Diplomarbeit

Die Bedeutung von Kundenorientierung für die Softwareentwicklung

von Boris Fittkau

Betreuer:

Prof. Dr. Heinz Züllighoven
Prof. Dr. Christiane Floyd

Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

Diese Diplomarbeit wurde eingereicht, um am Fachbereich Informatik der Universität Hamburg Teile der Anforderungen zu erfüllen, die für die Erlangung des Titels „Diplom-Informatiker“ gestellt werden.

Erklärung

Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, den 8.März 1998

Boris Fittkau
Forsmannstr.21
22303 Hamburg
Matrikel-Nr.: 2482945

Prof. Dr. Heinz Züllighoven (Erstbetreuer)
Prof. Dr. Christiane Floyd (Zweitbetreuer)

Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik
Vogt-Kölln-Straße 30
22527 Hamburg

Danksagung

Ich möchte all jenen danken, die mich bewußt oder unbewußt bei dieser Arbeit unterstützt haben.

Heinz Züllighoven als Erstbetreuer, nicht nur für seine konstruktive und sehr freundlich formulierte Kritik, sondern auch für die indirekte Ermunterung, die ich erfuhr, wenn er Teile meiner Überlegungen in seiner eigenen Arbeit weiterverwertete. Außerdem kam von ihm der entscheidende Strukturierungshinweis, der dieser Arbeit den Aufbau gab, den sie jetzt hat.

Christiane Floyd als Zweitbetreuerin dafür, daß sie mich mit ihrer anfänglichen Skepsis in bezug auf das Thema zwar kurzfristig frustrierte, langfristig jedoch zu höherer Leistung anspornte. Außerdem gebührt Christiane Floyd meine volle Anerkennung und Wertschätzung für die innovative und wegweisende Arbeit, die sie seit vielen Jahren auf dem Gebiet der Softwaretechnik leistet.

Meinem Vater, Prof. Dr. Bernd Fittkau, dafür, daß er letztlich die Idee zu diesem Thema gegeben hat. Sein umfangreiches Erfahrungswissen, das er immer gerne mit mir geteilt hat, war für wesentliche Bereiche dieser Arbeit unersetzlich.

Den Mitarbeitern des Softwarehauses, die mit Ihrer Offenheit und Kooperationsbereitschaft den für die wissenschaftliche Forschung so wichtigen ‘Blick hinter die Kulissen’ ermöglicht haben. Besonderer Dank an die Geschäftsleitung für die Erlaubnis, eine sehr kritische Bestandsaufnahme hier veröffentlichen zu dürfen.

Meiner Lebensgefährtin, Carola Lilienthal, sowohl für die wichtige fachliche Unterstützung als auch für die noch wichtigere nicht-fachliche Motivationsarbeit und die liebevollen Durchhalteparolen.

Meinem Freund, Cedric Parkin, dafür, daß er sich weder für diese Arbeit, noch für meine damit verbundenen Zweifel interessiert hat und in diesem Sinne eine echte Unterstützung gewesen ist.

Meiner Mutter, Antje Fittkau und auch Martina für die enorm wertvollen Korrekturarbeiten.

Birgit, Isabella und allen anderen hier nicht aufgeführten Freunden mein ehrlicher Dank. Ihr werdet es verstehen und, da Ihr mich kennt, höchstens für ein bißchen pathetisch halten, wenn ich sage: *„Es ist vollbracht“*.

He who knows not, and knows not that he knows not, is a fool - avoid him.
He who knows not, and knows that he knows not, is a child - teach him.
He who knows, and knows not that he knows, is asleep - wake him.
But he who knows, and knows that he knows, is a wise man - follow him.
(Sprichwort)

Inhaltsverzeichnis

1 Einleitung

Seit einiger Zeit ist der Begriff *Kundenorientierung* sowohl in der öffentlichen als auch in der wissenschaftlichen Diskussion auf großes Interesse gestoßen. Obwohl oft so gebraucht, ist Kundenorientierung mehr als nur ein Schlagwort bzw. Modetrend, sondern im Gegenteil als integrierendes Leitbild moderner Unternehmensführung geeignet. In vielen Unternehmen finden zur Zeit z.T. gravierende Umstrukturierungsmaßnahmen statt, die auch eine Verbesserung der Kundenorientierung zum expliziten Ziel haben. Auch in der Softwareentwicklung stellt sich die Frage nach Bedeutung und Umsetzung von Kundenorientierung.

Schaut man sich die Situation in der Softwareentwicklung an, so scheinen auch heute noch traditionelle, phasenorientierte Vorgehensweisen zu dominieren. Eine große Zahl von solchen Softwareentwicklungsprojekten scheitert oder nimmt einen unbefriedigenden Verlauf (vgl. [WelOrt92]). Dies hat in der Regel gravierende negative Konsequenzen sowohl für den Kunden als auch für den Softwarehersteller.

Die aus einer traditionellen Vorgehensweise resultierenden Probleme sind lange bekannt und trotzdem weiterhin präsent. Liegt hier u.U. ein Mangel an Kundenorientierung zugrunde? Kann Kundenorientierung als Grundlage für Veränderungen in der Praxis der Softwareentwicklung genutzt werden?

Diese Arbeit dient für mich dazu, nach nunmehr 10 Jahren aktiver Arbeit als Softwareentwickler ein vorläufiges Resümee zu ziehen. Meine praktischen Projekterfahrungen der letzten Jahre haben mich zu der Überzeugung kommen lassen, daß die traditionellen Formen von Softwareentwicklung überholt sind und durch eine neue Sichtweise ersetzt werden müssen, die am ehesten mit *kundenorientierter Softwareentwicklung* bezeichnet werden kann. Hierunter verstehe ich eine Vorgehensweise, die den Kunden kontinuierlich in den Entwicklungsprozeß einbezieht (Partizipation und Kooperation) und dazu einerseits eine evolutionäre Prototypingstrategie und Objektorientierung einsetzt, anderseits das Projekt selbst zum Gegenstand von Reflexion macht und moderne Erkenntnisse aus der Organisations- und Kommunikationspsychologie umsetzt (Prozeßorientierung, Teamorientierung, etc.).

1.1 Das Märchen vom „kundenorientierten Softwareprojekt“

Es war einmal in einem nicht allzu fernen Land in einer nicht allzu fernen Zeit, daß ein Softwareprojekt durchgeführt wurde, welches auf allen Ebenen zufriedenstellend verlief. Obwohl es sich um ein großes Projekt handelte, welches in einem neuen Umfeld den kompletten Arbeitsablauf des Kunden unterstützen sollte, wurde es rechtzeitig fertig. Es überschritt den Kostenrahmen nicht. Der Kunde war nicht nur mit Funktionalität und Handhabung des Programms zufrieden, sondern auch mit der Art und Weise, wie die Entwicklung selber durchgeführt wurde.

Wie konnte das geschehen?

Das Projekt wurde konsequent und kontinuierlich kundenorientiert durchgeführt.

Das System wurde evolutionär in sinnvollen und überschaubaren Teilschritten realisiert, wobei ein besonderer Schwerpunkt auf den vielfältigen Einsatz von Prototypen gesetzt wurde. Die Projektsprache, die in den Entwicklungsdokumenten oder in Sitzungen benutzt wurde, orientierte sich an der Sprache der Anwendungswelt des Kunden. Beides zusammen, evolutionäres Prototyping und sprachliche Orientierung an der Anwendungswelt, ermöglichten es den Vertretern des Kunden, an allen Phasen der Entwicklung gleichberechtigt

teilzuhaben und konstruktive Unterstützung zu geben. Durch die schrittweise Fertigstellung konnte nicht nur der Kunde (Management und Anwender), sondern auch der Softwarehersteller (Management und Entwickler) ständig den Fortschritt des Projektes kontrollieren.

Noch bevor man sich den fachlichen und technischen Fragen der Systementwicklung zuwandte, fanden externe Weiterbildungsmaßnahmen statt, die sowohl die Entwickler als auch die Vertreter des Kunden in bezug auf die Probleme und Schwierigkeiten von Softwareentwicklungsprojekten sensibilisierten und wo kooperative Zusammenarbeit praktisch erprobt wurde. Einige der Entwickler hatten sogar die Gelegenheit, über längere Zeit am Tagesgeschäft des Kunden aktiv teilzuhaben, d.h. sie beobachteten nicht nur oder führten Interviews, sondern erfuhren die Anwenderrealität ‘am eigenen Leib’.

Das Projekt wurde kontinuierlich und auf professionelle Weise „von außen“ beobachtet und begleitet. Externe Moderatoren leiteten regelmäßige Projekt-Reviews an, in denen der Projektfortgang reflektiert wurde. Diese Reviews fanden in größeren aber auch in kleineren Gruppen statt, je nach Notwendigkeit. Außerdem wurden Einzelcoaching und moderierte Konfliktlösungsgespräche durchgeführt. Indem auf diese Weise der Entwicklungsprozeß selbst kontinuierlich auf einer Meta-Ebene zum Gegenstand der Diskussion gemacht wurde, war es möglich, Probleme und Fehlentwicklungen früh- und damit rechtzeitig zu erkennen und geeignete Lösungsmaßnahmen einzuleiten. Außerdem führte die Prozeß-Reflektion zu einer kontinuierlichen Verbesserung der Kooperationsbeziehungen zwischen Kunde und Softwarehersteller, die u.a. in der Heranbildung von vielfältigen informellen Kommunikationsbeziehungen zwischen Entwicklern und Anwendern resultierte.

Innerhalb des Entwicklerteams fand ebenfalls kontinuierliche Metareflektion statt. Hier stand nicht nur die technische Realisierung des Systems im Blickpunkt, sondern auch die Kooperation und Kommunikation untereinander und mit dem Kunden. Auch hier waren positive Auswirkungen auf die Teamentwicklung zu bemerken. Als Konsequenz war bei den Entwicklern ein hohes Maß an Eigenverantwortlichkeit und Motivation zu erkennen und eine positive Identifikation mit dem entstehenden Produkt. Am Ende des Projektes waren die Entwickler nicht nur mit der "äußerlichen" Funktionalität, sondern auch mit der „inneren“ Beschaffenheit, d.h. der technischen Qualität des Produktes mehr als zufrieden - das System lief stabil, war pflegbar und erwies sich auch bei umfangreichen Änderungen als erweiterbar.

Aus Managementsicht zeichnete sich das Projekt durch ein hohes Maß an Kontrollierbarkeit und Planbarkeit aus. Indem von Anfang an konsequent auf unrealistische Versprechungen verzichtet wurde und das System in kleinen Teilschritten realisiert wurde, konnte die Validität der Terminschätzungen ständig kontrolliert und verbessert werden. Im Verlaufe des Projektes hatten deshalb nicht nur die Entwickler, sondern auch das Management eine immer genauere Vorstellung von der Komplexität des Systems und der Tragweite neu entstehender Anforderungen.

Beim Kunden entwickelte sich im Projektverlauf eine zunehmende Sensibilität für die Schwierigkeiten der „anderen“ Seite. Durch die enge Einbindung in den Entwicklungsprozeß, wurde das sich entwickelnde System immer mehr zu einem System der Anwender, die zunehmend mehr Toleranz für etwaige Schwachstellen entwickelten. Nach mehreren Iterationen und Pilotinstallationen zeichnet sich das System heute besonders dadurch aus, daß es den Anwender auf flexible Weise in seiner Arbeit unterstützt und ihn nicht durch ein Übermaß an Kontrolle und eine Vielzahl von technisch bedingten Workarounds behindert.

Die Erfahrungen, die sowohl auf technischer und fachlicher Seite aber auch in bezug auf die Prozeßgestaltung gemacht worden sind, sind lesbar dokumentiert und stehen für spätere Projekte als Unterstützung zur Verfügung. Errungenschaften konnten ausgebaut und wiederholt werden, Fehler sind bekannt und mußten kein zweites Mal gemacht werden.

Wie in anderen Projekten ähnlichen Ausmaßes, kam es auch in diesem Projekt zu z.T. schwerwiegenden Krisen. Diese hingen in der Anfangsphase insbesondere mit der engen Zusammenarbeit zwischen Entwicklern und Anwendern zusammen. Beide Gruppen waren aufgrund ihres Selbstverständnisses (und ihrer Ausbildung) wenig geneigt und in der Lage, Verständnis für die jeweils andere Seite aufzubringen. Mit zunehmendem Projektverlauf schienen die Auswirkungen dieser „Kulturbarriere“ jedoch schwächer zu werden. Die anstehenden Konflikte, die z.B. bei Auslieferungen von Pilot- und Produktionssystemen auftraten, konnten vor dem Hintergrund einer zunehmend stabileren Vertrauensbasis konstruktiv gelöst werden.

Wie bereits erwähnt: Dieses Projekt war ein voller Erfolg, nur ist es kein Tatsachenbericht, sondern eben nur ein Märchen. **„Klingt erstrebenswert!“,** werden denn auch viele rufen, **„nur leider gänzlich unrealistisch und naiv!“**. So einfach gehe es nicht, das sei unmöglich, die Realität sehe ganz anders aus. Das könne sich doch nur ein Theoretiker aus der Weltfremdheit des universitären Elfenbeinturms heraus ausgedacht haben, dem die Empfehlung **„Träum’ weiter!“** zugerufen werden sollte.

An dieser Stelle könnte auf weitere Überlegungen zu diesem Thema verzichtet werden, gäbe es nicht eindeutige Zeichen, die dafür sprechen, die eben skizzierte Vision zum erklärten Ziel einer Umorientierung in der Softwareentwicklung zu machen: Softwareentwicklung ist zu teuer, Projekte scheinen ab einer bestimmten Größenordnung nicht mehr kontrollierbar zu sein. Termine werden nicht gehalten, Kosten werden überschritten, Funktionalität entspricht nicht den Erwartungen bzw. wird im Projektverlauf kontinuierlich reduziert. Das Softwaresystem behindert den Anwender mehr als es ihn unterstützt und zwingt zu Anpassung des Menschen an das System und nicht umgekehrt. Die Programmeinführung verläuft problematisch in aufgereizter und gestreßter Atmosphäre mit Folgeschäden für den weiteren Verlauf der Entwicklung. Zunehmende Polarisierung zwischen Kunde und Softwarehersteller mit verhärteten Fronten ist die Folge. Ein Teufelskreis entsteht, bei dem die Schuldfrage für beide Seiten hinlänglich geklärt ist:

Der Kunde klagt: *„Das Softwarehaus* [bzw. die DV-Abteilung] *ist schuld. Wir werden nicht genug mit einbezogen. Termine werden nicht gehalten und wir bekommen instabile, unvollständige und unbrauchbare Software!“*.

Das Softwarehaus bzw. die DV-Abteilung reagiert mit: *„Der Kunde ist schuld. Er will die perfekte Software, möglichst ohne dafür zu bezahlen. Er kann seine Wünsche nicht präzise genug äußern, er ändert ständig seine Vorstellungen, hat vermutlich gar keine konkreten Visionen, denn er ist viel zu tief in eigenen Problemen verstrickt und versucht nun, Hausgemachtes auf uns abzuschieben!“*.

Die Liste gegenseitiger Schuldzuweisungen könnte beliebig fortgesetzt werden. Es scheint so zu sein, daß bei einem negativen Projektverlauf sowohl Kunde, als auch Softwarehersteller gleichermaßen Schaden nehmen. Beiden muß an positiven Veränderungen gelegen sein.

1.2 Entstehungsgeschichte und Zielsetzung der Arbeit

Die vorliegende Arbeit begann im Grunde bereits 1993, als ich als Entwickler an einem mittleren Softwareentwicklungsprojekt beteiligt war und dieses nach Abschluß im Rahmen meiner Studienarbeit kritisch analysierte (vgl. [Fittkau94]). Seit dieser Zeit setzte ich mich intensiv mit den Problemen von traditionellen Softwareentwicklungsprojekten auseinander.

Das in meiner Studienarbeit beschriebene Projekt resultierte in einem Folgeprojekt, welches im Januar 1994 begonnen wurde. Nach einer Projektdauer von annähernd 2½ Jahren ging das entwickelte System im Mai 1996 in Produktion. Unmittelbar darauf hatte ich die Gelegenheit, auch dieses Projekt zu analysieren, wobei ich mit vielen der Projektbeteiligten auf der Seite

des Softwarehauses persönliche Gespräche führte. Auch diese Analyse (die in Kapitel 4 dieser Arbeit wiedergegeben wird) fiel kritisch aus.

Da sich die Ergebnisse der Projektanalyse weitgehend mit den in der Literatur angeführten Problemen von Softwareentwicklung deckten, tauchte die Frage auf, wieso lang bekannte Probleme immer noch bestehen und wie sie gelöst werden können. Insbesondere stellte sich mir die Frage, ob der in den letzten Jahren populär gewordene Begriff der *Kundenorientierung* u.U. zum Ziel einer Umorientierung in der Softwareentwicklung genutzt werden könne.

In dieser Arbeit will ich daher im wesentlichen der Frage nachgehen, welche Bedeutung Kundenorientierung für die Softwareentwicklung hat. Im Rahmen der Arbeit soll deutlich werden, daß Kundenorientierung im Bereich der Softwareentwicklung tatsächlich eine entscheidende Rolle spielen kann und sollte. Es soll gezeigt werden, daß die Forderung nach Kundenorientierung im Rahmen von Softwareentwicklungsprojekten sowohl für die Produkt- als auch für die Prozeßgestaltung wesentliche Konsequenzen hat. Weiterhin sollen strukturelle und kulturelle Voraussetzungen diskutiert werden, die innerhalb einer softwareentwickelnden Organisation für eine kundenorientierte Softwareentwicklung notwendig sind.

Daraus ergibt sich der folgende Aufbau der Arbeit:

1.3 Aufbau der Arbeit

In **Kapitel 2** wird zunächst der Begriff der *Kundenorientierung* aus der Sicht der betriebswirtschaftlichen Forschung diskutiert. Dabei soll deutlich werden, daß Kundenorientierung als Leitbild moderner Unternehmensführung dienen kann und mehr als nur ein Modetrend ist. Als zentrales Ziel von Kundenorientierung wird die Erreichung von größtmöglicher Kundenzufriedenheit erkannt, mit der wiederum eine langfristige Kundenbindung angestrebt wird.

In **Kapitel 3** wird erläutert, welche Bedeutung Kundenorientierung für die Softwareentwicklung hat und welche Konzepte der Kundenorientierung sich nutzbringend auf diesen Bereich übertragen lassen. Es soll deutlich werden, daß Kundenorientierung in fast allen Bereichen von Softwareentwicklung Relevanz besitzt und als Grundlage von Umstrukturierungen und Veränderungen genutzt werden kann.

In **Kapitel 4** wird ein industrielles Softwareprojekt beschrieben und vor dem Hintergrund von Kundenorientierung analysiert und ausgewertet. Insbesondere sollen hier die negativen Auswirkungen eines traditionellen Vorgehens auf die Kundenorientierung des Softwareherstellers deutlich werden.

Abschließend wird in **Kapitel 5** eine Zusammenfassung der Ergebnisse und ein Ausblick auf mögliche zukünftige Forschungsarbeiten unternommen.

2 Kundenorientierung

Diese Arbeit will die Frage beantworten, welche Bedeutung der Begriff Kundenorientierung für die Informatik besitzt. Da Kundenorientierung im Rahmen der Informatik-Forschung bisher nur wenig behandelt wurde, stützt sich dieses Kapitel vorwiegend auf Forschungsarbeiten aus dem Gebiet der Betriebswirtschaftslehre. Hier gibt es eine Fülle von Literatur, die sich direkt oder indirekt mit Kundenorientierung auseinandersetzt. Die Forschungsergebnisse sollen im folgenden vorgestellt werden.

Ausblickend auf die späteren Kapitel werden bereits in diesem Kapitel an verschiedenen Stellen Fragen in bezug auf Kundenorientierung in der Softwareentwicklung gestellt.

2.1 Einleitung

Sowohl in der wissenschaftlichen Diskussion als auch in der Tagespresse taucht seit einiger Zeit der Begriff *Kundenorientierung* immer häufiger auf. In Stellenangeboten ist Kundenorientierung oft ein wichtiges Einstellungskriterium, und eine Vielzahl von Managementseminaren hat die Schaffung und Verbesserung einer kundenorientierten Haltung zum Ziel. In vielen Dienstleistungsunternehmen ist Kundenorientierung schon heute expliziter Bestandteil des Unternehmensziel: „Im Mittelpunkt unserer Bemühungen stehen die Kunden“ ist eines der Leitbilder des Daimler-Benz-Konzerns. „Die Wünsche unserer Kunden stehen an erster Stelle“ formuliert es die Deutsche Lufthansa. DEUTSCH und SCHMIDT zufolge ist Kundenorientierung für 75% der deutschen Versicherungsunternehmen und für 80% der europäischen Banken eine Zielsetzung mit größter strategischer Bedeutung für die Zukunft ([Deut93], [Schm92]). Von 800 befragten europäischen Unternehmen betreiben 50% z.T. gravierende Veränderungen ihrer Organisationsstruktur, „wobei als vorrangiges Ziel eine Verbesserung der Kundenorientierung genannt wird“ ([Droege94]).

Auch in der Softwarebranche kann man ähnliche Trends erkennen. Exemplarisch seien die beiden Softwarehersteller PLAUT und Siemens-Nixdorf genannt. Kundenorientierung wird beispielsweise von PLAUT als zentraler Bestandteil der Unternehmensphilosophie genannt („Kundenorientierung und Kundennutzen haben für PLAUT höchste Priorität“). Siemens-Nixdorf betreibt seit 1994 mit hohem Aufwand ein sog. „Culture Change Program“, welches u.a. die Steigerung der Kundenorientierung zum expliziten Ziel hat.

Während gleichzeitig viele Unternehmen nach mehr Kundenorientierung bzw. Kundennähe streben, und andere vorgeben, sie schon zu praktizieren, machen viele auch populärwissenschaftliche Veröffentlichungen auf einen Mangel an Kundenorientierung aufmerksam: „Störenfried Kunde“ lautete 1994 die vielbeachtete Titelgeschichte des Nachrichtenmagazins DER SPIEGEL, in der u.a. von einer „Servicewüste Deutschland“ gesprochen wurde. OGGERS Buch „König Kunde: Angeschmiert und Abserviert“ spricht eine deutliche Sprache in bezug auf den Grad der praktischen Umsetzung von Kundenorientierung. Letztlich steht einer als notwendig erkannten Forderung nach Kundenorientierung („No client - no company“, [Geffroy94]) die real existierende Überzeugung „Das einzige, was stört, ist der Kunde“ (ebd.) gegenüber.

Folgende Fragen sollen im weiteren beantwortet werden:

1. Woraus resultiert die gegenwärtige Diskussion um Kundenorientierung?
2. Ist Kundenorientierung lediglich ein Modetrend, oder steckt mehr dahinter?
3. Welche Forschungsbereiche beschäftigen sich mit Kundenorientierung und zu welchen Ergebnissen ist man dort gekommen?
4. Was genau ist Kundenorientierung?

2.2 Ursprung der Diskussion um Kundenorientierung

Die gegenwärtige Diskussion um Kundenorientierung wurde mit ziemlicher Sicherheit durch den Bestseller „In Search of Excellence“ von PETERS und WATERMAN (vgl. [PetWat82]) ausgelöst. In Interviews mit Managern von 43 als erfolgreich eingestuften amerikanischen Unternehmen hatten die Autoren Kundennähe als einen von acht Erfolgsfaktoren identifiziert:

> „The excellent companies really are close to their customers. That’s it. Other companies talk about it, the excellent companies do it.“ ([PetWat82]).

PETERS und WATERMANS Studie schlug seinerzeit besonders auf dem amerikanischen Markt wie eine Bombe ein und wurde mit über 5 Mio. verkauften Exemplaren zu einem der erfolgreichsten Managementbücher aller Zeiten.

In wissenschaftlichen Kreisen wurde „In Search of Excellence“ z.T. heftig kritisiert. Insbesondere wurde bemängelt, daß keiner der gewählten Begriffe exakt definiert bzw. einer Operationalisierung zugeführt wurde, sondern lediglich anhand von (allerdings eindrucksvollen) Beispielen illustriert wurde.

Außerdem wurden methodische Mängel in PETERS und WATERMANS Untersuchung festgestellt: Einerseits fehlte in der Untersuchung eine Kontrollgruppe nicht exzellenter Unternehmen, andererseits erschien die Auswahl der als exzellent eingestuften Unternehmen zumindest fragwürdig (auffällig viele der aufgelisteten Firmen wurden von der Unternehmensberatung McKinsey betreut, für die auch die Autoren lange Jahre tätig waren).

Weiterhin waren schon zwei Jahre nach Erscheinen des Buches 14 der 43 exzellenten Unternehmen in wirtschaftliche Schwierigkeiten geraten. „Who’s Excellent now?“ fragte die Zeitschrift BUSINESS WEEK in ihrer Ausgabe am 5.11.84 und identifizierte interessanterweise besonders Verstöße gegen Kundennähe als Gründe für die Schwierigkeiten.

Trotz der geäußerten Kritik wird PETERS und WATERMAN jedoch allgemein der Verdienst zugesprochen, die Bedeutung des Kunden wieder in das allgemeine Bewußtsein gerückt und die Diskussion um Kundenorientierung neu ausgelöst zu haben. Eine Vielzahl von Untersuchungen versuchte in der Folge, die von PETERS und WATERMAN aufgestellten Behauptungen zu beweisen bzw. zu widerlegen.

2.3 Kundenorientierung - ein Modetrend?

Was dem außenstehenden Betrachter der Kundenorientierungsdiskussion sofort auffällt, ist der vielfältige Gebrauch von Schlagwörtern, die gut klingen (z.B. „Der Kunde ist König“ oder „Nur wer dient, verdient!“), wo jedoch unklar bleibt, welche Bedeutung hiermit verknüpft ist. Liegt hierin ein Indiz, daß es sich bei Kundenorientierung lediglich um einen Modetrend handelt, um noch ein weiteres Management-Schlagwort unter vielen?

Ein Grund für den oft leichtfertigen Gebrauch von Schlagwörter scheint darin zu liegen, daß die gegenwärtige Diskussion um Kundenorientierung ihren Ausgangspunkt in der populärwissenschaftlichen Literatur genommen hat. Dort treten als Protagonisten oft erfolgreiche Managementberater (wie z.B. PETERS und WATERMAN) auf, die mit leicht-verdaulichen und plakativen Aussagen ein breites Publikum ansprechen wollen, um damit letztlich auch Werbung für ihre eigenen Beratungsangebote zu machen.

Um die Bedeutung von Kundenorientierung anzuerkennen, ist es deshalb wichtig zu erkennen, daß die Forderung nach Kundenorientierung an sich nichts Neues darstellt. Im Gegenteil: Schon 1954 formulierte der anerkannte Betriebswirtschaftler DRUCKER:

> „There is only one valid definition of business purpose: to create a satisfied customer. It is the customer who determines what the business is. Marketing…is

> the whole business seen from the point of view of its final result, that is, from the customer's point of view" ([Drucker54]).

Eine ähnlich lautende Aussage ist bei LEVITT zu finden, der 1960 die Aufgabe des Marketings definierte als

> „satisfying the needs of the customer by means of the product and the whole cluster of things associated with creating, delivering and finally using it." ([Levitt60]).

Damit liegt Kundenorientierung im Kern jeder Marketingkonzeption und ist mehr als nur ein Modetrend. Die immer wieder aufflammende Diskussion um mehr Kundenorientierung kann als ein Indiz dafür gelten, daß in vielen Unternehmen dieses Herzstück des Marketing-gedankens schlichtweg unterentwickelt ist, daß es also an einer konsequenten Umsetzung mangelt (vgl. u.a. [Althaus95], [Plinke96]).

Im Zuge eines immer härter werdenden wirtschaftlichen Konkurrenzkampfes scheint in der Rückbesinnung auf Kundenorientierung eine existentielle Voraussetzung für das Überleben von Unternehmen zu liegen. In diesem Zusammenhang wird auch von der „Wiederentdeckung des Kunden" gesprochen (vgl. [Plinke96]).

Fragen in bezug auf Kundenorientierung in der Softwareentwicklung

⇒ Wie sieht es mit der Umsetzung von Kundenorientierung in der Softwarebranche aus?

⇒ Gibt es auch in der Softwarebranche Indizien dafür, daß eine Steigerung von Kundenorientierung notwendig ist?

2.4 Begriffsabgrenzungen

Bevor das Phänomen Kundenorientierung näher erläutert werden kann, müssen zunächst einige Begriffsabgrenzungen erfolgen, damit die im weiteren vorgestellten Forschungsarbeiten besser eingeordnet werden können.

In der Management-Literatur gibt es eine Vielzahl von Begriffen, die direkt oder indirekt mit Kundenorientierung in Verbindung gebracht werden, wie z.B. Kundenfreundlichkeit, Kundenzufriedenheit, Kundennutzen, Servicequalität, etc. Diese Begriffe werden im weiteren Verlauf der Arbeit noch näher behandelt. Man kann aber schon vorab festhalten, daß diese Begriffe tendenziell dem Begriff Kundenorientierung untergeordnet werden, d.h. sie dienen der Umsetzung von Kundenorientierung.

Weiterhin gibt es eine Reihe komplexer Management-Strategien, bei denen zunächst unklar ist, ob sie mit Kundenorientierung identisch sind oder nicht. In der Literatur werden hier besonders die folgenden genannt (vgl. [Althaus95], [Homburg94], [Zollner94]):

- Marketingorientierung
- Marktorientierung
- Kundennähe

Ausgehend von der historischen Entwicklung des Marketings seit den 50er Jahren, kommt ZOLLNER zu dem Schluß, daß weder *Marktorientierung* noch *Marketingorientierung* mit Kundenorientierung gleichzusetzen sind (vgl. [Zollner94]). Und dies aus den folgenden Gründen:

Marketingorientierung löste in den 50er Jahren die bis dahin vorherrschende Produktionsorientierung ab, die das Marketing auf den Verkauf dessen beschränkte, was die Produktion zur Verfügung stellte. Mit Marketingorientierung ist die zentrale Forderung verbunden, dem Bereich Marketing im Unternehmen eine höhere Bedeutung zu geben bzw. ihm sogar eine Führungsaufgabe zuzubilligen. Kundenorientierung erscheint in diesem

Zusammenhang als historisch gewachsener Marketing-Schwerpunkt, d.h. als eine mögliche Strategie unter vielen. Damit kann Kundenorientierung nicht mit Marketingorientierung gleichgesetzt werden.

Der Begriff *Marktorientierung* kam in den 80er Jahren auf und ist eng mit dem Begriff der Wettbewerbsorientierung verbunden. Aus einer starken Orientierung auf die Mitbewerber, die es galt, aus dem Wettbewerb zu verdrängen oder gar zu 'vernichten', resultierte jedoch gerade eine Vernachlässigung des einzelnen Kunden. Insofern scheinen beide Konzepte nicht synonym zu sein, obwohl es darüber kontroverse Aussagen gibt. Beispielsweise spricht SHAPIRO von „no meaningful difference between 'market driven' and 'customer oriented'" ([Shap88]). In der Regel jedoch werden beide Begriffe unterschiedlich benutzt: „Marktorientierung ist nach unserem Verständnis ein umfassenderes Konstrukt, das neben der Kundenorientierung auch die Wettbewerbsorientierung umfaßt." ([Homburg94]). Oder etwas plakativer ausgedrückt: „A 'market' has never been observed paying a bill, 'customers' do that" ([PetWat82]).

Der Begriff, der am ähnlichsten wie Kundenorientierung verwendet wird, ist der der *Kundennähe*. Wenn man die relevante Literatur zum Thema Kundennähe sichtet, stellt man fest, daß beide Begriffe in der Regel synonym benutzt werden (vgl. u.a. [Zollner94], [Homburg94]), wobei Kundenorientierung manchmal als Folge von Kundennähe gesehen wird und manchmal umgekehrt. Eine Differenzierung beider Begriffe erscheint im Rahmen dieser Arbeit wenig sinnvoll.

Folgende Begriffsabgrenzungen können damit festgehalten werden:

Begriffsabgrenzungen:

1. ***Kundenorientierung als historisch gewachsener Marketing-schwerpunkt kann nicht mit Marketingorientierung gleichgesetzt werden.***
2. ***Kundenorientierung wird als Teil der Marktorientierung verstanden. Beide Begriffe sind deshalb nicht synonym.***
3. ***Die beiden Begriffe Kundenorientierung und Kundennähe werden im Rahmen dieser Arbeit synonym verwendet.***

Im folgenden werden deshalb auch Untersuchungen zum Thema Kundennähe herangezogen. Beide Begriffe werden im Rahmen dieser Arbeit synonym verwendet, jedoch wird in der Regel von Kundenorientierung gesprochen.

2.5 Forschungsergebnisse zum Thema Kundenorientierung

Der Begriff Kundenorientierung begegnet dem Beobachter in den folgenden Bereichen:

- Im Dienstleistungsbereich tritt Kundenorientierung als Streben nach Servicequalität und Kundenzufriedenheit auf.
- Im Bereich der Innovationsforschung wird Kundenorientierung mit dem Erfolg von Innovationen in Verbindung gebracht.
- In der Organisationsforschung wird Kundenorientierung als Leitbild von Unternehmens-entwicklung verstanden.
- In der Erfolgsfaktorenforschung ist Kundenorientierung als ein entscheidender Faktor des Unternehmenserfolges identifiziert worden.
- Als mögliche Umsetzung von Kundenorientierung wird eine Integration des Kunden in den Leistungserstellungsprozeß diskutiert.

Forschungsergebnisse aus diesen Bereichen sollen im folgenden näher beschrieben werden. Damit soll beim Leser nicht nur ein erster Eindruck davon entstehen, was unter Kundenorientierung zu verstehen ist, es soll auch deutlich werden, daß es sich hierbei um ein äußerst komplexes und vielschichtiges Konzept handelt.

2.5.1 Kundenorientierung und das Streben nach Servicequalität und Kundenzufriedenheit

Auslöser für das Streben nach höherer Servicequalität ist der bereits erwähnte zunehmend härtere Wettbewerb. Das Überleben eines Unternehmens ist durch einst erfolgreiche Konzepte, wie z.B. Produktdifferenzierung, Technologieführerschaft und Kostensenkung, nicht mehr gewährleistet (vgl. [Zollner94]). Ein Unternehmen muß deshalb nach anderen Bereichen suchen, um sich vom Mitbewerber zu differenzieren.

Da die Produkte in der Regel einen hohen Reifegrad besitzen (d.h. sie sind in den Augen des Kunden austauschbar geworden), versuchen viele Unternehmen durch einen überdurchschnittlich guten Service bzw. durch Anbieten zusätzlicher Dienstleistungen, Kunden zu gewinnen und langfristig an sich zu binden. Diverse Untersuchungen belegen, daß schlechter Service ein wesentlicher Grund für den Kunden sein kann, zu einem Konkurrenzunternehmen zu wechseln. Nach einer Untersuchung von WHITELEY hatten 70% der Gründe, daß Kunden abwanderten, nichts mit der Produktqualität zu tun (vgl. [White92]). Nach BERRY et al. ist das Brechen von Service-Versprechungen der wichtigste Einzelgrund, weshalb Dienstleistungsfirmen bei ihren Kunden versagen (vgl. [BerZeithPara90]).

Die naheliegende Gleichsetzung von Kundenorientierung und Servicequalität erscheint allerdings problematisch, wenn man z.B. auf Unternehmen wie Aldi oder IKEA schaut (vgl. [Eggert92], [Zollner94]). Diese Unternehmen zeichnen sich gerade durch ein Minimum an Service aus, verfügen anderseits aber über eine relativ hohe Produktqualität bei besonders geringen Preisen. Damit wird deutlich, daß das Streben nach hoher Servicequalität lediglich eine Möglichkeit ist, Kundenorientierung umzusetzen. Da aber ein niedriger Preis in bestimmten Marktsegmenten für Kunden interessant ist, kann vermutlich auch ein niedriger Preis kundenorientiert genannt werden. Wenn jedoch auch in diesen Niedrigpreisbereichen zunehmend härterer Wettbewerb stattfindet, wird vermutlich auch hier das Streben nach besserem Service eine höhere Bedeutung gewinnen.

Mit einer verbesserten Service-Qualität wird vorrangig eine höhere Kundenzufriedenheit angestrebt. In der Literatur besteht weitgehender Konsens darüber, daß man die Kundenorientierung eines Unternehmens am ehesten an der Kundenzufriedenheit messen kann. Kundenzufriedenheit wird als ein Maßstab dafür gesehen, inwieweit versprochene Leistungen mit wahrgenommener Leistung übereinstimmen. Ziel ist es, über Kundenzufriedenheit die Kundenbindung zu erhöhen, d.h. es geht um die Bildung einer langfristigen Stammkundschaft.

Die Bedeutung von Kundenbindung für den Unternehmenserfolg ergibt sich aus den folgenden Gründen:

- **Kostspielige Neuakquise** - Kunden neu zu akquirieren, scheint wesentlich teurer zu sein als das Halten alter Kunden (in der Literatur wird von einem Faktor 5 gesprochen)
- **Zusatzgewinne** - Langfristige Kundenbeziehungen versprechen zusätzliche Gewinne, über erhöhte Kauffrequenz, den Verkauf von Zusatzleistungen, die Durchsetzbarkeit höherer Preise und die Weiterempfehlung an andere Kunden (*reference selling*).

- **Gewinnsteigerung durch Verminderung der Abwanderungsrate -** Nach einer Untersuchung von REICHELD und SASSER bewirkt die Senkung der Abwanderungsrate um 5% je nach Branche eine Gewinnsteigerung von 25%-85%. Für die Softwarebranche geben die Autoren eine Gewinnsteigerung von 30% an (vgl. [ReichSass90]).

Unter dem Stichwort *Beziehungsmanagement bzw. Relationship Marketing* hat die Diskussion um Kundenorientierung/Kundennähe nicht nur im Investionsgütermarketing, sondern zunehmend auch im Dienstleistungsmarketing an Bedeutung gewonnen. Hier wird insbesondere der Begriff des Vertrauens und des Commitments eng mit Kundenorientierung in Verbindung gebracht (vgl. [Homburg94]).

In diesem Zusammenhang stehen auch die Bemühungen, Kundenzufriedenheit kontinuierlich zu messen und im Zuge eines Benchmarkings Vergleiche zu anderen Unternehmen anzustellen. Ziel ist es, von den 'Besten' zu lernen, um die Zufriedenheit der eigenen Kunden ständig zu verbessern.

Die Ergebnisse der Qualitäts- und Zufriedensheitsforschung lassen sich in folgender Kausalkette zusammenfassen:

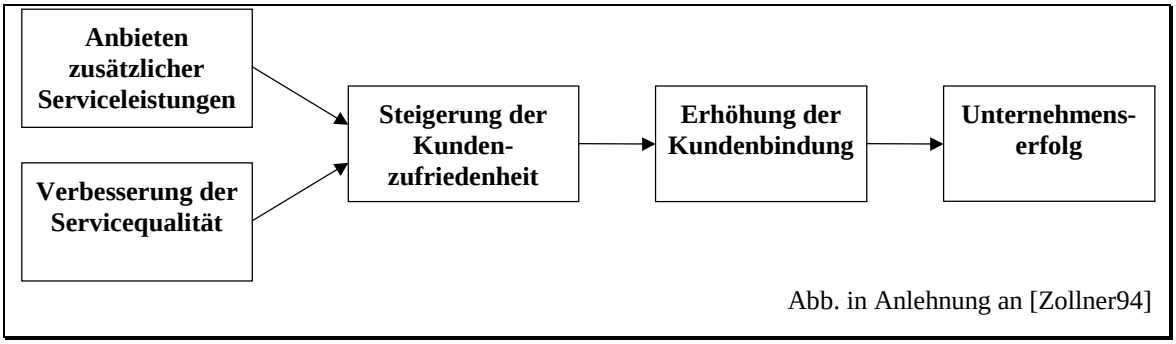

Abbildung 2-1

Mit der Verbesserung der Servicequalität und dem Anbieten von zusätzlichen Serviceleistungen versuchen die Unternehmen, sich von Mitbewerbern zu differenzieren und die Zufriedenheit der eigenen Kunden zu erhöhen, wovon sie sich eine stärkere Kundenbindung und damit langfristig einen höheren Unternehmenserfolg erhoffen.

Fragen in bezug auf Kundenorientierung in der Softwareentwicklung

⇒ Wie kann ein Softwarehersteller die Zufriedenheit seiner Kunden steigern?

⇒ Welche Gründe gibt es für Kundenunzufriedenheit in der Softwareentwicklung?

⇒ Wie sieht eine Steigerung der Servicequalität im Bereich der Softwareentwicklung aus?

⇒ Mit welchen zusätzlichen Serviceleistungen kann sich ein Softwarehersteller vom Mitbewerber differenzieren?

⇒ Wie kann ein Softwarehersteller dafür sorgen, Versprechungen gegenüber seinem Kunden einzuhalten?

⇒ Wie ändert sich die Geschäftsbeziehung zwischen Softwarehersteller und Kunde im Zuge einer kundenorientierten Softwareentwicklung?

2.5.2 Kundenorientierung und Innovationsentwicklung

Innovationen sind von größter Bedeutung für die Wettbewerbsfähigkeit von Unternehmen. Nach empirischen Untersuchungen machen Unternehmen im Durchschnitt 10% bis 25% ihres Umsatzes mit erst vor kurzer Zeit entwickelten Produkten, wobei diese Zahl in bestimmten

Branchen (wie z.B. der Softwarebranche) durchaus wesentlich höher sein kann (vgl. [Zollner94]).

Innovationstätigkeit scheint jedoch mit erheblichen Schwierigkeiten verbunden zu sein. ZOLLNER zitiert Untersuchungen, die auf Flopraten zwischen 70% und 95% hindeuten. In der Innovationsforschung gibt es zwei gegensätzliche Erklärungsmodelle für den Erfolg/Mißerfolg von Innovationen: Nach der Theorie des *Technologiedrucks* („technology push“) sind Innovationen im wesentlichen technikinduziert und kommen eher von der Herstellerseite als auf Grund einer Nachfrage des Marktes, wie es die Theorie des *Nachfragesogs* („demand pull“) unterstellt. Obwohl es bislang keine empirischen Nachweise für eine deutliche Überlegenheit der Nachfragesog-Hypothese gibt, besteht doch Konsens darüber, daß der Kunde selbst in der Innovationsentwicklung eine wichtige Rolle spielen kann.

Ausgehend von der Nachfragesog-Hypothese ist die genaue Kenntnis der Kundenbedürfnisse Grundvoraussetzung für erfolgreiche Innovationen. Zieht man die hohen Flopraten in Betracht, so dürfte es jedoch gerade in der Erhebung dieser Bedürfnisse gravierende Mängel geben. Traditionell wird die Ermittlung der Kundenbedürfnisse dem Bereich der Marktforschung zugeordnet. Dies wird von vielen Autoren kritisiert, da die Marktforschung erstens zu spät ansetzt (wenn das Produkt fast fertig ist) und zweitens einen zu reaktiven Charakter hat, da die Befragten immer nur ein konkretes Produkt bewerten und selten den von der Befragung gesteckten Rahmen verlassen. Somit kommt es eher zu Detailverbesserungen als zu wirklicher Innovation. Weiterhin hat sich die Vermittlung von Marktforschungsergebnissen über viele Organisationseinheiten innerhalb eines Unternehmens als problematisch erwiesen (vgl. [Zollner94]).

Im Rahmen des Innovationsmanagements wird deshalb gefordert, schon frühzeitig in der Entwicklung engeren Kundenkontakt herzustellen. Folgende Maßnahmen werden genannt:

- Mitarbeiter der Forschung und Entwicklung (F&E) sollen auf regelmäßiger Basis an Kundenbesuchen teilnehmen, um „realistische Vorstellungen von dem zu gewinnen, was wirklich Kundennutzen bedeutet“ ([Krub84]).
- Beteiligung von Kunden an Brainstorming-Meetings.
- Aktive Beteiligung von Kunden an der Produktentwicklung.
- Veränderung der Kommunikation mit dem Kunden vom einseitigen Monolog hin zu einem Dialog.
- Einrichtung von Beschwerdetelefonen, um die Kontaktaufnahme des Kunden mit dem Unternehmen zu erleichtern.
- Schaffung von finanziellen Anreizen, die den (potentiellen) Kunden zur Mitarbeit ermuntern sollen.

Zusammenfassend kann man sagen, daß in der Innovationsforschung zunehmend eine höhere Kundenbeteiligung gefordert wird, wobei es vor allem darum geht, den Entwicklern von Innovationen direkte Kontakte zu den späteren Nutzern zu ermöglichen. Damit wird Kundenorientierung in erster Linie als eine Intensivierung der Kommunikation zwischen dem F&E-Bereich und dem Kunden verstanden.

Fragen in bezug auf Kundenorientierung in der Softwareentwicklung

⇒ Gilt die Forderung nach stärkerer Kundenbeteiligung auch für den Bereich von Softwareentwicklung?

⇒ Welche Auswirkungen hat die Umsetzung von stärkerer Kundenbeteiligung für das Vorgehensmodell von Softwareentwicklungen?

