T U M

I N S T I T U T F Ü R I N F O R M A T I K

Theoretische und praktische Ansätze im Requirements Engineering für Standardsoftware und Anlagenbau

Bernhard Deifel

TUM-I9832
April 98

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-04-I9832-80/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Theoretische und praktische Ansätze im Requirements Engineering für Standardsoftware und Anlagenbau [*]

Bernhard Deifel

Institut für Informatik
Technische Universität München
http://www4.informatik.tu-muenchen.de

30. April 1998

Kurzfassung

Dieser Bericht gibt einen Überblick über allgemeine und anwendungsspezifische Forschungsaktivitäten des Requirements Engineerings von Software. Wir gehen dabei einerseits auf typische allgemeine Problemstellungen im Requirements Engineering ein und betrachten andererseits spezielle Probleme innerhalb der Anwendungsbereiche der Entwicklung komplexer Standardsoftware wie auch der Entwicklung von automatisierten Produktionsanlagen. Typische allgemeine Problemstellungen sind gegliedert in das Herausarbeiten, das Verhandeln, die Dokumentation/Spezifikation und die Validierung/Verifikation von Anforderungen. Spezifisch für die Entwicklung von Standardsoftware ist vor allem der Marktbezug, woraus sich einige Unterschiede zur Entwicklung der Individualsoftware ergeben. Das Requirements Engineering des Anlagenbaus zeichnet sich vor allem dadurch aus, daß Software nur ein Teil des zu entwikkelnden Systems ist und daher eine integrierte Hardware- und Softwareplanung notwendig ist. Wir zeigen den aktuellen Stand der Forschungsaktivitäten und ordnen sie bezüglich ihrer Praxistauglichkeit ein.

[*] Diese Arbeit entstand im Rahmen des FORSOFT Projektes A4 “Requirements Engineering in der Automatisierungstechnik” und wurde gefördert von der Bayerischen Forschungsstiftung und der Siemens AG

Inhaltsverzeichnis

1 Einführung und Zielsetzung

Im Teilprojekt A4 des Forschungsverbundes Software-Engineering (FORSOFT) beschäftigen wir uns schwerpunktsmäßig mit dem Requirements Engineering in zwei verschiedenen Forschungsschwerpunkten. Dies ist zum einen die Analyse und Verbesserung des Requirements Engineering Prozesses für automatisierte Produktionsanlagen. Dieser zeichnet sich vor allem dadurch aus, daß Software nur ein Teil des zu entwickelnden Systems ist und daher eine integrierte Hardware- und Softwareplanung notwendig ist. Zum anderen betrachten wir das Requirements Engineering komplexer Standardsoftware gemeinsam mit unserem Industriepartner. Deren Aufgabe sind die Entwicklung und Weiterentwicklung komplexer SPS-Steuerungen und der dazugehörigen Engineeringsoftware. Unser Projekt analysiert dabei die frühen Phasen des Entwicklungsprozesses für ein komplexes Softwaresystem und identifiziert dadurch besondere Problemstellungen, die durch den Marktbezug dieser Software auftreten. Für die genauere Eingrenzung der beiden Forschungsschwerpunkte siehe [BDS98].

Dieser Bericht soll einen Überblick über theoretische und praktische Ansätze im Requirements Engineering geben und den weiteren Forschungsbedarf innerhalb dieses Gebietes identifizieren. Wir gehen dazu auf unterschiedliche allgemeine Problemstellungen im Requirements Engineering ein und erläutern derzeit vorhandene Lösungsansätze und Forschungsaktivitäten. Dazu gehören das Herausarbeiten, das Verhandeln, die Dokumentation/Spezifikation und die Validierung/Verifikation von Anforderungen. Von besonderem Interesse sind für uns Aktivitäten, die im Zusammenhang unserer beiden Forschungsschwerpunkte laufen. Da wir uns hier hauptsächlich mit einer anwendungsnahen Forschung befassen, ordnen wir die Lösungsansätze jeweils bezüglich ihrer Eignung für den praktischen Einsatz ein und geben Hinweise für die mögliche spezifische Anpassung an unsere Anwendungsbereiche.

In dem folgenden Kapitel 2 führen wir unser Verständnis von Requirements Engineering ein. In Kapitel 3.1 stellen wir kurz einige Überblicksartikel über Requirements Engineering vor. Kapitel 3.2. befaßt sich mit verschiedenen allgemeinen Forschungsaktivitäten im Requirements Engineering, geordnet nach einer idealisierten Prozeßsicht. Die Kapitel 3.3 und 3.4 befassen sich mit den speziellen Forschungsaktivitäten im Requirements Engineering der betrachteten Anwendungsgebiete zur Entwicklung von automatisierungstechnischen Systemen und von Standardsoftware. Schließlich fassen wir in Kapitel 4 die gewonnenen Erkenntnisse zusammen.

2 Der Begriff Requirements Engineering

In der Literatur gibt es keinen einheitlichen Konsens darüber, was genau unter Requirements Engineering zu verstehen ist. Eine Auswahl verschiedener Definitionen hierzu befindet sich zum Beispiel in [Poh96].

In unserem Projekt verstehen wir unter Requirements Engineering einen systematischen Prozeß. Dieser hat zum Ziel, aus vagen Vorgaben in unterschiedlicher Form und Darstellung eine in sich konsistente, konfliktfreie und validierte Spezifikation als Ergebnis zu erzeugen. Dabei stellt die Spezifikation dar, was zu realisieren ist, jedoch nicht wie.

Im ersten Forschungsschwerpunkt soll die Spezifikation dabei verschiedene Teilaspekte einer zu automatisierenden Produktionsanlage darstellen. Für den zweiten Schwerpunkt stellt die Spezifikation die Anforderungen an ein neues oder ein weiter zu entwickelndes Softwaresystem dar.

Wesentlich für die Spezifikation ist eine adäquate und verständliche Darstellung für verschiedene an der Entwicklung beteiligten Personen, wie zum Beispiel Softwareentwickler, Manager, Benutzer und Käufer.

Wir verstehen in diesem Bericht unter Requirements Anforderungen an ein zu erstellendes System. Viele vorgestellte Methoden setzen einen Kontakt mit dem Kunden eines zu erstellenden Systems voraus. Unter Kunden verstehen wir sowohl den Auftraggeber als auch zukünftige Anwendergruppen. Je nach Problemstellung ist es erforderlich, mit dem Auftraggeber und/oder mit speziellen Anwendergruppen zusammenzuarbeiten.

3 Überblick über Forschungsaktivitäten

3.1 Übersichtsartikel

Das Forschungsgebiet Requirements Engineering ist eine sehr junge Disziplin und unterliegt noch einem ständigen Wandel. Deshalb erscheinen fortwährend neue Übersichtsartikel, die versuchen, die aktuellen Forschungsaktivitäten nach verschiedenen Kriterien zu ordnen. Zwei dieser Übersichtsartikel wollen wir hier kurz vorstellen, [Fin94] von Finkelstein und [Poh96] von Pohl.

Der Artikel von Finkelstein ordnet den Requirements Engineering Prozeß in sein spezifisches Umfeld ein. Dazu gibt er einen Überblick über die organisatorische Einordnung innerhalb einer Firma, betrachtet ihn unter Vertragsgesichtspunkten zwischen Auftraggeber und Auftragnehmer und stellt Überlegungen zur Personalbesetzung an. Zum Projektstart schlägt er Aktivitäten, wie eine Zuständigkeitsabgrenzung zwischen Requirements Engineering und Design, und eine Risiko- und Kostenabschätzung vor. Der Artikel geht kurz auf die Kernaktivitäten des Requirements Engineering, hier strukturiert nach Akquisition, Modellierung und Analyse, ein. Schließlich stellt er einen Bezug zu Rahmenthemen, wie die Fortschrittskontrolle, Kommunikation und Prozeßdokumentation her.

Pohl befaßt sich in erster Linie mit den Kernaktivitäten des Requirements Engineerings. Der Artikel enthält zunächst eine Übersicht über in der Praxis vorgeschlagene Standards und Verfahrensanweisungen und gibt einen kurzen Überblick über einsetzbare Beschreibungstechniken. Dann stellt er Forschungsaktivitäten nach einem Requirements Engineering Prozeßmodell, aufgeteilt in vier Phasen, dem Herausarbeiten (engl. Elicitation), der Verhandlung (engl. Negotiation), der Dokumentation/Spezifikation (engl. Documentation/Specification) und der Validierung/Verifikation (Validation/Verification) vor. Schließlich werden die Aktivitäten aus verschiedenen Anwendersichten beleuchtet.

Die beiden Artikel bieten einen guten Einstieg in das gesamte Gebiet des Requirements Engineerings und enthalten eine Reihe weiterführender Literaturverweise.

3.2 Allgemeine Aktivitäten

Wir haben zur Gliederung der Forschungsaktivitäten im Requirements Engineering den prozeßorientierten Ansatz von [Poh96] gewählt. Die Zuordnung der im Folgenden vorgestellten verschiedenen Ansätze zu den einzelnen Phasen erfolgt aufgrund ihrer hauptsächlichen Verwendung. Natürlich sind manche Ansätze nicht ausschließlich auf diese Phase beschränkt. Außerdem sind die Phasen nicht strikt voneinander zu trennen, sondern laufen meist stark iterativ und ineinander verzahnt ab. Im folgenden beleuchten wir jede der Phase genauer.

3.2.1 Herausarbeiten (Elicitation)

Der erste Schritt in unserem Modell ist das Herausarbeiten von Kundenanforderungen. In der Literatur über Requirements Engineering geht man dabei davon aus, daß ein direkter Kontakt zum Kunden besteht. Dies ist meistens jedoch nur der Fall, falls es sich um Individualsoftware handelt, da hier ein direkter Auftraggeber vorhanden ist. Wie wir weiter unten in diesem Abschnitt sehen werden, ist der

direkte Kundenkontakt im Bereich der Entwicklung von Standardsoftware nur eine Möglichkeit, Anforderungen an das Produkt zu sammeln.

Wir gehen zunächst in den weiteren Betrachtungen davon aus, daß bereits ein Kontakt zum Kunden besteht, jedoch noch keine Beschreibung der Anforderungen existiert. Diese Annahme ist nicht immer zutreffend, da der Kunde häufig bereits vor der Ausschreibung wie auch vor der Auftragsübergabe mehr oder weniger detaillierte Lastenhefte in textueller Form anfertigt, die Anforderungen an das zu erstellende System beschreiben. Diese bilden dann zumindest eine Grundlage, auf der die folgenden Betrachtungen aufbauen. Hierbei ist jedoch nicht zu übersehen, daß in solchen Fällen eben die Tätigkeiten des Herausarbeitens natürlich in ähnlicher Weise bereits beim Kunden erfolgt sein müssen.

Ziel des Herausarbeitens ist es, erste Anforderungen an ein zu erstellendes Softwareprodukt herauszufinden. Wir geben hierzu zunächst einen kurzen Überblick über allgemeine Techniken, wie sie auch für andere Produkte eingesetzt werden können. Anschließend betrachten wir die zwei Techniken Szenarioanalyse und Prototyping, da diese für den besonders schwierigen Teil des Herausarbeitens von Abläufenfür Software als gut geeignet erscheinen.

Die besondere Schwierigkeit beim Herausarbeiten liegt neben der rein technischen Vorgehensweise bei der Berücksichtigung verschiedener anderer Einflüsse. Dazu gehören einerseits die unterschiedlichen Beteiligten des Kunden und des Herstellers der Software, die alle ein unterschiedliches Hintergrundwissen und Interessen an der zu der betrachteten Problemstellung haben. So ist teilweise allein die Art der Kommunikation eine Schwierigkeit. Andererseits spielen verschiedene soziale, organisatorische und gesetzliche Einflüsse eine Rolle. Diese werden meist gar nicht oder nur teilweise in den Techniken berücksichtigt. Hinzu kommt noch das Herausfiltern der wahren Bedürfnisse des Kunden aus seinen Aussagen. Dies ist dann eine der Hauptaufgaben der Verhandlungsphase.

Allgemeine Techniken zur Wissensgewinnung

Zu den allgemeinen Techniken gehören zunächst verschiedene Fragetechniken. Dazu gehören Fragebogeninterviews, offene Interviews und die Selbstbefragung [GoL93]. Bei Fragebogeninterviews werden anhand eines vorgefertigten Fragebogens Kunden interviewt. Dabei werden strikt nur die Fragen des Fragebogens beantwortet. Man verfolgt damit das Ziel der Vergleichbarkeit der Ergebnisse verschiedener Befragungen bei verschiedenen Interviewpartnern. Umgekehrt geht man bei offenen Interviews davon aus, daß eventuelle Bedürfnisse eines Kunden spezifischer herausgearbeitet werden können. Bei der Selbstbefragung geht man davon aus, daß ein Entwickler durch seine einschlägige Erfahrung selbst weiß, wie ein Produkt auszusehen hat. Man verzichtet hier auf eine Befragung des Kunden. Vorteil dieser Vorgehensweise ist natürlich die Einsparung von Kosten. Der Erfolg aller Befragungstechniken hängt immer vom Erfahrungsschatz des Interviewenden ab. Dazu gehört einerseits Interviewerfahrung und andererseits das entsprechende Anwendungswissen.

