesse - Techniken - Werkzeuge).* 1. Auflage. München : Addison-Wesley, 2002 (Programmer's Choice)

[SH01] Stahlknecht, Peter ; Hasenkamp, Ulrich: *Einführung in die Wirtschaftsinformatik.* 10. Auflage. Berlin, Heidelberg : Springer-Verlag, 2001

[SMF03] Schwarz-Mehrens, Elisabeth ; Fliers, Frits: *Woran scheitern IT-Projekte?* Webseite. 2003. – `http://www.gulp.de/kb/it/projekt/itprojekteins_f.html`

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

[Sta02] Starke, Gernot: Agile Prozesse statt starrer Vorgehensmodelle. In: *Computerwoche* Nr. 27 (05. Juli 2002), S. 40

[Str77] Strunz, Horst: *Entscheidungstabellentechnik.* München : Carl Hanser Verlag, 1977

[TE77] Teichroew, Daniel ; Ernest A. Hershey: PSL/PSA: A computer-aided technique for structured documentation of information processing system requirements. In: *DBLP Bibliography Server* (1977). – `http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/t/Teichroew:Daniel.html`

[Tec03] TechTarget: *WhatIs.Com*. Needham, USA : TechTarget, Inc., 2003. – `http://www.whatis.com`

[Tho94] Thomas, Karl-Georg: *Die mittelständische Unternehmung im Entwicklungsprozeß*. Ludwigsburg, Berlin : Verlag Wissenschaft&Praxis, 1994 (Schriftenreihe Wirtschafts- und Sozialwissenschaften Band 19)

[V+02] Versteegen, Gerhard et al.: *Software Management - Beherschung des Lifecycles*. Berlin, Heidelberg : Springer-Verlag, 2002 (XPert.press)

[Ver00] Versteegen, Gerhard: *Projektmanagement mit dem Rational Unified Process*. Berlin, Heidelberg : Springer-Verlag, 2000 (XPert.press)

[VSH01] Versteegen, Gerhard ; Salomon, Knut ; Heinold, Rainer: *Change Management bei Software-Projekten*. Berlin, Heidelberg : Springer-Verlag, 2001 (XPert.press)

[Web92] Weber, Herbert: *Die Software-Krise und ihre Macher*. Berlin, Heidelberg : Springer-Verlag, 1992 (Springer-Compass)

[Wie03] Wiegers, Karl E.: *Software Requirements*. 2nd Edition. Redmond, WA : Microsoft Press International, 2003

[Win00] Winter, Andreas: *Referenz-Metaschema für visuelle Modellierungssprachen*. Wiesbaden : Deutscher Universitäts-Verlag, 2000