2.5.3 Kundenorientierung in der Organisationsforschung

Die Organisationsforschung beschäftigt sich mit Fragen der Struktur und der Kultur von Unternehmen. Hier wird Kundenorientierung nicht nur als Aufgabe bestimmter Unternehmensbereiche (wie z.B. des F&E oder des Marketings) verstanden, sondern als eine Herausforderung für das gesamten Unternehmen. DROEGE stellt dazu fest:

> „Kundenorientierung ist nicht nur als absatz- und servicebezogene Tätigkeit der Marketingabteilung, sondern vielmehr als Unternehmensaufgabe zu verstehen, die nach innen wie nach außen gelebt werden muß.“ ([Droege96])

Nicht nur eine Abteilung soll kundenorientierter werden, sondern das ganze Unternehmen. Damit wird Kundenorientierung zum Ziel von unternehmensweiten Umgestaltungen - Kundenorientierung wird als Vision verstanden.

Die Organisationsforschung schlägt eine Reihe von Maßnahmen zur Erreichung von Kundenorientierung vor. Diese betreffen vor allem *aufbauorganisatorische Maßnahmen* und Maßnahmen zur Schaffung einer kundenorientierten *Unternehmenskultur*.

Als *aufbauorganisatorische Maßnahmen* (die also die Struktur des Unternehmens betreffen) werden u.a. folgende genannt:

- Vermeidung bürokratischer Organisationsstrukturen
- Abbau von Hierarchiestufen bzw. flache Hierarchien
- Aufbrechen zentraler Strukturen (Dezentralisierung)
- Einführung eines Key-Account-, Kunden-, und/oder Kundengruppenmanagements
- Einrichtung von Beschwerdestellen
- Nutzung der Reklamationsabteilung, u.a. als Frühwarnsystem und als Impulsgeber für Innovationen

Darüber hinaus wird als aufbauorganisatorische Maßnahme auch die Einführung moderner EDV-Systeme als Voraussetzung für Kundenorientierung genannt (Die Schmalenbach-Gesellschaft richtete dazu Ende der 80er Jahre sogar einen Arbeitskreis mit dem Titel „Kundennähe durch moderne Informationstechnologien“ ein). Ziel solcher *kundenorientierten Informationssysteme* ist es, den Mitarbeiter darin zu unterstützen, kundenorientiert agieren zu können. Weiterhin wird durch die Informationstechnologie eine direktere Kopplung von Unternehmen an ihre Kunden und Lieferanten angestrebt.

Aufbauorganisatorische Maßnahmen wollen somit das gesamte Unternehmen näher an den Kunden heranführen. Ziel ist es, die Kommunikation bzw. den Kontakt mit dem Kunden zu erleichtern und zu intensivieren.

Die Änderung der *Unternehmensstruktur* allein scheint jedoch nicht auszureichen, will man dem Ziel Kundenorientierung näherkommen. Die Schaffung einer entsprechenden *Unternehmenskultur* wird von vielen Autoren als viel wesentlicher angesehen (vgl. [Homburg94], [KobiWüth85], [Krulis84], [Zollner94]). HOMBURG findet in seiner Auswertung mehrerer Untersuchungen Indizien dafür, daß

> „Kunden- bzw. Marktorientierung stärker von der Kultur als von der Organisationsstruktur eines Unternehmens abhängt.“ ([Homburg94])

Ziel ist es, eine Unternehmenskultur zu schaffen, in der sich jeder Mitarbeiter konsequent am Kundennutzen und an den Kundenbedürfnissen ausrichtet. Folgende Maßnahmen werden zur Schaffung einer kundenorientierten Unternehmenskultur vorgeschlagen:

- Verankerung von Kundenorientierung im Leitbild des Unternehmens.

- Vorbildfunktion der Führungskräfte („Vielleicht ist es als erster Schritt zu betrachten, wenn im Sinne der Kundenorientierung zuerst der Parkplatz vor dem Haupteingang des Unternehmens für Kunden anstelle für die Geschäftsleitung…reserviert wird.“ [GünHub96])
- Denken wie Kunden.
- Mitarbeiter, die keinen direkten Kundenkontakt haben, sollen zeitweise in kundennahen Bereichen arbeiten. „Verbringen Sie einen Tag im Leben Ihres Kunden“ empfehlen in diesem Zusammenhang GOUILLART und KELLY ([GouKelly95]).
- Förderung von Prozeßorientierung bis hin zur Bildung einer prozeßorientierten Sekundärorganisation. Damit ist z.B. die Bildung von abteilungsübergreifenden Teams gemeint, die Verantwortung tragen für bestimmte Prozesse, die „quer“ zu den Hierarchieebenen der Primärorganisation (F&E, Marketing, Produktion, etc.) ‘vom Kunden zum Kunden’ laufen.
- Interne Kundenorientierung. Kundenorientierung als Gestaltungsmerkmal von internen Kunden- und Lieferanten-Beziehungen.
- Gelebte Mitarbeiterorientierung. Ziel ist es, durch Steigerung der Mitarbeiterzufriedenheit eine Steigerung des kundenorientierten Verhaltens zu erreichen, denn ‘Wer innen weint, kann außen schlecht lächeln’.
- Offene Kommunikation, die Innovation und Eigeninitiative fördert.
- Schulungs- und Trainingsprogramme für alle Mitarbeiter, um interne und externe Kundenorientierung zu lernen und zu verbessern (Kommunikationstrainings, Teamentwicklung, etc.).

Zusammenfassend kann man festhalten, daß die Organisationsforschung ein Unternehmen genau dann kundenorientiert nennt, wenn „mittels aufbauorganisatorischer Maßnahmen sowie einer kundenorientierten Unternehmenskultur Bedingungen geschaffen werden, die einen regen Informationsaustausch zwischen den Mitarbeitern des Unternehmens und seinen Kunden ermöglichen“ ([Zollner94]).

Fragen in bezug auf Kundenorientierung in der Softwareentwicklung

⇒ Welche Anforderungen lassen sich für kundenorientierte Informationssysteme formulieren?

⇒ Welche Rolle spielt Mitarbeiterorientierung für die Gestaltung von Softwaresystemen?

⇒ Wie kann das geforderte „Denken wie Kunden“ in Softwareentwicklungsprojekten umgesetzt werden?

⇒ Welche Auswirkungen hat eine Zunahme von Prozeßorientierung für Softwareentwicklungen?

⇒ Wie kann der Softwarehersteller selbst Mitarbeiterorientierung umsetzen?

⇒ Welche strukturellen und kulturellen Konsequenzen hat die Forderungen nach Kundenorientierung für den Softwarehersteller?

2.5.4 Kundenorientierung als Erfolgsfaktor

Ein Konzept wie die Kundenorientierung wäre uninteressant, wenn es nicht direkt oder indirekt zum Unternehmenserfolg beitragen würde. Es ist die Aufgabe der empirischen Erfolgsfaktorenforschung, diejenigen Komponenten unternehmerischen Handelns zu identifizieren, die zum marktwirtschaftlichen Erfolg und Bestehen von Unternehmen beitragen.

In diesem Forschungsbereich hat der bereits erwähnte Bestseller von PETERS und WATERMAN eine intensive wissenschaftliche Diskussion um den Stellenwert von Kundenorientierung als Erfolgsfaktor ausgelöst (vgl. [PetWat82]). Die Autoren stellen vier Komponenten zur Erreichung von Kundenorientierung heraus:

- Servicebesessenheit
- Qualitätsbesessenheit
- Nischen-Denken
- Eingehen auf Kundenwünsche

EGGERT kommt in ihrer kritischen Würdigung zu dem Ergebnis, daß es sich bei keinem dieser vier Faktoren um wirkliche Neuentdeckungen handelt, sondern daß die einschlägige betriebswirtschaftliche Literatur sich seit langer Zeit (teilweise seit 1940) intensiv mit ihnen beschäftigt (vgl. [Eggert92]). Trotz aller Kritik an PETERS und WATERMAN löste ihre Arbeit eine Reihe von Untersuchungen aus, die die strategische Bedeutung von Kundenorientierung als Erfolgsfaktor letztlich bestätigen konnten:

CAVANAGH und CLIFFORD beispielsweise untersuchten 100 mittelgroße US-Unternehmen und kamen zu dem Ergebnis, daß sich die erfolgreichsten unter ihnen durch außergewöhnlich hohe Kundennähe auszeichneten (vgl. [ClifCav86]). Die Autoren verstanden darunter:

- häufige Kundenkontakte,
- Denken wie Kunden und
- Verstehen von Kundenbedürfnissen.

SIMON untersuchte 39 kleinere und mittlere Unternehmen, die er „Hidden Champions“ nannte, da sie Welt- oder Europamarktführer sind, ohne einer breiten Öffentlichkeit bekannt zu sein (vgl. [Simon90]). Er kommt zu dem Ergebnis, daß diese Unternehmen „ausgeprägte Wettbewerbsvorteile bei Produktqualität, Kundennähe und Service“ (ebd.) innehaben. Kundennähe kann nach SIMON u.a. erreicht werden durch:

- regelmäßige direkte Kundenkontakte,
- hohe Kontaktintensitäten mit Kunden und
- kürzere interne Kommunikationsbeziehungen.

Alle diese Arbeiten identifizierten Kundenorientierung/Kundennähe als Erfolgsfaktor, jedoch blieben die Autoren fast immer eine genaue Definition des Begriffes Kundenorientierung schuldig. Außerdem konzentrierten sich die Untersuchungen fast ausschließlich auf die Unternehmen selbst, d.h. die „Kundenperspektive“ wurde außer acht gelassen. Auf diesen überraschenden Umstand weist z.B. ZOLLNER besonders hin (vgl. [Zollner94]).

In der deutschsprachigen Literatur sind beide Lücken durch die Arbeit von HOMBURG geschlossen worden (vgl. [Homburg94]). Dieser befragte in großangelegten Untersuchungen zunächst Kunden, was sie sich unter Kundennähe vorstellten und bewertete danach die Unternehmen. HOMBURG kam zu folgender (durch komplexe statistische Verfahren abgesicherter) Konzeptionalisierung von Kundennähe:

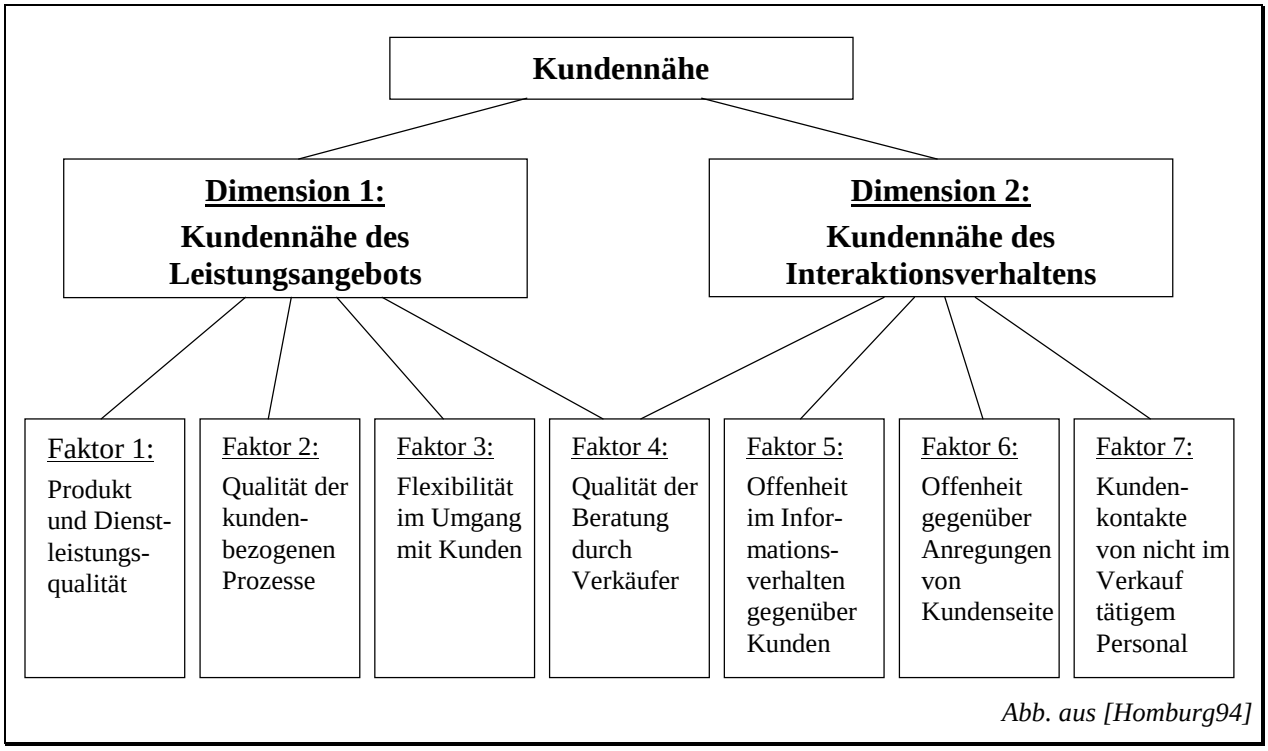

Abbildung 2-2

Die sieben aufgeführten Faktoren von Kundennähe bestehen dabei ihrerseits aus einer Reihe von Einzelkomponenten, die hier nicht weiter aufgeführt werden sollen. Aus der Fülle der von HOMBURG gefundenen Zusammenhänge sind besonders die folgenden für diese Arbeit interessant:

- Zunächst einmal konnte tatsächlich ein positiver Zusammenhang zwischen Kundennähe und Unternehmenserfolg festgestellt werden. Kundenorientierung ist demnach ein Erfolgsfaktor.
- Positive Auswirkungen hat Kundenorientierung auf die Geschäftsbeziehung und damit auf eine langfristige Zusammenarbeit zwischen Hersteller und Kunde. Dies gilt besonders in Bereichen, die sich durch eine hohe technologische Komplexität auszeichnen. Damit kommt Kundenorientierung im Bereich der Softwareentwicklung eine wesentliche Bedeutung zu.
- Für Kundenorientierung gibt es gewisse strukturelle und kulturelle Voraussetzungen:
  - Strukturell wirken sich besonders Entscheidungsdelegation in der Hierarchie und Koordination durch Selbstabstimmung (damit also selbstverantwortliches Arbeiten) positiv auf Kundenorientierung aus. Nachteilig wirken Formalisierung und Koordination durch persönliche Weisung.
  - Kulturell hat ein bürokratisches System gravierende Nachteile in bezug auf Kundenorientierung.

Zusammenfassend kann man festhalten, daß Kundenorientierung als Erfolgsfaktor identifiziert wurde. Kundenorientierung bezieht sich dabei einerseits auf Produkt- und Dienstleistungs-

gestaltung, als auch auf eine spezielle Art des Verhaltens. Außerdem legen die Untersuchungen nahe, daß es bestimmte organisatorische Grundvoraussetzungen für Kundenorientierung gibt, insbesondere eine nichtbürokratische Aufbauorganisaition und eine kundenorientierte Unternehmenskultur.

Fragen in bezug auf Kundenorientierung in der Softwareentwicklung

⇒ Wie kann bei einem Softwarehersteller eine kundenorientierte Struktur aussehen?

⇒ Wie sieht eine kundenorientierte Unternehmenskultur bei einem Softwarehersteller aus?

⇒ Wie kann beim Softwarehersteller die Forderung nach selbstverantwortlicher Arbeit umgesetzt werden?

2.5.5 Kundenorientierung durch Kundenintegration

Ein weiterer Ansatz aus der betriebswirtschaftlichen Literatur, der für diese Arbeit Relevanz besitzt, wird *Customer Integration* genannt (vgl. Artikelsammlung in [KleFliJac96]). In seiner Einleitung stellt KLEINALTENKAMP fest, daß in Wissenschaft und Praxis mittlerweile Konsens darüber besteht, daß „nur ein wirklich kundenorientiertes Unternehmen langfristig im Wettbewerb überleben kann“ ([Kleinalt96]). Gleichzeitig stellt er fest, daß die meisten Unternehmen von einer Realisierung dieses Anspruches weit entfernt sind. Appelle an die Kundenorientiertheit jedes einzelnen bewirken wenig, es „müssen Strukturen geschaffen werden, welche die Menschen lenken, sich entsprechend den Maximen zu verhalten.“ ([Kleinalt96]). Customer Integration versteht der Autor folglich als ein Konzept, welches über das bloße Predigen hinausgeht und der tatsächlichen Umsetzung von Kundenorientierung dient.

Ausgangspunkt ist die Tendenz im Business-to-Business-Sektor, zusätzlich zu fertigen Produkten verstärkt auch Dienstleistungen mit anzubieten. Der Charakter von Dienstleistungen macht es gerade aus, daß sie nicht ohne Mitwirkung des Kunden stattfinden können, d.h. daß per se immer eine mehr oder minder starke Integration stattfindet. Die „Integration des externen Faktors“ ist denn auch ein Definitionsbestandteil des Begriffes *Dienstleistung* (vgl. [Zollner94]). Informationen des Kunden gehen somit als Produktionsfaktoren in den Leistungserstellungsprozeß mit ein. In Anlehnung an KLEINALTENKAMP kann man deshalb folgende Begriffsdefinition formulieren:

> Customer Integration ist eine Vorgehensweise, die die Verschmelzung der Wertschöpfungsprozesse von Kunden und Anbietern fördern und damit zur Realisierung einer effektiven und gleichzeitig effizienten Kundenorientierung führen soll.
>
> Grundprinzip der Customer Integration ist es, das Problem des Kunden zusammen mit dem Kunden zu lösen.

Innerhalb der Diskussion um Customer Integration ist besonders der dort erkannte wesentlichste Hinderungsgrund für Customer Integration zu nennen: Das Fehlen von „Prozeßevidenz“ (vgl. [Fließ96]). Mangelnde Prozeßevidenz resultiert aus mangelndem *Prozeßbewußtsein* einerseits und aus mangelnder *Prozeßtransparenz* anderseits.

Fehlendes Prozeßbewußtsein bedeutet z.B., daß dem Kunden nicht klar ist, daß sein Mitwirken für den Leistungserstellungsprozeß von Bedeutung ist. Selbst wenn dieses Bewußtsein vorhanden ist, kann es unklar sein, an welchen Stellen im Prozeß der Kunde seine Leistungen einbringen kann. Dann spricht man von fehlender Prozeßtransparenz. Mangelnde Prozeßevidenz tritt auch auf Seiten des Anbieters auf, beispielsweise wenn er neue Produkte/Dienstleistungen anbietet oder wenn die Zusammenarbeit mit dem Kunden noch

nicht eingespielt ist. Als Ursachen für mangelnde Prozeßevidenz identifiziert FLIEß sowohl Fähigkeits- als auch Willensbarrieren (vgl. [Fließ96]).

Als Fazit kann man festhalten, daß innerhalb der betriebswirtschaftlichen Forschung die Integration des Kunden in den Leistungserstellungsprozeß als eine Möglichkeit zur Umsetzung von Kundenorientierung erkannt wurde.

Fragen in bezug auf Kundenorientierung in der Softwareentwicklung

⇒ Wie kann eine Kundenintegration im Rahmen von Softwareentwicklungsprojekten aussehen?

⇒ Sind die im Rahmen der Softwaretechnik vorgeschlagenen evolutionären und partizipativen Entwicklungsstrategien geeignet, diese Integration zu ermöglichen?

2.6 Was ist Kundenorientierung?

Nachdem der vorige Abschnitt dazu diente, einen Überblick über die Forschungsergebnisse zum Thema Kundenorientierung zu geben, soll jetzt eine Definition von Kundenorientierung erarbeitet werden. Als Vorbereitung werden zunächst die wesentlichen Ziele und die Voraussetzungen von Kundenorientierung diskutiert und wichtige Begriffsdefinitionen aus der betriebswirtschaftlichen Forschung vorgestellt. Dann werden die Ergebnisse dieses Kapitels zusammengefaßt und abschließend eine mehrteilige Definition des Begriffes Kundenorientierung gegeben.

2.6.1 Ziele von Kundenorientierung

Kundenorientierung war als Reaktion der Unternehmen auf einen zunehmenden Wettbewerbsdruck eingeführt worden. Das Unternehmensumfeld ist komplexer und unberechenbarer geworden, Veränderungen geschehen schneller. Märkte sind gekennzeichnet durch zunehmend geringe oder stagnierende Wachstumsraten, ruinösen Preiswettbewerb und steigende individuelle Anforderungen durch emanzipierte und qualitätsverwöhnte Kunden (vgl. u.a. [Zollner94], [DopLaut94]).

In dieser Situation suchen die Unternehmen nach neuen Erfolgsstrategien, die den langfristigen Erfolg (und Erhalt) des Unternehmens sichern sollen. Kundenorientierung ist eine von ihnen.

Damit kann folgendes Zwischenergebnis festgehalten werden:

> ***Kundenorientierung ist eine Strategie, die den langfristigen Unternehmenserfolg sichern soll.***

Die Strategie Kundenorientierung fokussiert deutlich auf die bestehenden Kunden einer Unternehmung. Hier ist die Zielsetzung, den Kunden an das Unternehmen zu binden, d.h. eine langfristige wechselseitige Geschäftsbeziehung aufzubauen.

Somit kann als zweites Zwischenergebnis festgehalten werden:

> ***Kundenorientierung ist eine Strategie, die die Erhöhung der Kundenbindung anstrebt.***

Der zentrale Begriff im Rahmen der Diskussion um Kundenorientierung ist die ***Kundenzufriedenheit***. Die Erreichung und Verbesserung von Kundenzufriedenheit ist das Ziel von vielen Maßnahmen, die in der Literatur zum Thema Kundenorientierung beschrieben werden. Von einer Erhöhung der Kundenzufriedenheit wird eine höhere Bereitschaft des Kunden abgeleitet, dem Unternehmen treu zu bleiben. SALZGEBER formuliert es so:

> „Langfristig ist der Kundenzufriedenheit die gleiche Bedeutung beizumessen wie dem Unternehmensgewinn.“ ([Salzg96])

Als drittes Zwischenergebnis halten wir deshalb fest:

> ***Kundenorientierung ist eine Strategie, die auf die Erreichung maximaler Kundenzufriedenheit ausgerichtet ist.***

2.6.2 Wer ist der Kunde?

Die Diskussion um Kundenorientierung hatte ihren Ursprung zunächst in solchen Unternehmen, die direkten Kontakt zu Endverbrauchern haben. Hier zeichnete sich schon einige Zeit lang eine Veränderung des Kundenbildes ab. Das Bild des passiven, reaktiven Kunden mit linearem, einschätzbarem Verhalten ist zunehmend durch dem Bild des emanzipierten Kunden mit hybrider, schwer prognostizierbarer aber ganzheitlicher Bedürfnisstruktur gewichen. Damit einher geht eine Verschiebung von sog. 'Verkäufermärkten', in denen sich Kunden praktisch alles gefallen lassen mußten, hin zu 'Käufer-Märkten', wo hohe Service- und Produktqualität immer entscheidender wurden und wo der Kunde zu einem „kritischen und umkämpften Marktpartner, und damit zur Herausforderung" ([Althaus95]) geworden ist.

Auch im sog. 'Business-to-Business'-Bereich ist eine ähnliche Tendenz erkennbar. Hier steht dem Anbieter nicht ein einzelner Kunde gegenüber, sondern ein ganzes Unternehmen. Die zunehmende Bedeutung von Kundenorientierung in diesem Bereich schlug sich in einer Höherbewertung der Geschäftsbeziehung nieder. Begriffe wie 'Relationship-Marketing' oder 'Beziehungsmanagement' stehen für diese Entwicklung. Auch hier geht es um die Schaffung von Kundenzufriedenheit mit dem Ziel, langfristige Kundenbindung zu erreichen.

Darüber hinaus wurde Kundenorientierung nicht nur auf den 'externen' Kunden bezogen, sondern auch auf den 'internen'. Im Rahmen von 'interner Kundenorientierung' werden die Ideen der Kundenorientierung jetzt auch auf die interne Zusammenarbeit in einem Unternehmen übertragen.

2.6.3 Ein kundenorientierter Qualitätsbegriff

Mit Kundenorientierung ist die Umsetzung eines kundenorientierten Qualitätsbegriffes verbunden. Die Qualität von Produkten und Dienstleistungen wird aus der Sicht des Kunden beurteilt. Damit wird Qualität zu einem explizit subjektivem Begriff, der eng mit dem Konzept der Zweckeignung und der Nutzenerwartung verknüpft ist. Andere Qualitätsbegriffe (wie z.B. produktorientierte Qualität oder herstellungsorientierte Qualität) werden damit nicht überflüssig, sondern sind lediglich einem kundenorientierten Qualitätsbegriff untergeordnet (vgl. [Zollner94]).

2.6.4 Kundenorientierung als Integrationsfaktor von Unternehmensführung

Die Literatur stimmt dahingehend überein, daß Kundenorientierung nicht nur Aufgabe einzelner Abteilungen (wie z.B. des Marketings oder des Vertriebs) ist, sondern als eine Herausforderung für das gesamte Unternehmen zu verstehen ist. Nicht nur eine Abteilung soll kundenorientierter werden, sondern das gesamte Unternehmen. Bildlich gesprochen, rückt das Unternehmen als Ganzes näher an den Kunden heran. Einerseits wird Kundenorientierung dadurch mehr zu einem Problem der Führung, andererseits auch zum Ziel von unternehmensweiten Umgestaltungen.

Kundenorientierung kommt der Stellenwert einer *Strategie* zu. Hiermit bestimmt das Unternehmen seine grundsätzliche Ausrichtung am Markt. Die Strategie Kundenorientierung, die zunächst einmal starken Visionscharakter hat, zielt dabei auf eine engere Anbindung des Unternehmens an den Kunden und umgekehrt.

Zur Umsetzung von Kundenorientierung bedarf es bestimmter Organisationsstrukturen und einer entsprechenden Unternehmenskultur, die letztlich ein kundenorientiertes Verhalten der Mitarbeiter fördern sollen (vgl. [Bleicher91], [DopLaut94], [Althaus95]).

Die folgende Abbildung zeigt die Bereiche, die von der Kundenorientierung betroffen sind:

Abbildung 2-3

Indem Kundenorientierung auf jeden dieser Bereiche wirkt, kann man von Kundenorientierung „als Integrationsfaktor ganzheitlicher Unternehmensführung“ ([Althaus95]) sprechen.

2.6.5 Begriffsdefinitionen aus der Literatur

ZOLLNER schlägt als primäre Maßnahme zur Erreichung von Kundenzufriedenheit die Ausweitung und Verbesserung der Kundenkontakte vor und definiert Kundennähe als „Qualität des Kundenkontaktes“. Da hierbei ein kundenorientierter Qualitätsbegriff zugrundegelegt wird, erfordert das Streben nach Kundenorientierung die „explizite Berücksichtigung der Kundenperspektive“ ([Zollner94]).

HOMBURG ermittelte in einer groß angelegten Untersuchung, was Kunden sich unter dem Begriff Kundennähe vorstellen. Als Ergebnis wird der Begriff Kundennähe als „Kundennähe des Leistungsangebotes“ und „Kundennähe des Interaktionsverhaltens“ operationalisiert ([Homburg94]).

Nach HERZWURM et al. bedeutet Kundenorientierung die

> „Ausrichtung der Unternehmensprozesse und der Organisation auf den Kunden“ ([HerzHier96]), mit dem Ziel, maximale Kundenzufriedenheit zu erreichen.

Während hier auf die strukturbildende Funktion von Kundenorientierung hingewiesen wird, betonen andere Autoren die Verhaltensebene und heben die Bedeutung jedes einzelnen Mitarbeiters hervor. Nach KLEINALTENKAMP etwa bedeutet Kundenorientierung, daß

> „alle Mitarbeiter eines Unternehmens täglich in den Kategorien der Kunden denken, sich für die Lösung der Kundenprobleme einsetzen und damit zur Herbeiführung von Kundenzufriedenheit beitragen.“ ([Kleinalt96])

Eine ähnliche Definition bringt KÜHN:

> „Kundenorientierung bedeutet eine grundsätzlich positive Grundeinstellung der Mitarbeiter einer Unternehmung zu den Kunden und ihren Bedürfnissen, die dazu führt, daß entsprechende Vorstellungen in ihre Entscheidungen einfließen und kundennahe Verhaltensweisen bewirken.“ ([Kühn91])

In der Hervorhebung einer bestimmten Grundhaltung deutet sich bereits an, daß die Umsetzung von Kundenorientierung eine tiefgreifende Veränderung in jedem einzelnen als Notwendigkeit (und als Konsequenz) hat.

Die oben angeführten Definitionen haben jedoch nach ALTHAUS eine gravierende Schwäche: Sie fokussieren zu sehr auf den Status Quo, d.h. auf die momentanen Bedürfnisse des Kunden. Damit stehen sich Kundenorientierung und Produktorientierung diametral gegenüber. Ein herkömmliches Verständnis von Kundenorientierung kann somit geradezu in einer Innovationsfeindlichkeit resultieren (vgl. [Althaus95]). Deshalb muß eine ‘emanzipierte Definition’ von Kundenorientierung nicht nur die gegenwärtigen, sondern auch die zukünftigen Kundenbedürfnisse mit einschließen. ALTHAUS definiert deshalb Kundenorientierung als

> „die Forderung nach einer bereichsübergreifenden Verpflichtung, den gegenwärtigen wie zukünftigen Erwartungen des Kunden, intern wie extern, gerecht zu werden.“ ([Althaus95])

Indem neben den aktuellen auch die zukünftigen Erwartungen des Kunden in die Definition eingehen, wird das traditionelle Spannungsfeld zwischen Produktorientierung und Kundenorientierung überwunden. Außerdem wird in dieser Definition explizit das Konzept des „internen Kunden“ angesprochen. Damit wird Kundenorientierung auch für die Zusammenarbeit innerhalb eines Unternehmen relevant.

2.7 Zwischenfazit und Begriffsdefinition *Kundenorientierung*

Aus dem Quellenstudium ist deutlich geworden, daß es sich beim Phänomen Kundenorientierung weder um einen Modetrend noch um ein gänzlich neues Managementkonzept handelt. Kundenorientierung bedeutet die Rückbesinnung auf den Kern unternehmerischen Handelns, der in der historischen Entwicklung vernachlässigt bzw. nur mangelhaft umgesetzt wurde.

Solange sich die Unternehmen in sog. „Verkäufer-Märkten“ bewegten, in denen sich die Kunden praktisch alles gefallen lassen mußten, wirkte sich dies nicht spürbar negativ auf den Unternehmenserfolg aus. Im zunehmend härteren Wettbewerb wurden aus diesen Verkäufer-Märkten jedoch mehr und mehr „Käufer-Märkte“, in denen Produkt- und Servicequalität für den Unternehmenserfolg immer entscheidender wurden. Die Wiederentdeckung der Kundenorientierung stellt demnach eine Anpassungsreaktion von Unternehmen dar und ist nicht nur eine Erfolgs-, sondern in erster Linie eine Überlebensstrategie. Anders ausgedrückt: Unternehmen wollen nicht kundenorientierter werden (z.B. weil es enormen Veränderungsaufwand bedeuten kann), sie müssen es!

Kundenorientierung kann zunächst als eine **Vision** verstanden werden, als

> „ein konkretes Zukunftsbild, nahe genug, daß wir die Realisierbarkeit noch sehen können, aber schon fern genug, um die Begeisterung der Organisation für eine neue Wirklichkeit zu erwecken“ ([Bosten88]).

Einerseits beschreibt Kundenorientierung die grundsätzliche Ausrichtung eines Unternehmens und seiner Mitarbeiter, andererseits bündelt der Begriff eine Vielzahl von Maßnahmen zur Erreichung und Verbesserung dieser Zielsetzung. Aufgrund der durchweg als mangelhaft bezeichneten Umsetzung ist die Zielsetzung Kundenorientierung untrennbar mit

Organisationsentwicklung verbunden und kann als Zielformulierung einer solchen Entwicklung dienen.

Kundenorientierung betrifft nicht nur einzelne Abteilungen eines Unternehmens (wie z.B. klassischerweise die Marketing-Abteilung), sondern das gesamte Unternehmen. Es dient nicht nur als Ausrichtung nach außen auf den „externen Kunden“ (und auf die Kunden des Kunden!), sondern bietet auch im Sinne des „internen Kunden“ Strukturierungshilfe für die Zusammenarbeit im Unternehmen.

Die Kundenorientierung kann es nicht geben. Deshalb kann es auch keine Patentrezepte zu einer Umsetzung geben, sondern es wird situationsgemäße und/oder branchenspezifische Ausprägungen geben.

Im weiteren wird folgende mehrteilige Definition von Kundenorientierung zugrundegelegt:

Kundenorientierung bedeutet

A) die Ausrichtung der Unternehmensprozesse und der Organisation auf den Kunden,

B) eine grundsätzlich positive Grundeinstellung der Mitarbeiter einer Unternehmung zu den Kunden und ihren Bedürfnissen,

C) die unternehmensweite, abteilungsübergreifende Verpflichtung, den gegenwärtigen und zukünftigen Erwartungen des internen und externen Kunden gerecht zu werden,

D) und eine kontinuierliche Bemühung, die Kundenzufriedenheit zu verbessern, um Kundenbindung zu erhöhen und damit den langfristigen Unternehmenserfolg sicherzustellen.

3 Kundenorientierung in der Softwareentwicklung

Nachdem im vorigen Kapitel der Begriff Kundenorientierung erläutert wurde, soll nun die Bedeutung von Kundenorientierung in der Softwareentwicklung diskutiert werden. Dabei wird ausdrücklich nicht der Anspruch einer umfassenden theoretischen Fundierung des Konzeptes *Kundenorientierte Softwareentwicklung* erhoben, sondern es wird der Versuch gemacht, erste Denkanstöße und Ansätze für ein solches Konzept zu entwickeln.

Dabei konzentriert sich die vorliegende Arbeit auf den Bereich der Individualsoftware-entwicklung innerhalb von Organisationen. Die Entwicklung von Standardsoftware wird aus den Überlegungen weitestgehend ausgeklammert; Näheres dazu findet sich u.a. bei HERZWURM et al. (vgl. [HerzHier96]).

Im folgenden wird zunächst erläutert, warum Kundenorientierung in der Softwareentwicklung notwendig ist. Dann soll diskutiert werden, welche Konsequenzen Kundenorientierung für die Produkt- und Prozeßgestaltung hat.

Es werden Anforderungen an eine kundenorientierte Produktgestaltung formuliert, die im wesentlichen eine Abkehr von der traditionellen ablaufsteuernden Sichtweise betonen. Das Ziel traditioneller Softwaresysteme liegt eher in einer Kontrolle des Benutzers als in einer Unterstützung. Im Gegensatz zu der ablaufsteuernden Sichtweise wird eine unterstützende Sichtweise als Grundlage kundenorientierter Softwaresysteme vorgeschlagen.

Dann werden Anforderungen an eine kundenorientierte Prozeßgestaltung formuliert, und es werden Vorgehensmodelle diskutiert, die diese Anforderungen umsetzen können. Dabei soll deutlich werden, daß traditionelle, phasenorientierte Vorgehensweisen als Grundlage für eine kundenorientierte Prozeßgestaltung ungeeignet sind. Es soll gezeigt werden, warum sich evolutionäre Vorgehensmodelle hier als erfolgversprechender darstellen.

Als Voraussetzungen für eine kundenorientierte Softwareentwicklung werden sowohl strukturelle als auch kulturelle Aspekte untersucht, wobei der Schwerpunkt auf die kulturellen Aspekte gesetzt wird. Es wird sowohl die Vision einer kundenorientierten Unternehmens-kultur, als auch das Bild eines „kundenorientierten Softwareentwicklers“ skizziert.

3.1 Die Notwendigkeit von Kundenorientierung in der Softwareentwicklung

Als Folge des sich verschärfenden Wettbewerbes geschehen zur Zeit in vielen Unternehmen z.T. grundlegende Umstrukturierungsbemühungen. Kundenorientierung ist dabei eine entscheidende strategische Zielsetzung. Auch für die softwareentwickelnden Organisations-einheiten, sowie die externen Software-Anbieter wird sich damit in zunehmendem Maße die Frage nach der eigenen Kundenorientierung stellen. So sind z.B. BAUKROWITZ et al. der Meinung, daß die Strukturveränderungen in Unternehmen auch Veränderungen in den DV-Abteilungen zur Folge haben werden:

> „Die Verantwortlichen sind gefordert, sich auch bei der Erarbeitung neuer IT-Strategien auf die neuen Erfordernisse einzustellen. Die DV-Abteilungen stehen angesichts dieser Entwicklung in einer vertrackten Lage: entweder sie stellen sich mit neuen IT-Strategien an die Spitze der Bewegung, die an der Reorganisation der Arbeit ansetzen, oder sie werden von den Fachabteilungen in ihrem ureigensten Feld überholt - die neuen Strategien werden dann von dort aus entwickelt.“ ([BauBoeEck94])

Softwareentwicklung findet in zunehmendem Maße im Rahmen von Umstruktierungs-maßnahmen statt. So wird z.B. im Rahmen vom Business Process Reengineering der

Unterstützung der neuen Unternehmensprozesse durch entsprechende Informationssysteme eine hohe Bedeutung beigemessen (vgl. [HammChamp94]).

An die entstehenden Softwaresysteme wird zunehmend die Anforderung gestellt werden, kundenorientierte Arbeit zu unterstützen. So wird innerhalb der Diskussion um Kundenorientierung explizit die Einführung sog. *kundenorientierter Informationssysteme* gefordert [vgl. [Althaus95]). Diese sollen das Unternehmen bzw. den einzelnen Mitarbeiter in seinem Bemühen unterstützen, die Nähe zum Kunden zu erhöhen und für Kundenzufriedenheit zu sorgen. Im Rahmen der betriebswirtschaftlichen Diskussion wird zwar die Bedeutung kundenorientierter Informationssysteme betont, jedoch wird die Frage, wie diese Softwaresysteme entstehen sollen, nicht thematisiert. In dieser Arbeit wird konsequenterweise gefordert, daß kundenorientierte Softwaresysteme selber auf kundenorientierte Weise entwickelt werden müssen.

Man kann damit sagen, daß Kundenorientierung für die Softwareentwicklung auf mehreren Ebenen Relevanz besitzt. Zunächst bietet sich Kundenorientierung als strategische Ausrichtung für Softwarehersteller genauso an, wie für andere im Wettbewerb stehende Unternehmen auch. Weiterhin werden viele Unternehmen selber kundenorientierter, d.h. sie werden in zunehmendem Maße auch von ihren Geschäftspartnern Kundenorientierung verlangen und dies im besonderen von denen, die (wie die Softwarehersteller) strategisch entscheidende Produkte und Dienstleistungen für das Unternehmen liefern. Richtet sich ein Softwarehersteller also auf mehr Kundenorientierung aus, so vollzieht er damit eine Entwicklung nach, die bei seinen Kunden geschieht und die diese über kurz oder lang auch von ihm fordern werden.

3.2 Wer ist der Kunde?

Die Orientierung zum Kunden erfordert zunächst einmal die Beantwortung der Frage, wer der Kunde überhaupt ist?

Auf der Ebene der beteiligten Organisationen ist diese Frage noch vergleichsweise einfach zu beantworten. Hier geht es bei Individualsoftwareentwicklung um die Entwicklung von maßgeschneiderten Anwendungssystemen für (zunächst) genau einen Kunden. Hier stehen sich als Lieferant/Hersteller die Organisation, die die Software entwickelt (im folgenden *Entwicklungsorganisation* oder *Softwarehersteller* genannt), und als Kunde diejenige, die sie schließlich anwendet (die *Anwendungsorganisation* oder der *Kunde*), gegenüber. Beide Organisationen können entweder in Gestalt von eigenständigen Unternehmen oder als Abteilung erscheinen. Beispielsweise kann die Entwicklungsorganisation sowohl ein selbständiges EDV-Unternehmen sein als auch eine DV-Abteilung innerhalb eines Unternehmens.

Die folgende Abbildung zeigt den Rahmen, in dem sich eine kundenorientierte Softwareentwicklung bewegen muß:

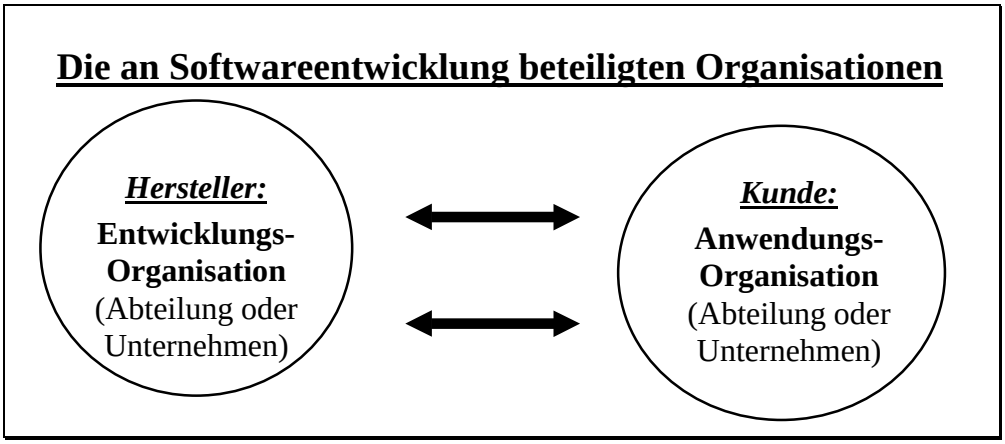

Abbildung 3-1

Kundenorientierung bedeutet hier zunächst einmal eine Veränderung der Geschäftsbeziehung zwischen Entwicklungs- und Anwendungsorganisation, die auf eine engere Kopplung und langfristigere Zusammenarbeit der beiden Systeme hinausläuft. Diese strategische Ausrichtung von Kundenorientierung muß in verschiedenen Bereichen umgesetzt und unterstützt werden. Auf der Seite der Entwicklungsorganisation müssen Organisationsstrukturen und eine entsprechende Unternehmenskultur geschaffen werden, die Kundenorientierung unterstützen. Organisationsstruktur und -kultur dienen demnach als Fundament, auf dem Vorgehensweisen von Softwareentwicklung aufsetzen müssen, die die angestrebte enge Bindung von Anwendungs- und Entwicklungsorganisation fördern bzw. erst ermöglichen.

Betrachtet man die Anwendungsorganisation detaillierter, so ist die Frage ‘Wer ist der Kunde?’ nicht mehr ohne weiteres zu klären. Auf der Seite der Anwendungsorganisation gibt es **den** Kunden meist nicht mehr. Vielmehr gibt es immer eine Vielzahl verschiedener Kundentypen mit unterschiedlichen Interessen, z.B.:

- Kunden als Auftraggeber bzw. Käufer, z.B. Abteilungsleiter, mittleres und höheres Management.
- Kunden als Benutzer, d.h. die Endanwender, die direkt mit dem System arbeiten.
- Kunden als indirekt Betroffene, z.B. Datenschutzbeauftragte, Betriebsräte, etc.

Als Zwischenergebnis kann man deshalb festhalten:

> ***Kunden sind alle direkt oder indirekt von Entwicklung und Einsatz des Softwaresystems Betroffene.***

Diese Definition deutet in ihrer Umfassenheit bereits auf eine Grundschwierigkeit von Softwareentwicklung hin: Die Zielgruppe von Softwareentwicklung kann nicht nur sehr groß werden, sondern zeichnet sich in der Regel durch eine Heterogenität der Interessen und Anforderungen der Betroffenen aus. Dazukommt, daß der Kundenkreis im Zeitverlauf nicht konstant bleibt. So muß sich beispielsweise die Akquise von Entwicklungsaufträgen mit z.T. anderen Kunden auseinandersetzen, als die eigentliche Projektdurchführung. Es deutet sich also an, daß Kundenorientierung in der Softwareentwicklung im Zeitverlauf unterschiedliche Ausprägungen haben wird.

In einer Orientierung auf den Kunden und seine Bedürfnisse muß sich die Entwicklungsorganisation schon frühzeitig konstruktiv mit diesen Konfliktpotentialen auseinandersetzen, die nicht ohne weiteres - und schon gar nicht über Patentrezepte - lösbar erscheinen.

Kundenorientierung bedeutet aus der Sicht der Entwicklungsorganisation jedoch nicht nur die Ausrichtung auf die Anwendungsorganisation. Wenn man sich anschaut, welche Arbeit durch Software unterstützt werden soll, so hat diese in der Regel mit den Kunden der Anwendungsorganisation zu tun. Das bedeutet, daß auch der Kunde des Kunden systematisch in die Überlegungen einbezogen werden muß.

3.3 Kundenorientierung und Geschäftsbeziehung

Wenn man sich die beiden an Softwareentwicklung beteiligten Organisationen anschaut, so kann man feststellen, daß beide z.T. völlig unterschiedliche Interessen haben. Extrem ausgedrückt, ist es das Interesse der Anwendungsorganisation, für einen möglichst geringen Preis in möglichst kurzer Zeit ein maßgeschneidertes Softwaresystem von der Entwicklungsorganisation zu bekommen. Dabei ist die Anwendungsorganisation nicht bereit, für zusätzliche Entwicklungskosten aufzukommen, und will vermeiden, in langfristige Abhängigkeit von der Entwicklungsorganisation zu geraten. Die Entwicklungsorganisation hat m.E. genau das entgegengesetzte Interesse. Beiden Seiten geht es darum, den jeweils anderen soweit wie möglich auszunutzen. Eine Zusammenarbeit muß unter diesen Umständen so strukturiert und reglementiert wie möglich ablaufen; Verträge dienen dabei der wechselseitigen Absicherung. Im Falle von Vertragsbrüchen wird im Extremfall gerichtlich gegeneinander vorgegangen. Dies läuft auf eine „Gewinner-Verlierer-Situation“ hinaus.