Ergebnis solcher Interviews kann nun eine Mischung "natürlicher" Requirements sein, die in verschiedenartigsten Formaten vorliegen kann. Dazu gehören zum Beispiel Text, Grafiken, Video, Tonaufnahmen usw. Hierzu erforscht man beispielsweise Mutlimediawerkzeuge, die mittels einer Datenbank die verschiedenen Formate abspeichern können und die Möglichkeit bieten, die Datensätze in verschiedener Weise in Beziehung zu setzen um diese dann graphisch darstellen zu können. Dann versucht man eine Strukturierung durch Filter und Suchmöglichkeiten zu unterstützen [WCS94].

Neben den Befragungstechniken versucht man, Anforderungen an ein Produkt aus dem organisatorischen Zusammenhang abzuleiten, in dem das zu erstellende Produkt eingesetzt werden kann. Hierzu werden in erster Linie zwei Techniken erwähnt, die Ethnographie und das unternehmenszielorientierte Herausarbeiten. Die Ethnographie basiert auf dem Prinzip, daß ein Mitarbeiter des Softwareherstellers über einen längeren Zeitraum in der Organisation des Kunden mitarbeitet, um dort die Arbeitsweise zu evaluieren und daraus Anforderungen an das Produkt abzuleiten [Som+93]. Das unternehmenszielorientierte Herausarbeiten versucht dagegen im Hinblick auf den globalen Kontext der Unternehmensziele einer Organisation durch sukzessive Verfeinerung der Ziele in Unterziele bezüglich des

betrachteten zukünftigen Einsatzbereiches Anforderungen an ein zu erstellendes Produkt abzuleiten. Hierzu zählt auch die Vorgehensweise des Business Process Engineerings (siehe z.B. [Dav93]), welches zur Optimierung von Unternehmensprozessen Software als Hilfsmittel ansieht und daher aus den Prozessen Anforderungen an diese ableitet.

Neuerdings versucht man, die obigen Techniken durch die Wiederverwendung von früher gesammelten Anforderungen zu unterstützen. Hierzu können einerseits Anwendungswissensbanken helfen und andererseits Musteranforderungen aus anderen Bereichen herangezogen werden, die vorher mittels Analogiebetrachtungen ([MaL97], [Rob97]) vom konkreten Forschungsschwerpunkt abstrahiert worden sind. Eine andere Art von Wiederverwendung ist das Betrachten bereits bestehender älterer Anwendungen, aus denen mit Hilfe von Reverse Engineering Methoden Anforderungen abgeleitet werden sollen. All diese Techniken befinden sich momentan noch in der Erforschung und haben noch keinen direkten methodischen Einsatz gefunden.

Die bisher betrachteten Techniken dienen in erster Linie zur Erfassung verschiedenartigster Rohdaten, die in den weiteren Phasen innerhalb des Requirements Engineerings weiter präzisiert und in eine für einen Entwickler eindeutige Form gebracht werden müssen. Die Rohdaten werden nach der Umsetzung in eine Spezifikation nur noch aus Zwecken der Verfolgbarkeit benötigt, jedoch nicht mehr weiterverwendet.

Die Eignung der Techniken für den praktischen Einsatz im Requirements Engineering hängt in erster Linie vom Erfahrungsschatz und der Intuition des Entwicklers ab. Eine gute Unterstützung für sich wiederholende ähnliche Projekte kann in Zukunft die Wiederverwendung bereits vorhandener, gut vorstrukturierter Anforderungen bringen. Wichtig ist hier vor allem eine Anpassung der verschiedenen Methoden an den spezifischen Anwendungsbereich der zu erstellenden Software. Spezifische Standardsoftware wird innerhalb einer Firma nur einmal entwickelt beziehungsweise durch unterschiedliche Inkremente erweitert. Aus diesem Grunde kann hier auf keine Spezifikationen ähnlicher Software zurückgegriffen werden. Wiederverwendung von Spezifikationen ist hier also nicht erfolgversprechend.

Insgesamt besteht unserer Meinung nach im Bereich allgemeiner Techniken des Herausarbeitens noch ein erheblicher Forschungsbedarf, der auf jeden Fall nicht aus einer einzigen Disziplin heraus geleistet werden kann, sondern sowohl Psychologie, Informatik als auch verschiedene konkrete Anwendungsgebiete miteinander vereinen muß.

Szenarioanalyse

Wie bereits oben erwähnt, versucht man, mit Hilfe der Szenarioanalyse Abläufe der Interaktion mit einem System herauszufinden. Ein Szenario stellt dabei in der allgemeinsten Definition eine exemplarische Nutzung eines Systems dar. In der Szenarioanalyse versucht der Entwickler gemeinsam mit dem Kunden exemplarische Abläufe des Systems zu erarbeiten.

Szenarien dienen dazu, ein grobes Verständnis über Abläufe eines zu entwickelnden Systems zu erarbeiten. Letztendlich will man jedoch auch aus einer Vielzahl beschriebener Szenarien zu einer vollständigen Verhaltensspezifikation kommen. Die Szenarien können neben dem Herausarbeiten von Verhalten auch zur Verhandlung eingesetzt werden, indem verschiedene alternative Szenarien zur Diskussion gegenübergestellt werden. Sie dienen auch zur Dokumentation von Entscheidungen durch die Darstellung von nicht gewählten Negativabläufen.

Szenariotechniken haben ein weites Anwendungsspektrum. So gibt es sowohl Techniken, mit deren Hilfe man Arbeitsabläufe herausarbeiten kann, um daraus erst in einem nächsten Schritt Anforderungen an ein Softwareprodukt zu erarbeiten. Ein Vorgehensmodell hierzu schlägt zum Beispiel der Artikel [PTA94] vor. Hier werden in einem zyklischen Frage/Antwort Prozeß Szenarien ermittelt beziehungsweise in Unterszenarien verfeinert.

Andere Techniken dienen dagegen dem Herausarbeiten von konkreten Interaktionen von Benutzern mit dem Softwaresystem. Durch eine entsprechende Formalisierung der Darstellung kann man diese Art von Szenarien auch zur Validierung, Verifikation oder auch zum Generieren von Testfällen einsetzen [Hsi+94].

Ein methodisches Vorgehen ist zum Beispiel in [Hsi+94] beschrieben. Man versucht hier zunächst den Kontext eines Szenarios einzuschränken, indem man beteiligte Anwender klassifiziert. Dann wird sukzessive je definierten Kontext einen sog. Szenariobaum konstruiert. Ein Szenario ist nach dieser Technik eine Folge von Einzelereignissen, die den Systemzustand des Programmes verändern. Die Einzelereignisse entsprechen im Szenariobaum den Kanten, die Systemzustände den Knoten. Ein Szenariobaum stellt damit eine Menge von einander ähnelnden Szenarien dar, die jeweils abhängig von der Ereignisabfolge durch genau einen Pfad von der Baumwurzel (entspricht dem Initialzustand des betrachteten Kontextes) zu einem Blatt (entspricht einem Endzustand) dargestellt werden.

Erfaßte Szenarien werden in den verschiedenen Methoden abhängig vom beabsichtigten Zweck unterschiedlich weiterverwendet. Um diese in eine Verhaltensspezifikation umzusetzen, werden verschiedene Techniken der Weiterverarbeitung vorgeschlagen: [RKW94] schlägt eine Synthese in eine Art Flußdiagramm vor. Andere Ansätze leiten teilweise formal und damit automatisierbar ([Hsi+94], [KoM93]) und teilweise methodisch informell ([RBP+91]) aus den spezifizierten Szenarien Zustandsdiagramme ab. Teilweise dienen Szenarien als direkter Teil der Spezifikation. Aus Zwecken der Verfolgbarkeit ist es unabhängig davon ratsam, Szenarien auch nach einer Überleitung in eine andere Spezifikation weiter zu erhalten.

Eine gute Referenz zum Thema Szenarien stellt [RBC+96] dar. Dort wird ein sehr ausführlicher Klassifizierungsrahmen von Szenarien vorgeschlagen und viele Beispiele bezüglich dieses Rahmens eingeordnet. Grobe Klassifizierungskriterien sind hier Form der Darstellung, Inhalt, Zweck und Lebenszyklus.

Da die Begriffe Szenarien und Use Cases häufig in ähnlicher Weise verwendet werden, weisen wir auf ein weiteres Klassifizierungskriterium, dem Abstraktionsgrad hin. Use Cases und Szenarien stellen im ursprünglichen Sinne von Jacobson [Jac+92] zwei unterschiedliche Abstraktionsebenen dar: Use Cases sind die Darstellung von einer Menge von gleichen Abläufen, die als Interaktionspartner nicht instanzierte Aktoren enthalten. Demgegenüber werden Szenarien immer von konkreten Objekten ausgeführt. Im allgemeinen Kontext des Klassifizierungsschemas von [RBC+96] ist diese Unterscheidung nicht eindeutig zu ziehen. Häufig werden auch Szenarien beschrieben, in denen manche Aktoren Objektinstanzen darstellen, andere jedoch einer Objektklasse zugeordnet werden können. (Beispiel (K: Objektklasse, I: Objektinstanz): "Ein Mann (=K) fährt gegen einen Gegenstand (=K)" vs. "Peter Maier (=I) fährt gegen einen Gegenstand (=K)" vs. "Peter Maier (=I) fährt gegen eine Wand (=I)").

Szenarien sind intuitiv verständlich und haben daher die Chance, vielfach akzeptiert zu werden. Ihr Einsatz ist dabei nicht nur auf die Requirements Engineering Phase beschränkt, sondern sie können genauso gut während des Designs verwendet werden. Unseres Erachtens besteht in erster Linie Forschungsbedarf in der Einführung einer Beschreibungssprache für Szenarien, in der sowohl der Kontext, die Startbedingungen und die Endbedingungen in einer einheitlichen Weise angegeben werden können. Auf diese muß dann eine entsprechende unterstützende graphische Darstellung aufbauen. Wesentlich ist auch die Darstellung des Zwecks und die richtige Einordnung in die Requirements Engineering Phase. Die unterschiedlichen Kategorien, wie in [RBC+96] dargestellt, können einen Anhaltspunkt geben, welche verschiedenen Aspekte in einer Definition einer vollständigen Szenariotechnik notwendig sind.

Im Hinblick auf den praktischen Einsatz in unserem Projekt könnte man versuchen, bereits vorhandene Darstellungstechniken für Szenarien eventuell noch anwendungsnäher umzugestalten, indem man gewisse Einschränkungen der Darstellungsmächtigkeit macht, wo diese nicht nötig ist, um so die Gesamtkomplexität zu verringern. Wenn man beispielsweise die Szenariodarstellung mit Hilfe von

MSCs betrachtet, wäre bereits bei der Konstruktion der Szenariotechnik denkbar, immer von einer konstanten Anzahl von Kommunikationspartnern (z.B. Anlagen: Mensch, Rechner, Maschine) auszugehen. Diese Einschränkung könnte eventuell für einen Anwender noch einfacher zu verstehen sein.

Prototyping

Eine weitere Möglichkeit, Verhaltensanforderungen herauszuarbeiten ist das Prototyping. Prototyping und Szenariotechniken sind nicht klar voneinander zu trennen beziehungsweise ergänzen sich oft gegenseitig. Sie unterscheiden sich in erster Linie darin, daß Szenariotechniken meist visionäre Abläufe herausarbeiten, während Prototyping sehr konkrete Abläufe an gegebenen Systemen im Mittelpunkt hat. Daher versteht man auch im Allgemeinen unter einem Softwareprototypen ein in irgendeiner Weise ablauffähiges Programm. Es gibt viele unterschiedliche Ansätze des Prototypings, die nicht nur in der Requirements Engineering Phase des Softwareentwicklungsprozesses eingesetzt werden können. Wir wollen uns hier jedoch nur auf Aspekte, die zur Ermittlung und Evaluierung von Anforderungen dienen, beschränken.

Der Zweck besteht nach [Som96] und [PoW97] darin, durch das ausführbare Programm frühzeitig mögliche Lösungen zu evaluieren. Man erhofft sich durch sie eine erhöhte Planungssicherheit, da sich durch die unmittelbare Sichtbarkeit des Programmes die Kommunikation zwischen Kunden und Entwickler wesentlich verbessert. Dadurch sollen frühzeitig Änderungswünsche des Kunden erkannt und somit das Risiko späterer Änderungen möglichst verringert werden. Umgekehrt soll sich durch die frühzeitige Abstimmung der vom Benutzer geforderten Eigenschaften mit den tatsächlich im Produkt umgesetzten Eigenschaften die spätere Benutzerakzeptanz des fertigen Produktes erhöhen.

