LEHRSTUHL FÜR BETRIEBSWIRTSCHAFTSLEHRE MIT SCHWERPUNKT LOGISTIK

Planung und Steuerung von Kosten der Softwareentwicklung

Dipl.-Wirtsch.-Ing. Carsten Jacobi

Vollständiger Abdruck der von der Fakultät für Wirtschafts- und Sozialwissenschaften der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Wirtschaftswissenschaften
(Dr. oec.)

genehmigten Dissertation.

Vorsitzender: .....

Prüfer der Dissertation: 1. .....
2. .....
3. .....

Die Dissertation wurde am ..... bei der Technischen Universität München eingereicht und durch die Fakultät für Wirtschafts- und Sozialwissenschaften am ..... angenommen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Seite

Abkürzungsverzeichnis

Abb.	Abbildung
Abs.	Absatz
ACES	Advanced Cost Estimating System
AG	Aktiengesellschaft
AQL	Acceptable Quality Levels
Anm.	Anmerkung
Aufl.	Auflage
Bd.	Band
BFuP	Betriebswirtschaftliche Forschung und Praxis
bzgl.	bezüglich
bzw.	beziehungsweise
ca.	circa
CASE	Computer Aided Software Engineering
CMM	Capability Maturity Model
COCOMO	Constructive Cost Model
CRM	Customer Relationship Management
CSCW	Computer Supported Cooperative Work
d. h.	das heißt
DBW	Die Betriebswirtschaft
DIN	Deutsches Institut für Normung
Diss.	Dissertation
EDI	Eletronic Data Interchange
EDV	Elekrtonische Datenverarbeitung
e. V.	eingetragener Verein
EFQM	European Foundation for Quality Management
EN	Europäische Norm
ERP:	Enterprise Ressource Planning
EQA	European Quality Award
engl.	englisch
erw.	erweitert
etc.	et cetera
et al.	et alii

evtl.	eventuell
F&E	Forschung und Entwicklung
f.	folgende
FAZ	Frankfurter Allgemeine Zeitung
ff.	fortfolgende
FMEA	Failure Mode and Effects Analysis
FPM:	Function-Point Methode
GENESIS	Grundlegende Effektivitätsverbesserung nach einer Sculung in schlanker Produktion, Organisation und Beschaffung
ggf.	gegebenenfalls
GmbH	Gesellschaft mit beschränkter Haftung
HGB	Handelsgesetzbuch
Hrsg.	Herausgeber
HWB	Handwörterbuch der Betriebswirtschaft
IEL	International Electronical Commision
IFPU	International Function Point User Group
IM	Information Management
insb.	insbesondere
io	Industrielle Organisation
ISO	Internationale Organisation für Normung
Jg.	Jahrgang
JUSE	Union of Japanese Scientists and Engineers
Kap.	Kaitel
KMU	kleine und mittlere Unternehmen
krp	Kostenrechnungspraxis
KVP	kontinuierlicher Verbesserungsprozeß
lat.	lateinisch
LOC	Lines of Code
min.	Mnimum
MIT	Massachusetts Institute of Technology
max.	Maximum
Nr.	Nummer
o. Jg.	ohne Jahrgang
o. g.	oben genannten
o. O.	ohne Ort

o. S.	ohne Seiten
o. V.	ohne Verfasser
OBaHP	Organizational Behavior and Human Performance
PRICE	Parametric Review of Information for Costing and Evaluation
QFD:	Quality Function Deployment
QJE	Quarterly Journal of Economics
QM	Qualitätsmanagement
QS	Qualitätssicherung
QZ	Qualität und Zuverlässigkeit
S.	Seite
s.	siehe
s.o.	siehe oben
SE	Simultaneous Engineering
Sp.	Spalte
SCM	Supply Chain Management
SPC	statistical process control
SPICE	Software Process Improvement Capability Determination
SW	Software
TOC	Total-cost-of-ownership
TQM	Total Quality Management
u. a.	unter anderem
USA	United States of America
USDP	Unified Software Developement Process
überarb.	überarbeitet
VDA	Verband der Automobilindustrie e.V.
VDE	Verband Deutscher Elektrotechniker e.V.
VDI	Verein Deutscher Ingenieure
vgl.	vergleiche
WiSt	Wirtschaftswissenschaftliches Studium
WISU	Wirtschaftsstudium
z. B.	zum Beispiel
ZfB	Zeitschrift für Betriebswirtschaft
zfbf	Zeitschrift für betriebswirtschaftliche Forschung
ZFP	Zeitschrift für Forschung und Praxis

1 Einführung

Systematische und methodengestützte Vorgehensweisen im innerbetrieblichen Rechnungswesen und im Software Engineering dienen dazu, komplexe Sachverhalte beherrschbarer zu machen. Die Fortschritte der letzten 30 Jahre im Software Engineering manifestieren sich in einer Vielzahl von Vorgehensmodellen, Sprachkonzepten und Entwicklungstools und fokussieren überwiegend auf den technischen Aspekt der Softwareerstellung.[1] Im innerbetrieblichen Rechnungswesen im allgemeinen und in der Kostenrechnung im besonderen führte die zunehmende Strategieorientierung der Unternehmen zu einem Kostenmanagement, das die Gesamtheit aller Maßnahmen für die frühzeitige und antizipative Beeinflussung von Kostenstruktur und Kostenverhalten sowie die Senkung des Kostenniveaus umfaßt.[2] Doch trotz der stetigen Weiterentwicklung im Bereich der Software-Techniken und der theoretischen Fundierung des Kostenrechnungs-Instrumentariums steht die Softwareentwicklung generell in dem Ruf, ihre Ziele nicht zu erreichen.[3] Standardsoftware kommt häufig später als angekündigt, kundenindividuell erstellte Software wird meist mit Termin- und Budgetüberschreitungen sowie zum Teil gravierenden Qualitätsmängeln ausgeliefert. Untersuchungen der Standish Group haben ergeben, daß 1995 in den USA nur 45% aller Ausgaben für Softwareentwicklungsprojekte geplant waren. Bei den realisierten Softwareprojekten wurde das Budget durchschnittlich um 50% überstiegen. Ein Drittel der Kosten für die Softwareentwicklung wurde für fehlgeschlagene Projekte ausgegeben.[4] Auch wenn der strategische Aspekt der frühzeitigen Vorbelegung eines Marktsegments eine Rolle spielt, um der Abwanderung von Kunden zu anderen Anbietern vorzubeugen, sind die Termin- und Budgetüberschreitungen eklatant, wenn nur jedes elfte Projekt innerhalb

---

1 1969 gründete eine Gruppe von Softwareentwicklern auf der Garmischer Konferenz die Disziplin des Software Engineering mit dem Ziel, Softwareentwicklung als ingenieurmäßiges Vorgehen zu betrachten. Vgl. Elzer, P. F. (1989), S. 183ff.; Baber, R. L. (1992), S. 3ff.