Kundenorientierung zielt innerhalb dieses Szenarios auf eine Veränderung der Geschäftsbeziehung zwischen den Beteiligten, die auf die Bildung von strategischen Allianzen zwischen Softwarehersteller und Kunde hinauslaufen. Die strikte Trennung der Systeme wird aufgehoben und es wird eine zeitweise oder sogar langfristige Verschmelzung der Systeme angestrebt:

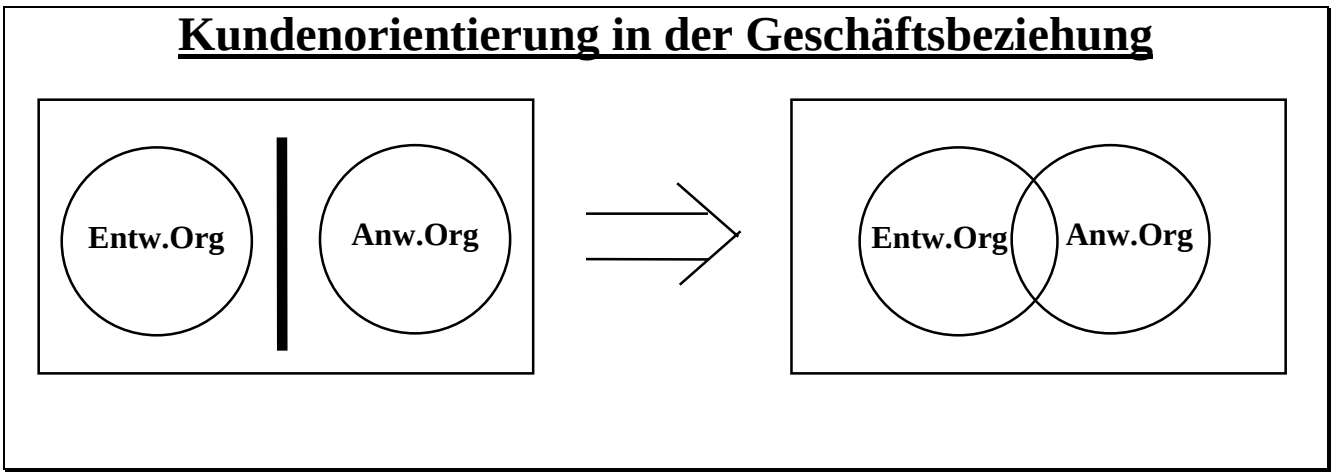

Abbildung 3-2

KLEINALTENKAMP et al. sprechen in diesem Zusammenhang von *Customer-Integration*, mit der die Verschmelzung der Wertschöpfungsprozesse von Kunden und Anbietern angestrebt wird, was zur Realisierung einer effektiven und gleichzeitig effizienten Kundenorientierung führen soll. Grundprinzip der Customer Integration ist es, „das Problem des Kunden zusammen mit dem Kunden zu lösen“ ([Kleinalt96]). Ziel ist es, im Rahmen einer gemeinsamen Zusammenarbeit eine Situation zu erreichen, in der beide gleichermaßen vom Erfolg profitieren, wo somit eine „Gewinner-Gewinner-Situation“ entsteht.

Für die Softwareentwicklung bedeutet dies, daß die Entwicklungsorganisation näher an die Anwendungsorganisation heranrückt und diese so weit wie möglich in den Entwicklungsprozeß integriert.

> Kundenorientierung „soll den Weg für eine intensive Kommunikation gleichberechtigter Partner aufzeigen, die letztlich das gleiche Ziel haben: Erfolgreiche Software für zufriedene Kunden zu entwickeln.“ ([HerzHier96])

Beide Organisationen verschmelzen in diesem gemeinsamen Ziel, behalten dabei jedoch ihre eigenen Funktionalitäten und Kernkompetenzen bei. Die Entwicklungsorganisation profitiert direkt z.B. in Form von Nachfolgeaufträgen, aber auch indirekt, indem sie z.B. von der Anwendungsorganisation an andere Unternehmen weiterempfohlen wird. Im Verlaufe einer langfristigen Zusammenarbeit bildet sich bei der Entwicklungsorganisation zusätzliches fachliches Know-How, wovon wiederum die Anwendungsorganisation profitiert, da Problemlösungen besser und schneller realisiert werden können.

3.4 Kundenzufriedenheit als Zielgröße

Das Erreichen und Verbessern von Kundenzufriedenheit spielt im Rahmen der Kundenorientierung eine entscheidende Rolle. Dies gilt sicherlich auch für den Bereich der Softwareentwicklung. Softwarehersteller (eigenständige Unternehmen, aber auch softwareproduzierende Organisationseinheiten) haben nur dann „Existenzberechtigung, wenn sie die Bedürfnisse ihrer Kunden befriedigen.…Keine Organisation mit unzufriedenen Kunden wird sich…langfristig am Markt halten bzw. ihre Existenz als Abteilung sichern“ ([HerzHier96]).

Kundenzufriedenheit ist dabei keine objektiv feststellbare Größe, sondern das subjektive Empfinden des Kunden. Kundenzufriedenheit drückt aus, „ob und wie gut die Erwartungen des Kunden mit der von ihm wahrgenommenen Leistung eines Softwareherstellers übereinstimmen“ ([HerzHier96]).

Ein Softwarehersteller ist also genau dann kundenorientiert, wenn seine Arbeit systematisch zur Kundenzufriedenheit führt. Für die Softwareentwicklung muß diese grundsätzliche Zielsetzung auf die beiden folgenden Bereiche bezogen werden:

- Auf das **Softwareprodukt** einerseits
- und auf den **Softwareentwicklungsprozeß** andererseits.

Beide hängen eng miteinander zusammen. Ohne Zweifel entscheidet letztlich die Qualität des Softwaresystems - das Produkt - darüber, ob der Kunde zufrieden ist oder nicht. Eine hohe Produktqualität ist jedoch nur durch eine hohe Prozeßqualität möglich (vgl. [Floyd93]). Kundenorientierung muß deshalb auf beide Bereiche bezogen werden. Dies kann in folgender Definition konkretisiert werden:

Definition:
Kundenorientierte Softwareentwicklung hat das Ziel, mittels einer kundenorientierten Prozeßgestaltung zu einer kundenorientierten Produktgestaltung zu gelangen, um eine möglichst hohe Kundenzufriedenheit sowohl mit dem Softwareprodukt als auch mit dem Softwareentwicklungsprozeß zu erreichen.

Im weiteren soll auf die Bedeutung von Kundenorientierung für die Gestaltung des Softwareproduktes und den Softwareentwicklungsprozeß eingegangen werden. Dabei sollen die Anforderungen an eine kundenorientierte Produktgestaltung und an eine kundenorientierte Prozeßgestaltung erarbeitet werden.

3.5 Anforderungen an eine kundenorientierte Produktgestaltung

Welche Bedeutung hat Kundenorientierung für die Produktgestaltung?

Zur Beantwortung dieser Frage werden zunächst grundsätzliche Qualitätsanforderungen formuliert. Dann wird der Zusammenhang zwischen Produktmerkmalen und Kundenzufriedenheit anhand des Kano-Modells diskutiert. Abschließend wird Kundenorientierung als konkretes Gestaltungsziel von Arbeitsplatzsystemen vorgestellt.

3.5.1 Qualitätsmerkmale

Damit durch das Softwareprodukt maximale Kundenzufriedenheit erreicht werden kann, muß dieses über eine hohe Qualität verfügen. Qualitätsmerkmale sind z.B. nach DIN 66234:

- Aufgabenangemessenheit - Der Benutzer wird bei der Ausführung seiner Aufgaben unterstützt, ohne durch die Eigenschaften des Systems unnötig eingeschränkt zu werden.
- Selbstbeschreibungsfähigkeit - Das System ist unmittelbar verständlich oder liefert dem Benutzer auf Verlangen Leistungsumfang und Einsatzzweck der jeweiligen Dialogbestandteile.
- Steuerbarkeit - Der Benutzer kann Geschwindigkeit des Ablaufes, Art und Umfang von Ein- und Ausgaben sowie Auswahl und Reihenfolge von Arbeitsmitteln beeinflussen.
- Erwartungskonformität - Das System entspricht den Erwartungen des Benutzers, die er aus seiner bisherigen Arbeit, in einer Schulung, im Umgang mit Handbüchern oder dem Dialogsystem erworben hat.
- Fehlerrobustheit - Das beabsichtigte Ergebnis kann trotz fehlerhafter Eingabe mit minimalem Mehraufwand erreicht werden. Fehler werden dem Benutzer zum Zwecke der Behebung verständlich gemacht.

Weitere Qualitätsanforderungen werden im Abschnitt 3.5.3 vorgestellt.

Die Forderung nach hoher Qualität stellt an sich nichts Neues dar, genausowenig wie Kundenorientierung etwas grundsätzlich Neues darstellt. Deshalb ist es auch nicht verwunderlich, daß viele der Anforderungen, die sich aus einer stärkeren Kundenorientierung ergeben, bereits lange bekannt sind. Im Zuge von Kundenorientierung bekommen diese Anforderungen (wie hier die nach Produktqualität) jedoch nochmals eine stärkere Bedeutung, da jetzt deutlich ist, wie eng sie mit Kundenzufriedenheit und damit Unternehmenserfolg zusammenhängen.

Festzuhalten ist, daß im Zuge von Kundenorientierung (was u.a. die Umsetzung eines kundenorientierten Qualitätsbegriffes einschließt) deutlich ist, wer die Qualität der Software beurteilt, nämlich der Kunde selbst. Hier spielen besonders die Anwender des Systems eine wichtige Rolle, da das System ihre Arbeit unterstützen soll. Die Qualität des Softwaresystems entscheidet sich letztlich erst im Einsatz bzw. im Gebrauch des Systems. Ein kundenorientierter Qualitätsbegriff ist daher eng mit dem von FLOYD eingeführten Begriff der „Gebrauchsqualität“ verbunden (vgl. [Floyd97]). Hier stellt sich die Frage, wie ein Softwaresystem mit hoher Gebrauchsqualität entwickelt werden kann. Dies ist Gegenstand des Abschnitts 3.6, wenn die Anforderungen an einen kundenorientierten Entwicklungsprozeß diskutiert werden. Hier kann jedoch festgehalten werden, daß eine kundenorientierte Produktgestaltung in jedem Fall ein Softwareprodukt von hoher Gebrauchsqualität anstreben muß.

3.5.2 Kundenzufriedenheit anhand des Kano-Modells

Welche Gründe gibt es für Kundenunzufriedenheit und wie kann Kundenzufriedenheit gesteigert werden?

Im Kano-Modell (vgl. [KanSerTak84]) werden drei Arten von Anforderungen (Basis-, Leistungs- und Begeisterungsanforderungen) unterschieden und der Einfluß beschrieben, den die Erfüllung bzw. Nichterfüllung dieser Anforderungen auf die Kundenzufriedenheit hat:

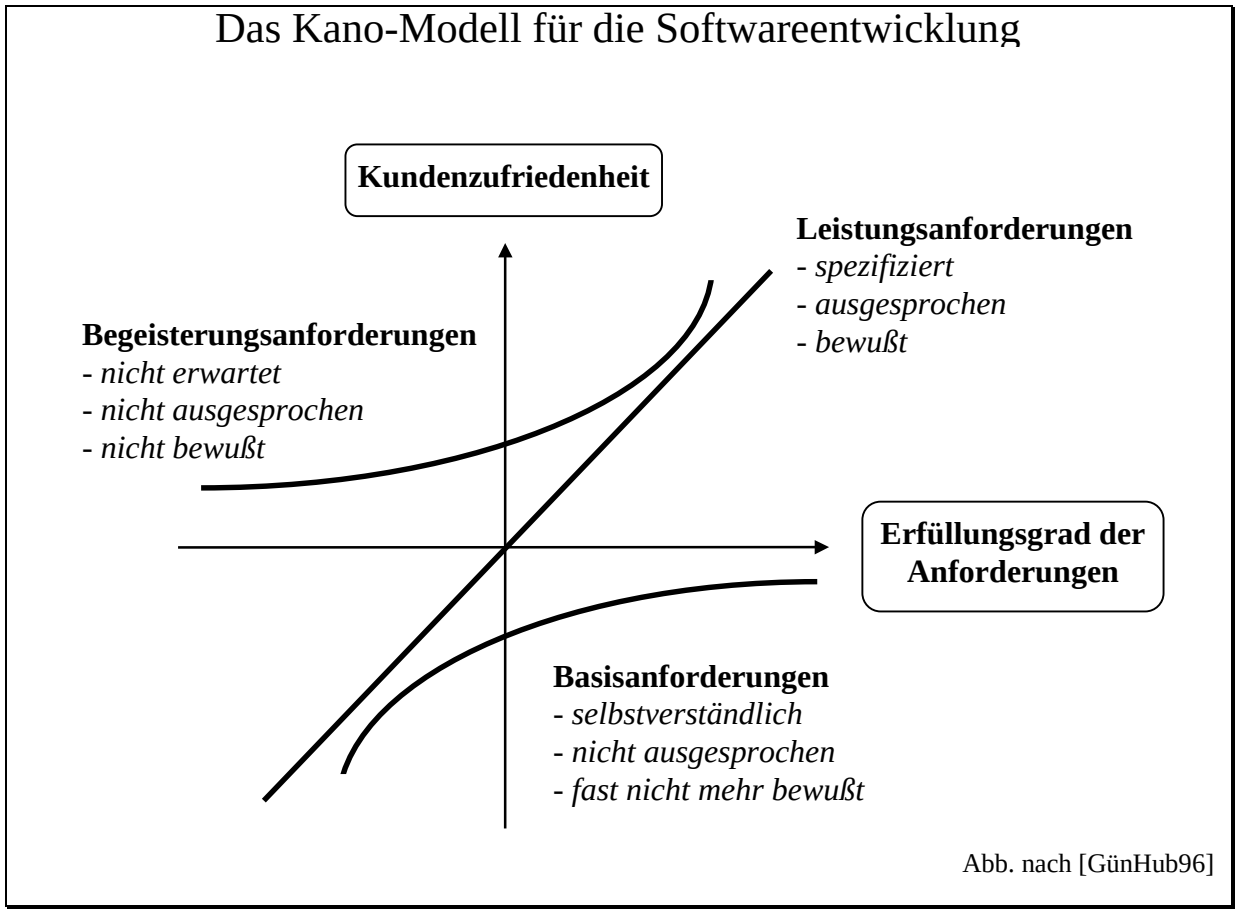

Abbildung 3-3

Basis- bzw. Grundanforderungen werden vom Kunden als selbstverständlich vorausgesetzt. Sie sind ihm allerdings fast nicht mehr bewußt und werden deshalb in der Regel auch nicht ausgesprochen. Die Erfüllung von Basisanforderungen bewirkt nicht zwangsläufig Kundenzufriedenheit, jedoch läßt die Nichterfüllung die Zufriedenheit rasch absinken.

Leistungs- bzw. Qualitätsanforderungen sind dem Kunden bewußt und werden in der Regel auch ausgesprochen. Je mehr diese Anforderungen erfüllt werden, desto höher steigt die Zufriedenheit und umgekehrt.

Begeisterungsanforderungen sind dem Kunden ebenfalls nicht bewußt und werden deshalb auch nicht ausgesprochen. Allerdings wird ihre Umsetzung nicht erwartet, weshalb eine Nichterfüllung auch keinen merkbaren Einfluß auf die Kundenzufriedenheit hat. Die Erfüllung von Begeisterungsanforderungen jedoch läßt die Zufriedenheit überproportional steigen, so daß u.U. sogar Mängel in der Umsetzung von Leistungsanforderungen ausgeglichen werden können. Begeisterungsanforderungen können hohen technischen Innovationsgrad besitzen, andererseits aber auch aus „Kleinigkeiten“ bestehen, die das tägliche Arbeiten enorm erleichtern.

Welche Bedeutung hat das Kano-Modell für Kundenorientierung in der Softwareentwicklung?

Zunächst einmal ist es wichtig zu erkennen, daß sowohl Basis- als auch Begeisterungsanforderungen dem Kunden nicht bewußt sind, d.h. auch nicht ohne weiteres genannt werden. Die Entwicklungsorganisation muß sich also mit der Frage auseinandersetzen, wie sie an diese Anforderungen „herankommt“. Die Ursachen dafür, daß Basis- und Begeisterungsanforderungen dem Kunden unbewußt sind, sind unterschiedlicher Natur:

- Basisanforderungen sind z.B. unbewußt, wenn sie stark die alltägliche Routinearbeit betreffen. Eine natürliche Betriebsblindheit verhindert, daß sie überhaupt als eigenständige Anforderungen erkannt werden. Um Basisanforderungen zu entdecken ist es notwendig, daß die Entwicklungsorganisation (und besonders die Entwickler selbst) ein genaues Verständnis der Arbeitsrealität des Kunden gewinnen. Spezielle Interviewtechniken müssen verwendet werden, die den Kunden in einen „Erzählzwang“ versetzen. Auch die von KRABBEL et al. erprobten „Wozu-Tabellen“ können bei der Entdeckung von Basisanforderungen hilfreich sein (vgl. [KrabWetzRat96b]). Um bei den Entwicklern ein Verständnis der Arbeitsrealität zu fördern, kann es sehr unterstützend sein, wenn diese für eine gewisse Zeit an der täglichen Arbeit beim Kunden aktiv teilnehmen. Da die mangelhafte Erfüllung von Basisanforderungen oft erst dann entdeckt wird, wenn der Kunde am Softwaresystem arbeitet, ist es wichtig, so früh wie möglich ausführbare Programmteile (z.B. in Form von Prototypen) zur Verfügung zu stellen.
- Begeisterungsanforderungen sind z.B. unbewußt, wenn die technischen Realisierungsmöglichkeiten dem Kunden unbekannt sind bzw. wenn der Kunde durch frühere Softwaresysteme an unhandliche Bedienungsabläufe gewöhnt ist. Um solche Begeisterungsanforderungen zu erkennen, ist es deshalb nicht nur nötig, daß die Entwickler die Arbeitsumstände des Kunden kennen, sondern auch, daß sie über die neuesten technischen Möglichkeiten unterrichtet sind. Auch das frühzeitige Bereitstellen von ausführbaren Programmteilen, die den Anwender mit alternativen Möglichkeiten der Systemunterstützung vertraut machen, kann hilfreich sein. Bereits durch den konkreten Umgang mit Prototypen werden beim Anwender Lernprozesse ausgelöst, die neue Gestaltungsideen erscheinen lassen und u.U. auch Begeisterungsanforderungen ‘ans Licht’ bringen können.

Das Kano-Modell bietet sich darüber hinaus auch dazu an, die unterschiedlichen Sichtweisen der Beteiligten zu thematisieren und den Softwarehersteller für etwaige Konfliktpotentiale zu sensibilisieren. Wenn man z.B. die Anforderung nach System-Stabilität betrachtet, so wird klar, daß dies aus Sicht des Kunden eine Basisanforderung darstellt, die als so selbstverständlich vorausgesetzt wird, daß ihre Erfüllung keine besondere Kundenzufriedenheit bewirken wird. Ist das System jedoch instabil, so sinkt die Zufriedenheit dramatisch ab. Betrachtet man nun die Sichtweise der Entwicklerseite, so stellt man fest, daß es sich hier genau umgekehrt verhält. Zwar ist Systemstabilität als Anforderung sehr wohl bekannt, jedoch (gerade in frühen Auslieferungsphasen) nur schwer vollständig umsetzbar. Die Erfüllung der Basisanforderung nach vollständiger Stabilität wird damit aus Sicht der Entwickler zu einer Begeisterungsanforderung. Kundenunzufriedenheit, die z.B. aus mangelnder Systemstabilität resultiert, wird bei den Entwicklern auf Unverständnis stoßen und das Bild des ewig unzufriedenen, nörgelnden Kunden bestätigen. In diesem Sinne kann das Kano-Modell also auch benutzt werden, um in der Kooperation zwischen Kundenseite und Entwicklerseite das gegenseitige Verständnis zu erhöhen.

Zusammenfassend kann man sagen, daß das Kano-Modell sowohl wichtige Impulse für eine kundenorientierte Produktgestaltung, als auch für eine kundenorientierte Prozeßgestaltung geben kann. Bezogen auf letzteres muß sich die Entwicklungsorganisation dann die Frage

stellen, welche Basis-, Leistungs- und Begeisterungsanforderungen der Kunde in bezug auf den Entwicklungsprozeß hat.

3.5.3 Arbeitsplatzsysteme zur Unterstützung kundenorientierter Arbeit

Die genannten Qualitätsmerkmale und das Kano-Modell dienen dem Softwarehersteller als Orientierung bei der Gestaltung des Softwareproduktes. Kundenorientierung bedeutet zunächst einmal, diese Qualitätsanforderungen ernster denn je zu nehmen und gezielte Bemühungen zur Steigerung von Kundenzufriedenheit zu unternehmen.

Dabei bleibt es jedoch nicht. Kundenorientierung bietet für den Softwarehersteller eine noch stärkere Orientierungshilfe, wenn man sich vor Augen führt, welche Arbeit Softwaresysteme zukünftig zunehmend unterstützen werden. Die in vielen Unternehmen durch das Streben nach Kundenorientierung ausgelösten Umstrukturierungen müssen von der Software in jedem Fall mitgetragen werden. Anders ausgedrückt: Die Mitarbeiter eines nach Kundenorientierung strebenden Unternehmens können nur dann kundenorientiert agieren, wenn sie an ihrem Arbeitsplatz eine entsprechende informationstechnische Unterstützung vorfinden (vgl. [Züllighoven98]). Bezieht man auch hier wieder den internen Kunden mit ein, so läßt sich folgendes Leitbild eines kundenorientierten Arbeitsplatzes formulieren:

> ***Kundenorientiertes Produktleitbild:***
> ***Arbeitsplatzsysteme zur Unterstützung kundenorientierter Arbeit sowohl gegenüber dem internen als auch gegenüber dem externen Kunden.***

Die Forderung, sich bei der Gestaltung des Softwaresystems an den Bedürfnissen des internen und externen Kunden zu orientieren, stellt meiner Meinung nach eine wesentliche Erweiterung der Aufmerksamkeitshaltung von Softwareentwicklern dar. Wo es bisher im wesentlichen darum ging, die Bedürfnisse und Anforderungen des Kunden zu ermitteln, wird die Sicht nun erweitert und auf die Kunden des Kunden ausgedehnt. In kundenorientierten Softwareprojekten könnte es durchaus dazu kommen, daß Entwickler sich direkt mit den Bedürfnissen der Kunden des Kunden auseinandersetzen, um daraus Rückschlüsse auf notwendige Systemeigenschaften zu ziehen. Damit wäre auch die Möglichkeit gegeben, Anforderungen an die Software zu ermitteln, die dem Kunden selber verborgen sind, die jedoch, indem sie die Bedürfnisse der Kunden des Kunden abdecken, durchaus den Stellenwert einer Begeisterungsanforderung haben könnten.

Welche Anforderungen werden an ein Softwaresystem zur Unterstützung kundenorientierter Arbeit gestellt?

Zunächst einmal ist es wichtig zu sehen, daß die Anforderungen, denen sich Unternehmen heute vom Markt her ausgesetzt sehen, einem raschen Wechsel unterliegen. Die einzige Konstante scheint der Wandel selbst zu sein. Daraus ergibt sich eine Notwendigkeit nach Flexibilität in Strukturen und Menschen. Als Folge des raschen Wandels nimmt bei vielen Arbeitsplätzen der Anteil unstrukturierter, sich ständig ändernder Aufgaben zu. Damit einher geht eine Zunahme an zu bewältigender Komplexität. Software kann diese Aufgaben nicht mehr vollständig abbilden, sondern muß sie mehr und mehr unterstützen (vgl. [BauBoeEck94]).

Es findet also eine Veränderung in der Sichtweise der Aufgaben von Softwaresystemen statt. GRYCZAN beschreibt in diesem Zusammenhang zwei grundsätzlich verschiedene Sichtweisen, die bei der Entwicklung von Anwendungssoftware eine Rolle spielen und die sich polar gegenüberstehen (vgl. [Gryczan95]):

- Die ablaufsteuernde Sichtweise, die eine Kontrolle und Automatisierung von menschlicher Arbeit durch Softwaresysteme anstrebt,
- und die unterstützende Sichtweise, die Softwaresysteme als Unterstützung von eigenverantwortlicher und qualifizierter menschlicher Arbeit sieht.

Die *ablaufsteuernde Sichtweise* könnte man auch als die traditionelle Sichtweise der Aufgaben von Softwaresystemen bezeichnen. Hier wird das Ziel verfolgt,

> „Operationen zu implementieren, die menschliches Arbeitshandeln regeln und kontrollieren. Die Kontrolle über den Ablauf der Arbeitsschritte des Menschen liegt bei der Maschine. Generell wird bei der Automatisierung das Ziel verfolgt, menschliches Arbeitshandeln durch Maschinen zu ersetzen oder bis auf notwendige Dateneingabe (Input) zu reduzieren. Dazu werden Pläne und Vorschriften verwendet, die im Idealfall durch Algorithmen auf Maschinen implementiert werden.“ ([Gryczan95])

Dagegen bedeutet die *unterstützende Sichtweise*,

> „daß Benutzer von Softwaresystemen als Experten ihres Arbeitsgebietes verstanden werden und daß in ihrem Arbeitshandeln Computer als Arbeitsmittel eingesetzt werden. Ein charakteristisches Merkmal dieser Sichtweise ist, daß die Initiative bei der Computerverwendung vom Benutzer ausgeht und daß die Kontrolle über den Ablauf der Arbeitsschritte beim Benutzer liegt.“ (ebd.)

Im Spannungsfeld zwischen ablaufsteuernder und unterstützender Sicht ordnet sich kundenorientierte Arbeit eindeutig in der Nähe letzterer ein. Die notwendige Flexibilität kann nicht durch starre, monolithische Systeme erreicht werden, die Arbeitsabläufe festschreiben. Ein Mitarbeiter, der an vom System vorgeschriebene Abläufe gebunden wird, ist u.U. gar nicht in der Lage, auf neue, unvorhergesehene Wünsche des Kunden reagieren zu können.

Auch die Forderung nach Mitarbeiterorientierung ist nur schwer im Rahmen der ablaufsteuernden Sicht umsetzbar. In der ablaufsteuernden Sicht wird der Mitarbeiter dequalifiziert und auf Routinetätigkeiten reduziert. Kundenorientierten Softwaresystemen muß demgegenüber ein emanzipiertes Anwenderbild zugrunde liegen. Der Anwender wird als Experte seines Arbeitsgebietes verstanden, den es zu unterstützen und nicht zu kontrollieren gilt.

Aus der Unterstützungssicht und dem damit verbundenen emanzipierten Anwenderbild resultieren weitere maßgebliche Anforderungen an ein Softwaresystem für kundenorientierte Arbeit:

- Entscheidungsspielraum - Der Anwender soll in größtmöglichem Umfang „innerhalb seiner Arbeitstätigkeit eigenständig über Ziele sowie Vorgehensweisen und Mittel zu ihrer Erreichung entscheiden“ können ([Gryczan95]).
- Kooperation und unmittelbare zwischenmenschliche Kommunikation - Das System muß die Kooperation der Anwender untereinander unterstützen und darf die Möglichkeiten, im Verlaufe der Aufgabenerledigung auf direkte Absprachen zurückzugreifen, nicht verhindern.
- Variabilität der Aufgaben - Das System muß den Anwender auch darin unterstützen können, Aufgaben mit variablen, nicht gleichförmigen und u.U. unvorhergesehenen Anforderungen zu erfüllen. Nur dadurch ist es möglich, auf Kundenwünsche flexibel reagieren zu können. Das System muß demnach auch für Aufgaben ausgelegt sein, die der Entwickler und auch der Kunde nicht vorhergesehen haben. Es ist zu vermuten, daß sich in diesem ‘Not-intended-use’ einige der Begeisterungsanforderungen der Kunden des Kunden verstecken.

Es ist deutlich, daß diese Anforderungen nicht oder nur schwer innerhalb traditioneller, ablaufsteuernder Softwaresysteme umgesetzt werden können. Als praktische Umsetzung bietet sich ein Softwaresystem an, das dem Benutzer je nach Arbeitsplatz eine bestimmte Auswahl von Software-Werkzeugen anbietet, die relativ lose miteinander gekoppelt sind, wo Arbeitsabläufe insofern nur relativ gering festgeschrieben sind und dem Anwender größtmöglicher Spielraum bei der Erledigung seiner Aufgaben eingeräumt wird. Näheres dazu findet sich bei GRYZAN und ZÜLLIGHOVEN (vgl. [Gryczan95], [Züllighoven98]).

Es sei hier nur am Rande erwähnt, daß die ablaufsteuernde Sichtweise, die vor allem auf traditionellen Hostsystemen zu finden ist, mit dem Einzug von modernen PC-Rechnern keinesfalls überwunden ist. Im Zuge von Modernisierung und Portierung von alten Programmen auf eine moderne Fensteroberfläche findet man die Ablaufsteuerung nun in Form von sog. „modalen Dialogen“ wieder, die in bezug auf die oben genannten Anforderungen an kundenorientierte Softwaresyteme m.E. keine nennenswerten Besserungen bringen.

Die Forderung nach Kundenorientierung impliziert jedoch noch weitere Anforderungen an ein Softwaresystem: Der konstante Wandel der äußeren Anforderungen macht es notwendig, die Software schneller als früher zu verändern und an geänderte äußere Ansprüche anzupassen. Kundenorientierte Softwaresysteme müssen demnach auf Veränderung hin angelegt sein. Änderbarkeit ist letztlich nur über eine entsprechende ‘innere’ Beschaffenheit des Softwaresystems zu erreichen. Hier kommen Technologien wie die Objektorientierung ins Spiel, die zumindest grundsätzlich in der Lage sind, eine verbesserte Änderbarkeit von Softwaresystemen zu gewährleisten. Eine nähere Diskussion wird im Rahmen dieser Arbeit nicht geführt. Ausführlich wird dies u.a. von KILBERTH et al. und von ZÜLLIGHOVEN behandelt (vgl. [KilGryZü93], [Züllighoven98]).

Auch die Anforderung an Systemstabilität und an eine hohe Systemperformanz bekommt vor dem Hintergrund von Kundenorientierung eine neue Dimension. Ersteres muß hier nicht näher erläutert werden. Eine hohe Systemperformanz muß u.a. deshalb gewährleistet sein, um die Kundenwünsche nicht nur mit vertretbarem Zeitaufwand (Basisanforderung), sondern sogar beeindruckend schnell (Begeisterungsanforderung) erfüllen zu können.

Neben diesen generellen Anforderungen an Softwaresysteme zur Unterstützung kundenorientierter Arbeit wird es je nach konkretem Arbeitsplatz ganz spezifische Anforderungen geben, die eine kundenorientierte Arbeit fordert. Für Arbeitsplätze mit direktem Kontakt zu externen Kunden (z.B. via Telefon oder im Schaltergeschäft) sind dies z.B.:

- Die reibungslose und umfassende Bereitstellung von Kundeninformationen, damit der Mitarbeiter nach außen kompetent und rasch reagieren kann. Hierzu gehören u.a. historische Kundeninformationen, damit Kunden individuell angesprochen werden können.
- Die Integration von Beschwerdemanagement im System. Die im Rahmen von Kundenorientierung geforderte Nutzung von Reklamationen als Impulsgeber für Verbesserungen an Produkten und Dienstleistungen muß vom Softwaresystem unterstützt werden (vgl. auch [GünHub96]).

Darüber hinaus ist auch darüber nachzudenken, ob man im System selbst Möglichkeiten verankert, mit denen die Anwender in Kontakt zu den Entwicklern treten können, um die geforderte enge Bindung zwischen Anwendungs- und Entwicklungsorganisation auch im Systembetrieb aufrechterhalten zu können. Ein interessanter Vorschlag kommt hierzu von LILIENTHAL, die ein Hilfesystem vorstellt, welches die Anwender auch dazu nutzen können, um Mißverständnisse und Fehler der Entwickler mitzuteilen sowie Vorschläge für System-verbesserungen zu machen (vgl. [Lilienthal95]).

Die hier gemachten Vorschläge sind sicherlich unvollständig und müßten im Rahmen von weiteren Arbeiten noch erweitert werden. Festzuhalten ist jedoch, daß das Konzept Kundenorientierung für die Produktgestaltung enorm hilfreich sein kann, indem es während der Systementwicklung als Orientierungshilfe dient und wertvolle Impulse für die Gestaltung des Informationssystems geben kann.

Im folgenden Abschnitt soll es um die Frage gehen, wie kundenorientierte Softwaresysteme entstehen können, d.h. wie die Forderung nach Kundenorientierung im Softwareentwicklungsprozeß umgesetzt werden kann.

3.6 Anforderungen an eine kundenorientierte Prozeßgestaltung

Welche Bedeutung hat Kundenorientierung für den Prozeß der Softwareentwicklung? Wie sieht eine kundenorientierte Prozeßgestaltung aus?

Aus den Ergebnissen des Kapitels über Kundenorientierung ergeben sich die folgenden Anforderungen an einen kundenorientierten Softwareentwicklungsprozeß:

- Erreichung und Verbesserung von Kundenzufriedenheit im und mit dem Entwicklungsprozeß.
- Umsetzung eines kundenorientierten Qualitätsbegriffes.
- Kunden-Integration in den Entwicklungsprozeß.
- Zusatzqualifizierung des Kunden durch den Entwicklungsprozeß.

Diese Anforderungen sollen im weiteren konkretisiert werden. Wie diese Anforderungen in der Praxis umgesetzt werden können, wird dann im darauf folgenden Abschnitt 3.7 erläutert.

3.6.1 Kundenzufriedenheit im Entwicklungsprozeß

Die Ausrichtung auf **Kundenzufriedenheit** im Entwicklungsprozeß hat zwei wichtige Konsequenzen:

- Die Einhaltung der von der Entwicklungsorganisation gemachten Zusagen.
- Die Messung und Verbesserung von Kundenzufriedenheit während des Entwicklungsprozesses.

Die *Einhaltung von Zusagen* richtet sich auf die Bereiche Kosten, Termine und Funktionalität. Um die Zusagen in allen drei Bereichen einzuhalten, ist es für die Entwicklungsorganisation notwendig, von Beginn an auf unrealistische Versprechungen zu verzichten. Außerdem muß die Planung im Prozeß selbst an die realen Gegebenheiten angepaßt werden. Dies bedeutet nicht, daß jegliche Planungssicherheit aufgegeben wird, sondern daß die Stufen der Entwicklung im Prozeß selber geplant werden und nicht komplett im Vorwege des Projektes.

Die kontinuierliche *Messung von Kundenzufriedenheit* während des Prozesses hat das Ziel, Probleme möglichst frühzeitig und damit rechtzeitig zu erkennen. Es reicht nicht, sich mit Kundenzufriedenheit erst bei Auslieferung des Softwareproduktes auseinanderzusetzen, weil es dann bereits zu spät sein kann und keine geeigneten Lösungsmaßnahmen mehr ergriffen werden können. Deshalb wird hier gefordert, Kundenzufriedenheit kontinuierlich zu messen und die Messung im Prozeß selbst zu verankern. Der etwas technisch klingende Begriff der ‘Messung’ könnte suggerieren, daß hier lediglich Fragebogenaktionen gemeint sind, deren Wert ohnehin zweifelhaft ist. Hier sind viel eher kontinuierliche Formen von Projektreviews gemeint, in denen u.a. auch die Kundenzufriedenheit thematisiert wird. Kundenzufriedenheitsmessung kann viele Formen annehmen. Wesentlich ist hier, daß der als notwendig erkannte Fokus auf Kundenzufriedenheit bereits im Prozeß umgesetzt wird, damit wahrgenommene Unzufriedenheit des Kunden schon frühzeitig als Möglichkeit genutzt werden kann, den Prozeß selbst anzupassen und zu verbessern.

3.6.2 Umsetzung eines kundenorientierten Qualitätsbegriffs

Die Einführung und Umsetzung eines kundenorientierten Qualitätsbegriffes bedeutet, daß Qualität (von Produkten, Prozessen, Dienstleistungen, etc.) aus der Sicht des Kunden beurteilt werden muß, d.h. daß letztlich der Kunde die Qualität beurteilt.

Im Bereich der Softwareentwicklung stößt man hier zunächst auf das Problem, daß es oft an Grundlagen einer frühzeitigen Qualitätsbeurteilung durch den Kunden mangelt, so daß erst bei Auslieferung eines Softwaresystems grundsätzliche Fehler und Unzulänglichkeiten erkannt

werden. Dies ist jedoch zu spät. Zur Umsetzung eines kundenorientierten Qualitätsbegriffes müssen deshalb schon frühzeitig im Entwicklungsprozeß Grundlagen (in Form von Zwischenprodukten und Dokumenten) für eine Qualitätsbeurteilung durch den Kunden bereitgestellt werden. Erst dadurch kann sichergestellt werden, daß die Qualität des ausgelieferten Softwaresystems den Erwartungen des Kunden gerecht wird. Als Ergebnis halten wir deshalb fest:

Die im Entwicklungsprozeß entstehenden Dokumente und Zwischenprodukte müssen frühzeitig und kontinuierlich vom Kunden auf Qualität geprüft werden, um eine letztendlich hohe Qualität des Softwaresystems zu gewährleisten.

Wenn eine Qualitätssicherung durch den Kunden gefordert wird, so müssen Voraussetzungen geschaffen werden, daß der Kunde die Qualität überhaupt beurteilen kann. Der Kunde muß schon in frühen Phasen der Entwicklung in der Lage sein, die entstehenden Dokumente und Zwischenprodukte bewerten zu können. Folgende Forderung muß deshalb erfüllt werden:

Die Umsetzung eines kundenorientierten Qualitätsbegriffes kann nur durch spezielle kundenorientierte Dokumente und Zwischenprodukte erfolgen, deren Qualität vom Kunden beurteilbar sind.

Welche Eigenschaften müssen solche kundenorientierten Dokumente und Zwischenprodukte aufweisen?

Es mag zwar banal klingen, jedoch gilt sicher das folgende: Damit Dokumente vom Kunden beurteilt werden können, müssen sie für ihn verständlich sein. Aus der Forderung nach Verständlichkeit ergeben sich die folgenden Anforderungen an kundenorientierte Dokumente und Zwischenprodukte:

- Sie müssen sich an der Sprache des Kunden orientieren.
- Sie müssen an der Arbeitsrealität des Kunden orientiert sein.
- Sie müssen die gegenwärtige Situation so beschreiben, daß für den Kunden deutlich wird, ob die Entwickler ein korrektes Verständnis von der Arbeitsrealität des Kunden haben.
- Sie müssen das Aussehen des zukünftigen Softwaresystems so beschreiben, daß für alle Beteiligten eine gemeinsam getragene Vision entstehen kann.

Kundenorientierte Dokumente müssen in der Sprache des Kunden verfaßt sein, d.h. sie müssen sich an der Begriffswelt des Kunden orientieren. Nur so kann der Kunde überprüfen, ob seine Arbeitsrealität wirklich von der Entwicklungsorganisation verstanden wird. Ein fundiertes Verständnis der Anwendungszusammenhänge durch die Entwicklungsorganisation ist notwendig, weil ohne ein solches Verständnis kein Softwaresystem entwickelt werden kann, welches diese Anwendungszusammenhänge unterstützt, geschweige denn verändert und verbessert. Eine Orientierung an der Sprache des Kunden bedeutet auch, daß auf eine techniklastige Sprache, die den Kunden tendenziell aus dem Entwicklungsprozeß ausgrenzt, verzichtet wird.

Weiterhin müssen kundenorientierte Dokumente und Produkte dem Kunden frühzeitig eine Vision des zukünftigen Systems vermitteln können. Dies geschieht primär deshalb, damit der Kunde prüfen kann, ob das Projekt ‘in die richtige Richtung geht’. Die frühzeitige Konfrontation der Erwartungen des Kunden mit überprüfbaren Realisierungsideen der Entwicklungsorganisation führt dazu, daß unterschiedliche Erwartungen und Interessen

rechtzeitig explizit benannt werden und nicht erst dann als unlösbare Konflikte auftauchen, wenn das Softwaresystem ausgeliefert wird.

Konkrete Arten von Dokumenten und Zwischenprodukten, die diese Anforderungen erfüllen, werden in Abschnitt 3.7 vorgestellt.

3.6.3 Kundenintegration im Entwicklungsprozeß

Kundenintegration als dritte Anforderung ergibt sich aus den in vorigen Kapiteln aufgeführten Forschungsergebnissen aus dem Bereich der Innovationsentwicklung (vgl. [Zollner94]) und der Customer-Integration (vgl. [KleFliJac96]). Diese legen nahe, daß Kundenorientierung gerade in einem hochinnovativen Bereich wie der Softwareentwicklung nur über eine starke Einbeziehung des Kunden in den Entwicklungsprozeß zu realisieren ist. Weiterhin wird Kundenintegration auch innerhalb der Informatik-Forschung (unter dem Begriff ‘partizipative Softwareentwicklung’) als vielversprechende Möglichkeit zur Überwindung der Probleme traditioneller Entwicklungen diskutiert [vgl. [BauBoeEck94], [Floyd93]).

Kundenorientierung im Softwareentwicklungsprozesses läuft deshalb auf eine verstärkte Integration des Kunden in den Entwicklungsprozeß hinaus. Hier wird die im Sinne einer „Customer Integration“ strategisch angestrebte Verschmelzung der Wertschöpfungsketten von Hersteller und Kunde praktisch umgesetzt. Das Leitbild eines kundenorientierten Entwicklungsprozesses kann man somit folgendermaßen formulieren:

Kundenorientiertes Prozessleitbild:
Softwareentwicklung gemeinsam mit dem Kunden

Welche Konsequenzen ergeben sich aus der Anforderung nach Kundenintegration?

Zunächst ergibt sich daraus die Struktur von Entwicklungsprojekten. Diese müssen innerhalb von interdisziplinären Teams durchgeführt werden. Kundenintegration bedeutet, daß Vertreter des Kunden Teil des Entwicklungsteams werden.

Die erfolgreiche Bildung von interdisziplinären Teams, die Vertreter des Kunden als gleichwertige Partner in den Entwicklungsprozeß integrieren, macht die folgenden Bedingungen nötig:

- Das Entwicklungsprojekt orientiert sich an der Sprache des Kunden.
- Das Projekt orientiert sich an der Arbeitsrealität des Kunden.
- Im Projekt werden kooperativ Visionen des zukünftigen Systems entworfen, die für alle Beteiligten verständlich und von allen Beteiligten getragen werden.
- Die Etablierung und Unterstützung eines direkten Kontaktes zwischen Entwicklern und Anwendern steht im Vordergrund der Bemühungen.
- Teamorientierung mit gezielten Teamentwicklungsmaßnahmen sind expliziter Bestandteil der Vorgehensweise.
- Die Vertreter des Kunden müssen für die Arbeit im Entwicklungsteam freigestellt werden.

Die ersten drei Forderungen (Orientierung an Sprache, Orientierung an Arbeitsrealität sowie kooperative Entwicklung von Visionen) sind bereits als Anforderungen an kundenorientierte Dokumente und Zwischenprodukte diskutiert worden. Hier bekommen sie nochmals eine zusätzliche Bedeutung, indem sie die Grundlage für eine Integration des Kunden in den Entwicklungsprozeß bilden.

Die Forderung nach einem **direkten Kontakt zwischen Entwicklern und Anwendern** leitet sich nicht nur aus der Forderung nach interdisziplinären Teams ab. Auch der im Rahmen der

Innovationsforschung geforderte engere Kontakt des F&E-Bereiches mit den Kunden legt Ähnliches nahe.