Bei der Durchführung des Prototypings reduziert man die Anforderungen an den Prototypen gegenüber einem späteren Produkt. Dadurch soll in erster Linie die Prototypenentwicklung beschleunigt werden. Zum Beispiel kann man einen Prototypen mit verringertem Funktionsumfang, geringerer Qualität oder einer verringerten Effizienz erstellen. Man versucht nach dem Bau eines initialen Prototypen durch Ausprobieren neue Anforderungen zu entdecken.

Zum Prototyping selbst existieren verschiedene eigene Lebenszyklen (siehe zum Beispiel [Som96], [Dav92]). Diese behandeln lediglich die Entwicklung und Erweiterung des Prototypen. Andere versuchen das Prototyping in Prozeßmodelle des gesamten Software-Engineering einzubauen (siehe zum Beispiel [PPS92]).

Zur weiteren Beschleunigung der Erstellung des Prototypen gibt es unterschiedliche Ansätze (siehe [Som96] und [PPS92]). Einerseits existieren spezielle höhere Programmiersprachen. Andere versuchen, bestimmte vorgefertigte Bausteine wiederzuverwenden und zusammen zu montieren oder Frameworks für bestimme Forschungsschwerpunkte zu definieren. Schließlich gibt es auch noch die Möglichkeit aus unterschiedlichen graphischen Sprachen automatisch Code zu generieren (siehe zum Beispiel [HuS97]).

Wie bereits bei den Szenarien wird in manchen Ansätzen der Prototyp Teil der Spezifikation oder man versucht ihn in eine Spezifikation zu überführen. Weiterhin unterscheiden sich verschiedene Ansätze darin, ob der Prototypencode in der späteren Implementierungsphase zumindest teilweise wiederverwendet oder verworfen wird.

In [PoW97] werden Erfahrungen im Praxiseinsatz von Prototyping vorgestellt. Die wesentlichen Vorteile von Prototyping sind danach eine erhöhte Produktqualität und eine erhöhte Produktivität, da nur Funktionalitäten in das Produkt umgesetzt werden, die vorher auch im Prototypen ermittelt wurden. Die Dauer des Entwicklungsprozesses der Software erhöht sich zwar, doch ist damit gleichzeitig eine Verringerung der Testzeiten zu verzeichnen. Problematisch hingegen ist zunächst bei der Einführung eines Prototypingverfahrens der Einarbeitungsaufwand in entsprechende Werkzeuge. Ansonsten verbergen sich gerade in der Erstellung von Prototypen Gefahren. Einerseits kann sich der Spezifikationsprozeß unnötig in die Länge ziehen, da aufgrund der einfachen Änderbarkeit des Prototypen zu

viel an Details herumgespielt wird. Dies kann andererseits dann aufgrund einer Verzögerung des Projektablaufs zur letztendlichen Auslieferung eines unausgereiften Prototypen führen.

Aufgrund der vielfachen Vorarbeiten im Bereich Prototyping und der insgesamt positiven Erfahrungen erscheint es sinnvoll, den Einsatz von Prototypingkonzepten im Hinblick auf unsere Forschungsgebiete näher zu betrachten und auf spezifische Bedürfnisse der Forschungsschwerpunkte anzupassen.

Spezielle Techniken für Standardsoftware

Standardsoftware zeichnet sich dadurch aus, daß sie nicht für einen einzelnen Auftraggeber entworfen wird, sondern für viele unterschiedliche Anwender gedacht ist. Dieser Unterschied wirkt sich nicht nur auf die konkrete Tätigkeit des Herausarbeitens von Anforderungen aus, sondern auch auf zusätzliche Aktivitäten, die im Umfeld der Anforderungsdefinition durchgeführt werden müssen.

Produktneuentwicklung

So geht dem Prozeß der Entwicklung eines neuen Produktes zunächst eine grobe Definition eines Zielmarktes voraus. Diese bestimmt mögliche Zielgruppen und mögliche durch das Produkt zu befriedigende Bedürfnisse. Weiterhin wird grob abgeschätzt, inwiefern sich in dem ausgewählten Zielmarkt die Positionierung eines neuen Produktes überhaupt rentieren könnte (siehe zum Beispiel [Kot97]).

Bevor ein konkretes Produkt entwickelt werden kann, müssen zuerst Ideen gesammelt werden. Eine Firma kann sich dabei interner oder externer Quellen bedienen. Diese werden in einem mehrstufigen Prozeß gefiltert, erprobt, nach Machbarkeit untersucht, in Prototypen evaluiert, nach möglichem Markterfolg überprüft und konkretisiert. Die Auswahl der Ideen orientiert sich dabei wesentlich an der Firmenstrategie. Ist beispielsweise eine Firma bereit, auch höhere Risiken einzugehen, so können auch Ideen mit höherem Innovationsgrad umgesetzt werden, deren Markterfolg meist nicht mit allzuhoher Sicherheit vorausgesagt werden kann.

Zum firmeninternen Finden von Ideen werden häufig Kreativitätstechniken wie Brainstorming, Brainwriting, der morphologische Kasten und die Synektik angewendet (näheres siehe zum Beispiel in [Hau93]). Firmenextern können prinzipiell die gleichen Quellen genutzt werden, wie sie anschließend bei der Produktweiterentwicklung erläutert werden, wobei die Informationsmenge mit wachsendem Innovationsgrad in der Regel sehr stark abnimmt, beziehungsweise manche Quellen oder Kanäle gar nicht erst genutzt werden können. So hat eine sehr innovative Firma neben dem großen Erfolgsrisiko auch eine sehr hohe Unsicherheit bezüglich der Erstellung von ersten Anforderungen. Hier findet meist kein reguläres Requirements Engineering statt, sondern es wird versucht, aufgrund von wenigen Ideen ein Produkt zu entwerfen, das eventuell erst neue Bedürfnisse auf einem Markt generieren wird.

Am Ende dieses Ideenfindungs- und Ideenauswahlprozesses steht ein grobes Produktkonzept, das Ausgangspunkt für die konkrete Entwicklung eines Produktes ist.

Produktweiterentwicklung

Um den Erfolg marktverkaufter Software über einen längeren Zeitraum hinweg sichern zu können, müssen neben der Entwicklung neuer Produkte bereits vorhandene Produkte kontinuierlich verbessert werden. Daher ist das Herausarbeiten von Anforderungen für Standardsoftware ein kontinuierlich über den gesamten Lebenszyklus eines Produktes hinweg andauernder Prozeß.

Die zu entwickelnde Standardsoftware soll in der Regel die Anforderungen mehrerer Zielgruppen erfüllen, so daß man sich einer größeren Menge von Informationsquellen bedienen muß. Alle Informationen sind mit einer mehr oder weniger großen Unsicherheit verbunden, kommen in unterschiedlich langen Zeiträumen überhaupt erst zum Tragen und haben unterschiedliche Granularitäten, so daß hier meist viel Gespür und Erfahrung des Informationssammlers notwendig ist, um Anforderungen an eine kommende Produktversion herauszuarbeiten.

Aus Sicht einer Standardsoftware entwickelnden Firma unterscheiden wir zwischen externen und internen Quellen von Anforderungen. Zu den externen Quellen zählen vor allem die Kunden, Konkurrenzprodukte, Basissysteme und Forschungseinrichtungen.

Die firmenextern gesammelten Informationen stammen nicht nur von verschiedenen Quellen sondern werden auch über unterschiedlich Kanäle an die Firma herangetragen. Falls noch keine Vorgängerversion eines Produktes existiert, können hiervon natürlich nicht alle genutzt werden. Abbildung 1 zeigt eine Übersicht über mögliche Quellen und Kanäle von Anforderungen.

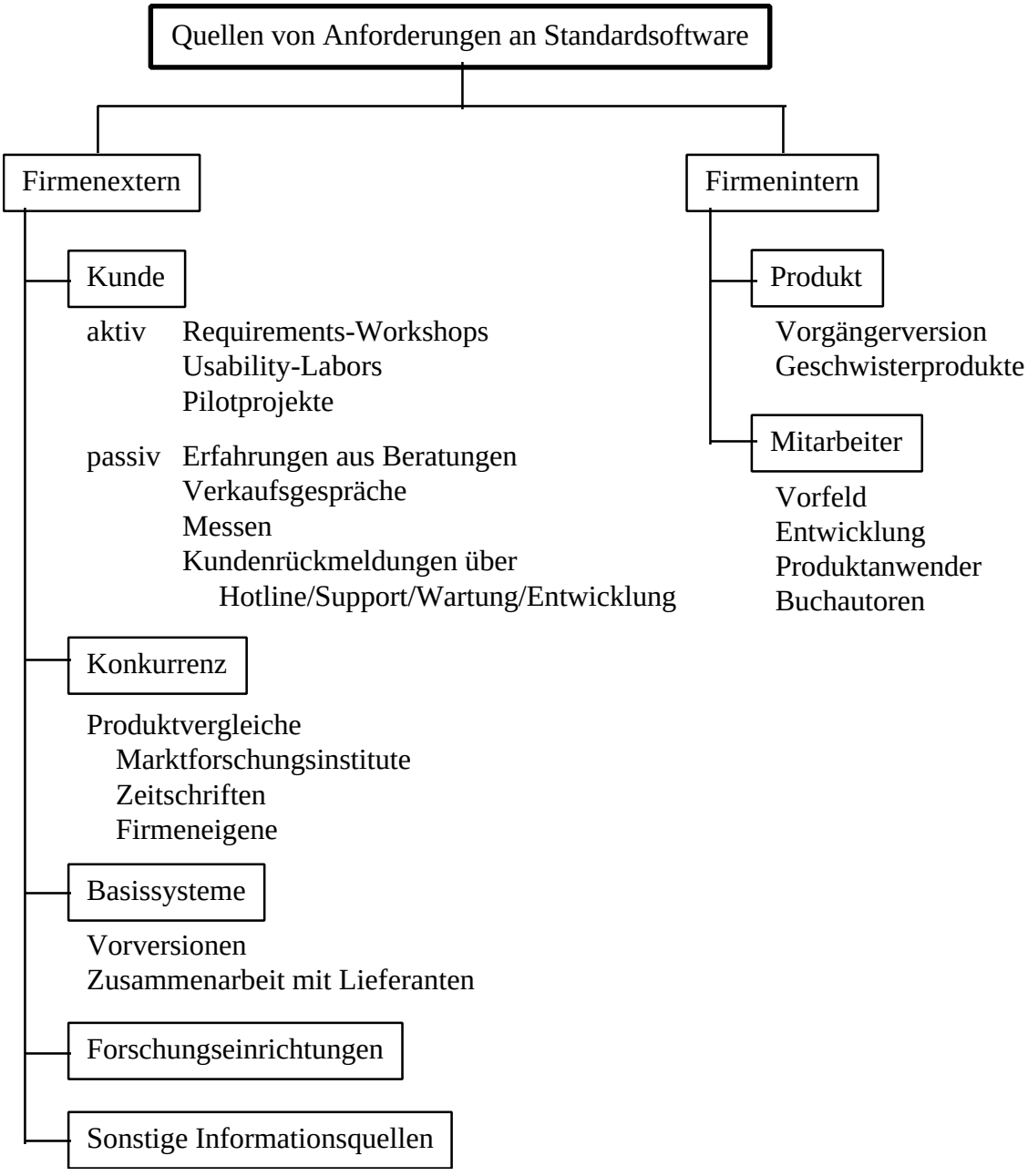

Abbildung 1 - Quellen und Kanäle von Anforderungen an Standardsoftware

Firmenexterne Quellen

Kunde

Zunächst zählen hierzu von der Firma gezielt zur Anforderungsdefinition mit Kunden durchgeführte Gespräche beziehungsweise Workshops. Dazu werden zuerst aus den festgelegten Zielgruppen mögliche Gesprächspartner ausgewählt. Mit diesen werden je nach Fortschritt der laufenden Produktdefinition der zu erstellenden Standardsoftware gezielte Gespräche zum Finden von neuen Ideen geführt, Folienpräsentationen, erste Mock-Ups und Prototypen von bereits vorhandenen Ideen vorgeführt sowie grobe Ablaufszenarien des entstehenden Produktes durchgespielt. Diese Sitzungen dienen häufig gleichzeitig dem Herausarbeiten, dem Verhandeln und der Validierung von Anforderungen.