2 Vgl. Reiß, M. / Corsten, H. (1992), S. 1479; Dellman, K. / Franz, K. P. (1994), S. 17; Männel, W. (1997a), S. 3

3 Vgl. Broy, M. et al. (2000), S. 3f.

4 Vgl. Feyhl, A. W. / Feyhl, E. (1996), S. 1; Wildemann, H. (2000a), S. 21 sowie Zillessen, W. (1992), S. 159

der Termin- und Budgetvorgaben realisiert werden kann. Die Kunden sind mit der Software nicht zufrieden und wünschen sich etwa eine andere Funktionalität oder eine bessere Stabilität der Programme. Prägnante Beispiele für eine mangelhafte Softwareentwicklung und deren gravierende Konsequenzen für die nutzenden Unternehmen waren die Explosion der Ariane 5-Rakete aufgrund eines Softwarefehlers, die Abrechnungspanne der Deutschen Telekom bei der Umstellung auf das neue Gebührenmodell zu Beginn des Jahres 1997 sowie der Zeitverzug bei der Einführung eines neuen Computersystems bei der Bundesanstalt für Arbeit im Jahr 2000, der zu monatlichen Mehrkosten in Millionenhöhe führte.

Die Notwendigkeit einer Beherrschung von Softwareentwicklungsaktivitäten wird verstärkt durch die zunehmende Bedeutung von Software. Zum einen nimmt der Einsatz von Software bei industriellen Anwendungen ständig zu. Je nach Funktionsbereich hat Software einen Durchdringungsgrad von 60% - 90% zur Unterstützung der Geschäftsprozesse erreicht.[5] Dieser Trend setzt sich fort mit den Internet-Anwendungen in den Bereichen Business to Business (B to B) und Business to Consumer (B to C), mit denen neuartige Formen von Geschäftsprozessen realisiert werden können, die zu erheblichen Kostensenkungspotentialen führen.[6] Zum anderen steigt der Anteil an Software in Hardwareprodukten zur Realisierung innovativer Produktfunktionalitäten. So haben die Kosten für die Softwareentwicklung beim Eurofighter bereits einen Anteil an 50% der gesamten Entwicklungskosten, der Großteil an Innovationen im Automobil ist auf Software zurückzuführen wie die Motorsteuerung, das Navigationssystem oder das Stabilitätssystem.[7] Software ist somit ein Wettbewerbsfaktor für effiziente Geschäftsabläufe, innovative Produkte und moderne Dienstleistungen.[8]

Die anhaltenden Abweichungen bei der Softwareerstellung führen jedoch zu der Vermutung, daß die Unternehmen offensichtlich mit den derzeitigen Vorgehensweisen und Werkzeugen nicht in der Lage sind, die Komplexität

---

5 Vgl. Wildemann, H. (2000a), S. 13

6 Vgl. Wildemann, H. (2000d), S. 55f.

7 Vgl. Riedl, J. E. / Wirth, W. / Kretschmer, H. (1985), S. 993; Quinn, J. B. et al. (1996), S. 11f.; Wildemann, H. (2000a), S. 14f.

8 Vgl. Lamberti, H.-J. (1998), S. 32ff.; Broy, M. et al. (2000), S. 1f.; Wildemann, H. (2001), S. 21ff.

der Softwareentwicklung zu beherrschen.[9] Neue Entwicklungsumgebungen, Entwicklungssprachen und Architekturkonzepte erleichtern zwar die schnelle Realisierung von Softwaremodulen, können jedoch eine entscheidende Komplexitätsreduzierung des Gesamtsystems zur erfolgreichen Softwareentwicklung nicht herbeiführen.[10] Dieser Trend zu immer umfangreicheren und damit auch komplexeren Softwaresystemen wird verstärkt durch die großen Leistungssteigerungen im Bereich der Computerhardware, die einen Umgang mit immer größeren Datenmengen erleichtern. Die vergangenen fünfzig Jahre galt das Mooresche Gesetz, daß sich alle 18 Monate die Rechnerleistung verdoppelt.[11] Führende Fachleute halten es für wahrscheinlich, daß dieses Gesetz auch die nächsten fünfzig Jahre Gültigkeit haben wird, so daß der Trend zu komplexeren und umfangreicheren Systemen anhalten wird.[12]

Die Intensivierung der Entwicklungsaktivitäten in den Phasen Analyse und Spezifikation[13] und deren Unterstützung mit Hilfe von technisch-orientierten Spezifikationsmethoden zur Erfassung der Kundenanforderungen[14] haben dieses Bild der Kosten- und Termineinhaltung bei Softwareentwicklungsprojekten auch kaum verändert. Die Bestrebungen, einen Großteil der Aufwendungen auf die Spezifikationsphase zu verwenden und auf Basis einer Feinspezifikation mit Hilfe von Entwicklungswerkzeugen eine automatische Programmgenerierung durchzuführen, sind noch in den Anfängen. Die angestrebte Flexibilität, bei Änderungen der Feinspezifikation mit Hilfe des Werkzeugs bedarfsgerecht geänderte Softwareprogramme zu erhalten, hebt die Notwendigkeit einer exakten Erfassung der Kundenbedürfnisse nicht auf. Die Erfahrung wird lediglich, unter der Prämisse, daß die Umsetzung mit Hilfe der neuen Entwicklungswerkzeuge schnell und fehlerfrei realisiert werden kann, in

---

9 Vgl. Hauer, R. (1995), S. 16

10 Vgl. z.B. die Erfahrungsberichte von DeMarco, T. / Lister, T. (1987); Elzer, P. F. (1989)

11 Das Gesetz wurde von Gordon Moore, dem damaligen Vorsitzenden von Intel aufgestellt. Vgl. Evans, P. / Wurster, T. S. (2000), S. 24f.

12 Vgl. Bell, G. / Gray, J. N. (1997), S. 6f.

13 Der Aufwand der Spezifikationsphase beträgt mittlerweile ca. 30% des Gesamtprojektes; vgl. Lannes, A. (2000), S. 295

14 Die Disziplin des Requirement Engineering befaßt sich mit der Umsetzung der Kundenanforderungen in eine technische Spezifikation, vgl. Deifel B. (1998); Für einen Überblick über gängige Spezifikationsmethoden vgl. Beer, J. (2000), S. 27ff.

eine andere Entwicklungsphase verlegt. Die Kernaufgabe, in den frühen Phasen der Softwareentwicklung ein gemeinsames Verständnis zwischen Anwender und Entwickler zu schaffen, bleibt bestehen.

Die Dimension und die Dauer der Software-Krise lassen erkennen, daß die Ursachen und die Lösungen nicht allein auf der technischen Seite zu suchen sind. Trotz einer rasanten Weiterentwicklung der Software-Technik kann die Softwareentwicklung den parallel dazu gestiegenen Anforderungen hinsichtlich Komplexität der Systeme, Zeitdruck und Kundenorientierung nicht genügen.[15] Offensichtlich besteht hier ein Managementproblem. Die Beherrschung von Zeit, Kosten und Qualität, die in vielen Märkten als selbstverständliche Voraussetzung für die erfolgreiche Teilnahme am Wettbewerb angesehen wird, ist in der Softwareentwicklung noch nicht gegeben. Die effiziente Durchführung von Softwareentwicklungsprojekten bedarf neben der Unterstützung durch neue Generationen von Programmiersprachen sowie Entwicklungswerkzeugen und weiterentwickelten Prozeßmodellen vor allem der Anwendung softwarespezifischer Managementkonzepte. Diese Managementkonzepte, die die Kosten als Kernelement zur Planung und Steuerung berücksichtigen, unterstützen softwareentwickelnde Unternehmen, den Entwicklungsprozeß planbar, meßbar und regulierbar zu machen. Sie erlauben eine Kommunikation zwischen Anwender und Entwickler in den frühen Phasen des Softwareentwicklungsprozesses zur kundengerechten Softwareproduktgestaltung und ermöglichen eine kostenorientierte Steuerung während des Entwicklungsprozesses im Sinne eines Controlling zur Einhaltung der geplanten und vorgegebenen Kosten.