Für die Umsetzung eines direkten Kontaktes zwischen Entwicklern und Anwendern bedarf es einer konstruktiven Basis in Form von wechselseitig verständlichen Dokumenttypen. Die oben eingeführten kundenorientierten Dokumente können als solche Basis dienen und haben deshalb nicht nur die Aufgabe, die Umsetzung eines kundenorientierten Qualitätsbegriffes zu unterstützen. Als Ergebnis halten wir fest:

> ***Kundenorientierte Dokumente dienen nicht nur zur Umsetzung eines kundenorientierten Qualitätsbegriffes, sondern auch als konstruktive Basis für Kundenintegration und direkte Entwickler-Anwender-Kommunikation.***

Daß interdisziplinäre Teams erfolgreiche Arbeit leisten, ist keine Selbstverständlichkeit. Die Voraussetzungen für kooperative Zusammenarbeit zwischen sehr unterschiedlichen Menschen (z.B. zwischen Entwicklern und Anwendern) können durchaus denkbar schlecht sein. Deshalb stellt sich **Teamorientierung mit gezielten Teamentwicklungsmaßnahmen** als Notwendigkeit dar. Ziel dieser Maßnahmen ist, aus einer Gruppe von Menschen, die für eine gewisse Zeit eng zusammenarbeiten sollen, ein produktiv arbeitendes Team zu formen und dieses in seiner Entwicklung zu fördern. In diesem Zusammenhang entwickeln DEMARCO und LISTER das Bild des „eingeschworenen Teams“. Teamorientierung liegt der einfache Gedanke zugrunde, daß das Ganze mehr sein kann, als die Summe seiner Teile. Allerdings nur, wenn die Randbedingungen stimmen. DEMARCO und LISTER nennen einige Punkte, die die Bildung von funktionierenden Teams verhindern, die also - wie die Autoren es nennen - für „Teammord“ verantwortlich sind (vgl. im folgenden [DeMarcoList91]):

- **Defensives Management** - Das Management vertraut dem eigenen Projektteam nicht. Es herrscht keine fehlertolerante bzw. experimentierfreudige, sondern eine ängstliche und fehlervermeidende Stimmung vor.
  „Wenn Ihre Mannschaft jedoch davon ausgeht, daß sie nie irgendwelche Fehler begehen darf, dann dringt die Botschaft, daß Sie dem Team nicht trauen, laut und deutlich an diese durch. Diese Botschaft ist besser als jede andere Botschaft geeignet, die Teambildung zu verhindern.“ ([DeMarcoList91])
- **Sinnlose Termine** oder **Scheintermine** - Termine, die aus Akquisegründen gesetzt werden und von denen jeder weiß, daß sie nicht zu halten sind, untergraben die Motivation und verhindern Teambildung:
  „Wenn der Manager wieder einmal seine gewichtige Stimme erhebt und verkündet: ‘Wir müssen bis … unbedingt fertig sein’, dann werfen sich die Teammitglieder schon oft gegenseitig Blicke zu. Das hatten sie doch schon öfter. Sie kennen die ganze Routine…Typischerweise laufen Scheinterminverhandlungen so: Der Manager kündigt zunächst an, daß die Arbeit bis dann und dann unbedingt fertig sein muß. Das genannte Datum kann unmöglich gehalten werden, und jeder Beteiligte weiß das. Die Arbeit dauert garantiert länger…Die Aufgabe wurde in einer Weise gestellt, daß ein Erfolg überhaupt nicht möglich ist. Die Botschaft an die Systementwickler ist klar:…Der Chef glaubt, daß sie ohne Druck absolut nicht arbeiten würden. In solch einem Projekt sollten Sie nie ein eingeschworenes Team erwarten.“ (ebd.)

- **Zersplitterung der Zeit der Mitarbeiter** - Hier liegt die Forderung zugrunde, daß Mitarbeiter nur in einem Team, d.h. also nur an einem Projekt zur Zeit arbeiten. Demgegenüber ist jedoch gängige Praxis, daß gerade die erfahrensten Mitarbeiter an mehreren Projekten gleichzeitig arbeiten.
  „Die Aufteilung der Zeit eines Mitarbeiters auf mehrere Projekte ist schlecht für die Teambildung, aber auch schlecht für die Effizienz… Keiner kann gleichzeitig Mitglied von vier eingeschworenen Teams sein. Die engen Beziehungen in einem eingeschworenen Team sind exklusiv. Bei zu großer Verteilung der Zeit kann sich kein eingeschworenes Team bilden…Der fest erklärte Wille, daß ein Mitarbeiter nur an einem Projekt arbeiten soll, kann im Großen gesehen schon zu einer drastischen Verringerung der Zeitaufteilung führen und gibt den Teams eine echte Chance zur guten Teambildung.“ (ebd.)
- **Qualitätsreduktion der Produkte** - Hiermit sind die Versuche gemeint, Abstriche in Qualität und Funktionalität des Produktes zu machen, um Termine halten zu können. DeMarco und Lister weisen auf negative Konsequenzen in bezug auf Motivation und Teambildung hin:
  „Die Maßnahmen, die wir üblicherweise ergreifen, um ein Produkt schneller fertigzustellen, führen zu verminderter Qualität. Oft stimmt der Endkunde diesem Kompromiß zu (weniger Qualität für frühere, billigere Verfügbarkeit). Aber solche Zugeständnisse können für die Entwickler ziemlich schmerzlich sein. Ihr Selbstwertgefühl und ihre Freude an der Arbeit werden durch die Notwendigkeit, ein Produkt von deutlich schlechterer Qualität zu erstellen als sie es könnten, unterminiert. Die Identifikation, die eine Gruppe mit dem Team und mit ihrer Arbeit aufbauen konnte, fällt als erstes der Qualitätsreduktion zum Opfer.“ (ebd.)
- **Physikalische Trennung** - Eine räumliche Trennung von Teammitgliedern verhindert Teambildung:
  „Die räumliche Trennung von Personen, die zusammenarbeiten sollen, ist ohnehin nicht sehr sinnvoll. Die Sitznachbarn sind die Ursache für Lärm und Störungen. Bei Kollegen im gleichen Team ist die Wahrscheinlichkeit höher, daß alle gleichzeitig Ruhe zum Denken und Arbeiten haben wollen, wodurch weniger Störungen entstehen. Wenn man sie gemeinsam unterbringt, dann erhöht man die Chancen zur zwanglosen Unterhaltung, die für die Teambildung so dringend notwendig ist.“ (ebd.)

Im Zusammenhang mit Kundenorientierung muß Teamentwicklung besonders auf eine Integration der Kundenvertreter in das Entwicklungsteam gerichtet sein. Dazu ist es notwendig, die mehr oder minder stark vorhandene „Kulturbarriere“ zwischen Entwicklern und Anwendern abzubauen. In beiden Gruppen gibt es (aus unterschiedlichen Gründen freilich) Vorbehalte gegen eine direkte Zusammenarbeit.

Zum Thema Teamentwicklung gibt es innerhalb der Management-Literatur eine große Zahl an Veröffentlichungen. Um den Fokus auf Kundenorientierung nicht zu verlieren, wird hier darauf verzichtet, konkrete Teamentwicklungsmaßnahmen zu beschreiben. Näheres dazu findet sich z.B. bei HANSEL und LOMNITZ (vgl. [HansLomn93]) und bei BOY, DUDECK und KUSCHEL (vgl. [BoyDudKus94]). An dieser Stelle ist nur noch zu fordern, daß Teamentwicklungsmaßnahmen kontinuierlich im Projektverlauf stattfinden müssen, d.h. daß sie integraler Bestandteil der Vorgehensweise werden.

Die **Freistellung der Kundenvertreter** für die Mitarbeit im Projektteam ist ebenfalls von einiger Bedeutung. Erfolgt diese nicht oder nicht im notwendigen Ausmaß, so wird die Motivation der Kundenvertreter zur Mitarbeit eher gering ausfallen. Im Rahmen des Fallbeispiels wird noch näher auf diesen Punkt eingegangen (vgl. Abschnitt 4.4.7).

3.6.4 Zusatzqualifikation des Kunden durch den Entwicklungsprozeß

Eine umfassende Integration des Kunden in den Entwicklungsprozeß ist meiner Meinung nach für die Entwicklung von Softwaresystemen, die menschliche Arbeit unterstützen sollen, unerläßlich. Nur so kann das enorme Fachwissen des Kunden genutzt werden, welches die Entwickler in der Regel nicht zur Verfügung haben. Ein weiterer wichtiger Grund für Kundenintegration liegt in der Möglichgkeit, den Kunden über seine Mitwirkung am Entwicklungsprozeß für zusätzliche Aufgaben zu qualifizieren (Stichwort *„Empowerment“*). Hier können auf Kundenseite z.B. neue Tätigkeiten als ‘Interne Systemtrainer’, ‘SuperUser’ etc. entstehen.

Diejenigen Mitarbeiter, die direkt an einer Systementwicklung teilnehmen, erweitern ihren Kompetenzbereich enorm. Die Kenntnis der Technologie, die durch Mitwirkung am Entwicklungsprozeß entsteht, ist dabei nur ein Aspekt. Dazu kommt beinahe zwangsläufig ein Lernen von teamorientierter Projektarbeit und ein signifikant besseres Verständnis der eigene Arbeit, d.h. über die Probleme und Schwachstellen einerseits und die Veränderungs- und Verbesserungsmöglichkeiten andererseits. Diese Mitarbeiter werden für ihr Unternehmen in Zukunft eine wichtigere Rolle übernehmen können. Insofern entsteht für die Anwendungsorganisation durch eine intensive Beteiligung am Entwicklungsprozeß der erfreuliche Zusatznutzen, die eigenen Mitarbeiter im Sinne einer angestrebten Mitarbeiter-orientierung höher zu qualifizieren, was wiederum positive Auswirkungen auf die eigene Kundenorientierung haben wird.

Aus Sicht des Softwareherstellers wird durch diese Höherqualifizierung auch eine höhere Akzeptanz des Softwaresystems in der Anwendungsorganisation angestrebt. Ziel ist es nicht nur, zufriedene Kunden zu haben, sondern darüber hinaus aus zufriedenen Kunden Advokaten des Systems und damit der Entwicklungsorganisation selbst zu machen.

Diese Zusatzqualifizierung des Kunden im Entwicklungsprozeß ist aus der Sicht des Softwareherstellers eine Möglichkeit, die im Rahmen von Kundenorientierung geforderten ‘zusätzlichen Serviceleistungen’ (vgl. Kapitel 2) umzusetzen.

3.7 Umsetzung einer kundenorientierten Prozeßgestaltung

Wie sollen die obengenannten Anforderungen an einen kundenorientierten Entwicklungsprozeß in der Praxis umgesetzt werden? Welche Vorgehensmodelle bieten sich hierfür an?

Wenn man sich die wissenschaftliche Diskussion innerhalb der Informatik anschaut, kann man im wesentlichen zwei unterschiedliche Arten von Vorgehensmodellen unterscheiden:

- Traditionelle, phasenorientierte Vorgehensmodelle
- und evolutionäre, partizipative Vorgehensmodelle.

Im Rahmen dieser Arbeit wird argumentiert, daß eine Prozeßgestaltung im Sinne des traditionellen Vorgehensmodell grundlegende Anforderungen an eine kundenorientierte Prozeßgestaltung verletzt und damit als unbrauchbar abgelehnt wird. Weiterhin wird gezeigt, warum evolutionäre Vorgehensweisen als wesentlich verträglicher mit den Zielen von Kundenorientierung einzuschätzen sind und deshalb als Grundlage für eine kundenorientierte Prozeßgestaltung dienen können.

3.7.1 Kundenorientierung und traditionelle Softwareentwicklung

Traditionelle Softwareentwicklung ist eng verknüpft mit dem sog. „Wasserfall-Modell“, das bereits 1970 beschrieben und später besonders durch die Arbeiten von BOEHM bekannt wurde (vgl. [Boehm76], [Boehm81]). Das Wasserfall-Modell empfiehlt die Durchführung von Softwareentwicklungsprojekten in einer Folge von sequentiell hintereinander und aufeinander aufbauenden Schritten bzw. Phasen. Im einzelnen werden im Wasserfall-Modell folgende Phasen identifiziert (vgl. [Meyer95]):

- **Machbarkeitsstudie** („feasibility study“) - mit dem Ziel herauszufinden, ob das geplante Softwareprojekt durchführbar bzw. realisierbar ist oder nicht. Nachfolgende Schritte werden nur unternommen, wenn diese Phase ein positives Resultat erbringt.
- **Anforderungsermittlung** („requirements analysis“) - Ermittlung der Leistungen, die die entstehende Software erbringen muß. Zielgruppe: Kunde.
- **Spezifikation** - Umsetzung der in der vorigen Phase ermittelten Anforderungen in präzisere und formalere Systemanforderungen. Zielgruppe: Entwickler.
- **Globaler Systementwurf** („global design“) - Definition der Systemarchitektur mit ihrer Aufteilung in Module und einen Überblick über die benutzten Datenstrukturen.
- **Detaillierter Entwurf** („detailled design“) - Exakte Beschreibung jedes Moduls und jeder Datenstruktur.
- **Implementation** - Umsetzung des detaillierten Entwurfes in die gewählte Programmiersprache und damit Realisierung des Softwaresystems.
- **Validation und Verifikation** („V&V“) - Prüfung, ob das entstandene System die Anforderungen erfüllt (Validation) und ob es intern fehlerfrei und konsistent entwickelt wurde (Verifikation).
- **Auslieferung** („distribution“) - Installation und Inbetriebnahme der Software beim Kunden.
- **Wartung** („maintenance“) - Fehlerbehebung und Programmverbesserungen nach Inbetriebnahme der Software.

Es gibt eine Reihe von Vorgehensmodellen, die am Wasserfallmodell angelehnt sind, und obgleich einzelne der obengenannten Phasen z.T. wegfallen oder unterschiedlich benannt werden, besteht doch Konsens, daß eine Phase erst komplett abgeschlossen sein muß (z.B. mit der Abnahme des zur Phase gehörenden Meilensteindokumentes durch den Auftraggeber),

bevor mit der nächsten Phase begonnen werden kann. Außerdem lassen die Modelle Rückgriffe nur auf die jeweils vorige Stufe und das auch nur in begrenztem Maße zu.

Der praktische Einsatz des Wasserfall-Modells resultiert in einer phasenorientierten Vorgehensweise, wobei der Abschluß einer Phase ein „Meilenstein“ im Projektverlauf ist und damit zum zentralen Bestandteil der Projektkontrolle wird.

Im folgenden soll gezeigt werden, warum man traditionelle Softwareentwicklung durchaus als kurzfristig kundenorientiert bezeichnen kann, langfristig jedoch schwerwiegende negative Auswirkungen auf die Kundenorientierung feststellen muß.

3.7.1.1 Die kurzfristigen Vorteile einer traditionellen Softwareentwicklung

Bevor die Kritik am Wasserfall-Modell hier referiert werden soll, kann es vielleicht hilfreich sein, den Kontext zu beleuchten, in dem das Wasserfall-Modell entstanden ist, um einerseits das Modell in seiner Berechtigung zu würdigen und andererseits die Frage zu untersuchen, warum sich dieses Modell bis heute als vorrangiges Modell von Softwareentwicklungsprojekten gehalten hat.

Gegen Ende der 60er Jahre kam es mit dem Einsatz von Rechnern der dritten Generation und neuen komplexen Betriebssystemen und Compilern im Bereich der Softwarebranche zu massiven Problemen. Symptome dieser sog. *„Softwarekrise“* waren drastische Kosten- und Terminüberschreitungen bis hin zum kompletten Scheitern ganzer Projekte. Die erstellten Softwaresysteme waren oft fehlerhaft, erfüllten die Anforderungen nur mangelhaft und waren umständlich zu bedienen. Der Prozeß der Softwareentwicklung erschien unkontrollierbar. BROOKS z.B. benutzte als Metapher für den Softwareentwicklungsprozeß das Bild eines Werwolfes und stellte fest, daß es keinen „silver bullet“ geben würde, um diesen Werwolf zu erlegen (vgl. [Brooks87]).

Ausgelöst wurde die Softwarekrise durch einen

> „Mangel an Methoden und Ausbildung auf Seiten der Softwareentwickler, zunehmende Komplexität der zu verwendenden Hilfsmittel (Sprachen, Betriebssysteme, Rechner) und der zu bearbeitenden Probleme sowie fehlende Einsichten in Probleme der Arbeitsorganisation und Kommunikation.“ ([Floyd93])

In vielen Entwicklungsprojekten herrschte laut MEYER der *„Code it now, fix it later“*-Ansatz vor, wo Entwickler unmittelbar mit der Implementierung begannen, ohne sich viel um Analyse bzw. Entwurf zu kümmern. Mit der Einführung des Wasserfall-Modells war demnach die deutliche Botschaft verknüpft, daß Softwareentwicklung keine Spielerei ist, sondern in professioneller und strukturierter Weise als Ingenieurstätigkeit ernst genommen werden muß (vgl. [Meyer95]).

Das Wasserfall-Modell stellte den Versuch dar, ein Modell der Softwareentwicklung explizit zu machen, um den Prozeß besser verstehen zu können und damit kontrollierbarer zu gestalten. BOEHM selbst identifizierte zwei grundsätzliche Argumente, die für das Wasserfall-Modell sprechen (vgl. [Boehm81], [Meyer95]):

- Die Aktivitäten, die im Wasserfall-Modell identifiziert werden, sind für die Softwareentwicklung nötig.
- Das Modell benennt die einzig sinnvolle Reihenfolge, in der diese Aktivitäten durchgeführt werden sollen.

MEYER stimmt damit überein und würdigt am Wasserfall-Modell darüber hinaus die folgenden Punkte (vgl. [Meyer95]):

- Benennung von Aktivitäten, die der Implementierung vorgelagert sein müssen, z.B. Analyse, Entwurf, etc.

- Die damit verbundene Empfehlung, die Aufmerksamkeit nicht zu früh auf Implementierungsdetails zu richten, sondern mehr Aufwand in Analyse und Entwurf zu investieren.
- Herausstellung der Bedeutung von Abstraktionen.
- Die Nichterwähnung einer separaten Dokumentationsphase und damit die implizite Forderung, daß Dokumentation phasenübergreifend entstehen muß.
- Die Einbeziehung einer separaten Validations- und Verifikationsphase um sicherzustellen, daß die Software den gestellten Anforderungen genügt.

Im Zusammenhang mit der Frage nach Kundenorientierung in der Softwareentwicklung liegt das entscheidende Argument für das Wasserfall-Modell jedoch in seiner **Klarheit** und **Einfachheit**: Das Modell ist „einfach strukturiert und für den Auftraggeber und das Projektteam unmittelbar transparent“ ([BauBoeEck94]). Dies ist vermutlich der entscheidende Grund, warum das Wasserfall-Modell sich relativ schnell als eine mögliche Lösung aus der Softwarekrise anbot und im folgenden dann über Jahre hinweg (und im Prinzip bis heute) in der einen oder anderen Form angewandt wird.

Eine solche Vorgehensweise erscheint sowohl für den Softwarehersteller als auch für den Kunden durchaus plausibel und erfolgversprechend und kann als solches kundenorientiert genannt werden. Mit Blick auf die nachfolgende Kritik an der traditionellen Vorgehensweise wird jedoch schon hier der Zusatz „*kurzfristig* kundenorientiert“ benutzt. Als Zwischenergebnis halten wird deshalb fest:

Ein Vorgehen im Sinne der traditionellen, phasenorientierten Softwareentwicklung erscheint für den Kunden sofort plausibel und erfolgversprechend.
Deshalb kann ein solches Vorgehen kurzfristig kundenorientiert genannt werden.

Das traditionelle Vorgehensmodell konnte in Teilbereichen der Softwareentwicklung (Programmierung, Spezifikation) wirksame Lösungen anbieten, jedoch

> „wurde die Softwarekrise im eigentlichen Sinne noch nicht bewältigt. Die ungelösten Probleme haben sich vielmehr vom technischen Kern der Softwareentwicklung in das Vorfeld der Anforderungsermittlung, in die Gestaltung der Mensch-Rechner-Interaktion und in die Einbettung der Software in menschliches Arbeitshandeln verschoben.“ ([Floyd93])

Diese Schwächen der traditionellen Softwareentwicklung sollen im weiteren erläutert werden.

3.7.1.2 Die langfristigen Nachteile einer traditionellen Softwareentwicklung

Die traditionelle Softwareentwicklung ist von vielen Seiten überzeugend kritisiert worden (vgl. u.a. [BauBoeEck94], [BuKaKuZü92], [Floyd93], [Pasch92], [PomBla93], [Spilln94], [Spitta89]). Die z.T. grundsätzlichen Kritikpunkte sind:

- Projekte folgen in der Realität selten einer sequentiellen Abfolge von Phasen. Iterationen bzw. Rückgriffe auf frühere Phasen sind die Regel und nicht die Ausnahme.
- Eine klare Trennung zwischen den Phasen ist in der Praxis nicht möglich. Die zu den Phasen gehörenden Tätigkeiten sind zu sehr ineinander verschränkt, als daß man sie trennen könnte. Im Rahmen objektorientierter Vorgehensweisen hat die Verknüpfung der Phasen sogar noch zugenommen. Eine versuchte Trennung wirkt geradezu schädlich. Besonders ist hier der „semantische Bruch“ zwischen Analyse, Entwurf und Implementierung zu nennen.

- Die Qualität des Produktes kann erst bei endgültiger Auslieferung vom Auftraggeber beurteilt werden. Insbesondere den Anwendern wird keine Möglichkeit gegeben, sich schrittweise an die neue Software zu gewöhnen. Das Wasserfall-Modell ist demnach zur Qualitätssicherung völlig ungeeignet. MEYER fragt zurecht: „What better recipe could one use to ensure *non*-quality?“ ([Meyer95])
- Die zum Ende einer Phase erzeugten Dokumente sind von ihrer Qualität her nicht beurteilbar und liefern somit auch keinen Überblick über den Stand des Projektes. Als zentrales Element einer Projektsteuerung sind sie daher nicht zu gebrauchen. Die Gleichbehandlung der erzeugten Dokumente hinsichtlich ihrer Bedeutung führt darüberhinaus „in der Praxis regelmäßig dazu, daß die hochgradig redundante Dokumentation überhaupt nicht gewartet wird“ ([Spitta89]).
- Das Modell erzwingt eine vollständige Erhebung der Anforderungen zu Beginn des Projektes. Daß gerade zu Anfang sowohl auf der Seite des Kunden als auch beim Softwarehersteller die größten Unsicherheiten vorliegen, wird vom Modell ignoriert. Dies kann zwei unterschiedliche Folgen haben:
  - Die Problemsicht der Entwickler verengt bzw. vereinfacht sich, sodaß man einem „illusionären Qualitätsverständnis“ ([BauBoeEck94]) erliegt, welches spätestens bei Auslieferung vom Kunden aufgedeckt wird.
  - Oder die Phase der Anforderungsermittlung wird extrem ausgedehnt, mit der gutgemeinten Intention, möglichst keine Anforderung zu vergessen. Wenn der Zeitdruck dann zunimmt, wird die Phase meist willkürlich abgebrochen, um im Projekt ‘weiterzukommen’.
- Es wird unterstellt, daß der Kunde seine Anforderungen an das System deutlich formulieren kann. Dies ist aufgrund einer natürlichen Betriebsblindheit jedoch nicht der Fall. Insbesondere die Basisanforderungen (vgl. Abschnitt 3.5.2), die das Fundament der täglichen Arbeit ausmachen, sind aus dem Bereich des Bewußten verschwunden und werden als selbstverständlich vorausgesetzt und nicht mehr artikuliert.
- Weiterhin wird unterstellt, daß die Anforderungen im Projektverlauf unveränderlich sind. Damit wird verhindert, daß „der Anwender die formulierten Anforderungen im Projektverlauf weiter konkretisiert und an die tatsächlich bestehenden Anforderungen annähert“ ([BauBoeEck94]). Weiterhin wird damit ignoriert, daß es durch die Einführung neuer Softwaresysteme immer zu z.T. tiefgreifenden Veränderungen und Neugestaltungen des Anwendungsbereiches selber kommt, daß somit die Entstehung neuer Anforderungen dem Softwareentwicklungsprozeß immanent ist. Daß Softwareentwicklung immer eine mehr oder minder stark ausgeprägte Komponente von Organisationsentwicklung hat, wird im Wasserfall-Modell nicht berücksichtigt.
- Die phasenorientierte Vorgehensweise verlagert den evolutionären Charakter von Softwareentwicklungsprojekten in die sog. Wartungsphase. Hier nimmt die Vorgehensweise dann zwangsläufig evolutionären Charakter an, dann allerdings meist unter nicht strukturierten bzw. chaotischen Bedingungen (vgl. [Fittkau94] und Fallbeispiel in Kapitel 4).

Eine phasenorientierte Vorgehensweise erzeugt die Illusion von Kontrolle, bewirkt in Wirklichkeit jedoch eine „Intransparenz und Bürokratisierung des Prozesses, die sich motivations- und kreativitätshemmend für das Projektteam auswirken“ ([BauBoeEck94]).

Welche Auswirkungen hat eine phasenorientierte Vorgehensweise auf die Kundenzufriedenheit?

In meiner eigenen Erfahrung (vgl. [Fittkau94], aber auch das Fallbeispiel in Kapitel 4) konnte ich in traditionellen Projekten den folgenden Verlauf der Kundenzufriedenheit beobachten:

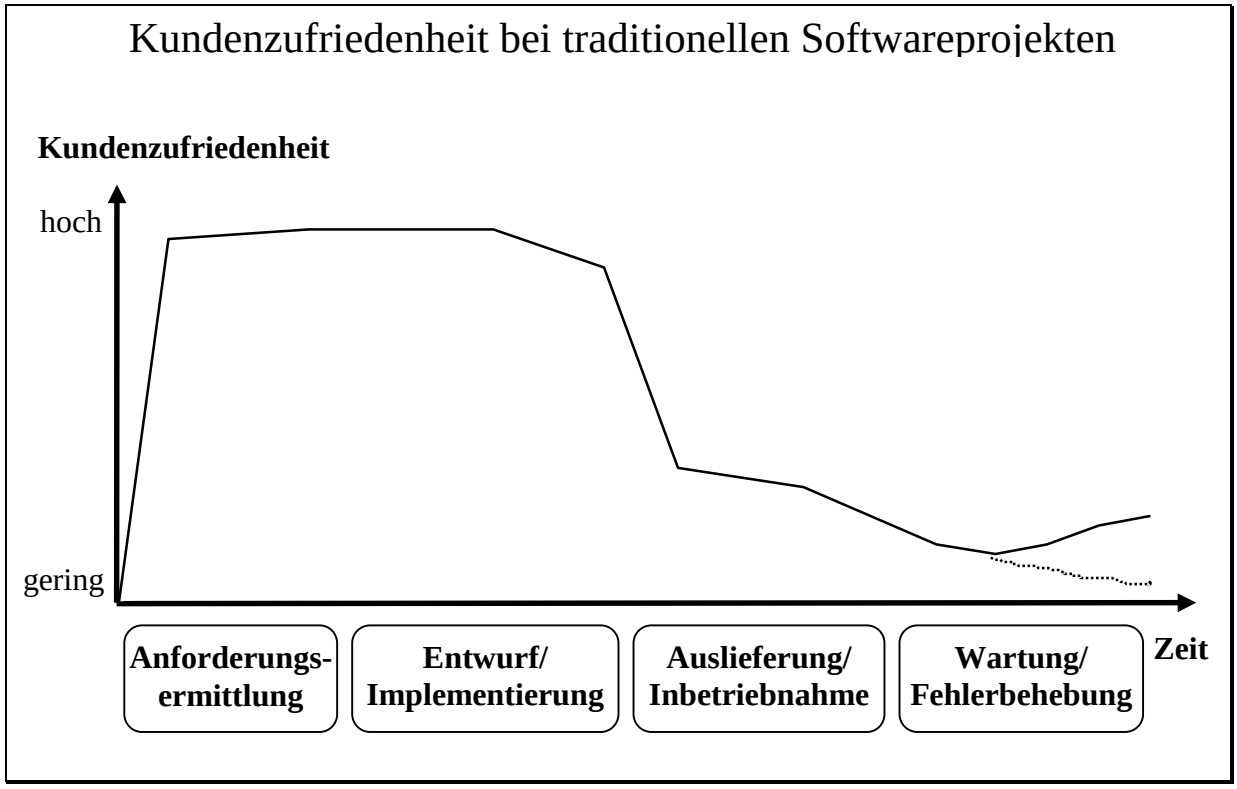

Abbildung 3-4

Die *Anforderungsermittlung* ist oft gekennzeichnet durch intensive Einbeziehung der späteren Benutzer des Softwaresystems. In Einzel- und Gruppengesprächen wird der Ist-Zustand ermittelt und die Anforderungen an das neue System erhoben. Die Vertreter des Kunden fühlen sich an der Entstehung des neuen Systems direkt beteiligt und in ihren Interessen sehr gut wahrgenommen. Sie können ihre aktuellen Probleme berichten und ihre Vorstellungen und Wünsche an die neue Software direkt oder indirekt über Benutzervertreter äußern. Es entsteht die Überzeugung, daß die neue Software viele der dringenden Alltagsprobleme lösen wird. Als Konsequenz entsteht eine relativ hohe Kundenzufriedenheit (vgl. Abschnitt 4.4.2).

In der *Entwurfs- und Implementierungsphase* gehen die direkten Kontakte des Kunden mit dem Entwicklungsprozeß zurück. Auf der Kundenseite nimmt man wieder das Alltagsgeschäft auf, in der Überzeugung, seinen Teil zur Systementwicklung beigetragen zu haben. Aus der Sicht des Kunden sind jetzt „die Techniker an der Reihe“, die das System schon „irgendwie hinkriegen“ werden. Gegen Ende der Implementierung tauchen jedoch erste Risse in der Kundenzufriedenheit auf, wenn z.B. - wie es in traditionellen Projekten häufig zu beobachten ist - Termine für die Inbetriebnahme oder für vorgelagerte Schulungen verschoben werden müssen, weil die Software noch nicht fertig war (vgl. Abschnitt 4.4.10).

Die *Inbetriebnahme* des Systems läuft dann meistens mehr oder minder chaotisch ab. Die Zeit zwischen Anforderungsermittlung und Produktionsbeginn ist zu lang gewesen, die Erinnerungen an die frühen Gespräche sind verblaßt und die Erwartungen an die neue Software dementsprechend hoch und unrealistisch. Die Benutzer haben eine funktionierende Software erwartet, die ihre Probleme löst und sind nun im Echtbetrieb mit instabiler und teilweise unvollständiger Software konfrontiert, die ihren Arbeitsalltag nur unzureichend

abbildet und oftmals geradezu behindert. Die Zufriedenheit der Anwender sinkt dramatisch, was direkte Auswirkungen auf das Management des Kunden hat, da es mitunter zu gravierenden Problemen im Tagesgeschäft kommt (vgl. Abschnitt 4.4.13)

Im Rahmen der *Wartungsphase* tritt nur langsam eine Verbesserung der Lage ein, wobei im wesentlichen schnelle Fehlerbehebungen an den entscheidenden Stellen des Programms für Entspannung und Zunahme der Kundenzufriedenheit sorgen. Werden jedoch in dieser Phase schwerwiegende konzeptionelle Schwächen am neuen System entdeckt, die nicht „über Nacht“ behoben werden können, so kann die Zufriedenheit des Kunden durchaus auch noch weiter absinken, was im Extremfall zu einer vollständigen Ablehnung der Software führen kann, zumindest jedoch eine erhebliche Belastung der Weiterentwicklung darstellt (vgl. Abschnitt 4.4.14 und 4.5).

Man kann festhalten, daß eine traditionelle Vorgehensweise langfristig beinahe zwangsläufig zu einer hohen Unzufriedenheit bei vielen am Projekt Beteiligten und vom Projekt Betroffenen führt. Im einzelnen sind dies:

- Das **Auftraggeber-Management**, weil zugesagte Termine nicht gehalten werden und weil das Produkt im Endeffekt mehr kostet und weniger leistet als erwartet.
- Die **Anwender**, weil die Software nicht den Erwartungen entspricht, weil sie schwer und umständlich zu bedienen ist und die tägliche Arbeit nur unzureichend unterstützt bzw. sogar behindert.
- Aber auch das **Management des Softwareherstellers**, weil der Herstellungsprozeß, besonders in späten Phasen des Projektes, zunehmend unkontrollierbar wird und mit viel zu hohen Kosten verbunden ist. Wenn der Kunde (z.B. im Rahmen eines Festpreisprojektes) nicht bereit ist, die entstehenden Mehrkosten mitzutragen, kann daraus für den Softwarehersteller mitunter eine existenzgefährdende Situation entstehen.
- Das **Marketing des Softwareherstellers**, weil dem Auftraggeber zugesagte Leistungen nicht oder nur unzureichend umgesetzt werden und somit eine fortgesetzte Konfrontation mit der Unzufriedenheit des externen Kunden erfolgt.
- Und schließlich das **Entwicklungsteam** (und der einzelne Entwickler), weil trotz hohen Arbeitseinsatzes wenig Belohnung erfolgt, selbstgesteckte Ziele nicht erreicht werden, zu viele Kompromisse eingegangen werden mußten und den „Sachzwängen“ grundsätzliche Ansprüche an Qualität geopfert werden.

Es herrscht also nicht nur hohe Unzufriedenheit beim ‘externen’ Kunden, sondern auch bei den ‘internen’ Kunden.

Entscheidend ist auch, daß sich die Unzufriedenheit nicht nur auf das letztendliche Softwareprodukt richtet, sondern auch auf den Entwicklungsprozeß selbst. Sowohl Produkt- als auch Prozeßqualität leiden also unter einer traditionellen Vorgehensweise.

3.7.1.3 Fazit zur traditionellen Softwareentwicklung

Das der traditionellen Softwareentwicklung zugrunde liegende Modell bildet die Realität von Softwareentwicklungsprojekten nur unzureichend ab. In der Anwendung einer phasenorientierten Vorgehensweise entstehen zwangsläufig eine Reihe schwerwiegender Probleme sowohl für Auftraggeber als auch für Auftragnehmer.

Die Anforderungen an eine kundenorientierte Prozeßgestaltung lassen sich im Rahmen traditioneller Vorgehensweise nicht hinreichend umsetzen. Insbesondere ist folgendes zu kritisieren:

- Die geforderte Einbindung des Kunden wird nicht im gesamten Projektverlauf umgesetzt, sondern auf die Randphasen (Anforderungsermittlung und Systembetrieb) beschränkt.
- Die Einbeziehung des Kunden dient für die Entwicklungsorganisation primär zur Informationsbeschaffung, d.h. es findet eine Instrumentalisierung des Kunden statt. Von einer Integration des Kunden in den Entwicklungsprozeß kann nicht gesprochen werden.
- Die Umsetzung eines kundenorientierten Qualitätsbegriffs findet nicht statt. Die eingesetzten Meilenstein-Dokumente sind in der Regel vom Kunden hinsichtlich ihrer Qualität zu schwer beurteilbar, da sie eine primär technische Ausrichtung haben.
- Eine Orientierung an der Sprache des Kunden findet nicht statt. Tendenziell ist der Kunde gezwungen, sich an die technikorientierte Sprache der Entwickler anzupassen.
- Frühzeitige Rückkopplungsprozesse finden nicht statt. Somit entstehen Erwartungen an das Softwaresystem, die nicht eingehalten werden.
- Die Nichteinhaltung von Zusagen bezieht sich nicht nur auf Funktionalität, sondern auch auf Termine und Kosten. Die Vorgehensweise, die zunächst Kontrollierbarkeit suggeriert hatte, entpuppt sich im Projektverlauf als unkontrollierbar.

Die genannten Probleme lassen sich aus meiner Sicht auch durch Anpassung des Modells innerhalb des Modells nicht mehr lösen. Ein phasenorientiertes Vorgehen im Sinne des Wasserfall-Modells hat die Softwarekrise letztlich nicht bewältigt. Im Gegenteil: Die fortgesetzte Anwendung traditioneller Vorgehensmodelle hat dazu geführt, daß in vielen Unternehmen die DV-Abteilung **„eine der am wenigsten bewunderten Unternehmens-funktionen“** ([Strassm95]) geworden ist.

Anders ausgedrückt: Die traditionelle Softwareentwicklung hat systematisch zur Kunden-unzufriedenheit geführt. Dieses schließt aus der Sicht des Softwareherstellers sowohl die externen als auch die internen Kunden mit ein.

In bezug auf die der Arbeit zugrunde liegende Fragestellung muß also festgehalten werden:

> ***Die traditionelle Softwareentwicklung in ihrer phasenorientierten Vorgehensweise nach dem Wasserfall-Modell führt langfristig zu Kundenunzufriedenheit und kann deshalb nicht kundenorientiert genannt werden.***

Obwohl in der Praxis selten ein Vorgehen nach einem reinen Wasserfall-Modell angewendet wird, so findet man jedoch sehr häufig organisatorische Maßnahmen, die eine Annäherung an dieses „Idealbild“ ermöglichen und damit letztlich ein Phasenkonzept aufrecht erhalten sollen. FLOYD führt dazu aus:

> „Dies führt zu Denk- und Arbeitsformen, mit Hilfe derer man das Idealbild zu simulieren versucht, etwa durch organisatorische Maßnahmen wie die Abschirmung des Entwicklungsteams von der Kommunikation mit dem Kunden, die Zulassung von Rückgriffen auf bereits abgeschlossene Phasen oder die Einführung von formalisierten Verfahren zur Berücksichtigung geänderter Anforderungen.“ ([Floyd93]).

Für eine kundenorientierte Softwareentwicklung muß das traditionelle Vorgehensmodell aus meiner Sicht weder verändert noch verbessert, sondern grundsätzlich in Frage gestellt und abgelöst werden. Im folgenden will ich erläutern, warum sich als Alternative evolutionäre und partizipative Entwicklungsstrategien anbieten.

3.7.2 Kundenorientierung und evolutionäre Softwareentwicklung

Die traditionelle Softwareentwicklung diente ihren Kritikern als Ausgangspunkt für die Entwicklung alternativer Vorgehensweisen (vgl. u.a. [Floyd93], [KilGryZü93]). Die Grundgedanken dieser evolutionären, partizipativen Vorgehensweisen sollen im folgenden kurz genannt werden, um sie anschließend in bezug zu den Anforderungen an eine kundenorientierte Prozeßgestaltung zu setzen.

3.7.2.1 Grundgedanken evolutionärer und partizipativer Softwareentwicklungen

Eine wesentliche Beobachtung bei traditioneller Softwareentwicklung sind die immensen Kosten, die in der Phase der Wartung entstehen. Diese sind oft um ein Vielfaches höher als die gesamten vorher angefallenen Entwicklungskosten. Eine Erklärung dieses Phänomens ist, daß die Bezeichnung ‘Wartung’, die auf ein Reparieren von kleineren Fehlern und Ungenauigkeiten hindeutet, im Grunde falsch gewählt ist, da sie den wahren Sachverhalt verschleiert: Tatsächlich geschieht in der Phase der Wartung etwas viel Wesentlicheres, nämlich die graduelle (d.h. evolutionäre) Anpassung der Software an die realen Erfordernisse der Anwendungswelt. Anders gesagt: Das, was bei traditionellen Projekten in der Phase der Wartung als Problem erscheint, deutet auf das Wesen von Softwareentwicklungen hin.

Tatsächlich erscheint die Erstellung eines Softwareproduktes im Sinne einer Fertigung in linearen Schritten gar nicht das entscheidende Problem von Softwareentwicklungen zu sein, sondern im Gegenteil die schrittweise Entwicklung im Zuge von Rückmeldungsprozessen aus dem Anwendungskontext. Innerhalb von traditionellen Vorgehensweisen kann diese Evolution des Systems erst in der Phase der Wartung stattfinden, weil sie vorher innerhalb einer strikten Phasenorientierung ausgeklammert wurde. Es ist deshalb naheliegend, diesen grundsätzlich evolutionären Charakter von Softwareentwicklung ins Zentrum der Vorgehensweise selber zu rücken.

Eine zweite wesentliche Beobachtung bei traditionellen Projekten ist die Art, wie Softwarehersteller und Kunde miteinander kommunizierten und welche Art von Kundenbeteiligung angestrebt wird. Wie bereits im Rahmen der Kritik an traditionellen Vorgehensweisen erläutert, wird hier tendenziell eine Ausgrenzung wichtiger Gruppen aus dem Entwicklungsprozeß bzw. eine phasenorientierte Beteiligung mit Schwerpunkt auf die Randbereiche des Projektes (Anforderungsermittlung und Systembetrieb) angestrebt. Dies betrifft nicht nur den Kunden, sondern auch die Entwickler der Software. Beispielsweise ist es im Sinne einer sequentiellen Arbeitsteilung durchaus üblich, die sog. Analysephase von anderen Personen durchführen zu lassen, als die Entwurfs- und Implementierungsphase. Der dadurch resultierende Informationsverlust hat jedoch schwerwiegende negative Konsequenzen. Evolutionäre Vorgehensweisen empfehlen demgegenüber eine engere und vor allem kontinuierliche Beteiligung des Kunden (und hier besonders der Anwender) einerseits und der Entwickler andererseits. Auch die Idee einer am Zeitverlauf orientierten strikten Arbeitsteilung wird aufgegeben. Analyse-, Entwurfs- und Implementierungstätigkeiten werden stärker ineinander verschränkt und in den Aufgabenbereich der Entwickler zurückverlagert. Die notwendige Arbeitsteilung findet anhand fachlich orientierter Aufgabenbereiche statt.

In traditionellen Projekten wird der Versuch unternommen, Kommunikationsbeziehungen zwischen Anwendern und Entwicklern zu minimieren (vgl. [KilGryZü93]). Im Gegensatz dazu betonen evolutionäre Vorgehensweisen die Bedeutung direkter Anwender-Entwickler-Kommunikation, weil nur so bei den Entwicklern ein ganzheitliches Verständnis von den Arbeitszusammenhängen in der Anwendungswelt entstehen kann. KILBERTH et al. formulieren es folgendermaßen:

> „Die Voraussetzung für die Entwicklung eines solchen Systems ist, daß alle Beteiligten, insbesondere die Entwickler, verstehen, was die Anforderungen im

> Anwendungsbereich sind und wie die dort anstehenden Aufgaben erledigt werden. Diese Notwendigkeit, eine Anwendungssituation zu verstehen,… steht im Widerspruch zu konventionellen Sichtweisen, bei denen es Aufgabe der Entwickler ist, aus einem vorgegebenen Pflichtenheft oder einer anderen vorgegebenen Beschreibung in schematischen, möglichst formalen Schritten ein Softwaresystem abzuleiten.“ ([KilGryZü93])

Softwareentwicklung wird nicht mehr als lineare Produktentwicklung, sondern als Arbeitsgestaltung im Rahmen eines wechselseitigen Lernprozesses verstanden.

Als dritter Themenkomplex erscheint die Frage nach Projektsteuerung. Softwareentwicklungsprojekte sind heute in ihrem Verlauf oft so kompliziert und undurchschaubar, daß steuernde Maßnahme eine Notwendigkeit darstellen, um Projekte nicht ins Chaos laufen zu lassen. In traditionellen Projekten findet Projektsteuerung anhand der schon beschriebenen Phaseneinteilung statt. Das Vorgehen ist dokumentengesteuert, indem in jeder Phase ein oder mehrere Dokumente entstehen, deren Fertigstellung das Ende der Phase bestimmt. Für die Projektsteuerung hat die Beendigung einer Phase „Meilensteincharakter“; dann kann die Planung mit der Realität abgeglichen werden. Oben wurde bereits gezeigt, daß diese Steuerung auf falschen Annahmen über die Realität von Softwareentwicklungsprojekten beruht und insofern lediglich die Illusion von Kontrolle erzeugen kann, in der Praxis jedoch regelmäßig scheitert (vgl. u.a. [WelOrt92], [Fittkau94]). In evolutionären Vorgehensweisen wird die Phasenorientierung grundsätzlich aufgegeben und durch ein zyklischen Vorgehensmodell ersetzt. Dem von FLOYD vorgestellte Ansatz STEPS („Softwaretechnik für Evolutionäre und Partizipative Systementwicklung“) kommt hierbei herausragende Bedeutung zu (vgl. [Floyd93]).

Eine statische Projektsteuerung über Meilensteindokumente wird in evolutionären Vorgehensweisen durch eine dynamische Prozeßsteuerung mit *Referenzlinien* ersetzt. REISIN definiert den Begriff ‘Referenzlinie’ als:

> „einen Projektzustand, den die Entwickler und Benutzer aus dem Entwicklungsprozeß heraus zur Synchronisation ihrer jeweiligen unterschiedlichen und gemeinsamen Arbeitsprozesse vereinbaren. Eine Referenzlinie wird vereinbart, indem ein angestrebter Projektzustand durch die Spezifikation eines (Ziel-) Produkts (oder eine Liste aus mehreren Produkten), auf das die Arbeitsprozesse der Entwickler und Benutzer gerichtet werden sollen, definiert wird und die Kriterien benannt werden, die es erfüllen muß, bevor der nächste Projektzustand in Angriff genommen werden kann.“ ([Reisin91])

Dadurch, daß Referenzlinien immer vor dem Hintergrund realer Projektzustände definiert werden, wird ein überschaubarer Planungshorizont geschaffen, der für die Beteiligten transparent und überprüfbar ist (vgl. [BauBoeEck94]). Die Planungssicherheit wird damit für beide Seiten erhöht und die Einhaltung von Zusagen der Entwicklungsorganisation erst möglich.

An traditionellen Vorgehensweisen wurde u.a. der Einsatz von Dokumenten kritisiert, die aufgrund ihres monolithischen und technikorientierten Charakters ein Verständnis des Entwicklungsstandes durch den Kunden erschwerten. Alternativ stellen KILBERTH et al. die folgenden Dokumenttypen und Zwischenprodukte vor (vgl. im folgenden [KilGryZü93]):

- Szenarios
- Glossare
- Systemvisionen
- Prototypen
- Pilotsysteme

Die momentane Arbeitsrealität der Anwender wird in Form von *Szenarios* festgehalten. Diese werden von den Entwicklern in Prosa geschrieben und orientieren sich an der Sprache der Anwender. Im *Glossar* wird die Begriffsbildung der Anwendungswelt festgehalten. Beide Dokumenttypen werden nach Erstellung von Anwendern beurteilt. Eine solch frühe Rückkopplung ist möglich, da es sich um Beschreibungen der täglichen Arbeit der Anwender handelt. Anders gesagt: Die Anwender müssen sich in den Szenarios „wiederfinden", sonst sind diese von mangelhafter Qualität. Die Auswertung von Szenarios erfolgt in direkten Gesprächen zwischen Anwendern und Entwicklern. In solchen „Autor-Kritiker-Zyklen" werden Mißverständnisse der Entwickler aufgedeckt, was letztlich dazu führt, daß das Verständnis der Entwickler für die Anwendungswelt wächst.