Mit den gleichen Kundengruppen können bei bereits weiter in der Entwicklung fortgeschrittenen Produkten auch Usability-Workshops und Pilotprojekte durchgeführt werden. Beide dienen in erster Linie des Feinschliffes des Produktes. Die Pilotprojekte sind meist auch zusätzlich ein gutes Instrument, Anforderungen für die übernächste Version zu sammeln, da die Integration neuer Anforderungen in die nächste Version in dieser späten Entwicklungsphase nicht mehr möglich ist, um deren Zeitpunkt des Markteintritts nicht zu gefährden.

Die gezielten Gespräche haben vor allem den Vorteil, daß sie vorbereitet werden können, mit den gewünschten Zielgruppen durchgeführt werden können und somit konzentriert die gewünschte Art von Information liefern können. Sie sind jedoch gleichzeitig verbunden mit einem hohen Aufwand der Vorbereitung und der Durchführung. Um den Aufwand der Durchführung zu verringern, setzt man teilweise auch schriftliche Umfragen an die ausgewählten Zielgruppen ein.

Neben dem aktiven Herangehen an den Kunden, um Informationen zu gewinnen, existieren auch verschiedene andere Kanäle, über die Anforderungen vom Kunden gesammelt werden können. Wertvolle Kanäle sind Erfahrungen aus Beratungsprojekten von Kunden unter Anwendung des eigenen Produktes (vergleiche zum Beispiel [CuS95]**)**, Informationen aus Verkaufsgesprächen, Gespräche mit Kunden auf Messen und aktive Kundenrückfragen sowie Anregungen über Hotline, Support, Wartung, Entwicklung. Der Nachteil dieser Kanäle ist, daß hier keine gezielte Vorauswahl strategischer Zielgruppen getroffen werden kann und somit die Streuung an Informationen wesentlich größer ist.

Praxiserfahrungen bei unserem Industriepartner haben gezeigt, daß Befragungen der Kunden häufig nicht den gewünschten Erfolg bringen. Bei offenen Fragen nach ihren Anforderungen werden häufig Dinge genannt, die entweder bereits umgesetzt sind oder so spezifisch für ihre speziellen Bedürfnisse sind, daß sie nicht in das Produkt umgesetzt werden können. Insbesondere sind nur sehr wenige vom Kunden geäußerte Anforderungen Innovationen. Häufig hingegen sind sich die Kunden vielmehr bewußt, was sie nicht haben wollen. Im Geschäftskundenbereich können jedoch umgekehrt Großunternehmen und kleine innovative Unternehmen gute Anregungen an ein Produkt liefern. Diese Erfahrungen sind allerdings sehr von dem durch das Produkt angesprochenen Markt abhängig.

Für alle direkt mit den Kunden geführten Gespräche und Workshops lassen sich die eingangs vorgestellten Techniken der spezifischen Literatur des Requirements Engineering anwenden.

Konkurrenz

Eine weitere wichtige Quelle von Informationen ist die Analyse von Konkurrenzprodukten. Man verfolgt damit einerseits das Ziel, aus diesen Produkten Anforderungen für das eigene Produkt aufzuspüren und andererseits Anhaltspunkte zu erhalten, wie sich das eigene Produkt von diesen differenzieren kann, speziell wie einzelne Anforderungen gewichtet werden sollen (vergleiche [Pfe93]).

Man kann die Vergleiche von Konkurrenzprodukten sowohl selbst durchführen als auch auf externe Quellen zurückgreifen. Externe Quellen können sowohl von Marktforschungsinstituten durchgeführte Studien als auch Vergleichstests in Zeitschriften sein. Während der interne Test den Vorteil bietet,

eigene Kriterien in den Vordergrund zu stellen, kann bei externen Vergleichen neben dem reinen Produktvergleich auch die Wahl der Vergleichskriterien Aufschluß über eine mögliche Gewichtung der Anforderungen geben.

Basissysteme

Abhängig von der Art der Software bettet sich diese in unterschiedliche Basissysteme ein, beziehungsweise nutzt deren Dienste. Dazu zählen zum Beispiel Betriebssysteme und Datenbanksysteme aber auch andere spezialisierte Anwendungen, die mit der zu entwickelnden Standardsoftware zusammenarbeiten sollen. Da sich diese Systeme selbst laufend in kurzen Versionszyklen weiterentwikkeln, sind sie auch Quellen für neue Anforderungen an die Standardsoftware.

Informationskanäle sind hier die Dokumentation und Vorversionen der Basissysteme, Gespräche mit Lieferanten oder auch die intensive Zusammenarbeit mit dem Lieferanten durch Personalaustausch zwischen dem Hersteller des Basissystems und dem Hersteller der Standardsoftware.

Eine weitere Forderung an manche Standardsoftware ist die Offenheit des Produktes. Offenheit heißt dabei im weitesten Sinne die Möglichkeit der Interaktion anderer Softwareprodukte mit dem Produkt, wobei diese selbst dem Produkthersteller im voraus nicht bekannt sein müssen. Die Offenheit wird häufig durch das Angebot von standardisierten Schnittstellen unterstützt. Da auch Standardschnittstellen sich im Laufe der Zeit verändern, ergeben sich auch von dieser Seite her neue Anforderungen. Informationen darüber erhält man über Standardisierungsgremien oder über entsprechende Fachpublikationen.

Forschungseinrichtungen

Forschungseinrichtungen können sowohl durch Produktvergleiche als auch durch spezifische Vorfeldprojekte die Produktweiterentwicklung unterstützen. Sie können außerdem durch wissenschaftliche Herangehensweisen weitere Verbesserungspotentiale an den Produkten aufspüren.

Sonstige Informationsquellen

Außer den oben genannten Quellen lassen sich je nach Branche und Produkt auch Meinungen zum Produkt aus der Tagespresse als Anforderungsquelle nutzen. Daneben können beobachtbare Trends im Produktumfeld als auch in anderen Bereichen Quellen von Anforderungen bieten.

Firmeninterne Quellen

Produkt

Wenn der Kunde eine neue Version eines Softwareproduktes kauft, erwartet er implizit die Kompatibilität zu seiner Vorgängerversion. Er setzt also voraus, daß gewisse Funktionalitäten des Programmes weiterhin vorhanden sind und sich genauso verhalten, wie in der letzten Version. Da sich eine Neuerung auch auf die Funktionalität und auf die Bedienbarkeit auswirkt, muß darauf geachtet werden, inwieweit auf die Kompatibilität Rücksicht genommen wird.

Bei komplexeren Produkten wird die Standardsoftware in mehrere isoliert verkaufte Produkte aufgeteilt. Dies hat den Vorteil, daß sie getrennt entwickelt werden können. Häufig nutzen die einzelnen Produkte auch Funktionalitäten der einzelnen Geschwisterprodukte, so daß sich daraus auch Anforderungen an das gegenseitige Zusammenspiel ergeben.

In der Praxis werden diese Anforderungen erst im Laufe der Konkretisierung aller anderen Anforderungen an das Softwareprodukt festgelegt. Eine gesonderte Methodik existiert hierfür bisher nicht.

Mitarbeiter

Neben Mitarbeitern, die aus externen Quellen Anforderungen sammeln, können auch innerhalb der Firma unterschiedliche Mitarbeiter Ideen für Anforderungen an das Produkt liefern. Die Ideen können sowohl systematisch gesucht werden, oder auch zufällig entstehen.

Beispielsweise in Vorfeldabteilungen wird versucht durch die Kombination von Vorfeldprojekten, extern gewonnenen Daten und Kreativitätstechniken sowohl Ideen für künftige Produkte als auch konkrete Anforderungen an die Weiterentwicklung vorhandener Produkte zu kreieren.

Auch die Entwicklungsabteilung, die sich eigentlich in erster Linie mit der konkreten Umsetzung der Anforderungen in das Produkt befaßt, ist eine mögliche Quelle von neuen Anforderungen. So versucht zum Beispiel Microsoft die Kreativität der Entwickler gesondert zu fördern, indem man in jeder Entwicklungsphase gesonderte Pufferzeiten eingeführt hat, währenddessen die Entwickler eigene Ideen entwerfen können (vergleiche [CuS95]).

Weitere interne Quellen können Mitarbeiter sein, die selbst mit dem Produkt arbeiten, und über das betriebliche Vorschlagswesen Ideen melden. Nicht zuletzt können auch Buchautoren, die Anleitungen zur Benutzung der Standardsoftware erstellen, wertvolle Informationen zur Produktgestaltung liefern.

Fazit

Zum Herausarbeiten von Anforderungen an Standardsoftware existieren für die unterschiedlichen Quellen und Kanäle verschiedene Methoden, die jede für sich noch weiter verbessert und mehr systematisiert werden kann. Da viele Informationen auf der Eigeninitiative von nicht unmittelbar am Entwicklungsprozeß beteiligten Personen basieren, ist es darüber hinaus notwendig, die Methoden immer mit adäquaten Anreizsystemen zu koppeln, um die Motivation zur Informationsübermittlung zu fördern. Dazu hilft beispielsweise die Ausschreibung von Prämien für innerbetriebliche Verbesserungsvorschläge.

Da die Informationen von einem größeren Personenkreis gesammelt und auch weiterverarbeitet werden, ist vor allem aber auch notwendig, alle gesammelten Informationen adäquat zu erfassen, um keine der gewonnenen Erkenntnisse verloren gehen zu lassen. Dazu ist die Erforschung einer entsprechenden Systematik dringend notwendig.

3.2.2 Verhandlung (Negotiation)

In der Regel werden die gesammelten Anforderungen von verschiedenen Personen zusammengetragen. Aufgrund deren unterschiedlichen Rollen und Ziele in einem Projekt ist es normal, daß dabei Konflikte zwischen verschiedenen Anforderungen auftreten. Diese sollen in der Verhandlungsphase aufgedeckt und verschiedene Lösungsalternativen gesucht werden. Bei der Entwicklung von Individualsoftware wird daraus schließlich eine der möglichen Alternativen ausgewählt. Für Standardsoftware wählt man gegebenenfalls mehrere Alternativen aus, um für unterschiedliche Kundengruppen jeweils adäquate Lösungen anbieten zu können. Falls es nicht möglich ist, alle ausgewählten Alternativen aufgrund von Zeit- oder Kostenrestriktionen zu realisieren, muß auch noch eine Priorisierung der einzelnen Anforderungen untereinander erfolgen.

Konfliktauflösung mit Viewpoints

Im Bereich der Verhandlung wird die Forschung um Viewpoints angesiedelt. Diese hat zum Ziel, Konzepte für Modellierungstechniken zu entwerfen, die es ermöglichen, bereits während der Erstellung einer Anforderungsspezifikation Konflikte automatisch zu erkennen und deren Auflösung zu unterstützen. Die Forschungsgruppe von Prof. Finkelstein beschäftigt sich sehr intensiv mit dieser Problematik. Eine Anforderungsspezifikation besteht nach seiner Forschungsarbeit aus unterschiedlichen Sichten auf ein zu entwickelndes System. Eine Sicht dient hauptsächlich dazu, jeweils bestimmte

Aspekte des Problembereichs in den Mittelpunkt der Betrachtung zu stellen. Solche Unterteilungen können einerseits aufgrund verschiedener Entwickler, andererseits aufgrund unterschiedlicher Beschreibungstechniken notwendig sein.

Eine Modellierungstechnik wird nach diesem Konzept aus unterschiedlichen Sichtenvorlagen (Viewpoint-Templates, [NKF94]) aufgebaut. Eine Sichtvorlage enthält Grundelemente einer Sicht und einen sogenannten Arbeitsplan, der Konstruktionsschritte vorschreibt, in welcher Weise die Grundelemente miteinander kombiniert werden dürfen, um sukzessive eine Sicht aufzubauen. Zum Beispiel kann eine Sichtenvorlage die Konstruktion von Zustandsautomaten beschreiben. Sie enthält unter anderem als Grundelemente Zustände und Transitionen. Ein Konstruktionsschritt ist hier beispielsweise das Einfügen einer neuen Transition, die nur zwischen zwei bereits vorhandenen Zuständen eingefügt werden darf.

Weiterhin enthält der Arbeitsplan Konsistenzbedingungen, die zwischen Grundelementen innerhalb einer oder zwischen unterschiedlichen Sichten gelten müssen und Prüfaktionen, die zu einem gegebenen Zeitpunkt die Erfüllung dieser Bedingungen nachprüfen sollen. Sowohl Konstruktionsschritte als auch Prüfaktionen sollen derart exakt formuliert sein, daß sie später in ein Werkzeug zur Unterstützung der Modellierungstechnik integriert werden können. Für die Modellierungstechnik insgesamt kann außerdem ein Prozeßmodell festgelegt werden. Dieses definiert zum Beispiel, wann welche Prüfaktionen der verschiedenen Sichten ausgeführt werden sollen, und wann beispielsweise Konfliktauflösungen stattfinden sollen.