A. Anhang

A.1. V-Modell 97: System-Anforderungsanalyse

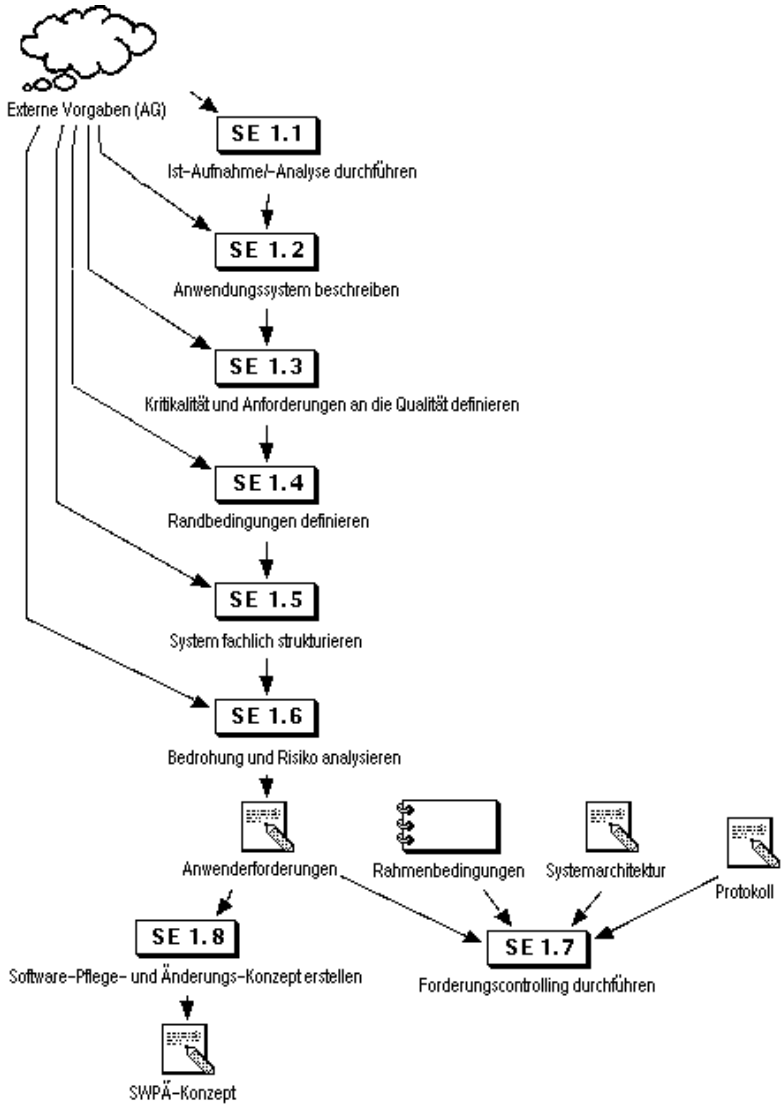

Abbildung A.1.: *SE1 - System-Anforderungsanalyse ([KBP03])*

A.2. Rational Unified Process: Change Requests

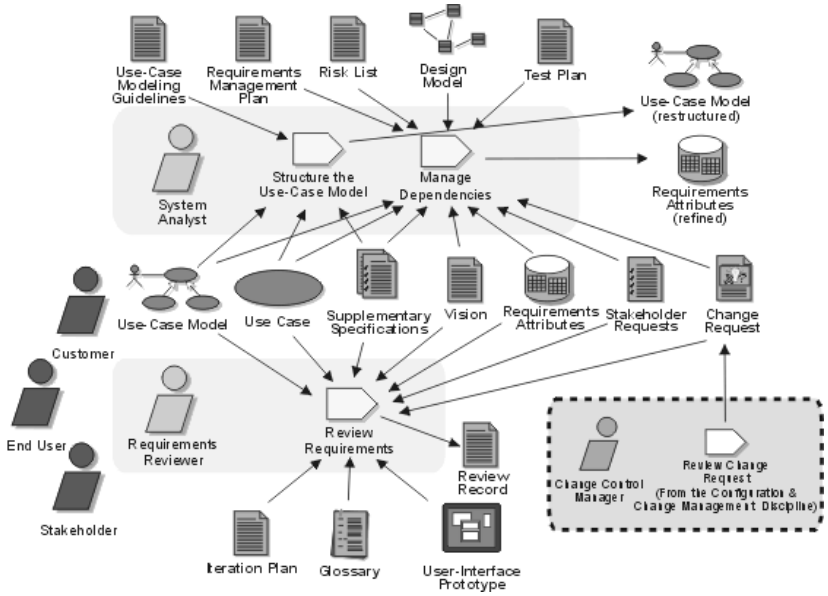

Abbildung A.2.: *Workflow Detail: Manage Changing Requirements ([RSC02])*

A.3. SWEBOK: Teilgebiete des RE

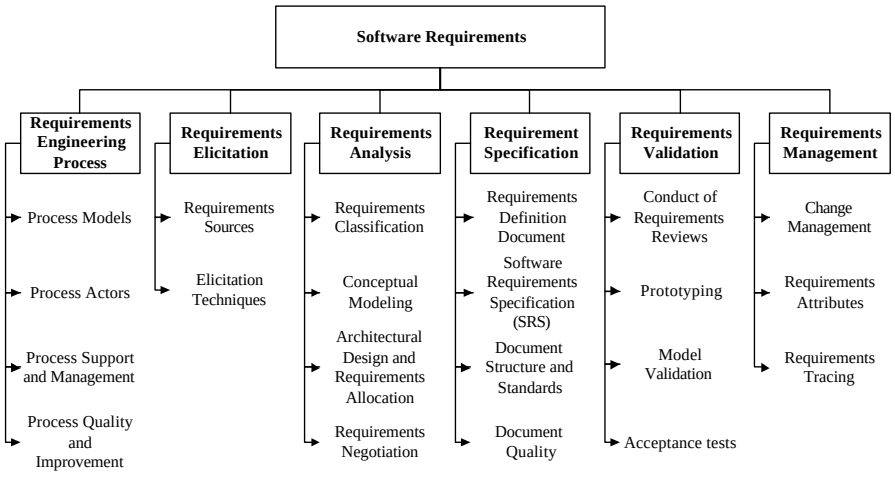

Abbildung A.3.: *Themenkomplex RE in SWEBOK([CS01])*

A.4. REP: Notationsübersicht der Diagramme

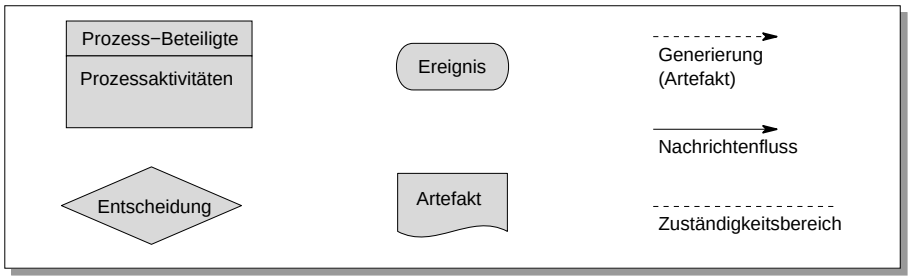

Abbildung A.4.: *Notationsübersicht der Diagramme*

A.5. Fragebogen

GFT

Fragebogen: Analyse zum Stand des Anforderungsprozesses bei Auftraggebern

(Offline-Version)

1 Allgemeines

Wählen Sie ein typisches Software-Projekt, das Ihr Unternehmen in der jüngeren Vergangenheit mit einem IT-Dienstleister durchgeführt hatte.

Welche der folgenden Aussagen beschreibt am besten die Priorität Ihrer Aufträge?

[ ] **Zeit**: Der Auslieferungstag ist unabänderlich. Wenn Probleme auftreten, wird die Funktionalität reduziert oder der Kostenrahmen erhöht.
[ ] **Funktionalität**: Die zu entwickelnde Funktionalität ist unabänderlich, bei Problemen werden Termine verschoben oder zusätzliche Mittel aufgebracht.
[ ] **Qualität**: Die Qualität ist der kritische Faktor, Auslieferungstermine und Kosten müssen angepasst werden, um die vorgegeben Qualitätsansprüche zu erfüllen.
[ ] **Kosten**: Die Kosten sind unabänderlich. Der Projekterfolg ist ausschließlich davon abhängig.
[ ] Keiner der genannten Punkte wird priorisiert. Die Faktoren Zeit, Funktionalität, Qualität und Kosten sind gleich wichtig.