Neben der Ist-Analyse, die durch Szenarios und Glossar abgedeckt ist, spielt die Schaffung einer gemeinsamen Vision des zukünftigen Softwaresystems im evolutionären Entwicklungsprozeß eine entscheidende Rolle. Hier schlagen KILBERTH et al. als Dokumenttypen *Systemvisionen* und *Prototypen* vor. Systemvisionen sind textuelle und graphische Beschreibungen des zukünftigen Systems. Sie sollen die „Vorstellung der Entwickler über das angestrebte Zielsystem vermitteln und entstehen im wesentlichen aus vorhandenen Szenarien" ([Bäum93]). Systemvisionen und Szenarios bilden die Grundlage für entstehende Prototypen. Innerhalb des Entwicklerteams dienen Prototypen u.a. dazu, technische Aspekte des zukünftigen Systems zu evaluieren. Nach „außen" dienen Prototypen dazu, das spätere System so früh und so praktisch wie möglich anzunähern, damit die späteren Benutzer ein Bild davon bekommen, wie ihre zukünftige Arbeitsrealität aussehen wird. Auch Prototypen werden im Rahmen von Autor-Kritiker-Zyklen vom Kunden beurteilt. Damit entsteht die Vision des zukünftigen Systems sukzessive im Prozeß.

Im Rahmen von evolutionärem Prototyping gibt es dabei sowohl Prototypen, die nur zur Klärung eines begrenzten Problemkomplexes eingesetzt werden und nach Fertigstellung weggeworfen werden, als auch Prototypen, die evolutionär weiterentwickelt und in Pilotsysteme überführt werden. Eine Typologie der möglicher Arten von Prototyping findet sich bei BUDDE et al. (vgl. [BuKaKuZü92]).

Wie bereits deutlich wurde, sind die vorgeschlagenen Dokumenttypen nicht isoliert voneinander zu verstehen, sondern bilden als Einheit die Grundlage einer evolutionären Entwicklungsstrategie, deren besonderes Ziel es ist, den Entwicklungsprozeß in enger Kooperation zwischen Entwicklungs- und Anwendungsorganisation durchzuführen.

Zusammenfassend lassen sich folgende Schlüsselelemente einer evolutionären und partizipativen Softwareentwicklung nennen:

- Ein zyklisches Vorgehensmodell, d.h. Entwicklung des Systems in überschaubaren und planbaren Schritten als eine Folge von aufeinander aufbauenden Systemversionen.
- Referenzlinien ersetzen Meilensteine in einer situativen Projektsteuerung.
- Verständnis des Softwareentwicklungsprozesses als Lernprozeß.
- Beteiligung der Betroffenen, d.h. frühzeitige und kontinuierliche Einbeziehung der
  - späteren Anwender des Systems
  - und der Entwickler des Systems.
- Direkte und kontinuierliche Kommunikation zwischen Entwicklern und Anwendern.
- Nutzung von Dokumenttypen, die für alle Beteiligten verständlich sind, die somit die wechselseitige Kommunikation unterstützen und eine frühzeitige Qualitätssicherung ermöglichen.
- Einsatz von Prototyping als zentraler Bestandteil der Entwicklungsstrategie.

3.7.2.2 Fazit zur evolutionären Softwareentwicklung

Wie müssen evolutionäre Entwicklungsstrategien unter der Fragestellung nach Kundenorientierung bewertet werden?

Zunächst einmal ist es wichtig zu sehen, daß evolutionäre Vorgehensweisen nicht primär deshalb angewendet werden, um kundenorientierter zu arbeiten, sondern schlichtweg, um überhaupt zu einem akzeptablen Ergebnis zu kommen. Die Entwicklung innerhalb eines zyklischen Modells stellt eine Notwendigkeit dar, die unabhängig von Kundenorientierung besteht. Nur so kann qualitativ hochwertige Software überhaupt entstehen. Ebenso verhält es sich mit der Forderung nach Partizipation. Partizipation bedeutet sowohl für Entwicklungs- als auch für Anwendungsorganisation zusätzlichen Aufwand und stellt erhöhte Anforderungen an die individuellen Kompetenzen der Teammitglieder. Partizipation stellt kein Ziel an sich dar, im Gegenteil: Softwareentwickler würden ‘gerne’ auf eine umfassende Mitwirkung des Kunden am Entwicklungsprozeß verzichten, wenn sie es denn könnten. In traditionellen Projekten ist die gesamte Vorgehensweise auf eine Minimierung des direkten Kontaktes des Kunden zum Entwicklungsprozeß ausgerichtet. Genau dies schlägt jedoch fehl. Partizipation und Evolution innerhalb evolutionärer Vorgehensweisen sind deshalb kein Selbstzweck, sondern stellen eine Notwendigkeit dar.

Interessanterweise sind die Mittel, die innerhalb einer evolutionären Vorgehensweise vorgeschlagen werden, sehr verträglich mit den Anforderungen an eine kundenorientierte Prozeßgestaltung (vgl. Abschnitt 3.6). Und dies aus folgenden Gründen:

- Die geforderte Einbindung bzw. Integration des Kunden in den Entwicklungsprozeß wird von evolutionären Vorgehensweisen über den gesamten Projektverlauf ausdrücklich angestrebt.
- Die Einbeziehung des Kunden dient primär dem Ziel, ihn zu einem gleichberechtigten Partner des Softwareherstellers zu machen. Die programmatische Forderung nach einer „Softwareentwicklung gemeinsam mit dem Kunden“ ist erfüllt.
- Die Umsetzung eines kundenorientierten Qualitätsbegriffes wird explizit angestrebt. Als kundenorientierte Dokumente und Zwischenprodukte bieten sich z.B. die oben vorgestellten Dokumenttypen (Szenario, Glossar, Systemvision, Prototyp) an.
- Die Projektsprache orientiert sich explizit an der Sprache des Kunden.
- Frühzeitige Rückkopplungsprozesse finden statt und sind sogar zentraler Bestandteil der Vorgehensweise. Die zur Projektkontrolle eingesetzten Instrumente (z.B. Referenzlinien) orientierten sich an realen Gegebenheiten und ermöglichen die im Rahmen von Kundenorientierung geforderte Flexibilität. Erwartungen des Kunden an das Softwaresystem werden im Prozeß schon lange vor dem eigentlichen Systembetrieb mit auswertbaren Zwischenprodukten und Vorstufen des realen Systems konfrontiert und abgeglichen. Dadurch wird die Entstehung von überhöhten Erwartungen verhindert und die Erreichung von Kundenzufriedenheit möglich.
- Die schnellen Rückkopplungsschleifen und die mögliche Anpassung der Projektplanung im Prozeß selbst erhöht die Planungssicherheit und führt zu einer kontinuierlichen Verbesserung von Termin- und Kostenschätzungen. D.h., daß die vom Softwarehersteller gemachten Zusagen zunehmend besser eingehalten werden können und Kundenunzufriedenheit weitestgehend vermieden werden kann.
- Beim Softwarehersteller wird den Entwicklern eine höhere Kompetenz zugetraut, da sie nun sowohl analysierende, entwerfende als auch programmierende Tätigkeiten übernehmen. Damit wird die im Rahmen von Kundenorientierung geforderte Mitarbeiterorientierung auf der Seite des Softwareherstellers unterstützt.

Zusammenfassend kann man damit festhalten, daß evolutionäre Vorgehensweisen die Anforderungen an eine kundenorientierte Prozeßgestaltung erfüllen, woraus sich folgendes Zwischenergebnis ergibt:

Evolutionäre Vorgehensweisen in der Softwareentwicklung erfüllen die Anforderungen an eine kundenorientierte Prozeßgestaltung. Sie haben das Potential, langfristige Kundenzufriedenheit zu erzielen und können deshalb kundenorientiert genannt werden.

Daß die Ziele von Kundenorientierung so stark mit denen evolutionärer Vorgehensweisen korrespondieren, hat den erfreulicher Nebeneffekt, daß sich die Interessen des Managements der Entwicklungsorganisation und die der Softwareentwickler weitgehend annähern. Wo das Management früher tendenziell eine Begrenzung und Reglementierung des Kundenkontaktes anstrebte, waren die Entwickler von jeher auf eine intensive Mitwirkung des Kunden am Entwicklungsprozeß angewiesen, da sie ohne dies nicht konstruieren konnten. Weiterhin waren sie auf frühzeitige und kontinuierliche Rückmeldungen im Prozeß angewiesen, weil sie nur so überprüfen konnten, ob sie ‘auf dem richtigen Weg sind’. Im Zuge von Kundenorientierung ändern sich jetzt die Interessen des Managements des Softwareherstellers. Kundeneinbeziehung bekommt eine strategische Bedeutung. Um es plakativ auszudrücken: *‘Keine Kundenbindung ohne Kundeneinbindung’*.

3.7.3 Zwischenfazit zur kundenorientierten Prozeßgestaltung

Zusammenfassend kann man festhalten, daß evolutionäre Vorgehensweisen (im Gegensatz zu traditionellen) von ihrer Grundanlage her für eine kundenorientierte Softwareentwicklung in Frage kommen.

Damit die angestrebte Kundenorientierung erreicht wird, bedarf es jedoch noch mehr: Kundenorientierte Softwareentwicklung braucht als Stützen eine kundenorientierte Organisationsstruktur und eine kundenorientierte Organisationskultur in der Entwicklungsorganisation (und vermutlich auch in der Anwendungsorganisation). Dies soll im folgenden Abschnitt behandelt werden.

3.8 Voraussetzungen für eine kundenorientierte Softwareentwicklung

In Abschnitt 2.6.4 wurden zwei Bereiche genannt, die entsprechend der Unternehmensstrategie Kundenorientierung gestaltet werden müssen:

- die Organisationsstruktur
- und die Organisationskultur bzw. das Verhalten der einzelnen Mitarbeiter.

Merkmale von kundenorientierten Organisationsstrukturen und einer kundenorientierten Organisationskultur in der Softwareentwicklung sollen im weiteren vorgestellt werden. Dabei werden die Überlegungen zur Struktur eher kurz gehalten und der Hauptschwerpunkt bewußt auf die Organisationskultur gelegt, da dieser in der Literatur die relativ größere Bedeutung beigemessen wird (vgl. [Homburg94]).

3.8.1 Strukturelle Voraussetzungen für eine kundenorientierte Softwareentwicklung

Im Sinne von „structure follows strategy“ ([DopLaut94]) muß Kundenorientierung als strategische Ausrichtung von den Strukturen innerhalb der Entwicklungsorganisation getragen und unterstützt werden.

Strukturelle Voraussetzungen für Kundenorientierung betreffen die **Aufbau-Organisation** einerseits und die **Ablauf-Organisation** anderseits. Beide müssen auf eine Heranführung der Entwicklungsorganisation an den Kunden bzw. eine Integration des Kunden in die Entwicklungsorganisation ausgerichtet werden.

Die Untersuchung von HOMBURG (vgl. [Homburg94]) legt nahe, daß die in Kapitel 2 aufgeführten **aufbauorganisatorischen** Maßnahmen (z.B. Vermeidung bürokratischer Strukturen, flache Hierarchien und Entscheidungsdezentralisierung) weitgehend branchenunabhängig sind und daher auch in der Softwareentwicklungsorganisation Gültigkeit haben.

Kundenorientierung bedeutet eine strukturanaloge Abbildung des Kundensystems in die Entwicklungsorganisation hinein. Softwareentwicklung findet schon heute fast ausschließlich innerhalb von Projekten und Teams statt. Deshalb ist aufbauorganisatorisch eine kundensystemanaloge Projektteam-Organisation zu fordern.

Eine Projektteam-Organisation ist eine am Kunden orientierte Prozeßorganisation und steht als solche konträr zu traditionellen Linienorganisationen. Die traditionelle Trennung z.B. von Vertrieb, Systemanalyse, Systemberatung und Entwicklung muß hier mehr und mehr aufgeweicht werden. Auch innerhalb der Entwicklungsabteilungen eines Softwareherstellers gibt es historisch gewachsene Trennungen, die nun überdacht werden müssen:

Als Beispiel sei die im Zuge von Client-Server-Architekturen zu beobachtende Trennung in Client-Entwicklerteams und Server-Entwicklerteams. Diese Trennung muß vor dem Hintergrund von Kundenorientierung in Frage gestellt werden. Tatsache ist, daß auch die Konstrukteure der Datenbank eine genaue Vorstellung von der Applikation haben müssen, die diese Datenbank schließlich benutzen soll, d.h. sie brauchen ebenfalls eine kundenorientierte Perspektive. Deshalb sollten die oben geforderten interdisziplinären Teams (vgl. Abschnitt 3.6.3) in jedem Fall sowohl Datenbank- als auch Client-Entwickler einschließen. Die Folgen einer strikten Trennung dieser Bereiche wird im Rahmen des Fallbeispiels noch diskutiert werden (vgl. Abschnitt 4.4.6).

Zusätzlich zu den am Kunden orientierten Projektteams, müssen projektübergreifende Strukturen geschaffen werden. Hier muß es sowohl Strukturen geben, die projektsteuernde Funktion haben (Stichwort *„Steuerungskreis“*), als auch Strukturen, deren Aufgabe es ist, das Wissen aus den einzelnen Projekten zu integrieren und im Unternehmen verfügbar zu machen.

Beispielsweise muß es im Zuge objektorientierter Vorgehensweisen Aufbaustrukturen geben, die die angestrebte Wiederverwendung auch über die Grenzen des einzelnen Projektes hinaus fördern. Hier empfehlen KILBERTH et al. z.B. die Schaffung einer sog. 'Architekturgruppe', deren explizite Aufgabe es ist, technisches und fachliches Know-How aus den einzelnen Projekten organisationsweit zu integrieren und zu konsolidieren (vgl. [KilGryZü93]).

Die **Ablauforganisation** ist bereits unter dem Aspekt der kundenorientierten Prozeßgestaltung beschrieben worden (vgl. Abschnitt 3.6 und 3.7). Diese beruht auf einer evolutionären Vorgehensweise, die eine umfassende Integration des Kunden in den Entwicklungsprozeß anstrebt.

Abschließend sei darauf hingewiesen, daß Kundenorientierung nicht nur eine kundenorientierte Projektdurchführung meint. Da Kundenorientierung eine langfristige, enge Zusammenarbeit zwischen Entwicklungs- und Anwendungsorganisation anstrebt, geht es um mehr als das einzelne Projekt. Kundenorientierte Softwareentwicklung muß als ein kontinuierlicher Veränderungsprozeß innerhalb der Anwendungsorganisation verstanden werden, den es von Seiten der Entwicklungsorganisation zu unterstützen gilt.

3.8.2 Kulturelle Voraussetzungen für eine kundenorientierte Softwareentwicklung

Neben „structure follows strategy" gilt DOPPLER et al. zufolge in zunehmendem Maße auch „culture follows strategy" ([DopLaut94]). Auch die Kultur einer Organisation muß sich an der übergeordneten Strategie ausrichten.

Unter dem Begriff Unternehmenskultur „ist die Gesamtheit von Normen, Wertvorstellungen und Denkhaltungen zu verstehen, die das Verhalten der Mitarbeiter aller Stufen und somit das Erscheinungsbild der Unternehmung prägen." ([Althaus95]). Darüber hinaus geht die Unternehmenskultur aus „Strukturen, Führungsinstrumenten, Personalauswahl und -entwicklungskonzepten, Know-How Standards, Infrastrukturgestaltungen, der Qualität und Modernität der Unternehmungsaktiva usw. hervor. So verstanden ist die Unternehmungskultur nicht nur 'eine psychologische Konstruktion', sondern ein Spiegelbild der ganzheitlichen Unternehmungsrealität" (ebd.)

Unternehmenskultur ist damit als ein äußerst vielschichtiges und komplexes Konstrukt einzuschätzen. Das Dilemma ist, daß es einerseits schwer präzise zu fassen ist, anderseits eine überragende Bedeutung bei der Umsetzung von Kundenorientierung zu spielen scheint (vgl. [Homburg94]). Die Unternehmenskultur erscheint hier einerseits als Schlüssel, andererseits als möglicher „Ansatzpunkt für eine konsequente Ausgestaltung der Kundenorientierung." ([Althaus95]).

Damit wird deutlich, daß eine 'kundenorientierte Unternehmenskultur' wiederum Visionscharakter hat. Die Aufgabe ist also einerseits, diese Vision relativ greifbar zu beschreiben, und anderseits Mittel und Wege aufzuzeigen, eine gegebene Unternehmenskultur in Richtung auf die angestrebte Zielvorstellung hin zu entwickeln. Wenn man sich darüber hinaus vor Augen hält, daß die Unternehmenskultur sich letztlich im individuellen Verhalten ausdrücken muß, geht es auch um die Entwicklung des einzelnen Mitarbeiters hin zu einem „kundenorientierten Mitarbeiter".

Im folgenden soll deshalb zunächst die Vision einer kundenorientierten Unternehmenskultur und danach die eines „kundenorientierten Softwareentwicklers" konkretisiert werden.

3.8.2.1 Elemente einer kundenorientierten Unternehmenskultur

In der Entwicklung der Vision einer kundenorientierten Unternehmenskultur kann zunächst die Abgrenzung vom Negativ-Bild einer „Bürokratiekultur" hilfreich sein. Es besteht in der

wissenschaftlichen Literatur Konsens darüber, daß sich eine bürokratisch geprägte Kultur konträr zu den Zielen von Kundenorientierung verhält. In der Bürokratiekultur fällt die starke Ausrichtung nach innen auf: auf die innere Struktur, auf klar reglementierte aber unflexible Abläufe, die auf Dauer festgeschrieben und nur schwer zu verändern sind. Besonders die Unflexibilität und die Trägheit gegenüber Neuerungen steht im Widerspruch zu den sich rasch ändernden äußeren Einflüssen, hier insbesondere den sich ändernden Kundenbedürfnissen. Damit wird schon deutlich, daß eine kundenorientierte Kultur eine starke Komponente von Veränderungsbereitschaft enthalten muß. In diesem Sinne kann man von ***'Veränderungskultur'*** sprechen. Hierunter fällt auch der Aspekt der Risikobereitschaft, der ebenfalls mit Kundenorientierung in Verbindung gebracht wird (vgl. [Homburg94]).

Unter dem Aspekt der Organisationsstruktur wurde bereits eine Aufhebung von strikten Abteilungsgrenzen hin zu einer Teamorganisation gefordert. Als Grundlage für teamorientiertes Arbeiten ist eine ***'Teamkultur'*** notwendig. Diese kann im wesentlichen nur durch kontinuierliche Teamentwicklungsmaßnahmen aufgebaut und gefestigt werden.

Vor dem Hintergrund von Kundenorientierung ist unmittelbar deutlich, von wo Veränderungen initiiert werden, nämlich vom Kunden, und zwar sowohl direkt (in Form konkreter Nachfrage) oder aber indirekt durch die Wahrnehmung von unbewußten und/oder langfristigen Kundenbedürfnissen durch den Hersteller. In diesem Zusammenhang scheint der Lernbereitschaft eine wichtige Bedeutung zuzukommen, d.h. es geht um die Schaffung einer ***'Lernkultur'***.

Damit haben wir drei aus meiner Sicht zentrale Elemente einer kundenorientierten Unternehmenskultur identifiziert:

- Veränderungskultur
- Teamkultur
- Lernkultur

Es deutet einiges daraufhin, daß letzteres eine übergeordnete Rolle spielt. Das Konzept der „Lernenden Organisation" (vgl. [Senge94]) wird von vielen Autoren als vielversprechende Möglichkeit auch bei der Umsetzung von Kundenorientierung gesehen (vgl. [Althaus95]). Auch für den Bereich der Softwareentwicklung kann dieses Konzept nutzbringend eingesetzt werden (vgl. [Glagda96]). Das weiter oben vorgestellte Verständnis des Softwareentwicklungsprozesses als Lernprozeß deutet ebenfalls auf die besondere Stellung des Lernens hin.

In bezug auf eine kundenorientierte Softwareentwicklung erscheinen mir die folgenden Bereiche wichtig:

- Vom Kunden lernen
- Mit dem Kunden lernen
- Den Kunden lehren

Vom Kunden zu lernen, bedeutet nicht, Informationen zu sammeln, sondern die Bedürfnisse des Kunden auf einer ganzheitlichen Ebene zu verstehen. Deshalb steht das Lernen vom Kunden im Gegensatz zur Instrumentalisierung des Kunden als schlichten Informationslieferanten. Gleichzeitig beinhaltet das Lernen vom Kunden das Eingeständnis, etwas nicht zu wissen und auf die Mithilfe des Kunden angewiesen zu sein.

Mit dem Kunden zu lernen, bedeutet einerseits, ihn in den eigenen Lernprozeß einzubeziehen und anderseits, ihm seinen eigenen Lernprozeß zuzugestehen. Die Forderung, mit dem Kunden zu lernen, steht damit in krassem Gegensatz zu einer Ausgrenzung des Kunden aus dem Lernprozeß. Eine Lernkultur ist demnach für die oben diskutierte evolutionäre Softwareentwicklungsstrategie von entscheidender Bedeutung.

Den Kunden zu lehren, bedeutet nicht, ihn zu belehren oder ihm einseitig eigene Vorstellungen aufzuzwingen. Hier geht es mehr darum, die Rolle eines Beraters in Fragen der Technologiegestaltung einzunehmen. Den Kunden zu lehren, hat darüber hinaus das Ziel, ihn zum eigenständigen Lernen und zum Lehren zu befähigen.

Die Elemente einer kundenorientierten Unternehmenskultur müssen vom einzelnen Mitarbeiter im täglichen Verhalten umgesetzt und gelebt werden. Aus der im vorigen Kapitel beschriebenen kundenorientierten Prozeßgestaltung wurde deutlich, daß hier insbesondere dem Softwareentwickler eine wichtige Rolle zukommt. Deshalb wird im folgenden das Bild eines „kundenorientierten Softwareentwicklers“ vorgestellt.

3.8.2.2 Der kundenorientierte Softwareentwickler

Ähnlich wie bei der Frage nach der Unternehmenskultur ist die Beschreibung, wie ein Softwareentwickler zu sein hätte, natürlich nicht ohne weiteres zu geben. Deshalb soll auch hier zunächst auf zwei (Negativ-)Idealbilder zurückgegriffen werden, um in Abgrenzung dazu das Bild eines „kundenorientierten Softwareentwicklers“ zu entwerfen.

BAUKROWITZ et al. stellen fest, daß die betrieblichen Rollen von Softwareentwicklern so diffus sind, daß „zu Recht von einer Berufsgruppe ohne Berufsbild gesprochen werden kann.“ ([BauBoeEck94]). Diese generelle Diffusität bedeutet jedoch nicht, daß keine berufliche Identität aufgebaut wird. Gerade die Diffusität der betrieblichen Rolle, verbunden mit einer hohen beruflichen Mobilität läßt die berufliche Identität „zum einzigen kontinuierlichen Selbstbezug in der Arbeit“ werden ([BauBoeEck94]). Die Autoren beschreiben zwei Idealtypen der beruflichen Identität von Softwareentwicklern:

- Der „Hacker“
- und der „rationale Techniker“

Ersterer wurde bereits 1977 von WEIZENBAUM beschrieben (vgl. [Weizenb77]) und gilt besonders außerhalb des DV-Bereiches (was allerdings die Vertriebsabteilung des Softwareherstellers durchaus einschließt) m.E. bis heute als Prototyp für den Informatiker. Charakteristisch für diesen Typus ist, daß er sich eher als Künstler, der einen Programmcode gestaltet, denn als Ingenieur sieht. Er betont den kreativen Aspekt des Programmierens, was einhergeht mit einem hohen Individualitäts- und Autonomiestreben. Daraus resultiert jedoch eine mehr oder weniger stark ausgeprägte Unfähigkeit zu kommunizieren und zu kooperieren, wodurch die z.T. genialen Beiträge zum Programmcode in ihrer Bedeutung stark schmälert werden. Der „Hacker“ (WEIZENBAUM nennt ihn den „zwanghaften Programmierer“) lebt in einer eigenen Welt, die reduziert ist auf Denkkonstrukte, die in Form von Programmcode Gestalt annehmen. Daß dieses Bild diametral zu den Ideen von Kundenorientierung liegt, ist offensichtlich. Für den Hacker gilt nicht nur „Störenfried Kunde“, sondern sogar m.E. „Störenfried Kollege“.

Das Bild des „rationalen Technikers“ kann als ausdrücklicher Versuch angesehen werden, das negative Image des Hackers zu überwinden. An die Stelle des Künstlertums tritt die Anwendung von prinzipiell allen zugänglichen technischen Methoden. Das dem Hacker innewohnende irrationale Moment weicht der Rationalität technischer Verfügbarkeit und Kontrollierbarkeit. Der Techniker befürwortet Standardisierung, sowohl bezüglich der Arbeitsprozesse, wie auch der Arbeitsprodukte (vgl. [BauBoeEck94]). WEIZENBAUM nennt dies den „Berufsprogrammier“ und stellt ihn explizit dem „zwanghaften Programmierer“ gegenüber (vgl. [Weizenb77]). Vor dem Hintergrund von Kundenorientierung muß jedoch auch das Bild des „rationalen Technikers“ aus den folgenden Gründen kritisch beurteilt werden:

- Es findet eine primäre Aufmerksamkeitsausrichtung auf die technischen Aspekte der Softwareentwicklung statt. Sowohl das Softwareprodukt, als auch der Entwicklungsprozeß selbst werden als rationale technische Problemkomplexe angesehen. Die Tatsache, daß Softwareentwicklung im wesentlichen ein sozialer Prozeß ist, indem Arbeit gestaltet und verändert wird, wird übersehen. Probleme, die aus diesem Mißverständnis erwachsen (wie z.B. ständig neue Anforderungen während der Entwicklung), werden als Störungen angesehen, die es zu minimieren gilt.
- Die vorwiegend technische Ausrichtung erschwert bzw. verhindert direkte Entwickler-Anwender-Kommunikation. Der Anwender (d.h. der Kunde) erscheint für den rationalen Techniker als Quelle von Störungen, die Interaktion mit ihm als „Schnittstellenproblem, das durch eine möglichst eindeutige Informationsübergabe zu bewältigen ist, und nicht als Interaktionssystem, in dem die eigentliche Software-Entwicklung stattfindet“ ([BauBoeEck94]).

Damit sollte deutlich geworden sein, daß weder das Bild des „Hackers“, noch das vom „Rationalen Techniker“ als Grundlage für den „kundenorientierten Softwareentwickler“ herangezogen werden können. Beide vernachlässigen nämlich mehr oder minder offen die Interessen und Bedürfnisse des Kunden bzw. sehen den Kunden als potentielle Störung der eigenen Arbeit an.

Das von BAUKROWITZ et al. vorgestellte Bild des „Prozeßmoderators“ erscheint mir hier wesentlich vielversprechender (vgl. im folgenden [BauBoeEck94]). Im Bild des „Prozeßmoderator“ sehen die Autoren den Kernbereich zukünftiger Aufgaben von Softwareentwicklern: Diese gestalten und verändern im Zuge der Softwareentwicklung die Arbeit des Kunden; sie verstehen den Softwareentwicklungsprozeß als primär sozialen Lernprozeß, der gemeinsam mit dem Kunden ausgestaltet werden muß; und sie sehen im Gegensatz zum rationalen Techniker mehr als nur die eigene *technische Verantwortung*, sondern übernehmen bewußt auch *Prozeßverantwortlichkeit* (der Begriff der ‘Verantwortlichkeit’ schließt den Begriff ‘Verantwortung’ mit ein, meint darüber hinaus jedoch die Fähigkeit, der Kombination von mehreren, möglicherweise konkurrierenden Verantwortungen gerecht zu werden).

Für den „Prozeßmoderator“ fordern die Autoren drei zentrale Kompetenzen, die meiner Meinung nach auch für den „kundenorientierten Softwareentwickler“ von zentraler Bedeutung sind: *Technikgestaltungskompetenz*, *Prozeßkompetenz* und *Kooperationskompetenz*.

Die Erreichung dieser Kompetenzen erfordert beim Softwareentwickler eine Reihe von individuellen *Sozialkompetenzen.* Vor dem Hintergrund von Kundenorientierung bedarf es beim Entwickler darüber hinaus einer bestimmten Grundhaltung, die hier als *„kundenorientierte Grundhaltung“* bezeichnet werden soll. Diese ist Dreh-und-Angel-Punkt seiner Tätigkeit und wird durch die vier vorher genannten umgesetzt. Damit sind die fünf Kernelemente des „kundenorientierten Softwareentwicklers“ die folgenden:

a) Kundenorientierte Grundhaltung
b) Technikgestaltungskompetenz
c) Prozeßkompetenz
d) Kooperationskompetenz
e) Sozialkompetenz

Obwohl in der Praxis nur schwer vollständig voneinander trennbar, sollen diese fünf Kernelemente im folgenden einzelnd vorgestellt werden:

a) Kundenorientierte Grundhaltung

In Anlehnung an die Definition von Kundenorientierung aus Kapitel 2 verstehe ich unter einer kundenorientierten Grundhaltung

> eine grundsätzlich positive Grundeinstellung des Entwicklers zu seinen internen und externen Kunden und die innere Verpflichtung, den gegenwärtigen und zukünftigen Erwartungen und Bedürfnissen dieser Kunden gerecht zu werden.

Eine kundenorientierte Grundhaltung resultiert in einem vom Traditionellen abweichenden Rollenverständnis des Softwareentwicklers. Im Zuge von Kundenorientierung muß sich dieser mehr und mehr als Dienstleister für die Lösung informationstechnischer Probleme seiner Kunden verstehen.

b) Technikgestaltungskompetenz

Dies ist die klassische Anforderung an Softwareentwickler überhaupt. Hierunter fallen die Fähigkeit zu programmieren und mit Hardware umzugehen (man könnte sagen, die positiven Qualitäten des „Hackers“), ebenso wie die Fähigkeit, in der Entwicklung systematisch, strukturiert und zielorientiert vorzugehen (die Tugenden des „rationalen Technikers“).

Darüber hinaus sind hier aber auch Fähigkeiten gemeint, die über den relativ begrenzten Bereich der Technik hinausgehen und das Umfeld einbeziehen, für das die Technik entwickelt wird:

- „Die Technik muß im Kontext von Unternehmensstrategien und Arbeitsgestaltungsprozessen interpretiert und gestaltet werden können.
- Der Technikeinsatz muß im sozialen Zusammenhang interpretiert und gestaltet werden können.
- Die Technik muß im Interessenbezug interpretiert und gestaltet werden können“ ([BauBoeEck94]).

Mit Blick auf Kundenorientierung bedeutet Technikgestaltungskompetenz insbesondere, die Technik in bezug zu den Anforderungen kundenorientierter Arbeit interpretieren und gestalten zu können.

c) Prozeßkompetenz

In dem Maße, wie Technikgestaltung immer auch Arbeitsgestaltung und Arbeitsveränderung ist, werden Softwareentwickler immer stärker in komplexe soziale Veränderungsprozesse involviert. Um auch hier handlungsfähig zu bleiben, benötigen sie Prozeßkompetenz. Dies bedeutet:

- „Die sozialen Prozesse in ihrer Dynamik verstehen können.
- Eigene Eingriffsmöglichkeiten und Verantwortung erkennen zu können.
- Interessen und Widersprüche als Motor der Entwicklung verstehen und nutzen können.
- Den Entwicklungsprozeß ohne Zielkonstanz regulierend gestalten zu können.
- Zyklizität und Rückkopplung als Methoden der Entwicklung nutzen zu können.“ ([BauBoeEck94])

d) Kooperationskompetenz

Im Rahmen einer kundenorientierten Prozeßgestaltung wurde die Integration des Kunden als zentrales Element vorgestellt. Damit rückt für den Softwareentwickler die Kooperation mit Experten anderer Fachbereiche in den Mittelpunkt der Aufmerksamkeit. Hier ist Kooperationskompetenz gefordert. Die bedeutet:

- „Die eigene Rolle und Aufgabe im Bezug zu anderen Akteuren im Innovationsprozeß bestimmen zu können.
- Das Expertenwissen und die Entwicklungsleistung anderer Akteure verstehen und bewerten zu können.
- Die eigene Arbeit auf das Expertenwissen und die Entwicklungsleistung der anderen Akteure beziehen zu können.“ ([BauBoeEck94])

Wesentlich erscheint mir, daß sich die Kooperationskompetenz des Entwicklers nicht nur im direkten Kontakt zum Kunde zeigen muß, sondern auch und gerade bei eigenständiger, d.h. zeitlich und räumlich vom Kunden getrennter Arbeit. Hier muß das geforderte „Denken wie Kunden“ praktisch umgesetzt werden. Gerade bei eigenständiger Arbeit muß der Kunde „virtuell“ anwesend bleiben. Das bedeutet auch, daß eigenständige Arbeit des Entwicklers immer auf zukünftige Kooperation mit dem Kunden ausgerichtet sein muß.

e) Sozialkompetenz

Als wesentliche Bestandteile von Sozialkompetenz nennen BAUKROWITZ et al. (in Anlehnung an HABERMAS) die folgenden (vgl. [BauBoeEck94]):

- Frustrationstoleranz
- Ambiguitätstoleranz
- Empathie
- Sprachkompetenz
- Rollendistanz

Frustrationstoleranz meint die Fähigkeit, die aktuelle Nichtbefriedigung der eigenen Bedürfnisse zu ertragen. Hier ist besonders der Moment gemeint, wo das vom Entwickler internalisierte Bild des Kunden nicht mit dem ‘realen’ Kunden übereinstimmt, wo der reale Kunde also zur Quelle von Frustration wird. Mangelnde Frustrationstoleranz führt in dieser Situation entweder dazu, daß der Kunde innerlich oder äußerlich abgewertet wird (‘Ich weiß es besser, was für den Kunden gut ist’) oder daß eine ‘Dienst-nach-Vorschrift’-Haltung eingenommen wird (‘Der Kunde hat es ja so gewollt’). Beides ist deutlich nicht kundenorientiert.

Ambiguitätstolerenz bedeutet die Fähigkeit, Unklarheiten und Widersprüchlichkeiten auszuhalten und damit auch in uneindeutigen Handlungssituationen handlungsfähig zu bleiben. Ambiguitätstoleranz ist ähnlich wie Frustrationstoleranz besonders unter dem Aspekt umfassender Kundenintegration und Abkehr vom traditionellen Vorgehen wesentlich. Die traditionelle Sicht versteht den Softwareentwicklungsprozeß im wesentlichen als Problemlösungsprozeß, wo es darum geht, globale Probleme möglichst rasch und systematisch zu verkleinern und in algorithmisierbare Problemlösungen zu überführen. Geringe Ambiguitätstoleranz resultiert in einem zu schnellen Verkleinern der Probleme, läßt potentiell die Kundenperspektive außer acht oder nimmt unangemessene Vereinfachungen an ihr vor. Kundenorientierte Softwareentwicklung bedeutet nicht nur das Lösen von Problemen, sondern auch das Generieren von neuen Problemen globaler Art.

> „Das heißt in letzter Konsequenz: Am Ende des Software-Entwicklungsprozesses stehen ein Softwareprodukt **und** ein Problemhorizont.“ ([BauBoeEck94])

Ambiguitätstoleranz bedeutet, diesen entstandenen Problemhorizont ‘ertragen’ zu können und ihn konstruktiv zu bearbeiten.

Empathie bedeutet zum einen das Einfühlungsvermögen in aktuelle Ausdrucksformen des Interaktionspartners und zum anderen das Verstehen seiner Handlungsorientierung. Man könnte auch von einem inneren Rollentausch sprechen, bei dem die Handlung und das

Verhalten des anderen nicht mehr aus dem eigenen Vorstellungssystem heraus beurteilt wird, sondern aus dem des Interaktionspartners.

> „Über Empathie zu verfügen, besagt, daß ich über meinen Interaktionspartner vorläufige Annahmen konstruiere, die ich in der direkten Kooperation überprüfe und korrigiere; über Empathie nicht zu verfügen, heißt, einem mehr oder weniger bewußten Vorurteil zu folgen (beispielsweise: Benutzer wissen nie, was sie wollen, wenn man ihnen nicht sagt, was sie wollen sollen)." ([BauBoeEck94])

Sprachkompetenz meint die Fähigkeit, sich verständlich ausdrücken zu können. Meiner Meinung nach ist hier im Zuge einer kundenorientierten Softwareentwicklung eine viel stärkere Anpassungsleistung des Entwicklers an den Kunden als umgekehrt nötig. Die Integration des Kunden wird nämlich genau dann unmöglich, wenn er gezwungen ist, auf einer ihm unvertrauten, technikorientierten Sprachebene kommunizieren zu müssen.

Sprachkompetenz drückt sich nicht nur in der direkten Interaktion aus. Sprachkompetenz bedeutet auch, bei der Erstellung von Dokumenten und Zwischenprodukten eine erfolgreiche spätere Interaktion mit dem Kunden vor Augen zu haben, d.h. die Hauptaufmerksamkeit auf Verständlichkeit aus der Sicht des Kunden zu legen. Sprachkompetenz bedeutet weiterhin die Fähigkeit, Sprache gemeinsam mit dem Kunden im Entwicklungsprozeß neu zu konstruieren ('Projektsprache', vgl. Abschnitt 3.7.2.1) und über Kommunikationsprobleme selber kommunizieren zu können ('Metakommunikation'). Für letzteres ist die Kenntnis von erprobten kommunikationswissenschaftlichen Instrumenten wesentlich. Ein Beispiel für ein solches Instrument ist das von SCHULTZ VON THUN vorgestellte „Quadrat der Nachricht", welches vier verschiedene Aspekte einer Nachricht differenziert (Sachinhalt, Selbstkundgabe, Beziehungshinweis und Appell). Auch die Arbeiten von WATZLAWICK et al. sind hier erwähnenswert (vgl. [WatzBeaJa67]).

Rollendistanz beschreibt die Fähigkeit, sich im sozialen Rollenhandeln als unverwechselbare Persönlichkeit zu behaupten. Der kundenorientierte Softwareentwickler kooperiert mit dem Kunden nicht als auswechselbare Rolle 'Entwickler', sondern als eigenständige unverwechselbare Entwicklerpersönlichkeit. Als solche erwartet er persönliche Achtung, die er auch bereit ist, seinem Gegenüber zukommen zu lassen. Zwar nimmt der kundenorientierte Entwickler tendenziell die Rolle eines Dienstleisters gegenüber dem Kunden ein, jedoch gibt er dabei nicht seine persönliche Identität auf. Um es mit den Worten des Geschäftsführers einer Softwareentwicklungsfirma zu sagen: „Die Parole, daß wir als Dienstleister dienen müssen, hat auch weiterhin Bestand. Wie ich es schon öfter ausgeführt habe, hört das Dienen beim Übergang zum Kriechen auf."

Gleichzeitig bedeutet Rollendistanz aber auch, daß eine Beziehung zwischen Entwickler und Kunde jenseits der jeweiligen Rollen durchaus erwünscht ist. Rollendistanz erscheint hier als notwendige Voraussetzung für Konfliktlösung zwischen Entwickler und Kunde: „Die Rollenbeziehung wird durch eine personale Beziehung gebrochen und deren Konflikte entschärft, ohne daß eine rein private Beziehung an ihre Stelle tritt." ([BauBoeEck94])

Die folgende Abbildung veranschaulicht nochmals das oben vorgestellte Qualifikationsprofil eines „kundenorientierten Softwareentwicklers“:

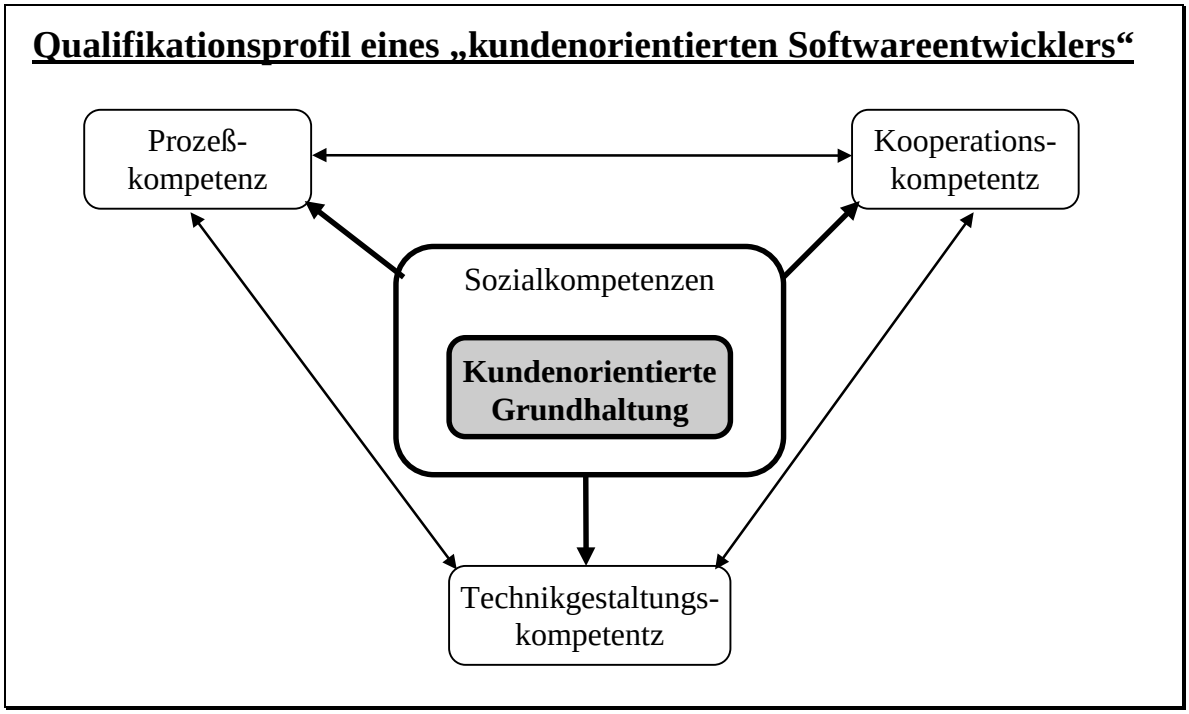

Abbildung 3-5

Es stellt sich hier natürlich die Frage, für welchen Teil der Informatik-Fachkräfte das oben erarbeitete Qualifizierungsprofil Gültigkeit besitzt. Sicherlich gilt es nur in sehr geringem Maße z.B. für Hardwarespezialisten oder die Mehrzahl der Rechenzentrumsberufe. Jedoch gilt es in besonderem Maße für alle anwendungsnahen Bereiche der Softwareentwicklung. Im Zuge von zunehmender Kundenorientierung scheint die Tendenz eindeutig dahinzugehen, die strikte Grenze zwischen DV-Abteilung einerseits und Fachabteilungen andererseits aufzuheben.

Insofern wird die zukünftige Tätigkeit von Informatikern zwangsläufig einen immer stärkeren direkten Anwendungsbezug aufweisen. Meine Vermutung ist deshalb, daß sich die Anforderungen an Softwareentwickler in Zukunft immer stärker dem Qualifizierungsprofil des „kundenorientierten Softwareentwicklers“ annähern werden.

3.9 Zusammenfassung

Dieses Kapitel hatte das Ziel, die Bedeutung von Kundenorientierung für die Softwareentwicklung zu diskutieren.

Es wurde gezeigt, daß Entwicklungsorganisationen vom Markt zunehmend an ihrer eigenen Kundenorientierung gemessen werden, u.a. weil die Kunden der Entwicklungsorganisation selber kundenorientierter werden.

Dann wurde gezeigt, daß Kundenorientierung sowohl Konsequenzen für die Produkt-, als auch für die Prozeßgestaltung hat.

Es wurden Anforderungen an eine kundenorientierte Produktgestaltung formuliert, die im wesentlichen eine Abkehr von der traditionellen ablaufsteuernden Sichtweise betonten. Softwaresysteme, die primär das Ziel haben, den Benutzer zu steuern und zu kontrollieren und ihn auf einen Datentypisten zu reduzieren, können kundenorientierte Arbeit nur schwer unterstützen. Große, monolithische Systeme mit starren Abläufen sind schwieriger anzupassen und zu unflexibel, um auf die raschen Veränderungen der äußeren Umstände reagieren zu

können. Im Gegensatz zu der ablaufsteuernden Sichtweise wurde eine unterstützende Sichtweise als Grundlage kundenorientierter Softwaresysteme vorgeschlagen.

In bezug auf eine kundenorientierte Prozeßgestaltung wurden die Probleme traditioneller, phasenorientierter Vorgehensmodelle diskutiert und gezeigt, daß diese langfristig konträr zu den Zielen von Kundenorientierung stehen und damit als Grundlage einer kundenorientierten Prozeßgestaltung nicht in Frage kommen. Als Alternative wurden evolutionäre Vorgehensmodelle diskutiert. Vor dem Hintergrund der Kundenorientierung wurde gezeigt, warum evolutionäre Vorgehensmodelle als Grundlage für eine kundenorientierte Prozeßgestaltung brauchbar erscheinen.