Mit Hilfe der Viewpointtechniken läßt sich eine Requirements Engineering Methode entwerfen, die eine verteilte Entwicklung erlaubt und über die Prüfaktionen Konflikte aufdeckt und eventuell auch Auflösungsalternativen anbietet. Die Arbeiten enthalten wertvolle Grundkonzepte zur Lösung des Verhandlungsproblemes, sind jedoch nicht unmittelbar zur Umsetzung in die Praxis geeignet.

Auswahl von Alternativen

Da ein Hauptaspekt der Verhandlungsphase die Entscheidung für eine oder mehrere Alternativen ist, werden in der Praxis manchmal auch entscheidungsunterstützende Systeme eingesetzt. Wie bereits erwähnt, können auch verschiedene festgehaltene alternative Szenarien zur Entscheidungsfindung in bestimmten Fällen verwendet werden.

Priorisierung von Anforderungen

Sowohl bei der Erstellung von Individualsoftware, als auch bei der Erstellung von Standardsoftware spielt die Priorisierung ausgewählter Anforderungen aufgrund von beschränkten Zeit-, Personal- und Kapitalressourcen eine wesentliche Rolle. Bei Individualsoftware befragt man hierzu im Zweifelsfalle unmittelbar den Kunden.

Bei der Entwicklung von Standardsoftware muß man sich wieder verschiedener Quellen und Kanäle bedienen. Wesentlich für die Priorisierung ist hierbei in erster Linie das mit einer Anforderung aus quantitativer Sicht verbundene Marktpotential und der durch Erfüllung einer Anforderung aus qualitativer Sicht erzielte Kundennutzen. Die Priorisierung soll dabei möglichst so feingranular sein, daß die Entwicklung unter Zuhilfenahme der jeweils geschätzten Aufwände und der Prioritäten zu der Entscheidung einer Anforderungsalternative kommen kann.

Zur Priorisierung der Anforderungen bezüglich des Kundennutzens gibt es grundsätzlich zwei unterschiedliche Möglichkeiten. Entweder man stützt sich auf die während des Herausarbeitens der Anforderungen gewonnenen Daten oder man ermittelt quantitativ meßbare Daten mittels Fragebogenaktionen oder Umfragen.

Aufgrund der großen Streuung der Anforderungen bezüglich ihrer Quellen und Kanäle lassen sich bei der ersten Methode keine quantitativ vergleichbaren Daten ermitteln, so daß die Priorisierung der

einzelnen Anforderungen in erster Linie von der Erfahrung und dem Gespür des Entwicklers abhängt. Er kann sich dabei vor allem an der Produktstrategie der Standardsoftware stützen, die bereits bei der ersten Produktplanung grob die einzelnen Ausbaustufen des Produktes umrissen hat. Weitere Anhaltspunkte können bereits erfüllte Anforderungen der Konkurrenz und Bewertungskriterien von Vergleichstests von Konkurrenzprodukten sein.

Mit der zweiten Methode ist ein erheblich höherer Aufwand verbunden, da hier Kundenbefragungen organisiert werden müssen. Es gibt unterschiedliche Methoden, in welcher Art Befragungen durchgeführt werden. Wichtig ist vor allem eine Abbildung der Kundenpräferenzen auf Zahlen, um später eine quantitative Auswertung vornehmen zu können.

Einige Methoden, wie zum Beispiel Checklisten und Category Scaling (siehe [Sab76] und [Shi92]), gehen von der gegenseitigen Unabhängigkeit einzelner Bewertungskriterien aus. Der Befragte bewertet hier verschiedene Anforderungen nach ihrer Wichtigkeit, indem er sie in eine vorgegebene geschlossene Skala einordnet. Aufgrund der Unabhängigkeit der einzelnen Anforderungen eignen sich diese Methoden auch für eine größere Anzahlen von Kriterien.

Andere Methoden gehen umgekehrt davon aus, daß einzelne Anforderungen nicht unabhängig voneinander sind. So werden zum Beispiel bei der Conjoint Analysis (siehe zum Beispiel [Mef92]) dem Befragten unterschiedliche Alternativen eines Produktes mit unterschiedlichen Erfüllungsgraden der verschiedenen Anforderungen präsentiert, woraus dieser die bevorzugte Variante auswählt. Eine Auswertung der Befragung läßt daraus Rückschlüsse für die Wichtigkeit der einzelnen Anforderungen der Produkte zu. Diese Art von Methoden lassen sich nur bei kleineren Anzahlen von Attributen anwenden.

Neben dem hohen Aufwand von Kundenbefragungen wird in der Praxis vor allem kritisiert, daß die Befragten häufig nicht wirklich differenzierte Antworten liefern. Meistens werden alle Anforderungen als sehr hoch prior eingeschätzt.

Dem Kundennutzen steht von der Entwicklungsseite her der für die Umsetzung einer Anforderung notwendige Aufwand gegenüber. Wesentlich ist hierbei für einen Hersteller von Standardsoftware vor allem auch die Konzentration auf seine Kernkompetenzen. Das heißt, es muß auch herausgearbeitet werden, ob die Erfüllung einer Kundenanforderung wirklich denjenigen Funktionalitäten entspricht, die er unterstützen will oder ob es sich hier um Anforderungen handelt, die ein Kunde auch durch andere Standardprodukte abdecken könnte.

So kann beispielsweise eine Kundenanforderung lauten, daß er eine Versionsverwaltung von Programmen in ein Entwicklungswerkzeug integriert haben möchte. Wenn dies ein Werkzeughersteller nicht als seine Kernkompetenz betrachtet, könnte er beispielsweise durch das Angebot einer Schnittstelle seiner Software zu käuflich erheblichen Versionsverwaltungssystemen der Kundenanforderung nachkommen, ohne ein eigenes komplettes Versionsverwaltungssystem integrieren zu müssen. Dies ist dann eine sozusagen eine von der Entwicklung zum Kunden hochgezogene ”make versus let-buy”-Entscheidung.

Die Aufwände für Anforderungen, welche die Kernkompetenz eines Herstellers von Standardsoftware betreffen, müssen geschätzt werden und in ein geeignetes Verhältnis dem Kundennutzen gegenübergestellt werden.

Insgesamt existieren noch wenige Ansätze, wie Anforderungen gegeneinander priorisiert werden können, die in die Praxis umgesetzt werden können. Es besteht daher hier noch dringender Forschungsbedarf.

Sonstiges

Wesentlich neben der technischen Unterstützung des Verhandlungsprozesses ist die Beteiligung der richtigen Personen an den einzelnen Entscheidungen. Sobald eine Entscheidung getroffen worden ist,

müssen auch Gründe für diese dokumentiert werden, um einerseits den Weg zu einer bestimmten Lösung transparent zu halten und andererseits auch später die Möglichkeit zu haben, Entscheidungen zu revidieren und abzuändern.

Für diesen Bereich existiert insgesamt noch sehr wenig konkrete praktische Unterstützung für das Requirements Engineering und es besteht noch ein großer Forschungsbedarf an Grundlagen, so daß man sich hier zunächst noch rein pragmatischer Hilfen bedienen muß.

3.2.3 Dokumentation/Spezifikation (Documentation/Specification)

Die Dokumentationsphase dient der Übersetzung der auf unterschiedlicher Weise erfaßten Anforderungen während der Herausarbeitungs- bzw. der Verhandlungsphase in eine Spezifikation.

Letztendlich bildet die Spezifikation die Grundlage für die Erstellung des Systems. Sie dient einerseits als Kommunikationsmittel zur Validierung und zur weiteren Herausarbeitung von Anforderungen zwischen Kunde und Entwickler. Andererseits soll sie dem Entwickler alle benötigten Informationen zur Systementwicklung liefern. Schließlich soll sie auch die Basis für eine Verifikation nach der Entwicklung sein. Spätestens, wenn man vom Requirements Engineering zur konkreten Entwicklung übergeht, muß die Spezifikation in sich konsistent sein.

Um diese Aufgaben erfüllen zu können, muß die Spezifikation einerseits in für verschiedene Personengruppen verständlichen Sichten dargestellt sein. Andererseits soll sie auf einer möglichst formalen Basis fundieren, um dem Entwickler eine eindeutige Vorgabe zur Erstellung des Systems zu geben und auch die Verifikation durch automatische Werkzeuge unterstützen zu können (siehe auch Abschnitt 3.2.4 Validierung/Verifikation).

In der gängigen Praxis wird heute meist noch beliebig gegliederter Prosatext eingesetzt. Es haben sich hierfür einige Standards und Vorgehensmethoden entwickelt, die in [DoT90] zusammen vorgestellt werden. Diese bieten in erster Linie Schemata zur Klassifikation verschiedener Typen von Anforderungen an und schlagen für das Dokument einheitliche Gliederungen vor.

Mehrfach wurden bereits formale Spezifikationssprachen vorgeschlagen (zum Beispiel SPECTRUM [BFG+93]), um Spezifikationen zu beschreiben. Da diese jedoch selten die gewünschte Akzeptanz bei Anwendern von Software finden, können diese in erster Linie zur Kommunikation zwischen den für unterschiedliche Phasen der Softwareentwicklung zuständigen Entwicklern eingesetzt werden.

Häufig wird versucht, Teilaspekte der Spezifikation mit graphischen Beschreibungstechniken zu unterstützen. Der Vorteil gegenüber einer rein textuellen Spezifikation liegt in erster Linie in der Übersichtlichkeit und in ihrer intuitiven Darstellung. In den meisten Software Engineering Methoden werden Beschreibungstechniken für Daten, Abläufe und Verhalten angeboten. Diese werden jedoch abhängig von der Methode teilweise sehr unterschiedlich eingesetzt. Vergleicht man aber zwischen den Methoden jeweils die Beschreibungstechniken für einen Aspekt, so stellt man nur geringe Unterschiede in ihrer Ausdrucksfähigkeit und in ihrer graphischen Darstellung fest.

Aus dieser Erkenntnis heraus ist im Bereich der zur Zeit aktuellen Objektorientierten Methoden die Unified Modeling Language (UML) entstanden. Diese stellt eine Sammlung und eine Vereinheitlichung von Beschreibungstechniken aus unterschiedlichen Methoden zur Darstellung der verschiedenen Aspekte in der Softwareentwicklung dar. Man versucht die UML als künftigen Standard für Beschreibungstechniken im Software-Engineering vorzuschreiben. Wie in [BHH+97] festgestellt wurde, fehlt den Techniken jedoch eine entsprechende formale Semantik. Diese würde einerseits den richtigen Umgang mit den Beschreibungstechniken unterstützen und andererseits könnten die Sichten ineinander integriert werden, so daß Konsistenztests in und zwischen den einzelnen Sichten durchführbar wären. In dem Projekt SysLab an unserem Lehrstuhl wurde an einer formal fundierten Integration von als zentral angesehenen Beschreibungstechniken gearbeitet (siehe [BGH+97]).

Durch die Einführung einer eindeutigen Semantik graphischer Beschreibungstechniken wird sich die Verständigung zwischen an verschiedenen Phasen beteiligten Entwicklern verbessern. Zur Kommunikation zwischen Kunden und Entwicklern können diese Beschreibungstechniken auch verwendet werden. Jedoch müssen die Entwickler dann darauf achten, daß Kunden meist ein individuelles intuitives Verständnis einer Beschreibung haben, das nicht genau mit deren eindeutigen Semantik übereinstimmt. Trotzdem kann die Beschreibung mit Hilfe einer graphischen Beschreibungstechnik die Verständigung zwischen Kunden und Entwicklern verbessern.

Inwieweit die Beschreibungstechniken aus dem Software-Engineering jedoch künftig in verschiedenen Anwendungsbereichen akzeptiert werden, ist zur Zeit noch offen. Es zeigt sich jedoch mittlerweile allgemein ein starkes Interesse in der Industrie an der UML, was erhoffen läßt daß hier eine Reihe von Anwendungsbereichen erprobt werden.

Zur Erhöhung der Akzeptanz könnte es sinnvoll sein, bereits im Anwendungsbereich für andere Zwecke eingesetzte Beschreibungstechniken für die Beschreibung von Software zu adaptieren und auf Software Engineering Beschreibungstechniken abzubilden. Möglicherweise erhöht auch die Anreicherung der Software Engineering Beschreibungstechniken um syntaktische Elemente, wie zum Beispiel Maschinensymbole anstatt bisher üblicher einfacher geometrischer Formen die Nähe zur Anwendung.

Die in diesem Bereich bereits umfangreichen grundlegenden Forschungsarbeiten versprechen in naher Zukunft eine gute Basis, auf die ein Praxiseinsatz aufsetzen kann. Daher erscheint hier auch bereits jetzt eine anwendungsorientierte Anpassung als sinnvoll. Bezogen auf unsere Anwendungsgebiete kann die Betrachtung von Beschreibungstechniken aus den Maschinenbau zu einer Verringerung der Kommunikationsbarrieren zwischen Ingenieuren und Softwareentwicklern führen. Da sowohl beim Anlagenbau als auch bei der Entwicklung eines SPS-Softwarewerkzeuges die Kunden zumindest teilweise aus dem Bereich Maschinenbau kommen, kann dies möglicherweise für beide Forschungsschwerpunkte gleichzeitig nutzbringende Ansätze liefern. Gleichzeitig ist jedoch weiterhin auch die Formalisierung der graphischen Software Beschreibungstechniken anzustreben.