1.1 Problemstellung

Der erreichte Stand zur methodischen Unterstützung für ein Kostenmanagement bei der Softwareentwicklung steht im Gegensatz zur hohen Bedeutung des Einsatzes von Softwareprodukten. Es existiert zwar

---

[15] Vgl. Weltz, F. / Ortmann, R. G. (1992), S. 158ff.; Wildemann, H. (1998b), S. 37

eine Vielzahl von Methoden, die durch eine aktive Kostensteuerung eine markt- und anforderungsgerechte Produktgestaltung[16] im Hardwarebereich erfolgreich sicherstellen, jedoch erfordert deren Einsatz im Bereich der Softwareentwicklung eine softwarespezifische Anpassung und Erweiterung. Die Defizite im Kostenmanagement bei der Softwareentwicklung lassen sich daher auf drei zentrale Problemfelder zurückführen:

Die Verfahren zur Kostenschätzung und Kostenverfolgung in der Softwareentwicklung sind nur bedingt geeignet: Ein Kostenmanagement bei der Softwareentwicklung beschränkt sich derzeit auf eine Kostenschätzung zu Beginn des Projektes, eine Kostenbetrachtung nach Ablauf des Softwareentwicklungsprojektes sowie eine Kostenverfolgung, die überwiegend auf einem Plan-Ist-Vergleich basiert und eine Beurteilung der erbrachten Leistungen unberücksichtigt läßt.[17] Eine mitlaufende Kostenkontrolle erfolgt nur in wenigen Fällen[18], eine Controllingkonzeption für Softwareentwicklungsprojekte[19] ist nicht vorhanden. Um die Kosten von Softwareentwicklungsprojekten prognostizieren zu können, existieren in der Praxis eine Reihe von softwarespezifischen Aufwandschätzverfahren.[20] Sie versuchen auf Basis von Schätzgleichungen, deren Einflußfaktoren anhand abgeschlossener Softwareentwicklungsprojekte ermittelt wurden, Aussagen über künftige Aufwände und Kosten zu treffen. Der rasante Fortschritt in der Computertechnik[21] und die hohe Veränderungsgeschwindigkeit in der Softwareentwicklung, die anhand von modernen Sprachkonzepten, neuen CASE-Tools und neuen

---

16 Vgl. insbesondere Horváth, P. / Seidenschwarz, W. (1992), S. 142ff.; Dellman, K. / Franz, K. P. (1994), S. 17ff.; Männel, W. (1997a), S. 5f.; Corsten, H. / Stuhlmann, S. (1997), S. 21f.; Becker, W. (1997), S. 40ff. Zur markt- und anforderungsgerechten Produktgestaltung in der Hardwareentwicklung wird vor allem die Methode des Target Costing eingesetzt; vgl. hierzu Horváth, P. / Niemand, S. / Wolbold, M. (1993), Seidenschwarz, W. (1993), Sakurai, M. (1997), S. 37ff.

17 Vgl. Wildemann, H. (2000c), S. 82

18 Vgl. Spitta, T. (1996), S. 473

19 Vgl. Baumöl, U. (1999), S. 151f.

20 In der Praxis existieren ca. 20 standardisierte Verfahren zur Aufwandschätzung von Softwareentwicklungen sowie entsprechende Abwandlungen (vgl. Wildemann, H. (1998b), S. 62). Gängige Aufwandschätzverfahren in der Softwareentwicklung sind das Constructive Cost Model (COCOMO) nach Boehm, B. (1986) sowie die Function-Point Methode nach Albrecht (vgl. IFPUG (1994)).

21 Vgl. Broy, M. et al. (2000), S. 1

Softwarefunktionalitäten ersichtlich ist, führt zu einer Neuartigkeit und damit verbunden zu Unsicherheiten im Softwareentwicklungsprozeß. Das notwendige Erfahrungswissen der Softwareentwickler für die Beurteilung der Einflußfaktoren in den Schätzgleichungen reicht dann nicht aus, um die Kosten von Softwareentwicklungsprojekten zuverlässig prognostizieren zu können, was anhand der durchschnittlichen Budgetüberschreitungen bei Projekten sichtbar wird.[22] Um eine hohe Aktualität der Schätzgleichungen zu erhalten, ist eine Kostenbetrachtung nach Abschluß der Softwareentwicklungsprojekte und eine permanente Rückkopplung der Ergebnisse zur Anpassung der Einflußfaktoren erforderlich. Da sich die Entwicklungsprojekte je nach Softwaretyp unterscheiden, konnte sich in der Praxis auch kein allgemeingültiges Aufwandschätzverfahren durchsetzen. Vielmehr existiert eine Vielzahl von Aufwandschätzverfahren und deren unternehmensspezifische Anpassung.[23] Aussagefähige Kostenprognosen können somit nur getroffen werden, wenn es sich um ähnliche Softwareprojekte handelt, deren Komplexität beherrschbar und Entwicklungsumgebung bekannt ist, so daß die Parameter der Aufwandschätzverfahren zur Beurteilung der Kosteneinflußgrößen entsprechend beurteilt werden können.[24] Da alle Aufwandschätzverfahren die Kosten von Softwareprojekten über den Umfang und die Bewertung von Einflußgrößen ermitteln, ist eine Marktorientierung bei der Kostenplanung nicht vorhanden. Diese Marktorientierung, die die vom Kunden erlaubten Kosten für die Produktgestaltung berücksichtigt, kann auch nicht durch Schätzgleichungen sichergestellt werden, denen für den Kunden sichtbare Einheiten zugrunde liegen wie die Funktionen bei der Function-Point Methode. Diese Schätzgleichungen erhöhen nur die Transparenz für Anwender und Entwickler bei der Ermittlung der Kosten.

---

22 Die Ergebnisse der dieser Arbeit zugrunde liegenden Untersuchung zeigen, daß der Großteil der befragten Unternehmen Softwareprojekte mit mehr als 20% Termin- und Budgetüberschreitungen realisieren. Marktforscher der Standish Group (Boston) belegen jedes Jahr die geringen Erfolgsquoten amerikanischer Softwareentwicklungsprojekte: Nur 25% der Softwareentwicklungsprojekte können Kosten und Termine einhalten; vgl. CW 29/1998, S. 17

23 Für einen Überblick gängiger Aufwandschätzverfahren und deren zugrunde liegende Methodik vgl. insbesondere Burghardt, M. (1988), S. 128; Knöll, H. D. / Busse, J. (1991), S. 44; Vollmann, S. (1990), S. 23ff.