Es wurde weiterhin gezeigt, daß kundenorientierte Prozeßgestaltung nur auf der Basis von strukturellen und kulturellen Voraussetzungen möglich ist. Als strukturelle Voraussetzungen wurde eine Abkehr von bürokratisch-geprägten Organisationsformen hin zu teamorientierten Strukturen vorgeschlagen. Bezogen auf die Unternehmenskultur wurden die Begriffe 'Lernkultur', 'Teamkultur' und 'Veränderungskultur' als Grundpfeiler einer kundenorientierten Unternehmenskultur vorgestellt. Bezogen auf den einzelnen Entwickler wurde im Gegensatz zu den beiden traditionellen Rollenbildern des 'Rationalen Technikers' und des 'Hackers' das Bild des 'Kundenorientierten Softwareentwicklers' vorgestellt.

Die folgende Abbildung veranschaulicht die oben genannten Zusammenhänge:

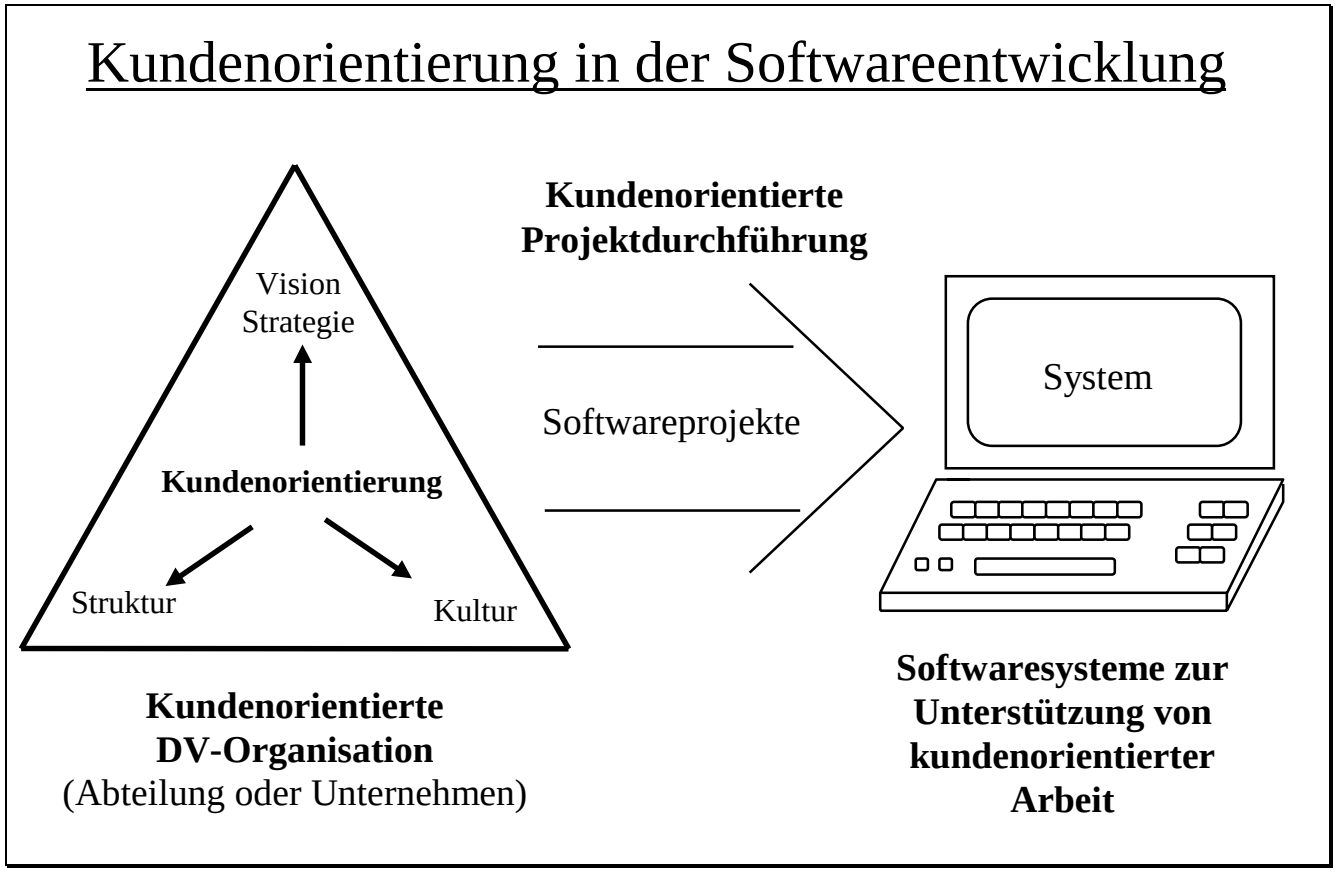

Abbildung 3-6

Eine Softwareentwicklungsorganisation, die sich in bezug auf Strategie, Struktur und Kultur kundenorientiert ausrichtet, entwickelt auf kundenorientierte Weise im Rahmen von Projekten Softwaresysteme, die die Benutzer dieser Softwaresysteme in ihrer eigenen Kundenorientierung unterstützen sollen.

Wie gezeigt wurde, besitzt Kundenorientierung in praktisch allen Bereichen von Softwareentwicklung eine hohe Relevanz. Kundenorientierung wirkt im wahrsten Sinne des Wortes

orientierend und integrierend. Kundenorientierung bietet sich deshalb auch für Softwarehersteller als Integrationsfaktor von Unternehmensführung an.

4 Fallbeispiel - Analyse eines Softwareprojektes

Die Überlegungen zum Thema Kundenorientierung in der Softwareentwicklung sollen nun anhand eines praktischen Fallbeispiels illustriert werden. Im folgenden wird ein großes industrielles Softwareentwicklungsprojekt beschrieben und vor dem Hintergrund von Kundenorientierung analysiert und bewertet. Dabei soll deutlich werden, daß traditionelle Vorgehensweisen einen signifikant negativen Einfluß auf die Kundenorientierung des Softwareherstellers haben.

Die vorliegende Analyse basiert auf einer Reihe von persönlichen Gesprächen, die ich in der Zeit von Juni bis August 1996 mit einem Großteil der auf Seiten des Softwarehauses am Projekt Beteiligten geführt habe. Die verschiedenen Standpunkte der Projektbeteiligten sind so authentisch wie möglich wiedergegeben worden. Um die Stimmung im Projektverlauf zu zeigen und den Gesamteindruck zu vervollständigen, sind an vielen Stellen Zitate (*„kursiv gedruckt“*) von Projektbeteiligten in den Text eingefügt.

4.1 Ausgangslage und Projektziel

Der Auftraggeber betreibt sein Kerngeschäft mit einer 1987 vom Softwarehaus entwickelten Software (genannt „Alt-System“). Das Alt-System ist geplanten Kapazitätserweiterungen nicht mehr gewachsen und muß abgelöst werden. Zusätzlich will der Auftraggeber sein Kerngeschäft beträchtlich ausweiten und weitergehende Leistungen verkaufen. Dazu benötigt er eine neue Software (genannt „Gesamt-System“), das alle seine jetzigen und zukünftigen Geschäftsbereiche unterstützt.

Ab 1993 geht es um die Frage, welches Softwarehaus dieses Gesamtsystem erstellen soll. Das Softwarehaus, welches das Alt-System entwickelt hatte, erhält den Auftrag, einen kleinen Ausschnitt des neuen Geschäftsbereiches in einer eigenständigen Software zu realisieren (genannt „Teil-System“). Die Erstellung des Teil-Systems war als Test gedacht und sollte klären, ob das Softwarehaus in der Lage ist, auch das Gesamt-System zu erstellen. Das Teil-System geht Ende 1993 in Produktion. Trotz Terminüberschreitungen und Problemen bei der Programmeinführung, wird die Fertigstellung des Teil-Systems als Erfolg gewertet und das Softwarehaus beginnt Anfang 1994 mit der Realisierung des Gesamt-Systems.

Das zu entwickelnde Gesamt-System hat demnach drei wesentliche Ziele:

1. Das Alt-System soll abgelöst werden. Die alte Funktionalität soll übernommen werden, sie soll jedoch modernisiert, alte Schwächen ausgebügelt und zusätzliche Anforderungen realisiert werden.
2. Das Teil-System soll abgelöst und in die neue Software integriert werden.
3. Das Gesamt-System soll neue Geschäftsbereiche unterstützen, die von den beiden laufenden Systeme nicht unterstützt werden.

Das Gesamt-System wird als Client-Server-Applikation erstellt. Die relationalen Datenbanken liegen zentral auf zwei leistungsstarken Unix-Maschinen. Die Anwender arbeiten verteilt auf ganz Deutschland an mehreren hundert PC's unter MS-Windows. Es ergibt sich die folgende Systemarchitektur:

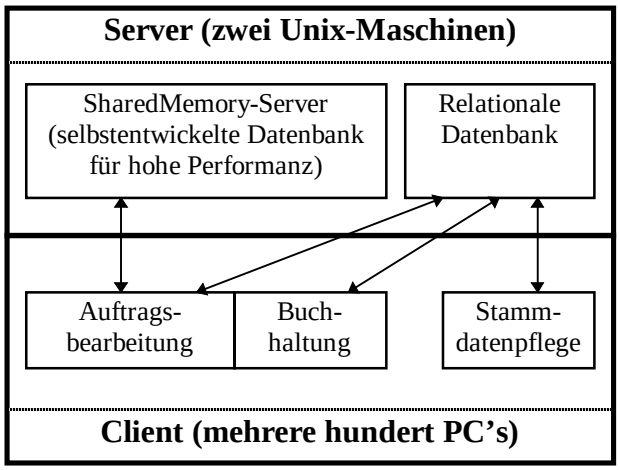

Abbildung 4-1

Das Entwicklerteam wird analog zu den in der Abbildung aufgeführten Bereichen in Teilteams aufgeteilt, d.h. es gibt ein Datenbank-Team, ein Client-Team, ein SharedMemory-Team, ein Buchhaltungsteam und ein Stammdaten-Team.

Das Projekt beginnt am 1.Januar'94 und soll im Sommer'95 beendet sein.

4.2 Der Projektverlauf im Überblick

Jan'94 - Juni'94	Ist-Analyse des Alt-Systems. Gespräche bei den Anwendern. Gesprächsprotokolle. Fachlicher Leistungsumfang. Meetings auf Abteilungsleiterebene. Parallel dazu Erstellung eines technischen Prototypen zur Evaluierung von bestimmten Datenbank-Konstruktionen und der Machbarkeit von Performanz-Anforderungen.
Juni'94-Sept'94	Konzeptphase. Freitagmeetings mit Anwendervertretern zur Vorstellung von Konzepten. Termin für Systemeinführung: Sommer'95.
Ab Sept.'94	Design und Implementierung von Datenbank und dem Shared-Memory-Server
Juli'94 - Sept'95	Anwender-Vertreter-Meetings (genannt AMT's).
Dez'94	Reduktion der geplanten Funktionalität unter Beibehaltung des Sommertermins.
Ab Jan'95	Einarbeitung neuer Entwickler auf der Client-Seite.
März'95	Terminverschiebung auf November'95 unter Hinzunahme der kompletten Funktionalität.
Nov./Dez.'95	Schulungen begonnen und vorzeitig abgebrochen.
Nov.'95-März.'96	„Heiße Phase": ständige Auslieferungen von neuen Programmversionen. Terminversprechungen und -verschiebungen. Hohe zeitliche Belastung der Entwickler. Kündigung/Entlassung von drei Entwicklern. Zwei weitere kündigen bis Juli.
8.April'96	Produktionsbeginn für einen Teilbereich vom Tagesgeschäft des Alt-Systems.
26.Mai'96	Produktionsbeginn für die Ablösung des Alt-Systems. Probleme beim Auftraggeber.
24.Juli'96	Krisensitzung beim Auftraggeber: Soll das Projekt abgebrochen werden? Das Projekt wird weitergeführt.

Abbildung 4-2

Anfang 1994 begann der Projektleiter des Softwarehauses mit der *Ist-Analyse* des Altsystems und der Erhebung der Anforderungen an das neue Gesamtsystem. Parallel dazu entstand ein technischer Prototyp, der die Machbarkeit von Performanz-Anforderungen evaluieren sollte.

Nach Abschluß der Ist-Analyse begann im Sommer'94 die *Konzeptphase*, an der neben dem Projektleiter des Softwarehauses und dem Projektleiter des Auftraggebers die späteren Leiter der Entwickler-Teilteams teilnahmen. Ziel dieser Phase, die in den Räumen des Auftraggebers durchgeführt wurde, war es, ein Grob-Design des Gesamt-Systems zu erstellen, was jedoch nicht erreicht wurde. Nach etwa drei Monaten endete die Konzeptphase und das Projektteam nahm seine Arbeit im Softwarehaus wieder auf. Neue Entwickler kamen zum Team hinzu.

Die ursprüngliche Zielvorgabe, bereits im Sommer'95 mit dem Gesamtsystem in Produktion zu gehen, stellte sich als nicht realisierbar heraus. Deshalb einigte man sich auf ein Vorgehen in mehreren Stufen. Zunächst sollten in *Stufe 1* das Altsystem und das Teilsystem abgelöst, die Unterstützung der neuen Geschäftsbereiche jedoch noch ausgeklammert werden. Für die Inproduktionsnahme der Stufe 1 wurde wiederum Sommer'95 angestrebt.

Auch dies erwies sich als unrealistisch. Es erfolgte eine Terminverschiebung auf November'95. Um diese Verschiebung durchzusetzen, mußte sich das Softwarehaus verpflichten, zu diesem Termin bereits das Gesamtsystem fertiggestellt zu haben. Auch dies wurde nicht erreicht. Im November wurden dann auf Auftraggeber-Seite Schulungen begonnen, die jedoch abgebrochen werden mußten, weil das Programm nur in Bruchteilen fertig war und instabil lief. Zu Beginn 1996 wurden die Schulungsaktivitäten wieder aufgenommen. Diverse Inproduktionsnahme-Termine wurden gesetzt und verschoben. Auch die Idee, gleich mit dem kompletten Gesamtsystem in Produktion zu gehen, wurde fallengelassen.

Anfang April'96 wurde das neue System für einen Teilbereich des Tagesgeschäftes in Produktion genommen. Das Alt-System lief parallel dazu weiter. Da es weniger Probleme gab als erwartet, wurde am 26.Mai die gesamte Produktion auf das neue System umgestellt. Jetzt stellten sich massive Probleme beim Auftraggeber ein. Das Programm hatte Performanz-Schwierigkeiten und war zudem instabil. Außerdem fehlte wichtige Funktionalität. Eine Vielzahl organisatorischer und technischer Probleme belastete das Arbeitsklima beim Auftraggeber.

Ende Juli gab es beim Auftraggeber ein Krisentreffen auf höchster Managementebene, wo die Frage erörtert wurde, ob das Projekt abgebrochen werden sollte oder ob man mit dem Softwarehaus weiterhin zusammenarbeiten wollte. Der Auftraggeber entschloß sich zunächst, die Zusammenarbeit fortzusetzen.

4.3 Projektresultat

Zum Zeitpunkt der Projektanalyse (August 1996) bietet sich dem Betrachter folgendes Bild:

Das entstandene Programm beinhaltet auf der Client-Seite in etwa 1300 Anwendungsklassen (entwickelt in Visual C++) und umfaßt ca. 300000 Lines of Code, nicht eingerechnet die etwa 500 Frameworkklassen, auf denen die Anwendungsklassen aufbauen (die Frameworkklassen wurden teilweise erst während des Projektes von einem anderen Entwicklerteam erstellt). Dazu kommen auf der Serverseite ca. 3000 Datenbank-Requeste, die als „Stored-Procedures" vom Client gerufen werden. Das Softwarehaus hat einen Aufwand von etwa 30 Mannjahren für die bisherige Entwicklung aufgewendet.

Das Projekt hat die ursprüngliche Zeitplanung um bisher 100% überzogen. Das Produkt wurde mit einer Verspätung von ca. einem Jahr ausgeliefert. Es hat in etwa doppelt so viel gekostet wie geplant. Der Funktionsumfang bei Auslieferung ist nur ein Teil des geplanten. Bis zur vollständigen Fertigstellung wird vermutlich ein weiteres Jahr vergehen. Die

ursprüngliche Zeitvorstellung und Kostenschätzung ist dementsprechend um einen Faktor 3 überschritten worden.

Auf Softwarehaus-Seite standen alle Projektmitglieder praktisch von Anfang des Projektes an unter enormem Zeitdruck und waren einer permanent hohen arbeitszeitlichen Belastung ausgesetzt, mit Spitzenwerten von 250 bis 300 Stunden pro Monat. Die überwiegende Zahl der Entwickler war während des Projektes mehr oder minder nahe an einer Kündigung. Fünf Entwickler haben die Firma während des Projektes verlassen oder wurden entlassen (vier davon waren erst für das Projekt neu eingestellt worden). Diese Fluktuationsrate ist durchaus nicht typisch im Vergleich zu früheren Projekten. Die Motivation ist bei praktisch allen Entwicklern im Projektverlauf immer geringer geworden. Erfolgserlebnisse fehlten oft gänzlich, Überstunden wurde nicht vergütet. Die Bereitschaft, in Zukunft eine ähnliche Belastung auf sich zu nehmen, ist bei den meisten gering.

Auch im privaten Bereich der Entwickler hatte das Projekt z.T. drastische Auswirkungen. Zwei Partnerschaften (darunter eine Verlobung) gingen bedingt durch das Projekt in die Brüche. Der „Beziehungs-Dispo“ ist bei den übrigen ausgeschöpft und wird kaum einer ähnlichen Belastung standhalten.

Das entstandene Programm ist von seiner „inneren“ Beschaffenheit in großen Teilen unzureichend konstruiert und wird von den Entwicklern als nur schwer erweiterbar und änderbar angesehen. Schon in der Zeit nach Produktionsbeginn hat man damit begonnen, Teile des Systems völlig neu zu entwickeln. Für die nahe Zukunft stehen wesentliche Erweiterungen an, für die es schon jetzt enge Termine gibt. Die personelle Zusammensetzung des Entwicklungs-Teams ist durch Kündigungen, aber auch durch Herausziehen von Mitarbeitern geschwächt und für diese Weiterentwicklung schlecht gerüstet. Es sind bereits jetzt Probleme angelegt, die mit hoher Wahrscheinlichkeit in der nahen Zukunft zu den gleichen Resultaten führen werden wie schon in der Vergangenheit.

Sowohl der technische als auch der fachliche Ausbildungsstand der Entwickler wird vom Auftraggeber und auch vom Softwarehausmanagement als nicht zufriedenstellend eingestuft.

Bei Produktionsaufnahme gab es beim Auftraggeber massive Probleme mit der neuen Software, die auch negative Auswirkungen auf die Kunden des Auftraggebers hatten. Die Folgen waren eine hohen Unzufriedenheit beim Kunden und ein verstärkter Druck auf das Softwarehaus.

Die Vertrauensbasis beim Auftraggeber gegenüber dem Softwarehaus hat im Projektverlauf stark gelitten. Das Softwarehaus hat durch die vielen Terminverschiebungen und -überschreitungen im Projektverlauf jegliche Glaubwürdigkeit verloren (Konsens beim Kunden: *„Das Softwarehaus hält keine Termine“*). Beim Auftraggeber hat man ernsthaft die Möglichkeit in Betracht gezogen, die Zusammenarbeit mit dem Softwarehaus endgültig zu beenden und das Projekt abzubrechen. Auf höchster Managementebene ist jedoch für eine weitere Zusammenarbeit entschieden worden, jedoch vermutlich nur deshalb, weil man von der neuen Software in hohem Maße abhängig ist und eine Neuentwicklung zu lange dauern würde und deshalb nicht mehr in Frage kommt.

Da die noch fehlende Funktionalität Teil des vertraglich vereinbarten Umfangs ist, wird die Weiterentwicklung des Produktes für das Softwarehaus ein Verlustgeschäft bleiben. Die vom Softwarehaus nachträglich durchgesetzten Budgeterhöhungen reichen nicht aus, um die Kosten zu decken. Einzige Perspektive für das Softwarehaus bleibt die Möglichkeit, das Produkt auch an andere Interessenten zu verkaufen.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Das Projekt hat dem Softwarehaus gravierende finanzielle, personelle und image-bezogene Verluste eingebracht. Beim Auftraggeber löste die Systemeinführung zum Teil chaotische

Zustände aus, führte zu finanziellen Einbußen und zu unzufriedenen Mitarbeitern und Kunden. Eine langfristige Zusammenarbeit zwischen Softwarehaus und Auftraggeber erscheint mehr als unwahrscheinlich.

Vor dem Hintergrund von Kundenorientierung muß folgendes festgehalten werden:

- Die Geschäftsbeziehung zwischen Softwarehaus und Auftraggeber hat sich durch das Projekt nachhaltig verschlechtert. Eine langfristige Kundenbindung scheint nicht erreicht worden zu sein.
- Das Softwarehaus konnte die meisten gegenüber dem Kunden gemachten Zusagen nicht einhalten. Dies betrifft die drei wesentlichen Bereiche Termine, Kosten und Funktionalität.
- Der Entwicklungsprozeß führte zu hoher Kundenunzufriedenheit, sowohl auf der Ebene des Managements auch auch der Anwender. Hier ist besonders die Phase der Schulungen und der Systemeinführung zu nennen.
- Im ausgelieferten Softwareprodukt scheint es sowohl Probleme mit der technischen Qualität (interne Programmbeschaffenheit), als mit der Gebrauchsqualität zu geben.
- Die im Rahmen von Kundenorientierung geforderte Mitarbeiterorientierung wurde anscheinend im Softwarehaus nur unzulänglich umgesetzt. Als Folge gab es hier eine hohe Unzufriedenheit im Entwicklungsteam, die nicht nur in ‘inneren’ Kündigungen mündete, sondern auch eine merkbar erhöhte Mitarbeiterfluktuation bewirkte.

Sowohl für das Softwarehaus als für den Kunden wird das Projekt negative Langzeitwirkung haben. Alles in allem muß man feststellen, daß das vorliegende Projekt nahe an einem Fehlschlag war bzw. als langfristiger Fehlschlag zu werten ist.

In den folgenden Abschnitten wird der Projektverlauf im Detail beschrieben und ausgewertet.

4.4 Der Projektverlauf im Detail

Die folgende Abbildung zeigt den Projektverlauf in detaillierterer Form und gibt gleichzeitig einen Überblick über die zeitliche Anordnung der verschiedenen Phasen und Tätigkeiten:

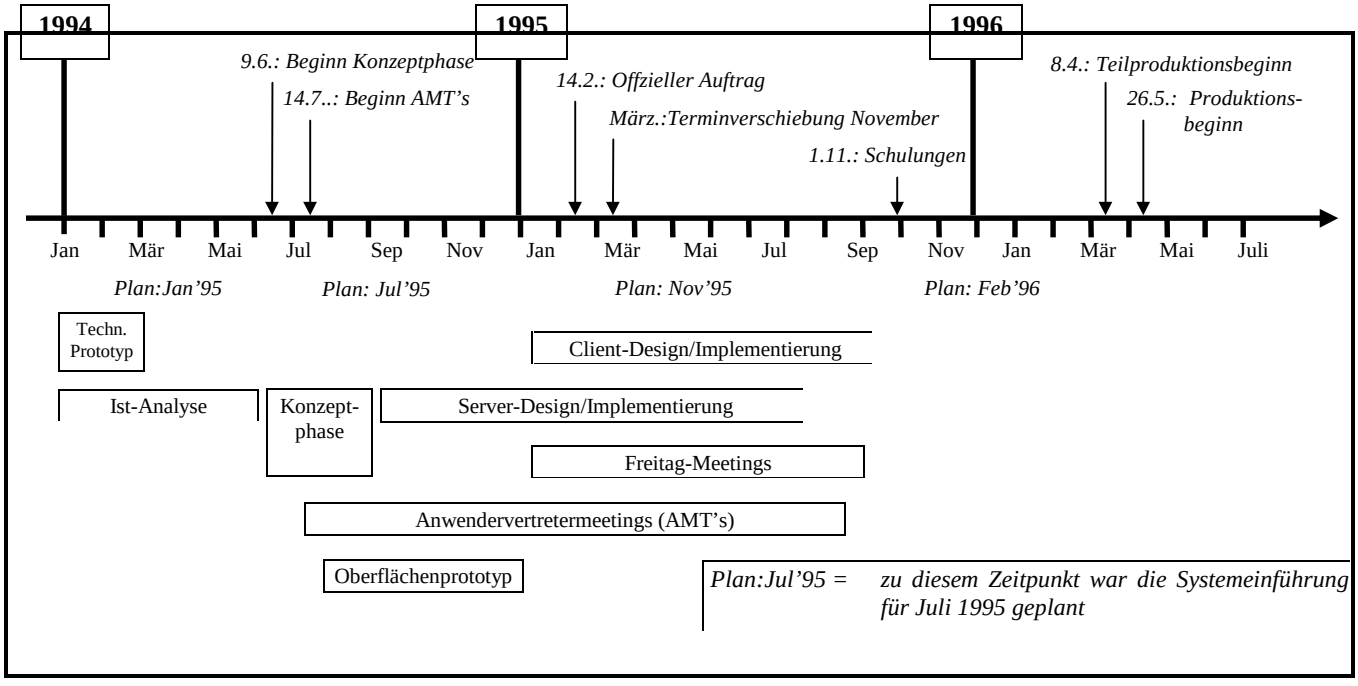

Abbildung 4-3

Wesentliche Elemente und Abschnitte des Projektverlaufes sollen im folgenden näher untersucht und vor dem Hintergrund von Kundenorientierung bewertet werden.

4.4.1 Die Vorgeschichte

Pläne, das neue Gesamt-System zu erstellen, gab es im Prinzip schon lange. 1987 hatte das Softwarehaus das Alt-System erfolgreich erstellt. 1989 hatten sich Auftraggeber und Softwarehaus im Unfrieden voneinander getrennt. Gleichwohl blieben persönliche Kontakte auf Managementebene erhalten. Als dann die Ablösung des Alt-Systems durch ein umfassenderes Gesamtsystem zur Debatte stand, meldete das Softwarehaus sein Interesse an einem solchen Auftrag an. Beim Auftraggeber gab es intern jedoch einige Vorbehalte gegen das Softwarehaus. Darüber hinaus gab es andere potentielle Mitbewerber. Außerdem konnte man beim Auftraggeber eine Entwicklung des Gesamt-Systems in einem Zuge nicht durchsetzen. Deshalb entschloß man sich, in Stufen vorzugehen. Mit der Realisierung des Teil-Systems, eines Ausschnittes des Gesamtsystems, wollte man auf Auftraggeberseite das Softwarehaus testen und gleichzeitig intern die Entwicklung des Gesamtsystems einen Schritt weiterbringen.

Schon bei der Realisierung dieses Teil-Systems hatte es erhebliche Probleme gegeben, jedoch wurde dieses Vorgängerprojekt insgesamt als Erfolg gewertet (vgl. [Fittkau94]). Viele der Vorbehalte des Auftraggebers gegenüber dem Softwarehaus blieben gleichwohl erhalten. Deshalb wurde dem Softwarehaus mit Ende des Teil-Systems auch nicht sofort der Auftrag über das Gesamt-System erteilt. Der Auftraggeber wollte sich Optionen offenhalten. Es erging zunächst ein Auftrag über die Ist-Analyse des bestehenden Alt-Systems. Man spekulierte darauf, daß die Ergebnisse dieser Ist-Analyse auch einem anderen Softwarehaus als Basis für eine Neuentwicklung hätten dienen können.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Das Projekt war bereits zu Beginn von der Vorgeschichte belastet. Das Vorgängerprojekt hatte beim Auftraggeber eine skeptische Grundhaltung gegenüber dem Softwarehaus zurückgelassen.

Vor dem Hintergrund von Kundenorientierung mußte es deshalb das vorrangige Ziel des Softwarehauses sein, das verlorene Vertrauen wieder zurückzugewinnen. Von den genannten Anforderungen an eine kundenorientierte Prozeßgestaltung sind hier besonders die Einhaltung von Zusagen und die Einbindung des Kunden in den Entwicklungsprozeß zu nennen (vgl. Abschnitt 3.6). Wie man jedoch sehen wird, versuchte das Softwarehaus ‘verlorenen Boden’ vorrangig durch überhöhte Versprechungen zurückzugewinnen, die dann im Projektverlauf nicht eingehalten werden konnten.

Eine starke Kundenbeteiligung lag im übrigen auch im Interesse des Kunden (allerdings bedingt durch Skepsis). Dieser wollte nicht nur eingebunden sein, um das Aussehen des Systems mit zu beeinflussen, sondern auch, um den erfolgreichen Fortgang zu überwachen und abzusichern.

Wie im folgenden noch zu sehen sein wird, bewirkte u.a. die skeptische Grundhaltung des Auftraggeber einen eher zögerlichen Projektanfang. Insbesondere das Herauslösen der Ist-Analyse als separater Auftrag erzwang eine Segmentierung des Projektverlaufes, die sich negativ auf die weitere Vorgehensweise auswirkte.

4.4.2 Die Ist-Analyse

Anfang 1994 reiste der Projektleiter des Softwarehauses durch die verschiedenen Niederlassungen des Auftraggebers und führte eine Vielzahl von Einzelgesprächen mit den Benutzern des Alt-Systems („einfachen“ Anwendern, Gruppenleitern, Abteilungsleitern, Management). Was ist gut am alten System? Was müßte verbessert werden? Was müßte

zusätzlich realisiert werden? Grundtenor war *„Ihr dürft träumen. Wie soll das neue System aussehen?“*.

Im Rahmen dieser Ist-Analyse entstanden eine Reihe von Gesprächsprotokollen, die dann auf Abteilungsleiterebene Grundlage für weitere Gespräche waren. Parallel dazu begann der Projektleiter mit der Erstellung des *„fachlichen Leistungsumfangs“*, einem Dokument, das die gesamten Anforderungen an das System enthalten sollte und auch als Vertragsgrundlage zwischen Auftraggeber und Softwarehaus diente (näheres dazu im Abschnitt 4.4.5).

Diese Phase wurde vom Auftraggeber insgesamt positiv aufgenommen. Die Anwender hatten das Gefühl, daß ihren Anliegen Interesse entgegengebracht wurde und daß ihre Wünsche in der neuen Software berücksichtigt werden würden.

Obwohl die Involvierung begrüßt wurde, scheint doch bei vielen der Interviewten Unklarheit über den Stellenwert der Gespräche in bezug auf das Gesamt-Projekt bestanden zu haben, schon allein deshalb, weil das eigentliche Projekt noch gar nicht begonnen hatte und es auch zu diesem Zeitpunkt weder dem Kunden noch dem Softwarehaus klar war, wer das Gesamtprojekt durchführen würde.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Es wurden direkte Gespräche mit den späteren Anwendern des Systems geführt. Insofern kann man die Ist-Analyse kundennah nennen. Die Vorgehensweise stieß beim Management des Auftraggebers auf Zustimmung. Es entstand eine hohe Erwartungshaltung sowohl an das neue System als auch in bezug auf eine direkte Beteiligung am Projekt. Beides wurde dann jedoch im Projektverlauf nicht oder nur zum Teil eingehalten.

Die geführten Gespräche wurden in Gesprächsprotokollen festgehalten, die an den Kunden geschickt wurden. Es gab jedoch nur spärliche Rückmeldung. Letztlich fehlte im Sinne eines „Autor-Kritiker-Zyklus“ eine kontinuierliche und konsequent betriebene Rückkoppelung mit denjenigen, die an den Gesprächen beteiligt waren. Eine direkte Rückkoppelung der Gesprächsprotokolle fand lediglich auf Abteilungsleiterebene statt. Dies muß deshalb kritisch bewertet werden, weil diese Gruppe wiederum eigene Interessen und Gestaltungsvorstellungen hat. Tatsächlich standen genau diese Interessen im Mittelpunkt der Treffen. Eine Evaluierung der Gespräche mit den Anwendern konnte so nur teilweise erfolgen.

Vor dem Hintergrund von Kundenorientierung muß diese Phase kritisch bewertet werden. Sie war eindeutig im Geiste einer traditionellen, phasenorientierten Vorgehensweise und hatte dementsprechend den Charakter von Informationsbeschaffung. Diese „Funktionalisierung“ des Kunden als Informationsgeber konnte nicht den Grundstein für eine erfolgreiche Integration des Kunden in den Entwicklungsprozeß legen. Im Gegenteil führte diese Phase zu falschen Erwartungen und zu mancherlei Verwirrung und Unklarheit bei vielen Vertretern des Kunden.

Die Ist-Analyse wurde nur von einer Person durchgeführt, was ebenfalls als nachteilig angesehen werden muß. Der ‘Know-How-Vorsprung’, den der Projektleiter durch die Ist-Analyse gewann, war von denen, die später zum Projektteam dazukamen, wenn überhaupt, nur schwer aufzuholen. Insbesondere fehlte dann bei den Entwicklern der direkte Bezug zur realen Anwendungswelt. Ein im Rahmen von Kundenorientierung gefordertes „Denken wie Kunden“ wurde auf diese Weise nicht erreicht.

Ein ‘Denken wie Kunden’ kann nicht über schriftliche Dokumente kommuniziert werden, sondern muß erfahren werden. Es fällt auf, daß diejenigen, die die Kernbereiche des Systems (Auftragserfassung auf der Client-Seite) schließlich entwickelten, nicht an dieser wichtigen Anfangsphase beteiligt waren. Noch heute wird sowohl vom Kunden als auch vom Softwarehaus-Management das mangelhafte fachliche Know-How vieler Entwickler kritisiert.

Es ist oben schon angemerkt worden, daß die Ist-Analyse im wesentlichen deshalb aus dem eigentlichen Projekt herausgelöst wurde, weil der Auftraggeber bedingt durch die Vorgeschichte nicht bereit war, ein 100prozentiges Commitment gegenüber dem Softwarehaus abzugeben. Damit stand mit der Ist-Analyse eine Phase außerhalb des Projektes, die für das Gesamtprojekt von zentraler Bedeutung hätte sein müssen. Das Projekt stand also bereits zu Beginn unter einem „schlechten Stern".

4.4.3 Der technische Prototyp

Parallel zur Ist-Analyse erhielt das Softwarehaus einen Auftrag über die Erstellung eines technischen Prototypen, der zeigen sollte, ob die zukünftigen Anforderungen (Durchsatz, Performanz) an das Gesamtsystem zu realisieren wären. Dieser Prototyp wurde erfolgreich in der vereinbarten Zeit fertiggestellt und brachte ein positives Ergebnis.

Der Prototyp wurde in der gleichen Architektur (Client-Server) wie das spätere Softwaresystem realisiert. Beim technischen Prototyp lag das Hauptaugenmerk auf der Server-Seite. Auf der Client-Seite wurde lediglich eine rudimentäre Oberfläche zur Steuerung des Servers realisiert. Der maßgebliche Entwickler des Server-Teils arbeitete auch im späteren Projekt an den Bereichen, die er im Prototyp realisiert hatte. Dies hatte positive Auswirkungen, da der Entwickler durch die Arbeit am Prototyp eine genaue Vorstellung über sein Aufgabengebiet gewonnen hatte. Diese Programmteile konnten dann verhältnismäßig plangerecht fertiggestellt werden.

Die Client-Seite „ruhte" nach Fertigstellung des Prototypen fast ein Jahr und wurde erst im Jan'95 von einem als Berufsanfänger neu zum Projekt hinzugekommenen Entwickler weitergeführt. Bedingt durch Unerfahrenheit und Termindruck versuchte der Entwickler, seinen Bereich durch Erweiterung des technischen Prototypen zu realisieren. Dieser Versuch scheiterte. Ende'95 wurde der Entwickler schließlich entlassen. Seine Programmteile mußten unter enormem Zeitdruck zu großen Teilen neu implementiert werden.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Der technische Prototyp kann mit Blick auf das Gesamtprojekt tendenziell positiv gewertet werden. Er wurde zum „richtigen Zeitpunkt" realisiert und hatte ein überschaubares Ziel. Er wurde in der geplanten Zeit fertiggestellt und brachte die geforderten Resultate.

Positiv wirkten hier vor allem die personelle Kontinuität auf der Server-Seite und das relativ zügige Überführen des Prototypen in das Zielsystem. Man kann erkennen, daß der Prototyp neben der Klärung technischer Fragen vor allem beim Entwickler der Server-Seite einen Lernprozeß initiiert hatte, der sich positiv auf seine weitere Arbeit auswirkte.

Negativ muß die Tatsache bewertet werden, daß ein entsprechender Lernprozeß auf der Client-Seite in doppelter Weise abgeschnitten wurde. Hier war zum einen die personelle Kontinuität nicht gewährleistet, und zum anderen lag zwischen Fertigstellung des Prototypen und Realisierung der entsprechende Teile im Zielsystem ein zu langer Zeitraum. Erschwerend kam noch die Unerfahrenheit des Client-Entwicklers hinzu. Einerseits muß man hier also die fehlende Einsicht in die Bedeutung von Prototypen für den Lernprozeß kritisieren, andererseits die mangelhafte Unterstützung bzw. Integration neuer und unerfahrener Entwickler ins Entwicklerteam.

Zusammenfassend kann man sagen, daß durch den technischen Prototypen die Vorteile von Prototyping nur teilweise für das Projekt genutzt wurden. Auf den weiteren Einsatz von Prototyping in diesem Projekt wird in Abschnitt 4.4.8 eingegangen.

4.4.4 Die Konzeptphase

Nach Beendigung der Ist-Analyse kamen auf der Seite des Softwarehauses diejenigen Entwickler zum Projektteam dazu, die die Teilbereiche des Systems als Teamleiter realisieren sollten. In den Räumen des Auftraggebers sollten nun Konzepte und Grob-Design für das Gesamtsystem erarbeitet werden. Diese Phase sollte als Basis für das gesamte Projekt dienen.

Ursprünglich hatte der Auftraggeber gefordert, das Projekt als In-House-Projekt in den eigenen Räumlichkeiten durchzuführen. Das Softwarehaus hatte dies abgelehnt. Man fürchtete, daß es durch die räumliche Nähe zu den Anwendern zu ständigen Störungen der Entwickler kommen würde. Man einigte sich darauf, lediglich die Konzeptphase beim Auftraggeber durchzuführen.

Die Gespräche während der Konzeptphase waren von hitzigen Diskussionen und persönlichen Animositäten zwischen einzelnen Projektbeteiligten geprägt (*„Was allen noch in Erinnerung ist, ist die Gesprächskultur, die nicht vorhanden war. Es wurde öfter mal geschrien. Es ging recht unzivilisiert zu.“*).

Jeder hatte hohe Erwartungen an diese Phase gehabt, die jedoch nicht in Bruchteilen erfüllt werden konnten. Es gab zu wenig konkrete Resultate (*„Es war insofern ergiebig, daß man durch die gesamte Thematik durchgegangen ist, aber es war nicht ergiebig in Form von Ergebnissen. Es haben sich alle wesentlich mehr davon versprochen.“*).

Daß die Erwartungen nicht erfüllt werden konnten, lag auch daran, daß das Gros der Teilnehmer zunächst einmal auf den Wissensstand des Projektleiters gehoben werden mußte, den dieser durch die Durchführung der Ist-Analyse gewonnen hatte. Vorher war nicht daran zu denken, mit einem Design für das neue System zu beginnen. (*„Der Projektleiter hatte die Gespräche mit den Anwendern geführt und kannte ja auch das alte System. Da sind dann auch viele Dinge reingeflossen, die er für selbstverständlich hielt, die aber nicht selbstverständlich waren. Für uns nicht. Das war sehr schwierig. Man konnte diesen Informationsstand in der Zeit nicht aufholen. Das war gar nicht möglich.“*)

Der Projektleiter und der Datenbank-Teamleiter verfügten zudem aus vergangenen Projekten über eine gemeinsame Wissensbasis, anhand der sie jetzt das neue System konstruieren wollten Dieser Wissensvorsprung einzelner schloß viele der Teilnehmer von den Diskussionen aus. Innerer Rückzug scheint eine Reaktion gewesen zu sein (*„Ich habe zu dem ganzen dann nichts mehr gesagt.“*).

Obwohl ausdrücklich ein objektorientiertes Vorgehen für Analyse und Design angestrebt war, verfügten nur wenige der Teilnehmer über praktisches objektorientiertes Know-how (*„Wir hatten ziemliche Sprachschwierigkeiten. Sowohl von der Terminologie, als auch von der Auffassung, was Analyse ist, was Objektorientierung ist. Ich hatte eine gewisse Erfahrung mit Objektorientierung, und ich wollte das jetzt auch umzusetzen. Aber ich sah, daß meine Erfahrung nicht auf fruchtbaren Boden traf, sondern eher Verwirrung auslöste. Es gab keine Ressonanzebene, wo ich das Gefühl gehabt hätte, man versteht, was ich sage.“*).

Allerdings verfügten besonders die maßgeblichen (bzw. tonangebenden) Teilnehmer über z.T. viel Erfahrung in relationaler Datenhaltung, was sich prägend auf die Gespräche auswirkte.

Die Konzeptphase wurde Ende September abrupt beendet und alle Beteiligten waren froh darüber (*„Und dann hörte diese Konzeptphase schlagartig auf. Wir waren es endgültig leid, sinnlose Diskussionen miteinander zu führen.“*).

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Rückblickend kann man sagen, daß die Entscheidung des Softwarehauses, das Projekt nicht beim Auftraggeber durchzuführen, ein folgenreicher Fehler war. Der Wunsch des Auftraggebers, das Projekt bei sich durchzuführen, resultierte natürlich primär aus Mißtrauen

und einem daraus folgenden Kontrollbedürfnis. Trotzdem hätte das Softwarehaus darauf eingehen sollen. Obwohl die Gründe für eine Ablehnung in gewisser Weise verständlich sind, so liegt ihnen doch im Kern eine Haltung von 'Störenfried Kunde' zugrunde (vgl. Abschnitt 2.1). Natürlich hätte ein Inhouse-Projekt eine stärkere Interaktion mit den Anwendern bedeutet, aber genau darin hätte eine große Chance liegen können. Die gewählte Lösung, lediglich die Konzeptphase beim Kunden durchzuführen, muß als 'fauler Kompromiß' bezeichnet werden, besonders, wenn man sich vor Augen führt, daß an den Sitzungen nur ein Kundenvertreter beteiligt war.

Die Konzeptphase wurde von beiden Seiten ausdrücklich als Grundstein für das beginnende Projekt angesehen. Tatsächlich wurde hier der Grundstein gelegt - allerdings im negativen Sinne. Aus heutiger Sicht ist deutlich, daß viele der damaligen Fehler und Versäumnisse in späteren Phasen des Projektes gravierende negative Folgen hatten und sich oft wie ein „roter Faden" durch das Projekt zogen. Besonders sind in diesem Zusammenhang die folgenden Punkte zu nennen:

- Mangelhafte Diskussionskultur.
- Keine Teambildung,
- Ausschluß von wichtigen Beteiligten und Betroffenen.
- Kein konsequenter Einsatz von Objektorientierung.

Die Konzeptphase war aus Sicht des Softwarehauses der eigentliche Beginn des Projektes. Zum ersten Mal arbeitete man in einem größeren Team zusammen. Die durchweg als mangelhaft bezeichnete Diskussionskultur kann als Indiz dafür angesehen werden, daß eine erfolgreiche Teambildung nicht erreicht wurde. Gerade in dieser Phase wären Teamentwicklungsmaßnahmen nötig gewesen.

Nicht nur der Kunde war zum großen Teil von dieser Phase ausgeschlossen, sondern auch ein Gros der späteren Entwickler. Die Zusammensetzung der Teilnehmer der Konzeptphase entsprach nicht der späteren Projektrealität. Das Datenbank-Team bestand im Projektverlauf aus vier Entwicklern, von denen zwei an der Konzeptphase teilnahmen, wohingegen von der Client-Seite lediglich ein Entwickler an der Konzeptphase teilnahm, obwohl dieses Team später aus bis zu neun Personen bestand. Deshalb verwundert es nicht, daß gerade im Client-Team während des Projektverlaufes die massivsten Probleme auftraten.

Vor dem Hintergrund von Kundenorientierung ist also im wesentlichen zu bemängeln, daß keine Teamentwicklungsmaßnahmen durchgeführt wurden und daß zu wenig Anstrengungen unternommen wurden, den Kunden ins Entwicklungsteam einzubinden (vgl. Abschnitt 3.6.3).

Der mangelhafte Einsatz von Objektorientierung sei hier ebenfalls erwähnt. Die an der Konzeptphase beteiligten Personen hatten nur wenig praktische Erfahrung mit Objektorientierung, wohl aber z.T. viel Erfahrung mit relationaler Datenhaltung. Wie noch zu sehen sein wird, resultierte dies in einer 'datenbank-lastigen' Denkweise, die auch dazu führte, daß auf der Client-Seite nur mangelhaft objektorientiert entwickelt wurde. Die Vorteile der Objektorientierung (wie z.B. Wiederverwendung und Änderbarkeit) kamen deshalb im späteren Projektverlauf nicht zum Tragen. Insofern hatte selbst eine eher technisch anmutende Problematik direkten negativen Einfluß auf den Ausgang des Gesamtprojektes und damit auch auf die Umsetzung von Kundenorientierung.