Requirements Engineering: In Anlehnung an "Softwaretechnik" würde eine mögliche deutsche Übersetzung "Anforderungstechnik" lauten. Gegenstand des Requirements Engineering ist es, im Rahmen eines systematischen Vorgehens, eine Dokumentation (inkl. Erfassung, Beschreibung und Prüfung) von Kundenanforderungen vorzunehmen.

Kennen Sie bereits das prinzipielle Anliegen des Requirements Engineering?

[ ] Völlig neu.
[ ] Die Problematik ist bekannt, aber keine direkten Erfahrungen.
[ ] Die Problematik ist klar, die wesentlichen Inhalte sind bekannt und es gibt eigene Ansätze zum Thema.

Abbildung A.5.: *Kunden-Fragebogen zum Anforderungsmanagement 1*

GFT

2 Anforderungsprozess

Verstehen Sie im Folgenden unter Anforderungsprozess alle Aktivitäten zur Erhebung, Analyse und Dokumentation ihrer Anforderungen an das vom Auftragnehmer zu entwickelnde Projektergebnis.

Welche der folgenden Aussagen charakterisiert ihre Sicht auf den Anforderungsprozess bei Software-Projekten am besten?

[ ] Der Anforderungsprozess ist eine **feste Phase zu Beginn des Projektes**. Spätere Änderungen von Anforderungen werden durch das Änderungsmanagement des Projektes bearbeitet.
[ ] Der Anforderungsprozess ist eine **feste Phase zu Beginn der Entwicklung einer Version**. Wesentliche Änderungen werden in zukünftige Versionen verlagert, kleinere Änderungen werden durch das Änderungsmanagement des Projektes bearbeitet.
[ ] Der Anforderungsprozess ist in enger Zusammenarbeit mit dem Auftragnehmer **eine kontinuierliche Aktivität**, die mit Beginn des Projektes anfängt und mit Auslieferung eines Systems endet.

Verstehen Sie im Folgenden unter Anforderungsprozess alle Aktivitäten zur Erhebung, Analyse und Dokumentation ihrer Anforderungen an das vom Auftragnehmer zu entwickelnde Projektergebnis.