24 Vgl. Großjohann, R. (1996), S. 134f.

Traditionelle Methoden zur Planung und Steuerung von Kosten für Hardware sind bei Software nur bedingt anwendbar: Software unterscheidet sich von Hardware vor allem durch die Immaterialität und die Komplexität des Produktes[25], die eine Anwendung der traditionellen Methoden im Kostenmanagement für Hardware nur teilweise erlauben.[26] Die betriebswirtschaftliche Kostentheorie und die auf ihr aufbauende Lehre basiert auf den Theorien der Produktionskosten.[27] Die Methoden für ein Kostenmanagement berücksichtigen dabei die Kosten der Entwicklung und die Kosten der Produktion in bezug auf die Kostenstrukturen, das Kostenniveau und den Kostenverlauf. Das Kostenbeeinflussungspotential erstreckt sich nicht nur auf die Entwicklungskosten, sondern auch auf eine regelmäßige Senkung der Produktionskosten.[28] Die Gegenständlichkeit bei der Hardwareentwicklung führt zu meßbaren Zwischenergebnissen und ermöglicht ein Lernen am Objekt. Der wirkungsvolle Einsatz der Methoden im Kostenmanagement basiert auf diesen meßbaren Zwischenergebnissen.[29] Im Softwareentwicklungsprozeß ist eine einfache Rückkopplung durch ein Lernen am Objekt aufgrund der Immaterialität nicht gegeben. Die sequentielle Abfolge des Entwicklungs- und Produktionsprozesses bei Hardwareprodukten wandelt sich zu einem Softwareentwicklungsprozeß, der als Ergebnis ein lauffähiges Softwareprodukt aufweist, das häufig während des Betriebes beim Kunden noch weiterentwickelt wird. Einerseits entfällt die in industriellen Entwicklungs- und Produktionsprozessen meist kritische Schnittstelle zwischen Konstruktion und Fertigung, andererseits entfällt damit auch die bei diesem Prozeßschritt gegebene Lernmöglichkeit am Produkt. Die Gegenständlichkeit in der Hardwareentwicklung und die damit verbundenen Meilensteine wie z. B. der Prototypenbau unterstützen die Meßbarkeit des Projektfortschritts. In der Softwareentwicklung

---

25 Vgl. Wildemann, H. (1993b), S. 155ff.; Broy, M. (1996c), S. 5f.; Zur Darstellung der Unterschiede von Informationsökonomie und Güterökonomie vgl. Evans, P. / Wurster, T. S. (2000), S. 21 sowie die dort angegebene Literatur.

26 Vgl. Sakurai, M. (1997), S. 77

27 Vgl. Dorn, G. (1992), S. 97ff.

28 Für die Ansätze zum Kostenmanagement in Produktentstehung und Produktion vgl. u. a. Dellman, K. / Franz, K. P. (1994); Jehle, E. / Willeke, M. (1996); Männel, W. (1997a); Hagenloch, T. (1997); Schuh, G. (1997).

29 Die Vorgehensmodelle zur Automobilentwicklung weisen alle fest definierte Meilensteine bzw. Quality Gates im Entwicklungsprozeß auf. Für einen Überblick der Vorgehensmodelle zur Automobilentwicklung vgl. Gentner, A. (1995), S. 49ff.

erschwert die Immaterialität des Produktes die Kontrolle der Leistungserstellung.[30]

Damit die Kosten als Basis zur Planung und Steuerung im Softwareentwicklungsprozeß dienen können, ist ein gemeinsames Kostenverständnis bei Kunde und Entwickler erforderlich. In der Hardwareentwicklung erfolgt diese Diskussion anhand der materiellen Komponenten bzw. der Subsysteme des Produktes.[31] Durch die fehlende Gegenständlichkeit von Software fällt es dem Entwickler schwer, dem Kunden die mit der Realisierung verbundenen Aufwendungen verständlich zu machen. Umgekehrt kann der Kunde nur schwerlich in die Diskussion der Systemkonfiguration mit einbezogen werden. Da für Softwareprogramme eine Vielzahl von individuellen Realisierungsmöglichkeiten besteht, ist der Spielraum sehr groß.[32] Erschwerend kommt hinzu, daß Software im Gegensatz zu Hardware keine Toleranzen zuläßt. Damit stellt die Softwareentwicklung höhere Anforderungen an die Fehlerfreiheit des Entwicklungsprozesses als andere technische Produkte. Schon kleine Fehler können unter gewissen Umständen zu unerwarteten, nicht kontrollierbaren Systemzuständen führen. Dabei bestimmen die Anwendungsgebiete, für die die Software eingesetzt wird, die hinsichtlich Fehlerfreiheit an den Entwicklungsprozeß zu stellenden Anforderungen.

Die Immaterialität von Software erfordert eine Kommunikationsplattform, die die Sicht der Anwender und Entwickler zusammenbringt. Die vom Kunden erlaubten Kosten können dann den geschätzten Kosten gegenübergestellt werden, damit ein gemeinsames Kostenverständnis die kostenorientierte Planung und Steuerung im Softwareentwicklungsprozeß sicherstellt. Abb. 1-1 zeigt die Unterschiede zwischen Hardware und Software. Sie verdeutlichen die softwarespezifischen Anforderungen an die Methoden im Kostenmanagement: Die fehlende Gegenständlichkeit sowie eine erschwerte Unterteilung der Software in Module bzw. Komponenten von Softwareprodukten tragen wesentlich zu einer erhöhten Komplexität

---

30 Vgl. Babini, M. (1993), S. 27

31 Zur Vorgehensweise der Zielkostenfestlegung vgl. insbesondere die Funktionswertmethode von Tanaka, M. (1989), S. 56ff. sowie Seidenschwarz, W. (1991), S. 200ff.; Tanaka, T. (1993), S. 4ff.; Claassen, U. / Hilbert, H. (1994), S. 120ff.; Nedeß, C. / Stalleicken, U. (1998), S. 207

32 Vgl. Döttinger, K. (1988), S. 970

und zu hohen Unsicherheiten im Softwareentwicklungsprozeß bei. Die durch die Eigenschaften des Produktes Software erhöhten Anforderungen an den Entwicklungsprozeß manifestieren sich in der Vielzahl unterschiedlicher Vorgehensmodelle zur Softwareentwicklung.[33] Mit Hilfe der Vorgehensmodelle soll eine hohe Prozeßqualität erreicht werden, die eine Produktqualität sicherstellt. Die in technischer Hinsicht erfolgte Anpassung an die spezifischen Merkmale des Produktes Software ist in betriebswirtschaftlicher Hinsicht noch nicht erfolgt. Für ein Kostenmanagement bei der Softwareentwicklung bedarf es daher einer Anpassung und Erweiterung der bestehenden Methoden zur Berücksichtigung der spezifischen Eigenschaften des Produktes Software.

	Hardware	Software
Gegenständlichkeit	**Gegenständlichkeit** Einfache Rückkopplung zw. Entwicklung und Produktion durch Lernen am Objekt	**Fehlende Gegenständlichkeit** Ein Lernen am materiellen Objekt ist nicht möglich
Komplexität	**Komplexität wird beherrschbar durch einen modularen Produktaufbau** Komponenten- / Systemstruktur orientiert sich an den Funktionen des Produktes	**Komplexität nur beherrschbar bei Unterteilung von Software in Komponenten / Objekte** Struktur nicht vorgegeben
Veränderbarkeit	**Schwere Veränderbarkeit** Eine Veränderung erfordert Anpassungen im Bereich Konstruktion und Produktionsmittel	**Leichte Veränderbarkeit** Quellcode ist leicht veränderbar Eine Überprüfung der Auswirkungen erfordert jedoch entsprechenden Testaufwand
Lösungsrahmen	**Lösungsrahmen** Physikalische Gesetzmäßigkeiten geben einen Rahmen für mögliche Lösungen	**Kein Lösungsrahmen** Dieselben Softwarefunktionen können auf unterschiedliche Art und Weise realisiert werden
Korrektheit	**Toleranzen** Qualitätssicherung mit Toleranzbereichen	**Keine Toleranzen** Qualitätssicherung prüft Fehlerfreiheit des Programms

Abb. 1-1 : Unterschiede zwischen Hardware und Software in Anlehnung an Wildemann[34]

Eine durchgängige Planung und Steuerung von Kosten der Softwareentwicklung existiert nicht: Für eine durchgängige Planung und Steuerung von Kosten der Softwareentwicklung ist eine systematische Kostenschätzung um Methoden zur Sicherstellung einer marktorientierten Kosten-

33 Für einen Überblick über gängige Vorgehensmodelle vgl. u. a. Wildemann, H. (1998b), S. 39f. sowie Broy, M. et al. (2000), S. 43ff.