Verfolgbarkeit (Traceability)

Neben der reinen Beschreibung der Anforderungen an ein zu erstellendes System muß die Dokumentation die Voraussetzung für die Verfolgbarkeit von Anforderungen schaffen. Ziel ist dabei eine konsistente und lückenlose Sicherstellung der Kundenanforderungen in der letztendlichen Umsetzung. Man will einerseits getroffene Entscheidungen dokumentieren um die Entstehung einer Lösung nachvollziehbar zu machen. Andererseits sollen bei der Änderung von Anforderungen die Auswirkungen auf eine Implementierung (und umgekehrt) einfach zu ermitteln sein. Somit muß also eine Verfolgbarkeit über alle Phasen der Softwareentwicklung hinweg sowohl vorwärts von den frühen Phasen in die späten Phasen als auch rückwärts möglich sein [Wie95].

Man unterscheidet zwischen der horizontalen und der vertikalen Verfolgbarkeit [Got95]. Die horizontale Verfolgbarkeit hat zum Ziel, den Verlauf der Entstehung der verschiedenen Versionen der Dokumente innerhalb einer Phase mit allen gefällten Entscheidungen nachvollziehbar zu machen. Dagegen soll die vertikale Verfolgbarkeit Verbindungen zwischen den verschiedenen aufeinanderfolgenden Phasen schaffen. In [GoF94] wird außerdem zwischen der Pre- und der Post-Verfolgbarkeit unterschieden.

Die Pre-Verfolgbarkeit befaßt sich mit der Verfolgbarkeit der Entstehung der Anforderungsspezifikation. Wesentlich ist hier die Herstellung eines Bezugs zwischen den Verursachern und der Anforderung selbst. Da der Verursacher nicht immer genau eine Person ist, versucht Gotel in [Got95] mit ihren Beitragsstrukturen ein Modell zu erstellen, welche die wechselseitigen Einflüsse verschiedener Personen und die verschiedenen Rollen der Personen bei der Entstehung einer Anforderung dokumentieren. Die Post-Verfolgbarkeit befaßt sich dann mit der Verfolgung der Umsetzung der Spezifikation in ein fertiges System.

Heutige Ansätze zur Lösung des Problems der Verfolgbarkeit setzen in erster Linie verschiedene Arten von Referenzen ein, die Artefakte verschiedener Versionen eines Dokuments beziehungsweise verschiedener Phasen zueinander in Beziehung setzen. Die Referenzen werden abhängig von der Methode nach verschiedenen Kriterien getypt. So gibt es beispielsweise in [RUP90] Strukturierungsreferenzen, die Artefakte zu einer übergeordneten Einheit zusammenfassen, historische Referenzen, die Artefakte verschiedener Dokumentenversionen verknüpfen und phasenorientierte Referenzen, die Artefakte verschiedener Projektphasen zueinander in Beziehung setzen. Über diese Referenzen lassen sich dann verschiedene Arten von Anfragen stellen.

Problematisch für den praktischen Einsatz erscheint mit dieser Lösung jedoch der Aufwand, der sich hinter der Verwaltung der zahlreichen Referenztypen verbirgt. Daher kommt Wieringa [Wie95] zu dem Schluß, daß der Aufwand, der in die Verfolgbarkeit gesteckt wird, sich an den verfügbaren Ressourcen und dem Reifegrad der Organisation (siehe Humphrey [Hum89]) orientieren muß. Seines Erachtens lohnt sich erst bei einem hohen Reifegrad diese Investition.

Das wesentliche Problem für die Verfolgbarkeit ist der meist fehlende klar definierte semantische Bezug zwischen den verschiedenen im Laufe der Entwicklung auftretenden Artefakten. Hier besteht noch dringender grundlegender Forschungsbedarf. Zumindest für einen Teil der Dokumentation wäre ein Lösungsansatz auf Basis formal fundierter Beschreibungstechniken zu verwirklichen. Grundlage wäre die Definition von klar definierten Konstruktionsregeln und Verfeinerungsregeln (wie zum Beispiel [SHB96] und [Rum96]) angefacht. Mit ihrer Hilfe könnte in einem Werkzeug durch ein automatisches Protokollieren der einzelnen Konstruktionsschritte jede Änderung an den verschiedenen Beschreibungen nachvollziehbar gemacht werden. Zusätzlich zu den einzelnen Schritten wäre dann lediglich zur Dokumentation von Entscheidungssituationen eine entsprechende manuelle Ergänzung von Informationen notwendig.

Die Verfolgbarkeit von Anforderungen ist ein generelles Problem im Software-Engineering, das in seinen Grundlagen noch nicht eingehend genug erforscht ist. Ein Heranführen von vorhandenen Ansätzen an den praktischen Einsatz in unseren Forschungsschwerpunkte erscheint uns daher nicht sinnvoll.

3.2.4 Validierung/Verifikation (Validation/Verification)

Die Begriffe Validierung und Verifikation werden häufig mißverständlich und vermischt gebraucht, obwohl sie in ihrer Intention zweierlei Dinge sind. Wir verstehen unter Validierung die Überprüfung, ob das zu entwickelnde System den Wünschen des Kunden entspricht. Eine Verifikation dagegen dient dazu, zu überprüfen, ob ein bereits erstelltes System den spezifizierten Anforderungen entspricht. Es existieren auch andere Definitionen für Validierung und Verifikation. So wird häufig Verifikation mit dem Beweisen von Eigenschaften des Systems verstanden, dagegen Validierung das nicht beweisbare Nachprüfen von Eigenschaften durch verschiedene Methoden. Wir beziehen uns hier jedoch auf die erste Unterscheidung (vergleiche zum Beispiel [Rak97]).

Validierung

Um Kosten zu reduzieren muß eine Validierung so früh wie möglich erfolgen, damit die weiteren Entwicklungsschritte auf einer möglichst sicheren Basis vollzogen werden können. Daher versucht man mit verschiedenen Techniken bereits nach der ersten Dokumentation von Anforderungen diese mit dem Kunden entsprechend zu validieren. Eine letztendliche Validierung kann erst anhand des fertigen Systems erfolgen.

Zur frühzeitigen Validierung betrachtet man das zu erstellende System aus einer Black-Box-Sicht (Dienstleistungssicht), wie es sich später einem Kunden präsentieren wird. Vorab kann eine vergleichende Analyse der dokumentierten Anforderungen auf Plausibilität gegenüber in bisherigen Projekten gesammeltem Anwenderwissen in einer Wissensbank vollzogen werden. Im Anschluß daran ist

eine intensive Zusammenarbeit mit dem Kunden ratsam um dessen Vorstellung mit denen des Entwicklers zur Deckung zu bringen. Diese kann beispielsweise durch ein schrittweises Durchsprechen einer Spezifikation im Rahmen eines mehrtägigen Workshops stattfinden. Hierzu können Prototyping- und Szenariotechniken eingesetzt werden.

Insbesondere für die Automatisierung von technischen Anlagen bieten sich auch Animationstechniken zur Visualisierung von Abläufen an (siehe auch Abschnitt 3.3 Aktivitäten für Automatisierungstechnische Systeme).

Der Bereich der Validierung ist noch nicht hinreichend erforscht. Wesentlicher Forschungsbedarf besteht hier vor allem, wie bereits im Abschnitt 3.2.3 erwähnt, in einer für den Benutzer verständlichen Spezifikationssprache als wichtiges Kommunikationsmedium. Für die praktische Umsetzung der Validierung selbst bieten sich bisher pragmatische Reviewtechniken an, die wohl für die bei uns betrachteten Anwendungsgebiete in gleicher Weise geeignet sind.

Verifikation

Zur Verifikation der richtigen Umsetzung der spezifizierten Anforderungen unterscheiden wir zwischen drei prinzipiell unterschiedliche Vorgehensweisen, der Inspektion, dem Testen und der formalen Verifikation. Für alle drei Tätigkeiten bildet die zuletzt erstellte Spezifikation die Grundlage.

Zur Inspektion finden im Laufe der Entwicklung der Software mehrere, meist klar strukturierte Reviewsitzungen statt. Daran nehmen vor allem einerseits die Softwareentwickler und andererseits Verantwortliche für die Anforderungsspezifikation teil. In den Sitzungen werden die Inhalte der verschiedenen Entwicklungsdokumente und der Programmlistings auf ihre Richtigkeit gegenüber der gestellten Anforderungen überprüft und gegebenenfalls entsprechende Korrekturmaßnahmen abgeleitet.

Das Testen basiert auf dem Überprüfen von eigens dafür erstellten Testfällen am gesamten oder an Teilen des lauffähigen Systems. Die Testfälle werden aus der Spezifikation abgeleitet. Falls die Spezifikation in einer formal definierten Form vorliegt, bietet sich auch hier die Möglichkeit einer automatischen Testgenerierung an.

Die formale Verifikation basiert auf dem automatischen Beweisen oder Nachprüfen von logischen Aussagen, die aus der Spezifikation abgeleitet wurden. Um diese durchführen zu können, ist es notwendig, daß sich die Anforderungsspezifikation und das fertige Programm in eine äquivalente logische Darstellung überführen läßt. Dies sollte möglichst automatisch passieren.

Zumindest für das Testen und der formalen Verifikation, ist auch aus Gründen der Verifizierbarkeit eine formale Repräsentation der Spezifikation notwendig. Die Durchführung der Verifikation selbst und der Test passieren in einer späteren Phase und werden deshalb in diesem Bericht nicht näher beleuchtet.

3.3 Aktivitäten für Automatisierungstechnische Systeme

Neben den angesprochenen allgemeinen Problemstellungen im Requirements Engineering hat [Bil97]mehrere Schwerpunkte zur Spezifikation von Anforderungen an automatisierten Produktionsanlagen identifiziert. Dazu gehören die Abbildung der Struktur der Anlagen, die Abbildung zeitbezogener Aspekte und die Abbildung sequentieller und paralleler Abläufe. Es gibt insgesamt sehr wenige speziell auf das Requirements Engineering ausgerichtete Aktivitäten in der Automatisierungstechnik. Wir wollen im Folgenden kurz die von uns identifizierten Schwerpunkte vorstellen.

In erster Linie zur besseren Abbildung von Struktur, Modularität, Hierarchie und zur Modellierung der Verteilung des realen Systems werden in der neueren Literatur verschiedene objektorientierte Ansätze (z.B. mit Shlear Mellor in [Dar95] oder mit Rubin Goldberg in [KoM94]) herangezogen.

Dabei werden technische Komponenten durch Objekte dargestellt. Die Kommunikationsbeziehungen und Aggregation werden durch Relationen im Objektmodell dargestellt.

Eine Mehrzahl der Forschungsaktivitäten im Requirements Engineering für die Automatisierungstechnik befaßt sich mit der Adaption von graphischen Beschreibungstechniken aus dem allgemeinen Software-Engineering. Neben der Nutzung von ER-ähnlichen Beschreibungsmitteln zur Strukturbeschreibung und Darstellung von Kommunikationsbeziehungen konzentriert man sich auf die Erweiterung von zustandsbasierten Sprachen um Konstrukte zur Beschreibung von Realzeitbedingungen. Dies wird beispielsweise durch die direkte Erweiterung von bereits üblichen Zustandsgraphen in [KoM94] versucht. Interessant erscheint hier der Ansatz in [DP95], der Viewpoint-Techniken einsetzt, um einerseits hierarchische Strukturen in Beziehung zu setzen und andererseits verschiedene Darstellungen von Zeitbedingungen verbindet.

Häufig wird in der Automatisierungstechnik Simulation eingesetzt. Diese dient meist zur Verifikation von Anforderungen und zur Optimierung von Umsetzungen (siehe beispielsweise [Bel95]). Während dies eher in der Designphase angesiedelt ist, wird auch versucht, Simulation in Kombination mit Animation zum Herausarbeiten und zur Validation von Anforderungen zu nutzen (siehe zum Beispiel [KrL97]). Dies erscheint insbesondere in der Automatisierungstechnik als vielversprechend, da gerade bei technischen Prozessen die realen Prozesse wie Bewegungen und Materialflüsse visuell nachgebildet werden können. Gleichzeitig erreicht man durch die Spezifikation, die der Simulation unterliegt eine formale Basis, die Grundlage für eine Verifikation bilden kann.