4.4.5 Der „Fachliche Leistungsumfang"

Der sog. „Fachliche Leistungsumfang" als zentraler Bestandteil der Vorgehensweise verdient an dieser Stelle besondere Aufmerksamkeit. Dieses Dokument entstand im Zuge der Ist-Analyse und wurde über den gesamten Projektverlauf (in der Regel vom Projektleiter) gepflegt und erweitert. Der „Fachliche Leistungsumfang" war im Projektverlauf nicht nur die

wichtigste Kommunikationsbasis zwischen Softwarehaus und Auftraggeber, er sollte auch den neu zum Projekt kommenden Entwicklern als Einarbeitungsgrundlage dienen.

Die Arbeit mit dem fachlichen Leistungsumfang erwies sich im Projektverlauf jedoch als schwierig. Er wurde recht schnell sehr umfangreich und hatte schließlich den beachtlichen Umfang von 600 Seiten erreicht (*„Das hat einen schon vom Umfang her erschlagen.“*).

Sowohl bei den Entwicklern als auch bei den Vertretern des Auftraggebers ist der Leistungsumfang, wenn überhaupt, nur von wenigen und dann auch nur unter Mühen gelesen worden (*„Ich habe mich da durchgequält.“*).

Im Softwarehaus war bekannt, daß der Leistungsumfang nur von wenigen gelesen wurde. Dies wurde besonders vom Management als Problem angesehen. Den Entwicklern wurde wiederholt gesagt, daß sie *„doch mehr im Leistungsumfang lesen sollten“*. Diese Anweisung wurde jedoch anscheinend nicht beherzigt.

Die Entwickler sagten später übereinstimmend, daß sie die wesentlichen Informationen für die Realisierung des Systems nicht aus dem Leistungsumfang, sondern im direkten Gespräch mit dem Projektleiter oder dem Datenbank-Teamleiter erhalten hätten.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Das Problem des Leistungsumfanges war in erster Linie nicht, daß er von den Beteiligten nicht gelesen wurde. Das Problem war, daß die wirklich relevanten Informationen nicht im Leistungsumfang zu finden waren und daß er deshalb nicht oder nur widerwillig gelesen wurde.

Die Auflistung von Anforderungen bringt für die Entwickler wenig, wenn die Arbeitsumstände nicht bekannt sind, die diese Anforderungen erst haben entstehen lassen. Auf Auftraggeberseite schließlich war der Leistungsumfang nicht geeignet, einen Gesamteindruck vom entstehenden System zu vermitteln. Eine frühzeitige Qualitätskontrolle durch den Kunden konnte so nicht stattfinden (vgl. Abschnitt 3.6.2).

Aus der Sicht von Kundenorientierung muß deshalb besonders das Fehlen von kundenorientierten Dokumenttypen kritisiert werden (vgl. Abschnitt 3.7.2.1). Der hier beschriebene „Fachliche Leistungsumfang“ hatte den Charakter eines Pflichtenheftes. Er war zu umfangreich, nicht in der Sprache des Kunden geschrieben und konnte keine ganzheitliche Sicht auf die Arbeitszusammenhänge in der Anwendungsorganisation vermitteln. Eine Integration des Kunden wurde damit nicht unterstützt bzw. sogar verhindert.

4.4.6 Design- und Einarbeitungsphase

Nach der Konzeptphase trennten sich die Wege der Mitglieder des Entwicklerteams. Der unbefriedigende Verlauf der Konzeptphase führte dazu, daß jeder im Grunde froh war, diese Phase hinter sich zu haben, um nun „endlich in Ruhe“ arbeiten zu können. Eine Folge war, daß die einzelnen Teilteams zunächst relativ isoliert voneinander arbeiteten, was besonders im Buchhaltungsbereich bei der späteren Systemintegration zu enormen Schwierigkeiten führte.

Der sog. ‘SharedMemory-Server’ (eine selbstentwickelte Datenbank, um die hohen Performanz-Anforderungen zu realisieren), der bereits innerhalb des technischen Prototypen angelegt war, wurde entworfen und implementiert. Wie schon oben erwähnt, hatte der Entwickler durch seine Arbeit am Prototyp eine genaue Vorstellung von der zu realisierenden Aufgabe (*„Im Grunde habe ich mich noch eine Woche hingesetzt und Design gemacht. Das war nicht so schwierig. Ich hatte den Prototyp gebaut und hatte eine sehr klare Vorstellung, wie das laufen muß. Und dann haben wir einfach angefangen. Das war eine sehr konstruktive Arbeit. Was Spaß gemacht hat, was vorangegangen ist und was funktioniert hat.“*).

Der SharedMemory-Server war im März'95 lauffähig. Als er fertig war, stellte der Entwickler mit Erstaunen fest, daß es noch keine Programmteile gab, die ihn benutzen wollten bzw. konnten. Die zentralen Client-Teile waren noch nicht einmal begonnen worden.

Die beiden maßgeblichen Datenbank-Designer (die beide an der Konzeptphase teilgenommen hatten) begannen mit dem detaillierten Design der Datenbank. Neue Mitarbeiter kamen zum Datenbank-Team dazu. Relativ schnell begann man mit konkreter Implementierung von Zugriffs-Requesten. Auch hier gab es keine Programmteile auf Client-Seite, die diese Requeste hätten benutzen können.

Die gravierendste Folge war, daß die Datenbank über das gesamte Projekt am weitesten fortgeschritten war. In jedem Fall war sie immer weiter, als die Client-Programmteile. Deshalb gab es für die Datenbank-Entwickler auch keine Ansprechpartner auf der Seite, die die Datenbank benutzen sollte.

Als die Client-Entwickler dann soweit waren, Requeste der Datenbank zu benutzen, orientierten sie sich zwangsläufig an dem, was dort bereits fertig implementiert vorlag. Es ist nicht verwunderlich, daß dies zu einer datenbankorientierten und damit mangelhaften objektorientierten Konstruktion führte.

Von den Client-Entwicklern hatte nur der designierte Teamleiter an der Konzeptphase teilgenommen. Nach Ende der Konzeptphase war er nicht nur in die Anwender-Meetings involviert, sondern mußte noch bis Ende'94 grundlegende Erweiterungen am Teilsystem (welches im Vorgängerprojekt entstanden war) implementieren, für das er ebenfalls zuständig war. Folge war, daß sein Hauptaugenmerk gar nicht auf dem Design des neues Systems, sondern in der Pflege des Teilsystems lag. Durch diese Doppelbelastung wurde der „Rückstand" der Client-Seite gegenüber der Datenbank-Seite noch verstärkt (*„Da sind dann viele Monate ins Land gegangen, die das Ganze im Nebel verschwinden ließen, was in der Konzeptphase gelaufen war. Es wurde auf sehr kleiner Flamme gekocht."*).

Zum Anfang des folgenden Jahres kamen zunächst drei Entwickler zum Client-Team dazu, die allesamt als Berufsanfänger unerfahren waren. Sie mußten sich zunächst einarbeiten und bekamen dann die Aufgabe, 'einfache' Stammdatenpflege-Dialoge zu entwickeln. Aus der zunächst als überschaubar und einfach eingestuften Stammdatenpflege wurde dann jedoch ein äußerst komplexes und umfangreiches Programmteil, das sich im Projektverlauf immer mehr zu einem *„Faß ohne Boden"* entwickelte.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Die Konzeptphase hatte im Entwicklerteam wenn auch unproduktive, so doch zumindest teilteamübergreifende Gespräche beinhaltet. Die fehlende Teambildung während der Konzeptphase führte jedoch dazu, daß die Teilteams nach der Konzeptphase voneinander isoliert arbeiteten. Die Tatsache, daß die Teilteams unterschiedlich weit fortgeschritten waren, verstärkte diese Isolationstendenz noch weiter. In späteren Phasen kam es dann - wenig überraschend - zu enormen Integrationsproblemen zwischen den Teilteams und somit zu unnötigen Reibungsverlusten in einem ohnehin schwierig zu bewältigenden Projekt.

In bezug auf Kundenorientierung muß hier besonders auf die Tatsache hingewiesen werden, daß sich die Aufteilung des Entwicklerteams in Teilteams an technischen Gegebenheiten orientierte. Es gab eine Client-Seite und eine Server/Datenbank-Seite. Damit waren die Teams nicht kundenorientiert zugeschnitten. Erst dieser Umstand machte es möglich, daß die Server-Seite der Client-Seite immer weit voraus war. Man muß sich vor Augen halten, daß die Datenbank-Entwickler, um die Datenbank realisieren zu können, ein genaues Bild vom späteren Anwendungssystem haben mußten. Dieses Bild konnte jedoch zum Zeitpunkt seiner Entstehung weder von anderen Teilteams, geschweige denn vom Kunden überprüft werden. Wenn die Teilteams kundenorientiert gebildet worden wären, dann wären Client-Entwickler

und Datenbank-Entwickler aufgabenorientiert in einem Teilteam zusammengefaßt worden. Ein isoliertes Vorgehen wäre damit verhindert worden.

Die überraschende Komplexität der Stammdatenpflege schließlich führte dazu, daß über eine lange Zeit eine Vielzahl Entwickler dort gebunden waren und erst spät in den zentralen Bereichen mithelfen konnten. Hier deutete sich bereits früh an, daß die Komplexität des Gesamtsystems wesentlich größer sein würde, als erwartet.

4.4.7 Die Anwendervertreter-Meetings (kurz AMT's)

Noch während der Konzeptphase begannen im Juli'94 die Anwendervertreter-Meetings (genannt AMT's). Hier wurden in Runden von 7-12 Personen (mit teilweise wechselnder Besetzung) Anforderungen an das System besprochen bzw. fertige Programm- oder Prototyp-Teile präsentiert. Vom Kunden nahmen 3-4 Anwendervertreter teil, dazu der Projektleiter und der Leiter der Schulungs- und Testabteilung. Die Anwendervertreter waren erfahrene Mitarbeiter, die das Tagesgeschäft gut kannten und für Sonderaufgaben zuständig waren. Ständige Teilnehmer des Softwarehauses waren der Projektleiter und der Client-Teamleiter. Später nahmen auch andere Entwickler an den Treffen teil, wenn fertig entwickelte Programm- oder Prototyp-Teile präsentiert wurden.

Die Ergebnisse der Gespräche wurden in Gesprächsprotokollen (sog. *„AMT-Protokolle"*) festgehalten und bis Ende'94 teilweise in Form eines Oberflächen-Prototyps umgesetzt.

Bei den Treffen fiel auf, daß die Anwender stark im alten System und in ihren gewohnten Arbeitsweisen verwurzelt waren. Ein Beispiel dafür waren die Diskussionen über die „Abschaffung" der Enter-Taste. Im zugrundeliegenden Fenstersystem (MS-Windows) hat die TAB-Taste die Funktion, von Feld zu Feld zu springen. Die Anwender waren jedoch vom Alt-System daran gewöhnt, daß diese Funktionalität auf der Enter-Taste liegt. Im Zuge der Neuentwicklung wollte das Softwarehaus auf den Windows-Standard zurück, weil jeder Nicht-Standard zusätzlichen Aufwand bedeutet und potentiell fehlerträchtiger ist. Deshalb propagierte das Softwarehaus in Diskussion den Windows-Standard. Die Anwender sahen ihre gewohnten Arbeitsabläufe in Gefahr und pochten auf eine maßgeschneiderte Software (*„Was kümmert uns der Windows-Standard"*). Für die Anwender waren die technischen Gründe des Softwarehauses nicht nachvollziehbar. Die Standpunkte verhärteten sich, und es wurde viel Zeit mit hitzigen Diskussionen verbracht.

Von den Anwendern scheinen wenig konstruktive Vorschläge für die Gestaltung des neuen Systems gekommen zu sein (*„Wenig kreative Ideen, wie man es hätte anders machen können, deshalb fand ich Anwender-Beteiligung fraglich."*).

Außerdem konzentrierten sich die Gespräche stark am zentralen Bereich des Systems, der Auftragserfassung. Andere (ebenfalls wichtige Bereiche) gerieten in den Hintergrund (*„Heute ist die Auftragserfassung der Programmteil, der mit am besten funktioniert. Im Gegensatz zur Buchhaltung, die wenig zur Sprache kam, und die heute mit die meisten Schwierigkeiten macht und am umständlichsten ist."*).

Die AMT's waren sehr einseitig geprägt. Das Softwarehaus war in der Rolle desjenigen, der Lösungen präsentieren mußte, die dann von den Anwendervertretern angenommen oder abgelehnt wurden (*„Wir haben Vorschläge gemacht, und die haben zugestimmt oder abgelehnt. Der kreative Anwender fehlte."*).

Während die Vertreter des Softwarehauses in technischen Realisierungs-Überlegungen steckten, waren die Anliegen der Anwender viel konkreterer Natur. Sie wollten schlichtweg oft wissen, ob bestimmte Dinge auch im neuen System funktionieren würden (*„Die Leute waren nicht in der Lage, diese tiefergehenden Konzepte zu verstehen. Sie fragten 'Geht diese oder jene Sache auch im neuen System?' Wenn wir bejahten, waren sie zufrieden."*).

Es schien für die Anwendervertreter unklar zu bleiben, wie das neue System aussehen würde. Auch wenn die vom Softwarehaus vorgestellten Lösungen von den Anwendervertretern akzeptiert wurden, so schien das doch keine Garantie für die Richtigkeit dieser Lösung gewesen zu sein (*„Das Problem ist nur, daß man dem Anwender etwas zeigen kann und man 'den Segen kriegt', und daß es später doch nicht handhabar ist. Das ist unbefriedigend, weil man sich fragt, wieso man es ihnen nicht nahebringen konnte.“*).

Die Gesprächsprotokolle, die nach den Sitzungen erstellt wurden, scheinen insbesondere von den Anwendervertretern nicht oder in nicht ausreichendem Maße gelesen worden zu sein (*„Die Protokolle kamen recht unpünktlich. Und sie wurden nicht gelesen. Deshalb konnte auch nicht im Folgemeeting darüber diskutiert werden.“*). Ein Grund dafür lag auch in der Tatsache, daß die Anwendervertreter neben den AMT's weiterhin ihrer normalen Arbeitstätigkeit nachgehen mußten, also mit einer Doppelbelastung fertig werden mußten. Letzteres unterstützt die Vermutung, daß die Bedeutung dieser Treffen auch vom Auftraggeber nicht in vollem Umfang erkannt worden war.

Ungefähr im September'95 endeten die AMT's. Das Softwarehaus war zunehmend in Schwierigkeiten mit zusätzlichen Anforderungen an das System gekommen (Tenor war *„Wir können nicht dauernd neue Sachen erfinden. Wir müssen jetzt mal anfangen.“*).

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Aus der Sicht des Kunden stellten die AMT's das wichtigsten Medium für Beteiligung und Einflußnahme am Projekt dar. Trotzdem scheint der Wert dieser Treffen gerade in bezug auf wechselseitige Kooperation und die Etablierung einer gemeinsamen Vision über das zukünftigen System eher gering gewesen zu sein.

In bezug auf Kooperation ist festzuhalten, daß herkömmliche Rollenverständnisse nicht aufgelöst werden konnten. Das Softwarehaus stellte seine Lösungen vor, der Kunde konsumierte. Die Beobachtung, daß selbst die Akzeptanz von bestimmten Lösungsvorschlägen keine Garantie für ihre Sinnhaftigkeit war, legt die Vermutung nahe, daß beim Kunden keine Vision des zukünftigen Systems entstanden war, vor dessen Hintergrund eine konstruktive Auseinandersetzung mit den Vorschlägen des Softwarehaus hätte erfolgen können (vgl. Abschnitt 3.6.3).

Die endlose Diskussion um Detailfragen (hier exemplarisch am Beispiel der Enter-Taste gezeigt) deutet darauf hin, daß beim Kunden eine grundsätzliche Unsicherheit bestand, die nicht artikuliert werden konnte. Der Streit um Details erscheint mir hier eher Symtomcharakter gehabt zu haben. Kommunikationspsychologisch ausgedrückt: Hitzige Diskussionen auf der „Sachebene“ überdeckten grundsätzliche Probleme auf der „Beziehungsebene“ (vgl. [SchulzvonThun81]). Letztere hätten jedoch vordringlich gelöst werden müssen.

Aus der Sicht der Entwickler erscheinen diese 'unnützen' Diskussionen als Störungen ('Störenfried Kunde'), was wiederum in einem Teufelskreis dazu führt, Kundenbeteiligung negativ zu bewerten und zu reduzieren.

Ähnlich ist es auch zu bewerten, daß die Gesprächsprotokolle von den Vertretern des Kunden nicht oder nur wenig gelesen wurden. Dies hatte meiner Einschätzung nach im wesentlichen die folgenden Ursachen:

- Die Kundenvertreter hatten zu wenig Zeit, da sie neben den AMT's ihrer normalen Beschäftigung nachgehen mußten. Insofern stellte die Mitarbeit am Projekt eine Zusatzbelastung dar.
- Bei den Kundenvertretern war es im Projektverlauf mehr und mehr zu einer resignativen Haltung von 'Das Softwarehaus wird schon wissen, was es tut' bis hin zu 'Ich kann hier sowieso nichts ausrichten' gekommen.

Zusammenfassend ist festzuhalten, daß die Anwendervertretermeetings gerade unter dem Aspekt der Kundenintegration nur wenig bewirkt haben.

4.4.8 Der Oberflächen-Prototyp

Im Zuge der Anwendervertreter-Meetings entstand ein Oberflächen-Prototyp (in Visual-Basic) zur schnellen Umsetzung der Gesprächsergebnisse. Hiermit wurde im wesentlichen die Funktionalität des Kerngeschäftes des Auftraggebers abgebildet.

Der Prototyp wurde in den Räumen des Auftraggebers erstellt. Die beiden Entwickler nahmen an den AMT-Meetings teil und präsentierten ihn dort regelmäßig. Allerdings waren beide Entwickler ausschließlich mit der Entwicklung des Prototypen befaßt und nahmen nicht an der späteren Systementwicklung teil.

Die Entwicklung des Prototypen wurde Anfang 1995 mit Beginn der Client-Entwicklung eingestellt. Der Prototyp sollte den Entwicklern als Anschauungsmaterial zur Verfügung stehen, ein Angebot, das jedoch praktisch nicht genutzt wurde. Interessanterweise gab es genau in dem vom Prototypen abgebildeten Bereich die schon oben beschriebenen gravierenden Probleme, die auch zur Entlassung des verantwortlichen Entwicklers und zur Neuimplementierung seiner Programmteile führten. Der Entwickler hatte gerade nicht den Oberflächenprototypen als Grundlage seiner Implementierung herangezogen, sondern die schon fertigen Teile des technischen Prototypen.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Positiv ist zu bewerten, daß der Oberflächenprototyp, ebenso wie der technische Prototyp, zum „richtigen“ Zeitpunkt erstellt, nämlich parallel zu den Anwender-Meetings, denen er als Anschauungsmaterial diente.

Allerdings scheint der Oberflächenprototyp in den Anwendermeetings schwerpunktmässig zu Präsentationszwecken benutzt worden zu sein. Eine Folge davon war, daß die Anwender eher über die Positionierung oder die Farbe einzelner Felder diskutierten, als über grundsätzlichere Fragen des gesamten Systems. Von den Anwendern wurde der Prototyp im Prinzip ‘abgenickt’, ohne daß eine wirkliche Vision des künftigen Systems entstehen konnte.

Es ist als schwerwiegender Fehler zu werten, daß die Entwickler des Prototypen vom übrigen Entwicklerteam getrennt waren und auch nicht die Aufgabe hatten, ihn sukzessive in das spätere Zielsystem zu überführen. Der Prototyp stand damit im Prinzip isoliert vom Rest des Projektes dar. Somit ist es auch nicht verwunderlich, daß er vom eigentlichen Entwicklerteam nicht als Anschauungsmaterial genutzt wurde.

Der Einsatz von Prototypen dient primär dazu, ein gemeinsames Verständnis des zukünftigen Systems entstehen zu lassen und den beidseitigen Lernprozeß anzuregen. Beides wurde hier nicht erreicht. Insbesondere der Lernprozeß, der bei den Prototyp-Entwicklern eingesetzt hatte, wurde abgeschnitten und stand dem Entwicklerteam nicht mehr zur Verfügung. Der Sinn des Oberflächenprototyps wurde hierdurch praktisch ins Gegenteil verkehrt.

4.4.9 Die Freitag-Meetings

Keine andere organisatorische Maßnahme hat ähnlich destruktive Wirkung auf die Entwickler gehabt, wie die sog. *Freitagmeetings*, die von Januar’95 bis Oktober’95 einmal wöchentlich im Softwarehaus stattfanden. Teilnehmer waren das gesamte Entwicklerteam, die Projektleitung, die Geschäftsführung und zwei Vertreter des Auftraggebers (Projektleiter und Schulungsleiter).

Ziel der Meetings war es, die Einhaltung der von den Entwicklern geschätzten Termine zu überwachen. Das Management des Softwarehauses und des Kunden wollten ähnliche

Erfahrungen wie im Vorgängerprojekt vermeiden, wo man von der zunehmenden Komplexität und den ständigen Terminverschiebungen überrascht worden war (vgl. [Fittkau94]).

Die Treffen gerieten schon frühzeitig zu einem Spießrutenlauf für die Entwickler (*„Es waren Demotivations-Sitzungen, wo Leute vorgeführt wurden.“*). In den ersten Meetings machten die Entwickler noch halbwegs ehrliche Aussagen, die jedoch häufig lauteten: *„Ich kann meine Schätzung nicht halten“*. Dies stieß bei der Geschäftsführung auf Unverständnis. Resultat war, daß die Entwickler schon relativ schnell begannen, die Situation zu beschönigen, um nicht als schwarzes Schaf dazustehen (*„Jeder wußte, daß einige gelogen haben, um ihre Ruhe zu haben. Wenn jemand gesagt hat, daß er Probleme hatte, dann wurde ihm gesagt, daß er mehr arbeiten solle.“*).

Um die Illusion aufrechtzuerhalten, ‘im Plan’ zu sein, haben die Entwickler Randbereiche aus ihrem Aufgabenbereich ausgegrenzt, neue Problemfelder, die erst während der Realisierung auftauchten, ignoriert bzw. verschoben und Bereiche wegdefiniert, die im Prinzip zum eigenen Bereich gehört hätten (*„Du läßt einfach irgendwas weg, weil es zu dem Zeitpunkt gar nicht sichtbar ist, ob es ganz fertig oder nur halbfertig ist. Es wurde einfach als fertig definiert. Damit fühlst Du Dich natürlich schlecht, weil Du auch denkst, daß es irgendwann rauskommt. Oder Du verschiebst Deine Aufwände einfach in andere Planungsposten. Weil die Zuordnung zu Planungsposten relativ unklar ist, weil es so viele sind. Es gab viele Ausweichposten.“*).

Außerdem wurde in der Regel bis zum letzten Moment gewartet, bevor man zugab, daß man seinen Termin nicht halten kann (*„Das konnte man auch daran sehen, daß man immer bis ein/zwei Tage vor Ende seiner Arbeit im Plan war, und dann hat man festgestellt ‘Ich brauche doch noch zwei Wochen’. Wenn es nicht mehr anders ging, kam man damit raus.“*).

Trotz der Schwierigkeiten hielt die Geschäftsführung des Softwarehauses an den Treffen fest (*„Der Chef ist in den Meetings sehr drastisch aufgetreten. Ich habe einmal mit ihm darüber diskutiert, und er war dann der Meinung, man müßte in dieser Phase das Projekt richtig treten und pushen, um zu vermeiden, daß es am Ende in eine Krise gerät.“*).

Letztlich mußten die versprochenen Termine dann doch immer wieder verschoben werden. Der Projektleiter des Auftraggebers prägte dazu den Begriff „General-Amnestie“ (*„Bei Überziehung von versprochenen Termine wurde mehrfach eine ‘General-Amnestie’ erlassen. Das Wort zeigt schon, wie gefangen wir waren und wie schrecklich das Klima war.“*).

Die Freitagmeetings sorgten für extrem schlechte Stimmung im Entwicklerteam. Der wahre Stand des Projektes wurde in keinster Weise transparent. Das Übermaß an Kontrolle führte zu Phänomenen, die noch am ehesten aus bürokratischen Systemen bekannt sind:

- Dienst nach Vorschrift
  (*„Ich machte eben nur das, was ich auf dem Plan hatte.“*).
- Nur ja keine Fehler machen, um nicht als Schuldiger darzustehen
  (*„Der Arbeitseinsatz auf der Client-Seite war bei allen in etwa gleich. Auch die Krankheitszahlen gingen runter. Es war so eine Art Gruppenzwang. Niemand wollte derjenige sein, an dem es scheitert.“*).
- Nur den eigenen Arbeitsbereich sehen.
- Keine ‘unnötige’ Kommunikation mit anderen, weil dafür keine Zeit ist.
- Polarisierung zwischen Projektleitung und Entwicklern.
- Unproduktives Konkurrenzdenken unter den Entwicklern
  (*„Diese Art von Planung fördert einfach das Konkurrenzdenken unter den Beteiligten. Wenn Du das Ziel hast, Deinen Plan nicht zu überschreiten, dann mußt Du einfach Sachen ausgrenzen.“*)

Es scheint sich auch negativ ausgewirkt zu haben, daß Vertreter des Auftraggebers an den Freitagsmeetings beteiligt waren. Daraus resultierte ein Mangel an Offenheit in bezug auf die Schwierigkeiten im Projekt. Es ging in diesen Treffen für das Softwarehaus eindeutig darum, „den Schein zu wahren". Beim Auftraggeber bezeichnet man die Treffen aus heutiger Sicht denn auch als *„Pseudo-Veranstaltungen zur Beruhigung des Kunden"*.

Trotzdem kann die Anwesenheit des Kunden nicht in erster Linie für die schlechte Stimmung verantwortlich gemacht werden. Auch gegenüber der Projektleitung und der Geschäftsführung wollten die Entwickler nicht zugeben, daß es im Projekt nicht lief. Versuche der Entwickler, in ihren Aussagen ehrlich zu sein, wurden nicht gehört bzw. durften nicht sein. Die Drohung 'Wochenendarbeit' sprach eine deutliche Sprache. Wer gibt schon zu, daß er nicht im Plan ist, wenn er fürchten muß, dann am Wochenende arbeiten zu müssen?

Die Freitagmeetings wurden in etwa bis September'95 durchgeführt (*„Und dann hörten die Freitagmeetings einfach auf. Sie hörten zunächst nicht auf, sie verloren einfach an Bedeutung. Das sah man daran, daß plötzlich kein Vertreter des Kunden mehr erschien und unser Chef auch nicht. Sie flauten ab und hörten dann irgendwann ganz auf. Aber ohne daß jemand definitiv sagte, daß sie jetzt aufhören. Sie fanden einfach nicht mehr statt. Es hat sie auch niemand vermißt."*).

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Die Freitagmeetings hatten zentrale Bedeutung für die Projektsteuerung. Sie sollten den Projektverlauf sowohl für das Softwarehaus als auch für den Auftraggeber kontrollierbar machen, bewirkten aber das genaue Gegenteil: Terminverzögerungen wurden aus den obengenannten Gründen nicht rechtzeitig genug erkannt und die Meetings verursachten ein zusätzliches Maß an Druck, besonders auf die Entwickler. Die Freitagmeetings machten den wahren Projektstand in keinster Weise transparent und verschärften ohnehin vorhandene Konflikte noch mehr, ohne eine Thematisierung dieser Konflikte anzuregen. Es ist deutlich, daß die Treffen in erster Linie für das Management bestimmt waren, jedoch in dieser Form und Intention zur Projektkontrolle nicht geeignet sind.

Es setzt sich ein Trend fort, der schon während der Konzeptphase deutlich war, nämlich die schlechte Stimmung und mangelnde kooperative Haltung innerhalb des Softwarehauses. Weiterhin kann man eine zunehmende Polarisierung zwischen Auftraggeber und Softwarehaus feststellen.

Besondere Aufmerksamkeit verdient in bezug auf Kundenorientierung die von einem Kundenvertreter stammende Bemerkung, die Freitagmeetings wären *„Pseudo-Veranstaltungen zur Beruhigung des Kunden"* gewesen. Aus einer Einrichtung, die ursprünglich für Projektsteuerung (gemeinsam mit dem Kunden) gedacht war, wurde ein 'Schein-wahren' des Softwarehauses. Die Tatsache, daß der Kunden dies durchschaute, macht deutlich, daß sich das Vertrauensverhältnis (und damit die Geschäftsbeziehung) zwischen Softwarehaus und Kunde denkbar schlecht entwickelt hatte (vgl. Abschnitt 3.3). Die Kundenorientierung des Softwarehauses war auf einem Tiefpunkt angelangt.

Bezogen auf das Gesamtprojekt kann man sagen, daß es eine Reihe von Alarmsignalen gab, aus denen eindeutig hervorging, daß es um das Projekt schlecht stand bzw. daß sich das Projekt unlösbar festgefahren hatte. Spätestens zu dieser Zeit hätten sich zumindest auf Softwarehausseite „alle an einen Tisch" setzen müssen, um die Probleme offen anzusprechen, wenn nötig, mit Unterstützung von „Außen". Aber auch eine Einbeziehung des Kunden wäre angezeigt gewesen. Dadurch wäre zwar zunächst 'viel Staub aufgewirbelt worden', mittel- und langfristig jedoch wären mit ziemlicher Sicherheit ein Großteil der Probleme entschärft worden. Auch die später erfolgten Kündigungen von Entwicklern hätten damit vermieden werden können.

Es wird deutlich, warum es für kundenorientierte Softwareprojekte von entscheidender Bedeutung ist, regelmäßige Projektreviews durchzuführen, in denen es nicht um die Reflexion der fachliche Probleme, sondern um die Reflexion des Prozesses selber, d.h. der Kooperation und Kommunikation zwischen Entwicklungsorganisation und Kunde, geht (vgl. Abschnitt 3.6.1 und 3.6.3).

4.4.10 Terminverschiebung auf November'95

Als das Projekt Anfang'94 begann, war als Fertigstellungstermin für das Gesamtsystem der Sommer'95 geplant gewesen. An diesem Termin wurde über lange Zeit festgehalten. Zunächst versuchte man noch, diesen Termin zu halten, indem man die bis dahin fertigzustellende Funktionalität reduzierte.

Im Frühjahr'95 zeichnete sich jedoch deutlich ab, daß der Sommer-Termin trotzdem nicht einzuhalten war. Man entschied sich, den Termin auf November zu verschieben. Dafür mußte sich das Softwarehaus jedoch verpflichten, bis dahin die gesamte Funktionalität fertigzustellen (*„Als sich abzeichnete, daß das mit dem Sommertermin nicht machbar war, hat man einen neuen Termin gemacht. Da hat man einen faulen Kompromiß gemacht, der nicht vernünftig war und in einer Nacht- und Nebel-Aktion entschieden wurde -- unter Einbeziehung der gesamten Funktionalität den Termin auf den 1.November zu verschieben. Klingt zunächst zwar fair, war aber irgendwie von Anfang an faul. Als ich es hörte, wollte ich es gar nicht glauben und habe es eher als Aprilscherz aufgefaßt. War aber dann einfach wahr. Es hatte einen Deal-Charakter. Man kann nicht einfach nur einen Termin verschieben, man muß auch etwas dafür bekommen.“*).

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Man konnte bereits zu diesem Zeitpunkt erkennen, daß die Komplexität des entstehenden Systems die bisherigen Planungen sprengen würde. Die Realisierung eines Ausschnitts des Gesamtsystems bot sich deshalb für das Softwarehaus an. De facto war dieser Ausschnitt jedoch nichts anderes als das bereits laufende Alt-System. Das war dem Auftraggeber zu wenig, denn dieser wollte für sein Geld „neue Funktionalität“ bekommen und nicht „nur“ die Ablösung seines Alt-Systems. Auch waren die Versprechungen des Softwarehauses von Anfang an wesentlich höher gewesen.

Es ist jedoch wichtig zu sehen, daß dann bei Produktionsbeginn (ein Jahr später als geplant!) tatsächlich nur die Funktionalität des Alt-Systems fertiggestellt war. Mit anderen Worten: Die Versprechungen hatten nichts bewirkt (außer den Druck auf die Entwickler weiter zu erhöhen). Die Strategie des Softwarehauses schien es weiterhin zu sein, den Kunden irgendwie „bei der Stange zu halten“. Dahinter stand die Angst, daß der Kunde sich dazu entscheiden könnte, das Projekt abzubrechen, wenn das Softwarehaus eine wahrheitsgetreue Darstellung des Projektstandes gegeben hätte. Rückblickend kann man vermuten, daß diese Angst unberechtigt war. Das Projekt war bereits zu weit fortgeschritten, als daß es hätte abgebrochen werden können.

Der Sommertermin, der jetzt nur unter Mühen verschoben werden konnte, war zu einer Zeit genannt worden, als das Softwarehaus noch nicht einmal in Ansätzen die Anforderungen an das neue System kannte. Dieser Termin hatte sich jedoch in den Köpfen des Kunden „festgesetzt“, und jede Abweichung davon war vom Softwarehaus nur unter Schwierigkeiten durchzusetzen. Dies deutet auf ein für Softwareentwicklung inhärentes Problem hin: Das, was die Akquise glaubt, versprechen zu müssen, um den Auftrag zu bekommen bzw. um einen Projektabbruch zu verhindern, läßt sich oft in der praktischen Umsetzung nicht realisieren. Die Folge davon ist immer dieselbe: Zunächst lastet auf dem Softwarehaus bzw. den Entwicklern ein immenser Druck, der zu neuen Terminverschiebungen und fehlerhaften

Produkten führt, woraus dann beim Auftraggeber massive Problemen bei der Programmeinführung resultieren. Letztlich werden also alle Beteiligte geschädigt.

Daraus lassen sich drei wichtige Rückschlüsse für eine kundenorientierte Softwareentwicklung ableiten, die im vorigen Kapitel als Anforderungen an eine kundenorientierte Prozeßgestaltung genannt wurden (vgl. Abschnitt 3.6.1 und 3.6.3):

- Erstens muß das Softwarehaus konsequent auf unrealistische Versprechungen verzichten. Diese sind i.d.R. nur kurzfristig kundenorientiert, nie langfristig.
- Zweitens muß die Planung und Steuerung von kundenorientierten Softwareprojekten immer dynamisch, d.h. am realen Projektstand orientiert sein (vgl. Referenzlinien im Abschnitt 3.7.2.1).
- Drittens muß der Kunde für die Probleme von Softwareentwicklung sensibilisiert werden. Dies geht letztlich nur über eine enge Kundeneinbindung in den Entwicklungsprozeß.

4.4.11 Die „heiße Phase“

Der November-Termin rückte zunehmend näher. Auf Auftraggeber-Seite sollten im November die ersten Schulungen des Systems anfangen. Deshalb wurde die Forderung des Kunden nach lauffähigen Programmteilen vehementer.

Ab Oktober begann das Softwarehaus dann, Programmteile auszuliefern, und es kam zu einer regelrechten Flut von regelmäßigen Auslieferungen. Bis Juli’96 (also in einem Zeitraum von nur 10 Monaten) wurden von den Entwicklern 160 interne Versionen fertiggestellt, wovon etwa 60 an den Kunden ausgeliefert wurden. Das bedeutet, daß im Schnitt alle 4-5 Tage, in besonders „heißen“ Zeiten sogar täglich, eine neue Version ausgeliefert wurde.

Ein positiver Nebeneffekt war, daß man im Entwicklerteam nach anfänglichen Schwierigkeiten das Problem der Systemintegration gut in den Griff bekam. Gravierender Nachteil jedoch war, daß das ständige Ausliefern neuer Versionen einen hohen Zeitaufwand bedeutete, ohne jedoch für die Entwickler die erhofften Erfolgserlebnisse zu bringen (*„Man sitzt da bis 5 Uhr morgens, nur um den Termin zu halten, und dann hört man, daß es beim Kunden doch nicht eingesetzt wird oder man hört nur Beschwerden.“*).

Die Auslieferung von Progammversionen führte darüber hinaus zu einen engeren Kontakt zwischen dem Testpersonal des Auftraggebers und den Entwicklern. Dies wirkte für die Entwickler oft störend, da sie ‘eigentlich’ weiterhin mit dem entwickeln nicht fertiger Systemteile beschäftigt sein sollten. Der Zeitaufwand, diese sog. „Altlasten“ zu beheben, lag ab dieser Zeit bei etwa 30-50%, ohne daß es in den Zeitschätzungen berücksichtigt wurde. Es ist deshalb nicht verwunderlich, daß das Entwicklerteam immer weiter in Zeitverzug geriet. An der Entwicklung der Arbeitszeiten kann man ab Oktober’95 eine deutliche Zunahme des Drucks erkennen.

Mit dem Beginn der heißen Phase hörten auch die unbeliebten Freitagmeetings auf und mit ihnen praktisch jede Projektkontrolle auf der Seite des Softwarehauses. Die Entwickler begannen, Tag und Nacht zu arbeiten (*„Jetzt begann die extrem fatale Phase. Da wurde versucht, Termine zu halten, die nicht zu halten waren.“).*

Insbesondere war es diese heiße Phase, die zu den späteren Kündigungen führte. Die meisten Client-Entwickler arbeiteten von Oktober’95 bis Januar’96 kontinuierlich über 250 Stunden pro Monat, manche sogar an die 300 Stunden. Wochenendarbeit war die Regel. Die Arbeitsbelastung war enorm und der Lohn dafür gering (*„Termine wurden vier-wöchentlich verlängert. Und der enorme Druck hielt an bis Februar. Keine Überstundenregelung. Manche sind einfach gegangen.*“). Die Auswirkungen auf das Privatleben der Entwickler tat ein übriges (*„Ich war am 1.Weihnachtstag da, ich war Silvester da. Heiligabend war ich nicht da*

und hatte ein schlechtes Gewissen deshalb. Ich habe meine Freundin oft tagelang nicht gesehen. Wenn ich nach Hause kam, hat sie schon geschlafen. Wenn es nach ihr ginge, hätte ich sofort kündigen müssen.“).

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Diese Phase „heiße Phase“ zu nennen, stellt fast schon eine Beschönigung der Umstände dar. Viel eher könnte man von „Chaos-“ oder „Panik-Phase“ sprechen. Das Bild, was dazu in den Sinn kommt, ist das eines Ruderbootes, in dem viele erfahrene, muskelbepackte Ruderer sitzen, die mit aller Kraft rudern, nur leider vergessen haben, das Tau vom Steg zu lösen. So wird zwar viel Energie (sprich Arbeit) eingesetzt, nur das Boot (sprich das Projekt) kommt nicht von der Stelle.

Ein für Softwareprojekte typischer Teufelskreis ist hier zu beobachten: Illusorische Termine führen zunächst zu Kundenzufriedenheit, dann später zu hektischem Aktionismus und mangelhafter konzeptioneller Arbeit bei den Entwicklern. Die Folge sind eine Vielzahl von Fehlern in der Software. Die Behebung dieser Fehler bedeutet eine zusätzliche Arbeitsbelastung, was wiederum den Druck und den Zeitverzug erhöht. Wird auch darauf mit neuen, wiederum illusorischen Terminen reagiert, so ist der Teufelskreis geschlossen. Die unter dem Aspekt von Kundenorientierung wichtigste Konsequenz ist, daß das Softwarehaus permanent seine Zusagen nicht einhalten kann und insofern die Vertrauensbasis zum Kunden verliert (vgl. Abschnitt 3.6.1). Der mittel- und langfristige Schaden kann beträchtlich sein.

In den meisten Softwareprojekten gibt es in der Nähe von angestrebten Auslieferungsterminen ähnliche „heiße Phasen“. Im vorliegenden Projekt war diese Phase jedoch mit 10 Monaten außergewöhnlich lang, und es gab besonders für die Entwickler keine hinreichende Motivationsbasis, die die enormen Belastungen hätte stützen können. Die Versäumnisse der anfänglichen Phasen (besonders das Fehlen von Teamentwicklung und Meta-Reflexion) wirkten sich deshalb besonders schwer aus.

Es war für die Entwickler ziemlich offensichtlich, daß das Management versuchte, die mangelnde Projektplanung durch die Forderung nach Mehrarbeit auszugleichen. Letztlich konnte dies nicht funktionieren, und auch im vorliegenden Projekt funktionierte es nicht. Das Programm wurde trotzdem nicht rechtzeitig fertig, und die Motivation der Entwickler sank derart, daß drei Client-Entwickler und ein Server-Entwickler kündigten. Die vorerst letzte Kündigung erfolgte sogar erst im Juli, also ein halbes Jahr nach dieser Phase. Hier wird die Langzeitwirkung solcher von den Entwicklern als sinnlos empfundenen Belastungen sichtbar. Aus „inneren“ Kündigungen werden dann auch „äußere“.

Hier wird auch deutlich, warum Mitarbeiterorientierung als Voraussetzung für Kundenorientierung genannt wurde (vgl. Abschnitt 2.5.3). Derart unzufriedene Mitarbeiter können nicht kundenorientiert handeln, im Gegenteil: Der Kunde wird zunehmend zum Ärgernis, weil er immer mehr zu verlangen scheint und (berechtigterweise) nicht zufrieden ist mit den erbrachten Leistungen. Auf der anderen Seite sehen die Entwickler ihren Arbeitseinsatz (berechtigterweise) nicht gewürdigt. Die Polarisierung zwischen Entwicklungsorganisation und Kunde verstärkt sich. Obwohl beide Seiten also aus ihrer Sicht ‘im Recht’ sind, entsteht doch deutlich eine „Verlierer-Verlierer-Situation“.

4.4.12 Schulungen

Ab November begannen erste Schulungen des Programms. Zunächst fand ein sog. „Train-The-Trainer“-Programm statt, d.h. diejenigen, die geschult wurden, sollten später die echten Anwender schulen. Im Prinzip war das Programm zu diesem Zeitpunkt nicht schulungsreif (*„Die haben angefangen zu schulen, obwohl das System noch gar nicht fertig war. Wie immer.*“). Das Softwarehaus war jedoch in Zugzwang geraten. Es mußte etwas präsentiert

werden. Die Schulungen wurden mit Verspätung begonnen und mußten vorzeitig abgebrochen werden.

Beim Auftraggeber bedeuteten Schulungen einen hohen Planungsaufwand, so daß einmal gesetzte Termine praktisch nicht mehr verschoben werden konnten. Die oft kurzfristig bekanntgegebenen Terminverzögerungen führten dort zu viel Unzufriedenheit. (*„Es ist irrsinnig viel Zeit in Planung investiert worden. Auf unserer Seite. Mehrere Mann-Monate sind vergeudet worden, weil Software nicht rechtzeitig fertig wurde. Man kann nicht mal eben locker 300 Leute schulen. Das muß vorbereitet werden.“*)

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Wie in den vorigen Kapiteln gezeigt wurde, entsteht Kundenzufriedenheit im wesentlichen dann, wenn die Erwartungen des Kunden erfüllt oder sogar übertroffen werden (vgl. Abschnitt 3.5.2). Unter diesem Aspekt müssen die Schulungen äußert kritisch bewertet werden. Letztlich hatten die Schulungen nämlich vielmehr den Charakter eines Systemtests. Das Programm war in einem Zustand, in dem es keine Schulungen hätte geben dürfen. Aber aus den obengenannten Gründen schien es unvermeidlich gewesen zu sein, mit den Schulungen zu beginnen.

Natürlich hat es negative Konsequenzen, wenn die Schulungsteilnehmer merken, daß sie im Prinzip als Softwaretester benutzt werden. An sich wäre das kein Problem, nur entsprach es eben nicht der Erwartungshaltung. Wenn Schulungen angekündigt werden, dann erwartet der Kunde zurecht ein fertiges und lauffähiges System. Alles andere ist dann eine Enttäuschung, was die Stimmung belastet und ein konstruktives Umgehen mit den Unzulänglichkeiten des Systems erschwert.

In diesem Zusammenhang wirkt es doppelt schwer, daß in diesen ersten Schulungen diejenigen Vertreter des Kunden geschult wurden, die ihrerseits die späteren Anwender trainieren sollten. Das im vorigen Kapitel geforderte ‘Empowerment des Kunden’ (vgl. Abschnitt 3.6.4), welches durch ‘Train-the-Trainer-Programme’ eigentlich angestrebt wird, kann so in keinster Weise erreicht werden.

4.4.13 Produktionsaufnahme

Der Beginn der Produktion fand in zwei Schritten statt. Anfang April’96 wurde das System für einen Teilbereich des Tagesgeschäftes in Produktion genommen. Zu dieser Zeit arbeitete das neue System parallel zum Alt-System. Beim Auftraggeber war das Mißtrauen gegenüber der Software erheblich. Man erließ u.a. die Arbeitsanweisung, daß alle mit dem neuen System erfaßten Aufträge auf Papier nachgehalten werden sollten, um evtl. wieder auf das Alt-System zurückwechseln zu können. Die Probleme hielten sich jedoch in Grenzen, was sowohl das Softwarehaus als auch den Kunden überraschte.