34 Vgl. Wildemann, H. (1999a), S. 23

politik[35] und einer kostenorientierten Steuerung zu erweitern. Ein Vergleich der gängigen Vorgehensmodelle der Automobilentwicklung mit der Softwareentwicklung verdeutlicht den fehlenden Einsatz von Methoden für ein Kostenmanagement bei der Softwareentwicklung.[36] Bei der Automobilentwicklung sind Methoden für eine marktorientierte Produktgestaltung fester Bestandteil des Prozeßmodells[37]: Eine Kostenprognose erfolgt dabei in Zusammenhang mit einem Target Costing für eine marktorientierte Vorgabe von Zielkosten. Mit Hilfe eines Cost Benchmarking werden frühzeitig Kostensenkungspotentiale erschlossen. Die Make or Buy-Fragestellung zu Beginn des Entwicklungsprozesses bestimmt die optimale Leistungstiefe, um eine kostengünstige Entwicklung und Produktion sicherzustellen. Die gängigen Vorgehensmodelle der Softwareentwicklung weisen nur vereinzelt auf Aktivitäten für die Kostenschätzung mit Hilfe von Aufwandschätzverfahren und für die Kostenverfolgung im Rahmen des Projektmanagement hin.[38] Eine methodisch durchgängige marktorientierte Kostenplanung und ein darauf aufbauendes Kosten-Controlling existiert in der Softwareentwicklung nicht.[39]

1.2 Stand der Literatur

Aus den bisherigen Ausführungen geht hervor, daß die Planung und Steuerung von Kosten der Softwareentwicklung die Erfüllung folgender Voraussetzungen erfordert:

---

35 Vgl. Männel, W. (1997a), S. 3

36 Vgl. Wildemann, H. (2000a), S. 25

37 Ein Target Costing ist fester Bestandteil vieler Automobilentwicklungsprozesse; Für Toyota vgl. Tanaka, T. (1993), S. 5; Für Volkswagen vgl. Claassen, U. / Hilbert, H. (1994), S. 118ff. Einen Überblick über die Vorgehensmodelle zu Automobilentwicklung gibt u. a. Gentner, A. (1995), S. 43ff. sowie Wildemann, H. (1999a), S. 9ff.

38 Insbesondere das V-Modell beschreibt Methoden zur Kostenschätzung und zur Kostenverfolgung im Rahmen des Projektmanagements; Vgl. Bundesministerium des Inneren, KBSt (1997), Methodenzuordnung.

39 Auch in der Hardwareentwicklung und –produktion wird auf die Lücke zwischen theoretischer Fundierung und praktischer Anwendbarkeit eines Kostenrechnungs-Instrumentariums hingewiesen; vgl. Fröhling, O. (1994b), S. 1ff.

- Kenntnis über die wesentlichen Einflußgrößen auf die Kosten im Softwareentwicklungsprozeß,
- Bereitstellung von Methoden zur Kostenplanung auf Basis systematisierter Kosteneinflußgrößen, die als Grundlage für eine durchgängige Planung und Steuerung dienen,
- Bereitstellung von Methoden zur marktorientierten Softwareproduktgestaltung,
- Bereitstellung von Meßgrößen zur kostenorientierten Steuerung im Softwareentwicklungsprozeß,
- Kenntnisse über Leitlinien und Strategien zur frühzeitigen Beeinflussung von Kostenstruktur, Kostenverhalten und Kostenniveau für eine anforderungsgerechte Softwareentwicklung,
- Gestaltung eines kostenorientierten Berichtswesens und
- Bereitstellung von Methoden zum Kosten-Controlling bei der Softwareentwicklung.

Eine Auswertung der Literatur dient der Überprüfung, inwieweit die bestehenden Ansätze die obigen Anforderungen an ein Kostenmanagement bei der Softwareentwicklung bereits erfüllen.

Ansätze der Aufwandschätzung in der Softwareentwicklung

Erste Ansätze für eine Kostenplanung von Softwareentwicklungsprojekten liefern die softwarespezifischen Verfahren zur Aufwandschätzung. Diese Verfahren lassen sich dadurch charakterisieren, daß sie auf der Kenntnis von Zusammenhängen zwischen Kostenelementen und Einflußgrößen aufbauen und die Möglichkeit bieten, in Form von „Wenn-Dann-Aussagen" Bilder zukünftiger Kostenverläufe zu entwerfen. Der Faktorverbrauch für die Entwicklung eines bestimmten Softwaretyps kann somit durch Schlußfolgerungen aus Gesetzmäßigkeiten hergeleitet werden, die das Verhalten des Systems und der Umwelt abbilden. Von den ca. 20 Verfahren für die Aufwandschätzung[40] von Softwareentwicklungsprojekten und den dazugehörigen Abwandlungen haben in der Praxis vor allem das

---

40 Für einen Überblick über die Aufwandschätzverfahren vgl. Noth, T. / Kretzschmair, M. (1986); Burghardt, M. (1988), S. 128ff.; Vollmann, S. (1990), S. 23ff.; Knöll, H. D. / Busse, J. (1991), S. 44;

Constructive Cost Model (COCOMO) und die Function-Point Methode breite Anwendung gefunden.[41] Das Verfahren von COCOMO ist ein parametrisches Aufwandschätzverfahren und benutzt als Ausgangsgröße die geschätzte Anzahl von Lines of Code. Der aufwandsbeeinflussende Effekt der verwendeten Programmiersprache und der Entwicklungsumgebung wird durch Einflußparameter berücksichtigt, die als Kostentreiber bezeichnet werden.[42] Die Function-Point Methode verwendet als Grundlage für die Aufwandschätzung die für den Benutzer sichtbaren Funktionen der Software, wodurch die Bewertung des Aufwandes unabhängig von der Programmiersprache erfolgt. Die Komplexität des Programms wird ebenfalls über Einflußparameter berücksichtigt. Anhand einer Erfahrungskurve kann dann aus dem mit Einflußgrößen bewerteten Umfang der Aufwand in Mann-Monaten abgeleitet werden.[43] Die weite Verbreitung der Function-Point Methode in Wissenschaft und Praxis manifestiert sich in der Vielzahl spezifischer Anpassungen zur Verbesserung der Einsatzfähgkeit.[44]

Die den Aufwandschätzverfahren zugrunde liegenden Umfangsmaße dienen gleichzeitig zur Beurteilung der Produktivität in der Softwareentwicklung.[45] Da die Produktivitätsmessung auf dem erstellten Quellcode bzw. den realisierten Softwarefunktionen basiert, kann sie erst sehr spät im Softwareentwicklungsprozeß erfolgen und bietet somit im Rahmen des Kosten-Controlling während der Softwareentwicklung wenig Unterstützung bei der Überprüfung der erbrachten Leistungen. Weitere Kritikpunkte an den Aufwandschätzverfahren sind der große subjektive Einfluß des Schätzers bei der Beurteilung der Einflußparameter sowie die nicht immer gegebene Vollständigkeit der Einflußparameter für die Aufwandsbeurteilung der verschiedenen Softwaretypen.[46] Die Berücksich-

41 Vgl. Knöll, H. D. / Busse, J. (1991), S. 43; Wildemann, H. (1998b), S. 62

42 Vgl. Boehm, B. (1986), S. 85ff.

43 Vgl. IFPUG (1994); Knöll, H. D. / Busse, J. (1991), S. 45ff.; Burghardt, M. (1988), S. 162; Lehner, F. (1992), S. 31f.