Der Bericht [FoR95] gibt eine Übersicht über den State of the practice in Automatisierungsprojekten in Schweden und gibt praxisrelevante Ratschlägen zur Verbesserung von kleinen Teilaspekten. Wir wollen hier die zwei wichtigsten Ergebnisse präsentieren. Zunächst wurde festgestellt, daß meistens die Wiederverwendung von Standardkomponenten angestrebt wird, um die Wartungskosten der Software niedrig zu halten. Dies führte jedoch in vielen Fällen nicht zum gewünschten Erfolg, da die Anpassungsarbeit an Standardkomponenten wesentlich teurer war, als die völlige Neuerstellung der Anlage gekostet hätte. Deshalb wird eine Kosten/Nutzen-Abwägung vorgeschlagen. Eine weitere interessante Feststellung wurde bei dem Herausarbeiten der Benutzeroberflächen festgestellt. Der Zeitaufwand und das Engagement der Kunden überstieg hier regelmäßig den durchschnittlichen Aufwand der Herausarbeitung anderer Teile der Software. Insbesondere verlor man sich hier in Details. Die Autoren interpretieren dies damit, daß gerade bei graphischen Oberflächen jeder mitreden kann. Sie schlagen zur Abhilfe des Problems eine klare Zielfestlegung für Bestandteile von Benutzeroberflächen und eine Vorgabe von Musteroberflächen vor.

Wir wollen hier noch kurz die Forschungsaktivitäten von zwei Forschungsgruppen erwähnen, deren Durchsicht unsererseits noch nicht abgeschlossen ist, sich jedoch als lohnend erweisen kann. Einerseits existiert eine Forschungsgruppe um Prof. Dubois, die eine formale RE-Sprache namens Albert (Agent oriented Language for Building and Eliciting Requirements for real-Time systems) entwickelt hat, die auf Grundlage der Integration der Datenaspekte aus dem Business-Bereich und der Aspekte im Bereich verteilter Realzeitsysteme entstanden ist. Unter den gleichen Integrationsprämissen ist auch bei Prof. Pomberger das Application-Framework ProcessTalk entstanden. Auf dieses aufbauend wurde ein Prototyping-Werkzeug entwickelt. In beiden Forschungsgruppen wurden jeweils eine Reihe von Fallstudien durchgeführt.

Die bisher untersuchte Literatur behandelt jeweils sehr isolierte und spezialisierte Fragestellungen, über deren Eignung für unser Forschungsprojekt wir zum momentanen Zeitpunkt noch keine Aussagen treffen können.

3.4 Aktivitäten für Standardsoftwareentwicklung

Unter Standardsoftware verstehen wir Software, die nicht für einen individuellen Kunden, sondern für einen Markt von vielen potentiellen Kunden entwickelt wird. Daraus ergeben sich eine Reihe von

Unterschieden zur Entwicklung von Auftragssoftware, die zum Beispiel in [CaB95] erörtert werden. Wir werden im Abschnitt 3.4.1 auf diesen Artikel etwas näher eingehen.

Die Problemstellungen für die Entwicklung von Standardsoftware sind in ähnlicher Weise auch bei anderen am Markt zu verkaufenden Produkten anzutreffen. Daher lassen sich Marketingaktivitäten, wie sie dort üblich sind, auf die Software übertragen. Hierzu gehören einerseits Marketingstrategien wie die Bestimmung von Zielmärkten, deren Größe und deren Verhalten, Gewinnziele, Preisbestimmung, Verkaufsstrategien, Marketingbudgets, langfristige Produktstrategien usw. und andererseits Verkaufsanalysen, das heißt die Abschätzung möglicher Umsätze aufgrund aktueller, vergangener und künftiger Umsatzzahlen anderer Produkte. Daß diese Strategien neben dem reinen Produkt auch in der Softwareindustrie einen wesentlichen Einfluß haben, zeigen zum Beispiel die unterschiedlichen Strategien von Microsoft, ihre Marktführerschaft beizubehalten beziehungsweise zu erkämpfen [Bag97].

Auch die Aktivitäten des Herausarbeitens von Anforderungen, wie wir sie bereits in Kapitel 3.2.1 ausführlich erläutert haben, sind nah verwandt mit dem Marketingresearch. Die Besonderheiten der Verhandlungsphase für Standardsoftware, insbesondere zur Priorisierung von Anforderungen wurden bereits in Kapitel 3.2.2 dargestellt.

Zur Minderung des Risikos und um den hohen Innovationsgrad von Software in den Griff zu bekommen, wird Standardsoftware in der Praxis in möglichst kurz aufeinanderfolgenden Versionen ausgeliefert. Man erhofft sich dadurch einerseits Rückmeldungen von Käufern des Produktes mit neuen Anforderungen für deren nächste Version. Andererseits kann man durch die kurze Zyklendauer schnell auf Innovationen und Konkurrenzprodukte reagieren.

Vor der Auslieferung der ersten Version eines Softwareproduktes wird meist eine Produktstrategie festgelegt. Dort wird grob aufgeschrieben, wie sich das Produkt von einer Ursprungsversion bis zu einer Endversion, in der alle anfänglich ausgewählten Anforderungen enthalten sein sollen, iterativ über die Versionen hinweg, entwickeln soll. Mittels der dadurch reduzierten Anforderungsmenge für die Ursprungsversion kann die Entwicklungszeit bis zum ersten Verkauf und auch zwischen zwei nachfolgenden Versionen erheblich verkürzt werden.

Bisher existieren sehr wenige Forschungsaktivitäten im Software-Engineering für Standardsoftware. In erster Linie wird versucht, aus dem Produktgeschäft anderer Produkte Methoden, wie zum Beispiel QFD (vgl. z.B.[Bro91]) zu adaptieren.

3.4.1 Prozeßmodell von Carmel und Becker

Daneben ist vor allem die Publikation von Carmel und Becker [CaB95] erwähnenswert. Hier wird ein Prozeßmodell für die Standardsoftwareentwicklung vorgestellt. Dazu werden zunächst schrittweise Teile der oben erwähnten Problemstellungen vorgestellt und daraus Anforderungen an Prozeßschritte abgeleitet. Diese sind meist aus bereits existierenden Prozeßmodellen adaptiert, die teilweise für die Produktentwicklung und teilweise für die Softwareentwicklung entwickelt wurden. Das Prozeßmodell besteht grob aus zwei iterativen und inkrementellen Zyklen, der Requirements-Schleife und der Qualitätsschleife. In der ersteren werden initiale Anforderungen an die erste Version und neue Anforderungen an Folgeversionen erarbeitet und in eine Spezifikation überführt. Nach Festschreibung einer fixen Spezifikation wird dann in der Qualitäts-Schleife die eigentliche Software entwickelt. Wesentliche Elemente innerhalb der Zyklen sind Entscheidungsmeilensteine, an denen jeweils anhand eines abgeschätzten Risikos über eine Weiterentwicklung der Software entschieden wird.

Dieses Prozeßmodell stellt eine grobe Grundlage für den generellen Prozeßablauf zur Verfügung. Für eine Umsetzung in eine praktisch nutzbare Methode muß diese um allgemeine RE-Techniken, wie im Abschnitt 3.2 erläutert, und um spezifische Techniken für die Entwicklung von Standardsoftware ergänzt werden.

3.4.2 Spezifische Fragestellungen

Wir erläutern jetzt kurz weitere Probleme im Requirements Engineering für Standardsoftware, die bisher in der Literatur nicht behandelt worden sind. Eine spezifische Fragestellung ist die technische Unterstützung von in Versionen ausgelieferter Software. So ist beispielsweise die Version, in der eine bestimmte Anforderung umgesetzt wird, nicht immer von Anfang an starr festgelegt. Eine Anforderung niedriger Priorität kann möglicherweise über ein paar Versionen nach hinten verschoben werden. Es ist auch möglich, daß sich die Realisierung einer Anforderung über mehrere Versionen erstreckt. Eine Anforderungsdokumentation kann daher nicht starr auf eine einzelne Version ausgelegt sein, sondern muß auch Anforderungen für künftige Versionen enthalten können. Dies ist insbesondere für eine vorausschauende Entwicklung des Designs notwendig, damit bei der Umsetzung von Anforderungen künftiger Versionen aufwendige Re-Designs verhindert werden können.

Interessant ist hierbei auch, inwiefern bisherige RE-Methoden an versionierte Softwareentwicklung angepaßt werden müssen, beispielsweise, ob und wie Prototypingkonzepte auf bereits fertiggestellte Software aufgesetzt werden können. Möglicherweise müssen hierfür in der ausgelieferten Software gesonderte Schnittstellen für den Anschluß von Prototypen geschaffen werden.

Eine weitere Problemstellung ergibt sich aufgrund der unterschiedlichen Kundenprofile auf dem Zielmarkt und der daraus resultierenden Notwendigkeit zur Variantenbildung. Wir unterscheiden hier zwischen der produktinternen und der produktexternen Variantenbildung. Von produktinternen Varianten sprechen wir, wenn der Kunde zwar ein Produkt erwirbt, sich jedoch durch seine individuelle Konfiguration bestimmte Programmteile anpassen kann. Bei der produktexternen Variantenbildung hingegen erstellt der Anbieter ein Gesamtsystem von unterschiedlichen einzeln kaufbare Bausteine, die sich ein Kunde nach seinen individuellen Bedürfnissen zusammenstellen kann.

Generell läßt sich feststellen, daß die Entscheidung für eine bestimmte Variante eine große Tragweite im Hinblick auf die weitere Entwicklung hat. Eine einmal eingeführte Variante erfordert zumindest Wartungskosten in künftigen Versionen, da auftretende Fehler möglichst beseitigt werden sollen. Auch wenig erfolgreiche Varianten lassen sich selten wieder vom Markt nehmen, da damit Kunden, die diese verwenden, verprellt werden könnten. Auch die Entscheidung, ob man aus unterschiedlichen Varianten jeweils eigene Produkte erstellt (externe Variante) oder diese innerhalb eines Produktes (interne Variante) integriert, ist nicht leicht zu fällen.

Für externe Varianten spricht die bessere Transparenz für den Kunden und die geringere Komplexität der einzelnen Produkte. Dabei wird jedoch umgekehrt die gegenseitige Vernetzung der unterschiedlichen Produkte untereinander größer. Bei der internen Variantenbildung erhöht sich die Produktinterne Komplexität mit jeder Variante. Dagegen kann bei geschickter Partitionierung der Varianten die externe Vernetzung der Produkte abnehmen. Die Wahl der Art der Variantenbildung hat damit unmittelbaren Einfluß auf die dem System zugrundeliegenden Architektur (zur Einführung in Softwarearchitekturen siehe beispielsweise [BDRS97]) und auf die gegenseitigen Abhängigkeiten der einzelnen Produkte. Zum Problem der Beherrschung der Komplexität im Zusammenhang mit Produktvarianten existiert in der BWL Literatur (siehe zum Beispiel [Wil97]), deren Inhalte eventuell an die Softwareentwicklung angepaßt werden müssen.

Die eben angesprochenen, für das Anwendungsgebiet spezifischen, Fragestellungen erfordern noch intensive Forschungsaktivitäten. Genauso offen ist die Frage, wie man notwendige Marketingaktivitäten mit Methoden des Requirements Engineerings noch stärker verknüpfen kann.

4 Zusammenfassung

In diesem Bericht haben wir verschiedene Forschungsaktivitäten im Requirements Engineering vorgestellt. Die allgemeinen Themen wurden nach dem Phasenschema von Pohl in Herausarbeiten, Verhandlung, Dokumentation/Spezifikation und Validierung/Verifikation aufgegliedert.

Beim Herausarbeiten allgemeiner Anforderungen werden in erster Linie Fragetechniken eingesetzt. Man versucht neuerdings diese durch Wiederverwendung bereits vorhandener Spezifikationen zu unterstützen. Es besteht hier noch dringender Forschungsbedarf, insbesondere in interdisziplinärer Zusammenarbeit mit der Psychologie und den einzelnen Forschungsschwerpunkten. Szenariotechniken und Ansätze zum Prototyping versprechen beide eine gute Unterstützung zum Herausarbeiten von Abläufen. Hier kann ein Schwerpunkt unserer Forschung auf eine anwendungsorientierte Anpassung der Techniken liegen. Das Herausarbeiten von Anforderungen von Standardsoftware zeichnet sich durch die große Zahl von Quellen und Kanälen aus. Jede der dort verwendeten Methoden läßt sich isoliert verbessern. Wesentlich ist aber hier auch die Systematik der ersten adäquaten Dokumentation aller gesammelten Anforderungen.

Zur Verhandlung über Anforderungen gibt es derzeit noch wenige Ansätze. Viewpoint-Techniken versuchen bereits bei der Methodenkonstruktion eine methodische Basis zur Aufdeckung von Konflikten zu schaffen. Insbesondere bei der Verhandlung von Anforderungen an Standardsoftware ist zusätzlich die Möglichkeit zur gegenseitigen Priorisierung von Anforderungen notwendig. Insgesamt besteht hier noch dringender Bedarf an Grundlagenforschung. Zur praktischen Verhandlung muß man sich daher bisher pragmatischer Ansätze bedienen.