Deshalb wurde bereits sechs Wochen später das Alt-System komplett abgelöst. Bei Produktionsaufnahme gab es beim Auftraggeber dann massive Probleme mit der neuen Software. Das Programm war unhandlich, schlecht getestet und instabil. Unter den Anwendern herrschte eine extreme Unzufriedenheit vor. Der Betriebsrat sprach von *„unzumutbaren Arbeitsumständen“*. Auch die Kunden des Auftraggebers reagierten mit heftigen Protesten auf die Probleme, die durch die neue Software entstanden waren (Fax vom Kunden: *„Wenn ich von Ihnen noch eine Änderungsbestätigung erhalte, sprenge ich ihren Buchungscomputer in die Luft!“*. Reaktion der Anwender: *“Wir nehmen Sie beim Wort! Versprochen!?! Wir würden Sie gern in Ihrem Vorhaben unterstützen!“*).

Außerdem war das System in wesentlichen Teilen noch nicht fertig implementiert. Fehlende Funktionalität führte z.B. dazu, daß in den ersten Monaten zwar Aufträge erfaßt, jedoch nicht abgerechnet werden konnten. Der Auftraggeber bekam schlichtweg kein Geld für die von ihm

erbrachten Leistungen. Nur wegen seiner enormen Liquidität konnte der Auftraggeber diese Phase ohne nennenswerten Schaden überstehen („*Jeder andere Laden wäre bankrott gegangen.*“).

Es stellt sich die Frage, warum mit dem Beginn der Produktion nicht länger gewartet wurde. Sowohl Auftraggeber als auch Softwarehaus hatten gewichtige Gründe, so schnell wie möglich mit dem Systembetrieb zu beginnen:

- Der Auftraggeber stand vor der Wahl, noch vor dem Sommer mit dem neuen System in Produktion zu gehen oder ein ganzes Jahr damit zu warten und mit dem Alt-System weiterzumachen (dies hängt mit der Art des Tagesgeschäfts des Auftraggebers zusammen). Dazu hätte man Hardware-Lizenzverträge verlängern und das Alt-System erweitern müssen. Dies hätte beträchtlichen finanziellen Aufwand bedeutet, den die EDV-Abteilung des Auftraggebers firmenintern nicht hätte durchsetzen können.
- Auch das Softwarehaus hatte ein Interesse, so früh wie möglich in Produktion zu gehen, da man durch die ständigen Terminverschiebungen in Argumentationsnot gekommen war und für spätere Verhandlungen einen Teil seiner Glaubwürdigkeit zurückbekommen wollte. Hinzu kam, daß die Stimmung innerhalb des Entwicklerteams auf einem Tiefpunkt angelangt war. Man mußte mit dem Echtbetrieb anfangen, um zu sehen, wo man stand, und um endlich Erfolgserlebnisse präsentieren zu können.

Der Produktionsbeginn markierte dann auch einen deutlichen Einschnitt im Projektverlauf. Der Druck auf die Entwickler nahm ab, dafür nahm der Druck für den Kunden erheblich zu. („*Seit wir im Produktion sind, hat der Kunde ja auch den schwarzen Peter.*“).

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Durch die ständigen Terminverschiebungen waren beide Seiten und insbesondere das Softwarehaus in Zugzwang geraten. Es mußte auf ‘Gedeih-oder-Verderb’ mit dem Produktionsbetrieb begonnen werden. Daher hatte der Produktionsbeginn aus Sicht des Softwarehauses deutlich den Charakter einer Stress-Umlastung zu Ungunsten des Kunden.

Programmeinführungen dieser Art bezeichnet man zutreffend als „Big-Bang“- oder auch als „Bombenwurf“-Strategie. Man konnte förmlich spüren, wie sich der Druck, der in den Jahren des Projektverlaufes entstanden war, nun wie in einer Explosion entlud. Unter dem Aspekt der Kundenorientierung kann man festhalten, daß das „Opfer“ der Explosion (wie in praktisch allen Projekten mit ähnlichem Verlauf) der Kunde selbst war. Man kann vermuten, daß auf Kundenseite die schlimmsten Vorurteile gegen EDV auf das Gründlichste bestätigt wurden und zukünftige Softwareentwicklungen unter enormen Schwierigkeiten stattfinden werden.

Man kann auch sehen, daß sich die negativen Auswirkungen dieser Big-Bang-Einführung nicht nur auf den Kunden bezogen, sondern auch auf die Kunden des Kunden (vgl. Abschnitt 3.2). Das bedeutet, daß das gewählte Vorgehen auch die Kundenorientierung des Kunden negativ beeinflußte.

Zusammenfassend kann man sagen, daß im Projektverlauf „Gewinner-Verlierer“- mit „Verlierer-Gewinner“-Situationen abwechselten um am Ende in eine „Verlierer-Verlierer-Situation“ zu münden. Die eigentliche Zielsetzung von Kundenorientierung, nämlich die Schaffung einer „Gewinner-Gewinner“-Situation ist nicht erreicht worden (vgl. Abschnitt 3.3).

4.4.14 Die Zeit nach Produktionsbeginn

Der Produktionsbeginn stellte einen wesentlichen Einschnitt im Projektverlauf dar. Der Druck, der auf dem Softwarehaus lastete, nahm spürbar ab, dafür häuften sich jetzt beim Auftraggeber die Probleme. Wo vorher der Kunde als Forderer aufgetreten war („*Wann seid*

ihr endlich fertig, wann kommt die neue Version?“), kam er jetzt mehr und mehr in die Rolle eines Bittstellers von Fehlerbehebungen.

Die Projektkontrolle, die nach Aussetzen der Freitagmeetings kaum noch vorhanden war, wurde jetzt wieder aktiviert. Es war seit Beginn der Auslieferungen zu unmittelbaren Kontakten zwischen dem Testpersonal des Auftraggebers und den Entwicklern gekommen. Dies war für die Entwickler durchaus nicht willkommen, da sie unter dem Druck standen, neue Programmteile fertigzustellen. Deshalb reglementierte man die Kontakte wieder und zentralisierte sie in Form einer Ansprechstelle im Softwarehaus, die die Probleme bzw. Fehlermeldungen dann gebündelt an die Entwickler weitergab.

Mit Produktionsbeginn nahm die Zeit wieder zu, die die Entwickler mit Fehlerbehebung zubringen mußten. Dies wirkte sich natürlich hinderlich auf die Fertigstellung der Programmteile aus, die noch realisiert werden mußten. Das Softwarehaus hatte vorher argumentiert, auch mit einer unfertigen Version in Produktion gehen zu können, weil die restliche Funktionalität schnell nachgeliefert werden konnte. Dies erwies sich dann in der Realität als nicht umsetzbar (*„Den ersten Monat, nachdem sie die Produktion komplett umgestellt hatten, war eine Weiterentwicklung überhaupt nicht möglich, weil es unendlich viele Bug-Fixes gab.“*).

Auf Auftraggeberseite gab es zwei Monate nach Produktionsbeginn eine Krisensitzung auf Top-Management-Ebene. Letztlich ging es dabei für den Auftraggeber um die Möglichkeit, die Zusammenarbeit mit dem Softwarehaus abzubrechen. Es wurde der Entschluß gefaßt, vorerst zusammenzuarbeiten.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Die Krisensitzung markierte einen vorläufigen Endpunkt im Projekt. Tatsache ist, daß die Zusammenarbeit zwischen Softwarehaus und Kunde nicht beendet wurde. Dies ist aus Sicht des Softwarehauses zunächst natürlich positiv zu werten.

Aus meiner Sicht hat der Auftraggeber die Zusammenarbeit mit dem Softwarehaus aber nur deshalb nicht beendet, weil er in einer extremen Abhängigkeitsposition gefangen war. Er selbst hätte die Software nicht weiterentwickeln können, war aber auf Weiterentwicklung und Fehlerbehebungen angewiesen. Es ist jedoch davon auszugehen, daß der Auftraggeber die Zusammenarbeit beenden wird, sobald er dazu in der Lage ist. Hier werden die langfristigen Auswirkungen fehlender Kundenorientierung deutlich.

4.5 Epilog - Ein Jahr danach

Die bisherigen Ausführungen basierten auf der im August 1996 fertiggestellten Projektanalyse. Ein Jahr später hatte ich die Möglichkeit, im Rahmen informeller Gespräche (sowohl mit Entwicklern als auch mit Vertretern des Kunden) einige meiner Prognosen zu überprüfen.

Inzwischen scheint das System relativ zufriedenstellend zu arbeiten. Es fehlen jedoch auch weiterhin wichtigste Funktionalitäten. Zum Beispiel wird der Auftraggeber auch 1997 seinen Jahresabschluß noch manuell durchführen müssen (*„weil das System das nicht packt“*).

Die Zusammenarbeit zwischen Auftraggeber und Softwarehaus ist weiterhin durch etliche schwerwiegende Krisen gegangen. Dabei war eine Auflösung der Zusammenarbeit immer wieder Thema. Der Auftraggeber hat es schließlich durch massiven Druck (*„Wir haben den Knüppel gezeigt, aber nicht zugeschlagen“*) erreicht, daß Entwickler des Softwarehauses jetzt vor Ort beim Auftraggeber arbeiten, um die noch fehlende Funktionalität nachzuliefern und die Fehlerbehebungen schnell und unbürokratisch durchzuführen. Die Entwickler empfinden diese Arbeit jedoch eher als *„Nachsitzen“* bzw. als *„Strafarbeit“*.

Das Klima zwischen Auftraggeber und Softwarehaus ist besonders auf höherer Managementebene denkbar schlecht geworden und hat sich letztlich festgefahren. Eine Konsequenz ist, daß der Auftraggeber wieder engeren Kontakt zu einem früheren Mitbewerber des Softwarehauses aufgenommen hat. In diesem Zusammenhang ist sogar die Ablösung des gerade erst fertiggestellten Systems im Gespräch. Eine Beendigung der Zusammenarbeit von Auftraggeber und Softwarehaus ist deshalb weiterhin aktuell.

Während das beschriebene Projekt in bezug auf die Zusammenarbeit zwischen Auftraggeber und Softwarehaus deutlich negative Auswirkungen hatte, so hatte es doch innerhalb des Softwarehauses auch positive Folgen. Die Projektanalyse wurde im August'96 mit der Empfehlung vorgelegt, ein Projekt-Review mit dem gesamten Entwicklungsteam unter Einbeziehung eines externen Moderators durchzuführen. Dieses Projektreview fand - geleitet durch einen externen Unternehmensberater - schon wenige Wochen später statt und wurde von den Beteilgten als überaus positiv bewertet. In der Folge erhielt der Unternehmensberater den Auftrag, Schulungen für die gesamte Belegschaft des Softwareherstellers zu konzipieren. Diese Schulungen, die ihren Schwerpunkt in Kommunikation und Teamorientierung hatten, wurden noch im selben Jahr begonnen und Anfang des folgenden Jahres abgeschlossen. Gleichzeitig brachte eine parallel durchgeführte ISO-9000-Zertifizierung einiges an Veränderungen in Kultur und Struktur des Softwarehauses.

Fazit und Bewertung vor dem Hintergrund von Kundenorientierung

Die negativen Prognosen aus der Projektanalyse haben sich bestätigt. Insbesondere gilt dies für die interne Struktur des Softwaresystems und die Geschäftsbeziehung zwischen Softwarehaus und Kunde. Interessant ist auch, daß jetzt die ursprüngliche Forderung des Kunden, die Software im eigenen Haus zu entwickeln, erfüllt wurde. Allerdings sind die Voraussetzungen für eine produktive kundenorientierte Arbeit jetzt denkbar schlecht, so daß auch dieses kundennahe Agieren der Entwickler nur wenig Veränderungen in bezug auf die Geschäftsbeziehung haben wird.

Auch wenn nicht alle Ursachen des problematischen Projektverlaufes vom Softwarehaus zu verantworten waren, so müssen doch die heutigen Gegebenheiten die Entscheidungsträger nachdenklich stimmen. Wechselseitige Schuldzuweisungen führen hier nicht weiter. Unterm Strich wird das Softwarehaus vermutlich einen wichtigen Kunden verlieren, bei dem ein großes Potential für langfristige Zusammenarbeit gelegen hätte. Deshalb ist es aus meiner Sicht gerechtfertigt, das vorliegende Projekt als langfristigen Fehlschlag zu werten.

Die zum Ende des Projektes einsetzenden Veränderungen im Softwarehaus sind zum Zeitpunkt dieser Betrachtungen noch zu sehr in einem Anfangsstadium, um sie abschließend beurteilen zu können. Trotzdem sind positive Ansätze durchaus erkennbar. Unabhängig davon, wie diese weiterverfolgt werden, kann man erkennen, daß sich selbst Fehlschläge bei 'richtiger' Behandlung in erfolgversprechende Veränderungen umsetzen lassen.

4.6 Zusammenfassung der Projektanalyse vor dem Hintergrund von Kundenorientierung

Wenn man das vorliegende Projekt in bezug zu den in Kapitel 3 aufgestellten Anforderungen an eine kundenorientierte Prozeßgestaltung setzt, so ergeben die folgenden Kritikpunkte:

- Die Zusagen des Softwarehauses wurden nicht eingehalten.
- Ein kundenorientierter Qualitätsbegriff wurde nicht umgesetzt.
- Es fand keine hinreichende Integration des Kunden in den Entwicklungsprozeß statt.
- Es wurden keine Teamentwicklungsmaßnahmen durchgeführt.

- Ein direkter Anwender-Entwickler-Kontakt wurde nicht unterstützt.
- Kontinuierliche Projekt-Reflexion war kein Bestandteil der Vorgehensweise.
- Das Entwicklungsteam war strukturell ungünstig konfiguriert.
- Es fehlte auf der Seite des Softwarehauses eine kundenorientierte Kultur.

Die von der Entwicklungsorganisation im Projektverlauf gemachten Zusagen wurden in der Regel nicht eingehalten. Termine wurden praktisch immer und z.T. erheblich überzogen. Gleiches gilt für den Bereich der Kosten, wobei dieser Bereich (da es sich um ein Festpreisprojekt handelt) eher den Softwarehersteller trafen. In bezug auf die Funktionalität des Systems kann man festhalten, daß hier im Projektverlauf eine kontinuierliche Reduktion an Funktionalität stattfand. Insofern sind auch die Zusagen, die die Funktionalität betrafen, nicht gehalten worden.

Die geforderte Umsetzung eines kundenorientierten Qualitätsbegriffes mit Hilfe von kundenorientierten Dokumenten und Zwischenprodukten fand nur unzureichend statt. Das zentrale Dokument des Entwicklungsprozesses - der „fachliche Leistungsumfang“ - war für den Kunden weder geeignet, das Verständnis der Entwickler zu überprüfen, noch eine Vision vom zukünftigen System entstehen zu lassen. Der eingesetzte Oberflächenprototyp konnte in Teilbereichen des Systems eine Vision schaffen, jedoch wurden diese Vorteile dadurch nivelliert, daß der Prototyp im Prinzip außerhalb des Entwicklerteams entstand. Damit wurde der Lernprozeß abgeschnitten und der Wert des Prototypen ins Gegenteil verkehrt.

Es wurde keine Integration des Kunden in den Entwicklungsprozeß erreicht. Gemeinsame Sitzungen dienten dem Softwarehaus zur Informationsbeschaffung, zu einseitigen Präsentationszwecken oder gar zum Verschleiern des realen Projektzustandes.

Im Gegensatz zu einer Integration des Kunden ins Entwicklungsteam war im Projektverlauf eine zunehmende Polarisierung zwischen Softwarehaus und Kunde zu beobachten, die kundenorientiertes Handeln unmöglich machte.

Es fanden keine Teamentwicklungsmaßnahmen statt. Im Gegenteil, es wurde mit organisatorischen Maßnahmen (allen voran die Freitagmeetings) die Bildung eines funktionierenden Teams über weite Strecken des Projektes sogar verhindert.

Die Etablierung und Unterstützung eines direkten Kontaktes zwischen Entwicklern und Anwendern stand nicht im Vordergrund der Bemühungen. Im Gegenteil, dieser Kontakt wurde tendenziell eher limitiert und fand intensiv nur dann statt, wenn es (wie z.B. in Zeiten der Systemeinführung) nicht zu vermeiden war.

Eine kontinuierliche Projektreflexion, z.B. in Form von Projektreviews, fehlte. Im Gegensatz war bei vielen Beteiligten eher eine „Kopf-in-den-Sand“-Strategie zu beobachten. Die Folge war eine generelle Intransparenz des Projektfortschritts und eine Verschleppung von Problemen. Die Probleme wurden vor sich hergeschoben und erst angegangen, wenn sie unübersehbar geworden waren. Dann allerdings war es zu spät, grundlegende Änderungen vorzunehmen, denn dann existierten bereits neue „Sachzwänge“. So waren auch die Lösungen zumeist situativ und kurzfristig angelegt. Es ist deutlich, daß dem Projekt ein ‘Frühwarnsystem’ gefehlt hat. Ein solches hätte aber im Rahmen von kontinuierlicher Projektreflexion fast automatisch etabliert werden können.

Bezogen auf die Struktur des Entwicklungsteams kann man sagen, daß dieses nicht kundenorientiert zugeschnitten war, sondern sich an technischen Gegebenheiten orientierte. Insbesondere ist hier die Trennung von Datenbank und Client zu kritisieren. Kundenorientierung wäre nur möglich gewesen, wenn beide Teile nicht getrennt, sondern gemeinsam (unter Einbeziehung des Kunden) vorgegangen wären.

Bezogen auf die Kultur kann man das Fehlen von Teamkultur und von Lernkultur feststellen. Die für eine Lernkultur erforderliche ‘fehlerfreundliche’ bzw. ‘fehlernutzende’ Haltung war

besonders beim Softwarehausmanagement zu wenig vorhanden. Ein besonders krasses Beispiel hierfür sind die Freitagmeetings, die in ihrer tendenziell repressiven Form eine offene Diskussion über die Probleme verhinderten und damit ein Verstecken von Fehlern bzw. ein Verschleppen von Problemen förderten.

Wenn man sich abschließend die der Kundenorientierung zugrunde liegenden Zielsetzungen, nämlich langfristige Kundenbindung über das Erreichen maximaler Kundenzufriedenheit, anschaut, so muß man festhalten, daß diese Zielsetzungen im vorliegenden Projekt nicht erreicht worden sind.

Zusammenfassend ist festzuhalten, daß sowohl Kunde als auch Softwarehaus aus vergangenen Projekten nicht oder nur wenig gelernt haben. Im Prinzip war die Vorgehensweise und der Projektverlauf identisch zum Vorgängerprojekt (vgl. [Fittkau94]) und führte zu ähnlichen, teilweise sogar massiveren Problemen und Konsequenzen. Insofern haben beide - Softwarehaus und Auftraggeber - mit diesem Projekt eine wichtige Lernchance verpaßt.

5 Zusammenfassung und Ausblick

Diese Arbeit hatte die Zielsetzung, die Bedeutung von Kundenorientierung in der Softwareentwicklung zu klären. Diese Zielsetzung wurde erreicht. Es ist deutlich geworden, daß Kundenorientierung in praktisch allen Bereichen von Softwareentwicklung Relevanz besitzt und zum Ziel von Veränderungsbemühungen werden kann.

In Kapitel 2 wurde die betriebswirtschaftliche Sicht auf Kundenorientierung diskutiert. Es wurde deutlich, daß Kundenorientierung zu einem wichtigen Erfolgsfaktor von Unternehmen geworden ist und weitreichende Konsequenzen für die Organisationstruktur und die Unternehmenskultur hat.

Im Kapitel 3 wurde die Bedeutung von Kundenorientierung für die Softwareentwicklung untersucht. Dabei ist deutlich geworden, daß Kundenorientierung von Softwareherstellern in Zukunft immer stärker gefordert werden wird und insofern auch in dieser Branche nicht nur eine erfolgsversprechende sondern auch eine notwendige Strategie darstellt. Es wurde argumentiert, daß Kundenorientierung im Rahmen von Softwareentwicklungen sowohl Auswirkungen auf die Produktgestaltung, als auch auf die Prozeßgestaltung haben muß. Weiterhin sind wichtige strukturelle und kulturelle Voraussetzungen für eine kundenorientierte Softwareentwicklung vorgestellt worden.

In Kapitel 4 schließlich wurden anhand eines industriellen Softwareprojektes die negativen Auswirkungen gezeigt, die ein traditionelles Vorgehen auf die Kundenorientierung des Softwareherstellers haben kann.

Zusammenfassend kann man sagen, daß die Forderung nach Kundenorientierung in den genannten Bereichen der Softwareentwicklung jeweils eine Abkehr von traditionellen Vorstellungen beinhaltet:

- Bezogen auf das Produkt, die Abkehr von einer ablaufsteuernden, den Kunden kontrollierenden Sichtweise hin zu einer unterstützenden, den Kunden qualifizierenden Sichtweise,
- bezogen auf den Entwicklungsprozeß, die Abkehr von phasenorientierten, den Kunden ausgrenzenden Vorgehenweisen hin zu evolutionären, den Kunden integrierenden Entwicklungsstrategien,
- bezogen auf die Organisationsstruktur, die Abkehr von einer bürokratischen Linienorganisation hin zu einer am Kunden orientierten Projektteam-Organisation,
- bezogen auf die Organisationskultur, die Abkehr von einer technisch ausgerichteten Bürokratiekultur hin zu einer veränderungsbereiten und teamorientierten Lernkultur
- und bezogen auf den einzelnen Softwareentwickler, die Abkehr von den kundenfeindlichen Rollenbildern des „Hackers“ bzw. des „rationalen Technikers“ hin zu einem Selbstverständnis als „kundenorientierter Softwareentwickler“, d.h. als Dienstleister und Berater für die Informationsprobleme des Kunden.

Die Entwicklung, die bei einem Softwarehersteller im Zuge vom mehr Kundenorientierung geschehen muß, wird in folgender Abbildung nochmals dargestellt:

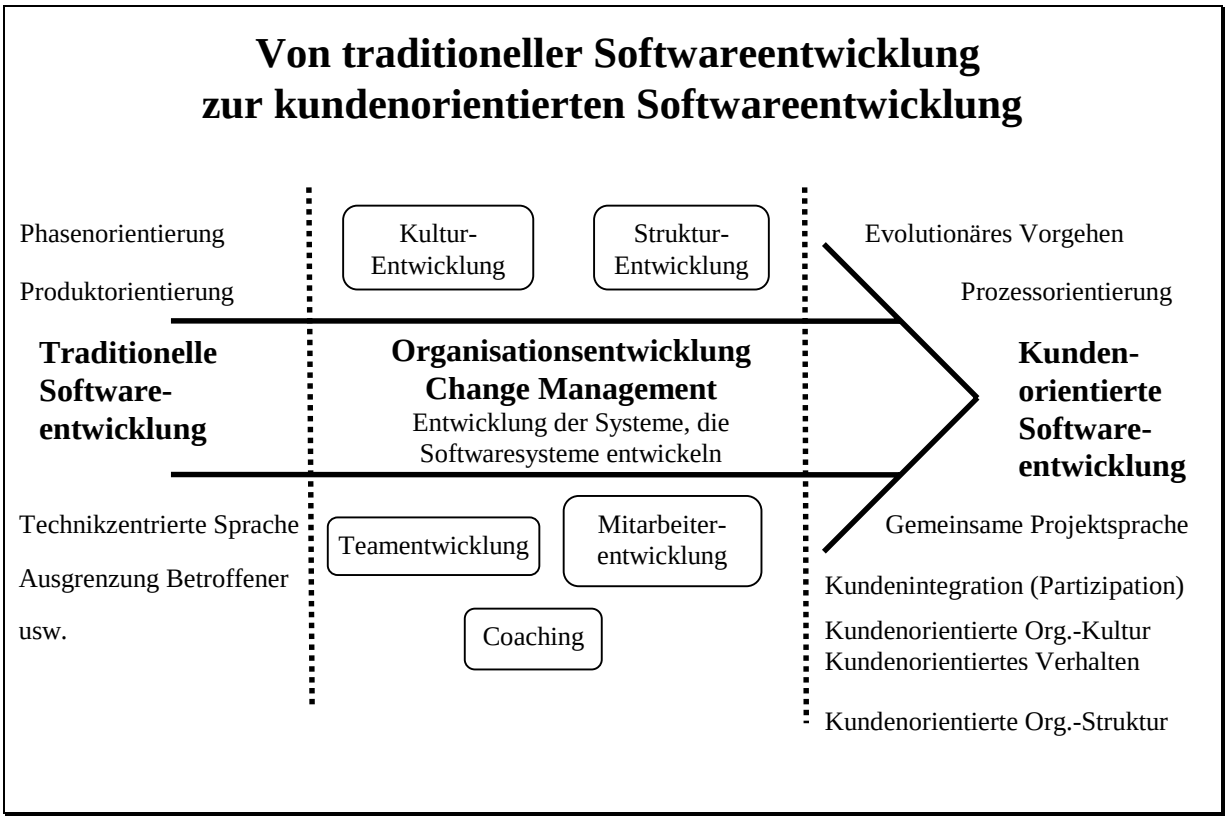

Abbildung 5-1

Aus der obigen Abbildung wird auch deutlich, wo zukünftige Forschungsarbeiten ansetzen müssen. Diese Arbeit konnte in Abgrenzung zur traditionellen Softwareentwicklung die Vision einer kundenorientierten Softwareentwicklung skizzieren. Lediglich angedeutet wurden die Instrumente, die beim Softwarehersteller im Rahmen einer Organisation-entwicklung hin zu mehr Kundenorientierung eingesetzt werden können.

Es ist mir bewußt, daß es letztlich nicht ausreicht, bestehende Vorgehensweisen in der Softwareentwicklung zu kritisieren, ohne den Erfolg alternativer Vorgehensweisen praktisch nachzuweisen. Zukünftige Arbeiten müssen sich deshalb auch mit der praktischen Umsetzung von kundenorientierter Softwareentwicklung befassen. Hier bietet sich die systematische Aufarbeitung von industriellen Projekten (die explizit den Anspruch Kundenorientierung erheben) in Form von Fallbeispielen an.

Ein wesentlicher Aspekt dieser Arbeit war die Betonung einer kundenorientierten Unternehmenskultur auf Seiten des Softwareherstellers. Wie eine solche Kultur praktisch erreicht werden kann, muß am ‘lebenden Objekt’ untersucht werden. Auch müssen Wege zur Erreichung der hohen Qualifikationsanforderungen, die an Softwareentwickler zukünftig gestellt werden, diskutiert werden. Dabei müssen sich zukünftige Arbeiten auch konstruktiv mit den Hindernissen auseinandersetzen, die einer kundenorientierten Softwareentwicklung im Weg stehen.

Ein Aspekt, der im Rahmen der Arbeit zwar angesprochen, jedoch nicht weiter diskutiert wurde, ist die Frage, ob es auch beim Kunden selber Voraussetzungen für eine kundenorientierte Softwareentwicklung geben muß. Ich glaube ja. Gerade die Forderung nach Kundenintegration in den Entwicklungsprozeß legt dies nahe. Eine hierarchisch und bürokratisch geprägte Kundenorganisation wird nur schwer in einen solchen kunden-

orientierten Entwicklungsprozeß eingebunden werden können, wie er hier beschrieben wurde. Heutige Entwicklungstendenzen, wo sogar der traditionell kundenfeindliche Beamtenapparat kundenorientierter werden soll (Stichwort „Das kundenorientierte Rathaus“, vgl. [KiBogWiech95]), deuten eher darauf hin, daß solche Organisationsstrukturen über kurz oder lang verschwinden werden. Bevor dies jedoch erreicht ist, muß ein kundenorientierter Softwarehersteller auch im Umfeld von ‘nicht-kundenorientierten’ Unternehmen erfolgreich handlungsfähig bleiben. Insofern muß es mehr und mehr die Aufgabe eines Softwareherstellers werden, auch auf dieser Ebene beim Kunden Entwicklungsarbeit zu leisten. Wenn sich also ein Softwarehersteller selber in Richtung Kundenorientierung entwickelt, dann hat das immer auch den Nebeneffekt, daß die Mitarbeiter am ‘eigenen Leib’ mit der Problematik von Organisationsentwicklung konfrontiert werden. Insofern werden sie nicht nur sensibilisiert, sie erweitern auch diejenigen Kompetenzen, die bei Veränderungen innerhalb der Kundenorganisation benötigt werden.

Generell müssen meiner Ansicht nach die Forschungergebnisse aus dem Bereich der Organisationsentwicklung stärker in den Bereich der Softwareentwicklung integriert werden. Nicht nur bedeutet Kundenorientierung Organisationsentwicklung für den Softwarehersteller, auch die Softwareentwicklung, die ein Softwarehersteller leistet, bedeutet letztlich Organisationsentwicklung beim Kunden. Insofern deutet die Frage nach Kundenorientierung in der Softwareentwicklung auf Organisationsentwicklung im doppelten Sinne hin.

Meiner Ansicht nach wird ein Softwarehersteller über kurz oder lang nur dann kundenorientiert arbeiten können, wenn er seine Kernkompetenzen in den Bereich der Organisationentwicklung hinein erweitert und Softwareentwicklung und Organisationsentwicklung als Einheit anerkennt.

6 Literaturverzeichnis

[Albach88] H.Albach: *Maßstäbe für den Unternehmenserfolg,* in: H.A.Henzler (Hrsg.): Handbuch Strategische Führung, Wiesbaden 1988

[Althaus95] S.Althaus: *Kundenorientierung als Integrationsfaktor ganzheitlicher Unternehmungsführung*, Dissertation an der Hochschule St.Gallen, 1995

[Bäum93] D.Bäumer: *Gebos - Ein Praxisbeispiel zu objektorientierter Analyse und objektorientiertem Entwurf,* HMD 170, 1993

[BauBoeEck94] A.Baukrowitz, A.Boes, B.Eckhard: *Software als Arbeit gestalten. Konzeptionelle Neuorientierung der Aus- und Weiterbildung von Computerspezialisten*, Westdeutscher Verlag 1994

[BerZeithPara90] L.L.Berry, V.A.Zeithaml, A.Parasuraman: *Five Imperatives for Improving Service Quality,* in: Sloan Management Review, Summer 1990

[Bleicher91] K.Bleicher: *Das Konzept Integriertes Management*, Campus 1991

[Boehm76] B.W.Boehm: *Software Engineering,* IEEE Transactions on Computers, 25(12) 1976

[Boehm81] B.W.Boehm: *Software Engineering Economics*, Prentice Hall 1981

[Boehm88] B.W.Boehm: *A Spiral Model of Software development and Enhancement,* in: Computer 5/1988

[Booch91] G.Booch: *Object-Oriented Design with Applications,* Benjamin/Cummings 1991

[Bosten88] The Bosten Consulting Group: *Vision und Strategie,* Die 34.Kronberger Konferenz, München 1988

[BoyDudKus94] J.Boy, C.Dudek, S.Kuschel: *Projektmanagement,* Gabal 1994

[Bråten73] S.Bråten: *Model Monopoly,* in: Acta Sociologica, 16(2), 1973

[Brooks75] F.Brooks: *The Mythical Man-Month*, Addison-Wesley 1975

[Brooks87] F.Brooks: *No Silver Bullet,* in: IEEE Computer 20(4) 1987

[BüGryZü92] U.Bürkle, G.Gryczan, H.Züllighoven: *Erfahrungen mit der objektorientierten Vorgehensweise bei einem Bankenprojekt*, Informatik-Spektrum, Band 15, Heft 5, Okt.1992

[BuKaKuZü92] R.Budde, K.Kautz, K.Kuhlenkamp, H.Züllighoven: *Prototyping - An Approach to Evolutionary System Development*, Springer-Verlag 1992

[BuZü90] R.Budde, H.Züllighoven: *Software-Werkzeuge in einer Programmierwerkstatt*, Oldenbourg Verlag 1990

[ClifCav86] D.K.Clifford, R.E.Cavanagh: *Spitzengewinner. Strategien erfolgreicher Unternehmen,* Düsseldorf, Wien 1986

[CopSchm95] J.O.Coplien, D.C.Schmidt: *Pattern Languages of Program Design,* Addison Wesley 1995

[Dahl92] B.Dahlbom: *The Idea that Reality is Socially Constructed,* in: [FloZüBuKe92]

[DahlMath93] B.Dahlbom, L.Mathiassen: *Computers in Context,* Blackwell Publishers 1993

[DeMarcoList91] T.DeMarco, T.Lister: *Wien wartet auf Dich! Der Faktor Mensch im DV-Management,* Hanser Verlag 1991

[DopLaut94] K.Doppler, C.Lauterburg: *Change Management: den Unternehmenswandel gestalten,* Campus 1994

[Droege94] W.Droege: *Den Weg für die Zukunft freimachen,* in: Absatzwirtschaft 5, 1994.

[Droege96] Droege & Company, *Customer Banking. Kundenorientierung als Unternehmensaufgabe,* ISBN 3-409-14096-4, April 1996

[Drucker54] P.F.Drucker: *The Practice of Management.* New York 1954

[DzidaKon95] W.Dzida, U.Konradt: *Psychologie des Software-Entwurfs,* Verlag für Angewandte Psychologie 1995

[Eggert92] K.Eggert: *Die Strategie Kundennähe. Komponenten, Konzept, Erfolgspotentiale.* Dissertation Universität Lüneburg 1992

[Fittkau94] Boris Fittkau: *Erfahrungen bei einem objektorientierten Softwareprojekt,* Studienarbeit an der Uni Hamburg, Fachbereich Informatik, 1994

[FittkauBe97] Bernd Fittkau: *Kommunikation. Ein bestimmendes Moment von Organisationskulturen,* Integrative Therapie 1-2/1997

[Fließ96] S.Fließ: *Prozeßevidenz als Erfolgsfaktor der Kundenintegration,* in: [KleFliJac96]

[FloZüBuKe92] C.Floyd, H.Züllighoven, R.Budde, R. Keil-Slavik: *Software Development and Reality Construction,* Springer-Verlag 1992

[Floyd92] C.Floyd: *Software Development as Reality Construction,* in: [FloZüBuKe92]

[Floyd93] C.Floyd: *Einführung in die Softwaretechnik,* Script zur gleichnamigen Vorlesung an der Universität Hamburg, 1993

[Floyd94] C.Floyd: *Software-Engineering - und dann?,* Informatik-Spektrum(1994)17, Springer-Verlag 1994

[Floyd97] C.Floyd: *Einführung in die Softwaretechnik,* Script zur gleichnamigen Vorlesung an der Universität Hamburg, 1997

[GamHelJohVli94] E.Gamma, R.Helm, R.Johnson, J.Vlissides: *Design Patterns. Elements of Reusable Object-Oriented Software,* Addison Wesley 1994

[Geffroy94] E.K.Geffroy: *Das einzige was stört ist der Kunde: Clienting ersetzt Marketing und revolutioniert Verkauf,* Verlag Moderne Industrie, 1994

[Glagda96] M.Glagda: *Organisatorisches Lernen nach Senge: Ein Weg zur Softwareprozeßverbesserung?,* in: [SteMüllKunz96]

[Gold90] A.Goldberg: *Informations, Models, Views and Controllers,* in: Dr.Dobb's Journal, Juli 1990

[GoldRub95] A.Goldberg, K.S.Rubin: *Succeeding with Objects,* Addison Wesley 1995

[Goletz96] O.Goletz: *Bestandsaufnahme: Praktische Probleme der Softwareprozeßverbesserung,* in: [SteMüllKunz96]

[GouKelly95] F.J.Gouillart, J.N.Kelly: *Business Transformation,* Ueberreuter 1995

[GryZü92] G.Gryczan, H.Züllighoven: *Objektorientierte Systementwicklung. Leitbild und Entwicklungsdokumente,* Informatik-Spektrum, Band 15, Heft 5, Okt.1992

[Gryczan95] G.Gryczan: *Situierte Koordination computergestützter qualifizierter Tätigkeit über Prozeßmuster.* Dissertation Uni Hamburg 1995

[GünHub96] B.Günter, O.Huber: *Beschwerdemanagement als Instrument der Customer Integration,* in: [KleFliJac96]

[HammChamp94] M.Hammer, J.Champy: *Business Reengineering,* Campus 1994

[HansLomn93] J.Hansel, G.Lomnitz: *Projektleiter-Praxis,* Springer 1993

[Henders92] B.Henderson-Sellers: *A Book of Object-Oriented Knowledge,* Prentice Hall 1992

[HersTsch91] C.Herstatt, H.Tschirky: *Kundennähe der technologieorientierten Unternehmen,* in: io Management-Zeitschrift 11/1991

[HerzHierMell96] G.Herzwurm, A.Hierholzer, W.Mellis (Hrsg.): *Kundenorientierte Softwareherstellung,* Reihe: Studien zur Systementwicklung, Band 9, Köln 1996

[HerzHier96] G.Herzwurm, A.Hierholzer: *Kundenorientierung durch Software Customer Value Management (SCVM),* in: [HerzHierMell96]

[Hinterh89] H.H.Hinterhuber: *Strategische Unternehmensführung,* Band I, Berlin, New York 1989

[Homburg94] C.Homburg: *Kundennähe von Industriegüterunternehmen,* Habilitationsschrift Universität Mainz 1994

[Homburg95] C.Homburg: *Produkte gut, Service mangelhaft,* in: VDI-Nachrichten, Jg. 49, 1995, Nr. 20

[Imai93] M.Imai: *Kaizen: Der Schlüssel zum Erfolg der Japaner im Wettbewerb,* Ullstein 1993

[KanSerTak84] N.Kano, N.Seraku, F.Takahashi: *Attractive quality and must-be quality,* in: Quality Nr.2, 1984

[KieLiSchZü92] A.Kieback, H.Lichter, M.Schneider-Hufschmidt, H.Züllighoven: *Prototyping in industriellen Software-Projekten,* Informatik-Spektrum (1992) 15: 65-77, Springer Verlag 1992

[KilGryZü93] K.Kilberth, G.Gryczan, H.Züllighoven: *Objektorientierte Anwendungsentwicklung,* Verlag Vieweg 1993

[KiBogWiech95] Kißler, Bogumil, Wiechmann: *Das kleine Rathaus: Kundenorientierung und Produktivitätssteigerung durch den Bürgerladen Hagen,* 1995

[KleFliJac96] M.Kleinaltenkamp, S.Fließ, F.Jacob: *Customer-Integration: von der Kundenorientierung zur Kundenintegration.* Gabler 1996

[Kleinalt96] M.Kleinaltenkamp: *Customer Integration - Kundenintegration als Leitbild für das Business-to-Business-Marketing,* in: [KleFliJac96]

[KobiWüth85] J.M.Kobi, H.Wüthrich: *So beurteilen wir die Unternehmenskultur,* in: io Management-Zeitschrift 1/1985

[KönVol93] E.König, G.Volmer: *Systemische Organisationsberatung,* Deutscher Studien Verlag 1993

[KrabWetzRat96] A.Krabbel, I.Wetzel, S.Ratuski: *Participation of Heterogeneous User Groups: Providing an Integrated Hospital Information System,* Universität Hamburg Fachbereich Informatik 1996

[KrabWetzRat96b] A.Krabbel, I.Wetzel, S.Ratuski: *Objektorientierte Analysetechniken für übergreifende Aufgaben,* in: Beiträge der GI-Fachtagung Softwaretechnik '96, Koblenz 1996

[Krub84] E.Krubasik: *Technologiemanagement für überlegene Innovationsstrategien,* in: H.Henzler (Hrsg.): Handbuch Strategische Führung, Wiesbaden 1988

[Krulis84] J.S.Krulis-Randa: *Reflexionen über Unternehmenskultur,* in: Die Unternehmung 4/1984

[Kühn91] R.Kühn: *Methodische Überlegungen zum Umgang mit der Kundenorientierung im Marketing-Management,* in: Marketing ZFP (13), Nr.2, 1991

[Lewis95] T.Lewis et al.: *Object Oriented Applications Frameworks,* Manning 1995

[Levitt60] T.Levitt: *Marketing Myopia,* in: Harvard Business Review Vol.38, Jul/Aug 1960

[LilienZülligh96] C.Lilienthal, H.Züllighoven: *Techniques and Tools for Continuous User Participation,* Universität Hamburg Fachbereich Informatik 1996

[Lilenthal95] C.Lilienthal: *Konzeption und Realisierung eines an der Anwendungssprache orientiertes Hilfesystem nach der Werkzeug-Material Methapher,* Diplomarbeit Uni Hamburg, Fachbereich Informatik 1995

[LorKidd94] M.Lorenz, J.Kidd: *Object-Oriented Software Metrics,* Prentice Hall 1994

[MamOppTep86] P.Mambrey, R.Oppermann, A.Tepper: *Computer und Partizipation,* Westdeutscher Verlag 1986

[Meixner95] H.Meixner: *Zum Einfluß partizipativer Entwicklungstechniken auf die Benutzerfreundlichkeit von Programmsystemen - Eine Feldstudie -,* Dissertation an der RWTH Aachen, 1995

[Meyer95] B.Meyer: *Object Success,* Prentice Hall 1995

[NagelRasn93] K.Nagel, C.Rasner: *Herausforderung Kunde: Neue Dimensionen der kunden- und marktorientierten Unternehmensführung,* Landsberg/Lech 1993

[Naranjo94] C.Naranjo: *Character and Neurosis,* Gateways 1994

[Naur85] P.Naur: *Programming as Theory Building,* In: Microprocessing and Microprogramming, Vol. 15, 1985

[Pasch92] J.Pasch: *Dialogischer Software-Entwurf.* Dissertation, Technische Universität Berlin, 1992

[PetWat82] T.Peters, R.H.Waterman: *In Search of Excellence.* New York 1982

[Plinke96] W.Plinke: *Kundenorientierung als Voraussetzung der Customer Integration,* in: [KleFliJac96]

[PomBla93] G.Pomberger, G.Blaschek: *Grundlagen der Software engineering: Prototyping und objektorientierte Software-Entwicklung,* Hanser Verlag 1993

[ReichSass90] F.F.Reichelt, W.E.Sasser Jr.: *Zero Defections: Quality comes to Services,* in: Harvard Business Review Sep./Okt. 1990

[Reisin91] F.-M.Reisin: *Kooperative Gestaltung in partizipativen Softwareprojekten,* Dissertation, Technische Universität Berlin, 1991

[Riemann74] F.Riemann: *Grundformen der Angst. Eine tiefenpsychologische Studie.* Reinhardt 1974

[Salzg96] F.Salzgeber: *Kunden- und Prozessorientierung im Versicherungsunternehmen,* Dissertation, Universität Zürich, 1996

[Salzm89] H.Salzman, S.Rosenthal: *Organizational Dimensions of the Software Design Process,* Center for Applied Social Science, Bosten, 1989 (zitiert nach [WelOrt92])

[SchulzvonThun81] F.Schulz von Thun: *Miteinander reden 1 - Allgemeine Psychologie der Kommunikation,* Rowohlt 1981

[SchulzvonThun91] F.Schulz von Thun: *Miteinander reden 2 - Differenzielle Psychologie der Kommunikation,* Rowohlt 1991

[SeiwGay96] L.J.Seiwert, F.Gay: *Das 1x1 der Persönlichkeit,* Gabal 1996

[Senge94] P.M.Senge: *The Fifth Discipline: The Art and Practise of the Learning Organisation,* New York 1994

[Shap88] B.P.Shapiro: *What the Hell is 'Market Orientation'?,* in: Harvard Business Review Nov./Dez. 1988

[Simon90] H.Simon: *Weltmeister*, in: Manager Magazin 11/1990.

[SteMüllKunz96] D.Stelzer, U.Müller, M.Kunz: *Softwareprozeßverbesserung durch organisatorisches Lernen*, Köln 1996

[Spilln94] A.Spillner: *Kann eine Krise 25 Jahre dauern?*, in: Informatik-Spektrum 17 (1994)

[Spitta89] T.Spitta: *Software-Engineering und Prototyping*, Berlin 1989

[Strassm95] P.Strassmann: *Outsourcing: A game for loosers*, in: Computerworld, 21.8.95

[ThomSchulz90] C.Thoman, F.Schulz von Thun: *Klärungshilfe - Theorien, Methoden, Beispiele*, Rowohlt 1990

[WatzBeaJa67] P.Watzlawick, J.H.Beavin, D.D.Jackson: *Pragmatics of Human Communication*, Norton & Company 1967

[Weizenb77] J.Weizenbaum: *Die Macht des Computers und die Ohnmacht der Vernunft*, Frankfurt a.M. 1977

[WelOrt92] F.Weltz, R.G.Ortmann: *Das Softwareprojekt*, Campus Verlag 1992

[Wendt93] S.Wendt: *Defizite im Software Engineering*, in: Informatik Spektrum, Febr. 1993

[WirfWiWi90] R.Wirfs-Brock, B.Wilkerson, L.Wiener: *Designing Object-Oriented Software*, Prentice Hall 1990

[White92] R.Whiteley: *Ihr Kunde ist der Boss*, Freiburg 1992

[Yourd93] Edward Yourdan: *Die westliche Programmierkunst am Scheideweg*, Prentice Hall 1993

[ZeithParaBer92] V.A.Zeithaml, A.Parasuraman, L.L.Berry: *Qualitätsservice: Was Ihre Kunden erwarten - was Sie leisten müssen.* Campus 1992

[Zollner94] G.Zollner: *Kundennähe in Dienstleistungsunternehmen.* Dissertation Universität München 1994

[Züllighoven98] H.Züllighoven: *Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material Ansatz*, dpunkt.verlag 1998