44 Zur Anpassung der Function-Point Methode in bezug auf objektorientierte Softwaretechniken vgl. u. a. Kemerer, C. F. / Porter, B. S. (1992), S. 1011; Jeffery, D. R. / Low, G. C. / Barnes, M. (1993), S. 529ff.

45 Vgl. Boehm, B. (1986), S. 568ff; Jones, C. (1991), S. 123ff.; Müller, R. / Ruland, D. / Hoch, D. J. / Klosterkemper, B. (2000), S. 84

46 Vgl. Vollmann, S. (1990), S. 73ff.; Knöll, H. D. / Busse, J. (1991), S. 38ff.

tigung der unterschiedlichen Einflußgrößen der Softwareentwicklungstypen und der Entwicklung höherer Programmiersprachen haben in der Praxis zu einer Vielzahl von Abwandlungen der Verfahren wie die Object Point Methode zur Schätzung der Entwicklungskosten objektorientierter Software geführt.[47] Alle Aufwandschätzverfahren sind jedoch geprägt durch eine traditionelle Betrachtungsweise bei der Kostenplanung, die auf der Gleichung „Kosten + Gewinn = Preis" aufbaut. Eine Marktorientierung für die anforderungsgerechte Softwareproduktgestaltung wird mit diesen Verfahren nicht unterstützt.[48]

Ansätze der Kostenplanung

Neben den Verfahren zur Aufwandschätzung in der Softwareentwicklung existieren in der Praxis weitere allgemeine Ansätze zur Kostenschätzung. Die grundsätzlichen Vorgehensweisen zur Kostenschätzung können in analytische und synthetische Vorgehensweisen unterschieden werden.[49] Analytische Methoden basieren auf einer Input-Output-Orientierung. Die Software wird in Elemente / Aktivitäten und deren Beziehungen zueinander aufgegliedert und der zugehörige Faktorverbrauch prognostiziert. Die Bewertung dieser Inputfaktoren und die Addition der Kostenelemente ergeben die direkten Kosten. Zur Berechnung der Gesamtkosten können die indirekten Kosten über differenzierte Zuschlagsfaktoren bestimmt werden. Der synthetischen Vorgehensweise liegt dagegen eine ganzheitliche Betrachtung der Software mit wenigen Kenngrößen zugrunde. Ziel ist die Definition einer Funktion, die eine Beziehung zwischen den technischen Systemparametern als unabhängige Variablen und den Softwarekosten als abhängige Variablen herstellt. Diese Vorgehensweise basiert auf einer Output-Input-Orientierung, die als Voraussetzung die Definition und Meßbarkeit von Variablen hat.[50] Die Bestimmung der Parameter erfolgt mit Hilfe statistischer Methoden und Kalkulationsverfahren auf Basis abgeschlossener Entwicklungsprojekte. Die allgemeinen Kostenschätzmodelle wie z. B. PRICE-H oder ACES

47 Vgl. Sneed, H. M. (1996), S. 133
48 Vgl. Wildemann, H. (1998b), S. 63
49 Vgl. Wildemann, H. (1982), S. 128ff.
50 Vgl. Wildemann, H. (2000a), S. 40f.

liefern auch bei der Softwareentwicklung gute Schätzergebnisse, wenn entsprechende Erfahrungsdaten vorhanden sind und ein ähnliches Softwareprodukt bereits entwickelt wurde.[51]

Neben den parametrischen Kostenschätzmodellen existieren noch Ansätze zur Kostenschätzung im Bereich der Expertensysteme und der neuronalen Netze. Ein Expertensystem wird definiert als ein „Programmsystem, das Wissen über ein spezielles Gebiet speichert und ansammelt, aus dem Wissen Schlußfolgerungen zieht und zu konkreten Problemen des Gebietes Lösungen anbietet."[52] Expertensysteme basieren auf Erfahrungen ähnlicher Gegebenheiten. Die Summe der Erfahrungen, die der Kostenschätzer durch die subjektive Beurteilung der Kostentreiber in die Kostenschätzung mit einfließen läßt, kann dabei nicht mit berücksichtigt werden.[53] Die Wissensakquisition ist somit ein zentraler Schwachpunkt des Expertensystem-Projektes[54], das vor allem bei dem hohen Neuigkeitsgrad von Softwareentwicklungsprojekten diese Vorgehensweise zur Kostenschätzung als ungeeignet erscheinen läßt. Neuronale Netze finden seit einigen Jahren vor allem in der konstruktionsbegleitenden Kalkulation eine nennenswerte Beachtung.[55] Dabei ähnelt die Vorgehensweise bei neuronalen Netzen den parametrischen Verfahren, da Regressionsanalysen die Grundlage für das Kostenschätzmodell bilden. Prinzipielle Vorteile neuronaler Netze gegenüber parametrischen Kostenschätzverfahren sind daher nicht erkennbar.[56] Neuronale Netze werden gerne favorisiert, da sie die Chance bieten, auch bei völlig unbekanntem Einfluß auf die Schätzgröße eine Vielzahl von Parametern zu modellieren. Jedoch geht die Transparenz, die eine systematische und damit auch nachvollziehbare Vorgehensweise mit sich bringt, bei der Kostenschätzung mit neuronalen Netzen verloren.[57] Diese Transparenz ist in der Softwareentwicklung vor dem Hintergrund möglicher Iterationen bei Änderungen der Kundenwünsche und gegenüber dem Auftraggeber jedoch von großer Bedeutung.

---

51 Vgl. Prem, H. (1994), S. 27f.; Rechberg, U. v. (1997), S. 99ff.
52 Vgl. Claus, A. / Schwill, V. (1993), S. 244
53 Vgl. Rechberg, U. v. (1997), S. 85
54 Vgl. Zelewski, S. (1991)
55 Vgl. Schaal, S. (1991), Kap. 7.4.; Zarefar, H. / Goulding, J. R. (1992), S. 180; Becker, J. / Prischmann, M. (1994), S. 167ff.
56 Vgl. Rechberg, U. v. (1997), S. 96
57 Vgl. Rechberg, U. v. (1997), S. 98

Es läßt sich festhalten, daß sowohl die softwarespezifischen als auch die allgemeinen Ansätze zur Kostenschätzung aussagefähige Kostenprognosen bei vorhandenen Erfahrungsdaten für ähnliche Softwareentwicklungsprojekte treffen können. Die hohe Innovationsdynamik in der Softwareentwicklung führt dazu, daß die bestehenden Kostenschätzmodelle die zu erwartenden Kosten nur sehr vage abschätzen können und die subjektive Beurteilung der Einflußgrößen durch den Kostenschätzer von großer Bedeutung bleiben wird. Daher kann auch nicht ein Kostenschätzmodell als besonders geeignet identifiziert werden. Hinzu kommt, daß bei den diskutierten Ansätzen zur Kostenschätzung eine frühzeitige Marktorientierung während der Kostenplanung nicht erkennbar ist. Auch wird die Planung der Kosten als ein isolierter Prozeß betrachtet. Eine durchgängige Planung und Verfolgung der Kosten über den gesamten Entwicklungsprozeß hinweg auf Basis der systematisierten Kosteneinflußgrößen wird derzeit nicht unterstützt.