Wird für ihre Aufträge ein Anforderungsprozess explizit geplant und als eigenständige Aufgabe in der Projektabwicklung verstanden?

[ ] Nein
[ ] Ja

Sie können diese Frage überspringen, wenn Sie keinen eigenen Anforderungsprozess definiert haben.

Gibt es in Ihrem Unternehmen Richtlinien, Ablaufbeschreibungen oder Leitfäden, die den Anforderungsprozess explizit dokumentieren?

[ ] Nein
[ ] Ja

Sie können diese Frage überspringen, wenn Sie keinen eigenen Anforderungsprozess definiert haben.

Stimmt der dokumentierte Anforderungsprozess mit dem gelebten Prozess überein?

[ ] Nein
[ ] Ja

Ihre Anforderungen helfen dem IT-Dienstleister bei der Umsetzung der Software-Projekte. Diese müssen sowohl von Ihnen als auch mit dem IT-Dienstleister abgesprochen werden.

Gibt es eine verantwortliche Person (z.B. Spezialist aus der IV-Abteilung, Anforderungsingenieur, Experten), die für die Formulierung von Anforderungen zuständig ist?

[ ] Nein
[ ] Ja

Abbildung A.6.: *Kunden-Fragebogen zum Anforderungsmanagement 2*

Wenn Sie Aktivitäten explizit ausführen, nehmen Sie sich in der Regel bewusst mehr Zeit. Wenn Sie Aktivitäten im Gegensatz dazu aus dem Gefühl heraus steuern, handeln sie implizit.

Welche der folgenden Aktivitäten werden im Zuge der Anforderungsermittlung von Ihnen durchgeführt?

[ ] Dokumentation der Anforderungen in einem Lastenheft
[ ] Anforderungsermittlung und Konsensbildung mit IT-Dienstleister
[ ] Dokumentation der Anforderungsherleitung (Nachweise)
[ ] Inspektionsberichte und Reviews der Anforderungsdokumente
[ ] Workshops, Präsentationen und Vorträge

Sie können bei dieser Frage auch mehrere Antworten angeben. (Mehrfachauswahl)

Welche Dokumente entwickeln Sie typischerweise im Rahmen des Anforderungsprozesses?

[ ] Visionsdokument / Beschreibung der Projektidee
[ ] Modelle der relevanten Geschäftsprozesse
[ ] Glossar über Fachbegriffe
[ ] Risikoliste
[ ] Machbarkeitsstudie
[ ] Lastenheft
[ ] Grob- und Feinkonzepte
[ ] Keine Dokumente, der Auftragnehmer ist gleichzeitig Berater und erstellt diese selbständig.

Abbildung A.7.: *Kunden-Fragebogen zum Anforderungsmanagement 3*

GFT

3 Techniken zum Requirements Engineering

Im Folgenden werden typische Techniken zur Anforderungsanalyse und Dokumentation genannt. Wenn Sie bereits mit einigen gearbeitet haben und entsprechende Erfahrungen sammeln konnten, sollen Sie hier die Nützlichkeit kurz bewerten. Beachten Sie, dass die Techniken auch unabhängig von IT-Projekten anwendbar sind, also nicht spezifisch etwas mit Requirements Engineering zu tun haben.

Kreativitätstechniken (1=sehr hilfreich ... 5=wenig hilfreich; 6=unbekannt)

[ ] Brainstorming: Ideensammlung und Aufdeckung unterbewusster/impliziter Anforderungen
[ ] Mind Mapping: baumartige/visuelle Verknüpfung von Begriffen und Assoziationen

---

Beobachtungstechniken (1=sehr hilfreich ... 5=wenig hilfreich; 6=unbekannt)

[ ] Feldbeobachtung: Arbeitsabläufe werden verfolgt und analysiert von Unabhängigen
[ ] Apprenticing: ein Analytiker erlernt die Tätigkeit und erhält dadurch ein genaues Bild der Vorgänge

---

Befragungstechniken (1=sehr hilfreich ... 5=wenig hilfreich; 6=unbekannt)

[ ] Fragebogen
[ ] Interview: Fragenliste mündlich durchgehen und direkte Klärung der Probleme im Gespräch
[ ] Selbstaufschreibung: Verfassen einer Tätigkeitsbeschreibung durch denjenigen, der Tätigkeit ausführt
[ ] On-Site-Customer: Kunde stellt einen Berater ab der vor Ort bei den Entwicklern Fragen beantworten kann