Die Hauptaufgabe einer Dokumentation im Requirements Engineering ist die unmißverständliche und Nutzungsrechte Spezifikation von Anforderungen. Dazu muß sie sowohl einerseits eine semantisch exakte Beschreibung für den Entwickler gewährleisten andererseits auch dem Anwender verständlich sein. Hier laufen momentan umfangreiche Forschungsaktivitäten, die versuchen, durch die Entwicklung graphischer Beschreibungstechniken eine gewisse Anwendernähe und durch deren formale Fundierung und Integration notwendige semantische Exaktheit zu erreichen. Forschungsbedarf besteht hier vor allem in der weiteren Intensivierung der Anpassung der Spezifikationstechniken auf spezifische Anwendungsbereiche. Um Anforderungen verfolgbar zu machen, müssen einerseits Entscheidungen dokumentiert werden, andererseits Wege von einer Anforderung bis hin zu einer fertigen Implementierung erkennbar sein. Momentane Ansätze sind aufgrund des hohen Verwaltungsaufwandes nur teilweise für die praktische Anwendbarkeit geeignet. Hier herrscht noch dringender grundlegender Forschungsbedarf.

Die hauptsächliche Basis zur Validierung und Verifikation bildet eine in entsprechender Form vorhandene Spezifikation. Während zur Validierung die Spezifikation in erster Linie die Kommunikation zwischen Anwender und Entwickler unterstützen soll, steht zur Verifikation die exakte Beschreibung im Vordergrund. Beide Aktivitäten unterstreichen erneut die gleichzeitigen Forderungen an Anwendungsnähe und einer semantischen Fundierung einer Spezifikationssprache.

Wir betrachteten neben den allgemeinen Themen des Requirements Engineering insbesondere auch die Forschungsaktivitäten in den Forschungsschwerpunkten Automatisierungstechnik und Standardsoftwareentwicklung.

In der Forschung des Requirements Engineering in der Automatisierungstechnik findet man sehr wenige und jeweils sehr spezifische Literatur. Die Hauptaktivitäten befassen sich in erster Linie mit der Anpassung von allgemeinen objektorientierten Beschreibungstechniken auf sehr spezielle Beschreibungsanforderungen und mit vereinzelten Simulationsansätzen. Inwiefern diese Ansätze in unsere Forschung einfließen können und wo konkreter Forschungsbedarf besteht, können wir zum momentanen Zeitpunkt noch nicht ausreichend beurteilen.

Bezüglich der Standardsoftwareentwicklung wird meist versucht, Prozeßmodelle und Methoden aus der Produktentwicklung an die spezifischen Bedürfnisse der Softwareentwicklung anzupassen. Dabei kommen insbesondere softwaretechnische Fragestellungen, wie beispielsweise Anpassung von verschiedenen Softwaretechnikansätzen an die Entwicklung von Versionen und Varianten zu kurz. Außerdem ist die komplexe Fragestellung der Verknüpfung von Marketingaktivitäten mit der technischen Anforderungsdefinition und auch die Auswirkungen von marktbedingten Entscheidungen auf

die Softwareentwicklung nicht ausreichend behandelt. Hier ist noch einiges an anwendungsorientierter Forschungsarbeit zu leisten.

Insgesamt haben wir festgestellt, daß eine große Menge von Forschungsthemen noch nicht entsprechend durchdrungen worden ist. Die im Folgenden im Teilprojekt A4 von FORSOFT behandelten Aspekte sind im Bericht [BDS98] festgehalten. Gerade für die anwendungsorientierte Forschung kann die Analyse von Methoden aus von uns nicht zentral betrachteten Anwendungsgebieten nützlich sein, um diese eventuell auf unsere spezifischen Bereiche anzupassen. Beispielsweise könnte es möglich sein, Business Process Modeling zur Modellierung von Abläufen in Produktionsanlagen einzusetzen.

5 Literaturverzeichnis

[Bag97] Bager J., "Die Redmond-Strategie - Schlüssel für Microsofts Erfolg", c`t, Nr. 14, S. 88, 1997

[BDS98] Billing G., Deifel B., Sandner R., "Zielsetzung und weitere Vorgehensweise des Teilprojekts A4", interner Bericht, FORSOFT A4, 1998

[Bil97] Billing G. "Steuerungsentwicklung automatisierter Produktionsanlagen", interner Bericht, FORSOFT A4, 1997

[Bel95] Belschner R., "Simulation Based Analysis on the Example of a Distributed Real-Time System and a Robot Control by Using the ECS Evolutionary Strategy", European Simulation MultiConference, Prague, 1995

[BDRS97] Broy M., Denert E., Renzel K., Schmidt M., "Software Architectures and Design Patterns in Business Applications", Technischer Report TUM-I9746, Technische Universität München, 1997

[BFG+93] Broy M., Facchi C., Grosu R., et. al., "The Requirement and Design Specification Language SPECTRUM An informal Introduction", Technischer Report TUM-I9311/TUM-I9312, Technische Universität München, 1993

[BGH+97] Breu R., Grosu R., Huber F., et. al., "Towards a Precise Semantics for Object-Oriented Modeling Techniques", in Proceedings ECOOP'97, Jyväskylä, 1997

[BHH+97] Breu R., Hinkel U., Hofmann C., et. al., "Towards a Formalization of the Unified Modeling Language ", ECOOP 97,LNCS 1241, 1997

[Bro91] Brown, P.G., "QFD, echoing the voice of the customer", AT&T Technical Journal, S. 18-32, März/April, 1991.

[CaB95] Carmel, E., Becker S., "A Process Model for Packaged Software Development", IEEE Trans. on Eng. Manag., Vol. 42, No. 1, Februar, 1995

[CuS95] Cusumano M. A., Selby R. W., "Microsoft Secrets", Free Press, New York, 1995

[Dar95] Darscht, P., "Rechnergestützte, objektorientierte Entwicklung von Automatisierungssystemen: Ein Erfahrungsbericht", Universität Stuttgart, 1995

[Dav92] Davis A., "Operational Prototyping: A new Development Approach", IEEE Software, Sept., 1992

[Dav93] Davenport T., "Process Innovation: Reengineering Work through Information Technology", Harvard Business School Press, Boston, Ma, 1993

[DFP95] Darscht P., Frigeri A., Pereira C., "Building up Object-Oriented Industrial Automation Systems: Experiences Interfacing Active Objects with Technical Plants", Proceedings of WFCS'95 , 1995, Leysin

[DoT90] Dorfmann M., Thayer R., "Standards, Guidelines and Examples on System and Software Requirements Engineering", IEEE Computer Society Press - Tutorial, 1990

[DP95] Darscht P., Pereira C. E., "An Object-Oriented Approach to Handle Complex Real-Time Industrial Automation Projects, ICECCS'95, Ft. Lauderdale, 1995

[Fin94] Finkelstein A., "Requirements Engineering: a review and research agenda", APSEC 94, pp. 10-19, 1994

[FoR95] Forsgren P., Rahkonen T., "Specification of Customer and User Requirements in Industrial Control System Procurement Projects", ISRE'95, 1995

[GoF94] Gotel O., Finkelstein A., "An Analysis of the Requirements Traceability Problem", ICRE'94, Colorado Springs, 1994

[GoL93] Goguen J. A., Linde C., "Techniques for Requirements Elicitation", RE 93, pp. 152-164, San Diego, Ca, 1993

[Got95] Gotel O, "Contribution Structures for Requirements Traceability", Ph. D. Thesis, University London, 1995

[Hau93] Hauschildt J., "Innovationsmanagement", Vahlen Verlag, 1993

[Hum89] Humphrey, W., "Managing the Software Process", Addison-Wesley, 1989

[HuS97] Huber F., Schätz B., "Rapid Prototyping with AutoFocus", in Formale Beschreibungstechniken für verteilte Systeme, GI/ITG Fachgespräch 1997, pp. 343-353, GMD Verlag, 1997

[Hsi+94] Hsia P. et. al. "Formal Approach to Scenario Analysis", IEEE Software, S. 33- 49, März 1994

[Jac+92] Jacobson I. et. al. "Object-Oriented Software Engineering - A Use Case Driven Approach", Addison Wesley, 1992

[Kas95] Kastin K. S. "Marktforschung mit einfachen Mitteln", Beck Verlag, 1995

[KoM93] Koskimies K., Mäkinen E., "Inferring State Machines from Trace Diagrams", Report A-1993-3, Univ. of Tampere, 1993

[KoM94] Koch F., Metz J., "Objektorientierte Problemanalyse als Grundlage für die Entwicklung von Automatisierungssoftware", 39. Internationales Wissenschaftliches Kolloquium, TUIllmenau, 1994

[Kot97] Kotler P., "Marketing Management - Analysis Planning, Implementation and Control", Prentice Hall, 1997

[KrL97] Krone M., Lemmer K., "Kundenorientierter Entwicklungsprozeß im spurgebundenen Verkehr", EKA'97, Braunschweig, 1997

[MaL97] Massonet P., van Lamsweerde A., "Analogical Reuse of Requirements Frameworks", Proceeedings RE'97, 1997

[Mef92] Meffert, H. "Marketingforschung und Käuferverhalten., Gabler Verlag, 2. Auflage, Wiesbaden, 1992

[NKF94] Nuseibeh B., Kramer J., Finkelstein A., "A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification", IEEE Trans. SE, 20 (10), 760-773, Oct., 1994

[Pfe93] Pfeifer T., "Qualitätsmanagement. Strategien, Methoden, Techniken", Hanser Verlag, München - Wien, 1993

[Poh96] Pohl K., "Requiremens Engineering: An Overview", Aachener Informatik-Berichte, Nr. 96-05, RWTH Aachen, 1996

[PoW97] Pomberger G., Weinreich R., "Qualitative und quantitative Aspekte prototypingorientierter Software-Entwicklung - Ein Erfahrungsbericht", Informatik Spektrum, 20, 33-37, 1997

[PPS92] Pomberger G., Pree W., Stritzinger A., "Methoden und Werkzeuge für das Prototyping und ihre Integration", Informatik Forsch. Entw., 7, 49-61, 1992

[PTA94] Potts C., Takahashi K., Antón A., "Inquiry-Based Requirements Analysis", IEEE Software, S. 21- 32, März 1994

[Rak97] Rakitin S. R., "Software Verification and Validation - A Practioner's Guide", Artech House, 1997

[RUP90] Ramamoorthy, C. V., Usuda, Y., Prakash A., et. al., "The evolution support environment system", IEEE Transactions on Software Engineering, 16 (11), 1225-1234, Nov., 1990

[RBC+96] Rolland C., Ben Achour C., Rlyté J. et. al., "A Proposal for a Scenario Classification Framework", CREWS Report 96-01, http://SunSITE.Informatik.RWTH-Aachen.DE/CREWS/reports.htm#1996, 1996

[RBP+91] Rumbaugh J., Blaha M., Premerlani W., et. al., "Object-Oriented Modeling and Design", Prentice Hall, 1991

[RKW94] Regnell B., Kimbler K, Wesslén A., "Improving the Use Case Driven Approach to Requirements Engineering", ICRE'94, 1994

[Rob97] Robertson S., "Requirement Patterns via Events/Use Cases", http://www.atlsysguild.com/Site/Suzanne/Requirements_Patterns, London, 1997

[Rum96] Rumpe B., "Formale Methodik des Entwurfs verteilter objektorientierter Systeme", Dissertation, Technische Universität München, 1996

[Sab76] Sabel H. "Wirtschaftlichkeitsanalysen von Produkten", Zfbf-Kontaktstudium, 28, S. 38, 1976

[Shi92] Shilito M. L., De Marle D. J.. "Value. Its Measurement, Design and Management", New York, 1992

[SHB96] Schätz B., Hußmann H., Broy M.,"Graphical Development of Consistent System Specifications", in FME'96: Industrial Benefit and Advances In Formal Methods, S. 248-267, M. Woodcook (ed.), Springer, 1996

[Som+93] Sommerville I. et. al., "Integrating ethnography into the requirements engineering process", RE 93, pp. 165-173, San Diego, Ca, 1993

[Som96] Sommerville I., "Software Engineering", Addison-Wesley, 1996

[WCS94] Wood, D. P., Christel, M. G., Stevens S. M., "A Multimedia Approach to Requirements Capture and Modeling", ICRE 94, IEEE CS, pp. 53-56, Los Almanitos, 1994

[Wie95] Wieringa, R., "An Introduction to Requirements Traceability", TR IR-389, Vrije Universiteit, Amsterdam, 1995

[Wil97] Wildemann H., "Leitfaden Variantenmanagement - Leitfaden zurKomplexitätsbeherr-schung", Technische Universität München, München, 1997