Ansätze der Kostenrechnung

Die Ansätze der Kostenrechnung basieren auf den Theorien der Produktionskosten.[58] Die Kostenrechnung mit den Teilgebieten Kostenarten-, Kostenstellen- und Kostenträgerrechnung bilden mit den Kosteninformationen die Grundlage zur Planung, Steuerung und Kontrolle.[59] In der Kostenträgerrechnung werden somit sämtliche in der Kostenartenrechnung erfaßten und in der Kostenstellenrechnung verrechneten Kosten den Kostenträgern zugeordnet. Kostenträger können dabei Softwareprojekte oder Softwareprodukte sein. Aufgabe der Kostenrechnung ist die Kalkulation, die Wirtschaftlichkeitskontrolle und die Fundierung und Überwachung von Entscheidungen. In der Softwareentwicklung findet die Kalkulation im Rahmen der Kostenschätzung zu Beginn eines Projektes Anwendung. Eine aussagefähige Kalkulation hängt dabei entscheidend von der Identifizierung und Ermittlung der relevanten Kosten und ihrer Zurechnung zu einem Objekt

---

58 Vgl. Dorn, G. (1992), S. 97ff.

59 Vgl. Männel, W. / Hummel, S. (1986), S. 3; Coenenberg, A. G. (1993), S. 25; Schweitzer, M. / Küpper, H. U. (1995), S. 4f.

ab.[60] Die Ermittlung und Modifikation der Kalkulationsgrößen erfolgt nach Abschluß eines Projektes für eine Verbesserung der Kostenschätzung bei weiteren Projekten. Da die Softwareentwicklungskosten durch Personalkosten dominiert werden, setzen sich die Kalkulationssätze aus den Personalkosten je Mengeneinheit zusammen. Eine methodische Unterstützung zur Ermittlung der Kalkulationssätze bietet die Kostenträgerrechnung mit ihren unterschiedlichen Kalkulationsverfahren[61] sowie die Prozeßkostenrechnung mit der Identifikation der relevanten Kostentreiber.[62] Die Bestimmung der Mengeneinheit hängt von der Beurteilung des Entwicklungsaufwandes ab, der durch die Kosteneinflußgrößen der Softwareentwicklung bestimmt wird. Ein Defizit besteht in der Systematisierung der Kosteneinflußgrößen in bezug auf Produkt- und Prozeßeinflußgrößen, damit eine Kostenrechnung eine kostenorientierte Planung und Steuerung der Aktivitäten zur Softwareentwicklung unterstützen kann. Die Unterteilung in Produkt- und Prozeßeinflußgrößen verdeutlicht die Gestaltungsparameter für die kundengerechte Softwareproduktentwicklung. Ein geschlossenes System der Kostenrechnung für Software von der Kostenarten- über die Kostenstellen- bis hin zur Kostenträgerrechnung in Verbindung mit einem Aufwandschätzverfahren, das die relevanten Gestaltungsparameter transparent darstellt, wird in der Praxis kaum unterstützt.[63] Somit kann eine Kostenrechnung auch keine Ziele auf Basis von Produkt- und Prozeßmerkmalen vorgeben, um die Kosten in Verbindung mit den erbrachten Leistungen zu verfolgen. Eine Kostenverfolgung besteht heute überwiegend aus der Verrechnung der geleisteten Stunden der Entwickler in Form einer Mitkalkulation.[64] Die verursachungsgerechte Zuordnung der Kosten der Softwareentwicklung in Verbindung mit den erbrachten Leistungen weist somit ein weiteres Defizit auf. Im industriellen Bereich hat sich gezeigt, daß die klassische

---

60 Vgl. Wildemann, H. (1982), S. 6

61 Für einen Überblick über gängige Kalkulationsverfahren vgl. Männel, W. / Hummel, S. (1986), S. 265ff.; Kilger, W. (1980), S. 305.

62 Zur Anwendung der Prozeßkostenrechnung in der Kostenplanung vgl. Kulawiak, W. (1992), S. 205ff.; Schiller, U. / Lengsfeld, S. (1998), S: 532ff.

63 In der Literatur kann nur INVAS (Integriertes Verfahren zur Aufwand-Schätzung) als ein Ansatz für ein Aufwandschätzverfahren identifiziert werden, das ein geschlossenes System von Aufwandschätzung und Kostenrechnung bietet; Vgl. Berkhoff, H. / Blumenthal, P. (1983), S. 407ff.

64 Vgl. Riedl, J. E. / Wirth, W. / Kretschmer, H. (1985), S. 994; Wildemann, H. (1999c), S. 74f.

Kostenarten-, Kostenstellen- und Kostenträgerrechnung durch den überproportionalen Anstieg der Gemeinkosten an Aussagekraft verliert. Bisher wurden die angefallenen Gemeinkosten über Verrechnungssätze pro Bezugsgrößeneinheit über die Kostenstellenrechnung den Kostenträgern zugeordnet. Die Methode der Prozeßkostenrechnung ermöglicht über die Identifikation von Kostentreibern eine verursachungsgerechte Kostenzuordnung.[65] Aufgrund der Immaterialität von Software ist bei der Wahl der Bezugsgrößen für eine verursachungsgerechte Zuordnung der Kosten noch Forschungsbedarf erkennbar.

Ansätze des Kostenmanagement

Die zunehmend strategische Ausrichtung der Kostenrechnung führte zu den Ansätzen des Kostenmanagements. Diese Ansätze basieren auf den Grundlagen der Kostenrechnung und erweitern diese um eine langfristige Betrachtungsperspektive zur „Gestaltung der Programme, Potentiale und Prozesse in einer Unternehmung nach Kostenkriterien.“[66] In der Literatur wird der Begriff des Kostenmanagements mit seinen Anwendungsbereichen sehr breit diskutiert: Die Ansätze des strategischen Kostenmanagements erweitern die klassische Produktkalkulation um eine frühzeitige Marktorientierung zu einer produkt-, prozeß- und ressourcenorientierten Kostenpolitik.[67] Instrumente wie beispielsweise das Target Costing, das Benchmarking, die entwicklungsbegleitende Kostenrechnung sowie die Wertanalyse unterstützen ein frühzeitiges Kostenmanagement. Das operative Kostenmanagement entspricht den Ansätzen der traditionellen Kostenrechnung mit den Funktionen des Controlling aller Aktivitäten im Wertschöpfungsprozeß.[68] Instrumente, die zur Kostenverfolgung Anwendung finden, sind die Prozeßkostenrechnung, die Earned-Value-Analyse oder die Meilensteintrend-Analyse zur Projektverfolgung, der ausgewogene Berichtsbogen sowie eine Visualisierung und ein Monitoring der erbrachten Leistung. Die in der Literatur beschriebenen Anwendungsbeispiele der Instrumente im