---

Vergangenheitsorientierte Techniken (1=sehr hilfreich ... 5=wenig hilfreich; 6=unbekannt)

[ ] Systemarchäologie: Ableitung von Anforderungen mittels Untersuchung eines abzulösenden Systems
[ ] Reuse: Anforderungen aus ähnlichen Projekten ableiten / Wiederverwendung von Anforderungen

---

Feedback-Techniken (1=sehr hilfreich ... 5=wenig hilfreich; 6=unbekannt)

[ ] Simulationen / Szenarien / Prototypen: Demonstrationen zu Dynamik, Zeitverhalten und Oberfläche
[ ] Anforderungsreview : durch Zustimmung/Ablehnung die Anforderungen auf Korrektheit / Verständlichkeit prüfen

---

Unterstützende Techniken (1=sehr hilfreich ... 5=wenig hilfreich; 6=unbekannt)

[ ] NLP/Bedeutungsanalyse: Anforderungsermittlung unter Einsatz der sprachlichen Analyse
[ ] Essenzbildung: Sachverhalte werden auf ihre fachliche Essenz zurückgeführt und im Kern hinterfragt
[ ] CRC-Karten: Modellierung durch spielerisches Erkunden von Anwendungsfällen
[ ] Snowcards: gruppenbasierte Erarbeitung von Anforderungen durch sukzessive Ergänzung im Team
[ ] Anwendungsfälle: Sammlung/Modellierung von Geschäftsereignissen und Gruppierung in logisch überschaubare Einheiten
[ ] Problem Frames: Komplexe Probleme werden in Teilprobleme zergliedert und Domänen zugeordnet
[ ] Anforderungspriorisierung und Risiko-Top-10: Aufstellen von Kriterien für die Dringlichkeit

Abbildung A.8.: *Kunden-Fragebogen zum Anforderungsmanagement 4*

GFT

4 Software-Projekte

Für die Beantwortung der folgenden Fragen, wählen Sie bitte ein typisches Projekt. Nach Möglichkeit sollten Sie bei der Erarbeitung des Anforderungsdokumentes mitgearbeitet haben.

Sie können bei dieser Frage auch mehrere Antworten angeben. (Mehrfachauswahl)

Welche der hier genannten Kritikpunkte, sind Ihrer Meinung nach häufig der Fall bei Software-Projekten?

[ ] zu viel Funktionalität umgesetzt als tatsächlich benötigt
[ ] zu wenig Funktionalität umgesetzt als gewünscht
[ ] falsche oder unbrauchbare Funktionalitäten
[ ] unklare Zielstellung und Umfang des Projektes
[ ] fachliche Missverständnisse und Kommunikationsprobleme
[ ] fachliche Bedürfnisse wurden nur unzureichend einbezogen
[ ] Anforderungsdokument war zu kompliziert
[ ] sonstige (Freitext)

Abbildung A.9.: *Kunden-Fragebogen zum Anforderungsmanagement 5*

Deutsche Begriffe zum Requirements Engineering

englisch	deutsch
artifact	Artefakt, Ergebnis oder Produkt
best practices	Erfahrungen
business use case	Geschäftsvorfall
change management	Änderungsmanagement
change request	Änderungsauftrag
elicitation techniques	Erhebungstechniken
estimation force	Schätzgruppe
repository	(Daten-)Quelle
requirement	Anforderung
requirement feature	funktionale Anforderung
requirement constraint	geforderte Nebenbedingung
requirements engineer	Anforderungsingenieur
requirements engineering	Anforderungstechnik
requirements engineering process	Anforderungsprozess
requirements management	Anforderungsmanagement
requirements model	Anforderungsmodell, Anforderungskatalog
requirements sources	Anforderungsquellen
requirements specification	Anforderungsdokument
scope	Geltungsbereich
scope of the system	Systemumfang
software engineering	Softwaretechnik
stakeholder	Beteiligter, Geschäftsinteressent
system vision	Vision
tailoring	Zurechtschneiden
task	Maßnahme
template	Dokumentvorlage
trigger event	Auslöser
tool	Werkzeug
use case	Anwendungsfall
use case diagram	Anwendungsfalldiagramm

Tabelle A.1.: *Übersetzungstabelle für häufig verwendete Begriffe [NMR03]*

Erklärung über Hilfsmittel

Die vorliegende Arbeit habe ich selbstständig und ohne Benutzung anderer als der angegebenen Quellen angefertigt. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Ilmenau, 13. Oktober 2003, Thomas Havemeister