---

65 Vgl. Horváth, P. / Renner, A. (1990), S. 100ff.; Wildemann, H. (1997b), S. 72f.
66 Vgl. Dellman, K. / Franz, K. P. (1994), S. 17; Reiß, M. / Corsten, H. (1992), S. 1478f.
67 Vgl. Horvárth, P. (1990), S. 178f.; Männel, W. (1997a), S. 3;
68 Vgl. Horvárth, P. (1990), S. 189; Dellman, K. / Franz, K. P. (1994), S. 18

Kostenmanagement beziehen sich vor allem auf den Hardwarebereich. Als etablierteste Methode für ein frühzeitiges Kostenmanagement gilt sicherlich das Target Costing. Wohingegen dies in Japan weite Verbreitung findet, sind in Europa und Amerika vor allem Anwendungen in den Bereichen der Automobilindustrie und der elektrotechnischen Industrie zu finden.[69] Erste Ansätze zur Übertragung der Methodik auf den Dienstleistungsbereich sind nur vereinzelt erkennbar.[70] Ein Kostenmanagement in der Softwareentwicklung weist somit noch Forschungsbedarf auf.[71] Zur Anwendung eines frühzeitigen Kostenmanagements und eines anschließenden Kosten-Controlling ist eine Anpassung und Erweiterung der Instrumente erforderlich. Dabei müssen die spezifischen Eigenschaften des Produktes Software, die sich durch die Komplexität und die Immaterialität und die damit verbundenen Schwierigkeiten bei der Abbildung der Kundenanforderungen ergeben, Berücksichtigung finden.[72]

Ansätze des Projektmanagements

Weitere Ansätze für eine kostenorientierte Planung und Steuerung von Softwareentwicklungsprojekten liefert das Projektmanagement. Das Projektmanagement in der Softwareentwicklung dient der Steuerung des Entwicklungsprozesses, wie es für die termingerechte Abwicklung von einem oder mehreren Softwareentwicklungsprojekten erforderlich ist.[73] Das Softwareprojektmanagement umfaßt dabei die Planung, die Organisation der Abwicklung, das Controlling des Softwareentwicklungsprozesses sowie das projektspezifische Informations- und Berichtswesen. Für ein aussagefähiges Controlling sind dazu die Zieldimensionen Zeit, Qualität (sowohl Prozeß- als auch Produktqualität) und Kosten heranzuziehen.[74] Aufgaben in der Planungsphase sind die Definition der am Projekt beteiligten Bereiche, die Abschätzung und Reservierung der benötigten

---

69 Vgl. Horváth, P. / Niemand, S. / Wolbold, M. (1993), S. 23; Rössler, F. (1996), S. 79ff.
70 Vgl. dazu die Arbeiten von Niemand, S. (1996) und Baumöl, U. (1999).
71 Vgl. Sakurai, M. (1997), S. 77
72 Vgl. Rössler, F. (1996), S. 94ff.
73 Vgl. Aeberhard, K. (1992), S. 385; Wildemann, H. (1998b), S. 45; Broy, M. et al. (2000), S. 55ff. sowie die dort angegebene Literatur.
74 Vgl. Wildemann, H. (1997b), S. 27

Ressourcen sowie die detaillierte Planung der Vorgehensweise und des Zeitablaufs des Projektes. Die Aufgaben der Realisierungsphase sind die Koordination und die Steuerung der Prozesse des Software-Engineering. Neben der Überwindung der Schnittstellenproblematik ist in der Realisierungsphase das Projektcontrolling von besonderer Bedeutung, das zu jedem Zeitpunkt des Entwicklungsprozesses den Leistungsstand eines Projektes, gegliedert nach den Controllinggrößen, feststellen muß.[75] Die Probleme im Projektmanagement für Software liegen in dem Vorgehen bei der Anwenderintegration in den Entwicklungsprozeß, was bislang weder methodisch ausreichend berücksichtigt wurde noch konnten geeignete organisatorische Lösungen entwickelt werden. Problematisch ist ferner die Messung des Projektfortschrittes, die durch die Immaterialität der Software und eine unscharfe Phasendefinition des Entwicklungsprozesses erschwert wird, um Soll-Ist-Abweichungen frühzeitig zu erkennen. Forschungsbedarf existiert daher in der Beschreibung eines adaptiven, kostenorientierten Steuerungsprozesses, dessen Ablauf sich an die konkreten Anforderungen des jeweiligen Projektes dynamisch anpassen kann und der eine konsequente Kundenorientierung auch innerhalb der Entwicklung ermöglicht. Des weiteren sind die durch ein marktorientiertes Kostenmanagement vorgegebenen Zielkosten für den Softwareentwicklungsprozeß in eine Controlling-Konzeption zu integrieren, um eine durchgängige kostenorientierte Planung und Steuerung zu ermöglichen.[76]

Ansätze des Qualitätsmanagements

Ansätze zur Unterstützung einer Marktorientierung im Kostenmanagement sind im Qualitätsmanagement zu finden. Ziel des Qualitätsmanagements ist die Ableitung der Qualitätsanforderungen und die Sicherstellung der Erfüllung der Kundenwünsche durch die Betrachtung der Produkt- und Prozeßqualität. Dies umfaßt alle Maßnahmen zur Sicherung der geforderten Prozeßqualität für eine markt- und anforderungsgerechte Softwareproduktgestaltung.[77] Das Qualitätsmanagement unterteilt sich

---

75 Vgl. Michel, R. (1996), S. 17; Wildemann, H. (1999c), S. 63

76 Vgl. Wildemann, H. (1998b), S. 45f.

77 Vgl. hierzu Döttinger, K. / Höhler, B. (1994), S. 877f.; Wildemann, H. (1998b), S. 52; Broy, M. et al. (2000), S. 85f.; Mieth, A. (2000), S. 275

dabei in die Qualitätsplanung und das Qualitätscontrolling. Die Qualitätsplanung gibt die Soll-Werte der zu erreichenden Qualität entsprechend den Kundenanforderungen vor und legt den Einsatz der Instrumente für das Qualitätsmanagement fest. Das Qualitätscontrolling übernimmt die Steuerung der Qualität über den gesamten Entwicklungsprozeß eines Softwareprodukts. Hierzu sind die Ist-Werte der zu erreichenden Qualität zu bestimmen, mit den Soll-Vorgaben zu vergleichen und Maßnahmen bei Abweichungen festzulegen. Das Qualitätsmanagement liefert somit Ansätze für ein Kostenmanagement bei der Softwareentwicklung in bezug auf die Ermittlung und Dokumentation der Kundenanforderungen, deren Umsetzung in Produktmerkmale und bei der Steuerung des Softwareentwicklungsprozesses. Methodische Unterstützung für ein Qualitätscontrolling bieten Quality Gates.[78] Sie stellen Entscheidungspunkte dar, um eine Projektsteuerung im Sinne der definierten Qualitätskriterien sicherzustellen. Mit Hilfe eines dreistufigen Freigabeprozesses in Analogie zu einer Ampelschaltung kann eine kontrollierte Weiterentwicklung der Software gesteuert werden. Zur Umsetzung der Kundenanforderungen in Produktmerkmale bietet das Quality Function Deployment (QFD) methodische Unterstützung.[79] QFD kommt häufig in Zusammenhang mit einem Target Costing zum Einsatz, um entsprechend den Anforderungen des Kunden erlaubte Kosten für Produktfunktionalitäten abzuleiten.[80] In der Literatur konnten Beispiele für die integrierte Anwendung von QFD im Rahmen des Target Costing in der Softwareentwicklung nicht gefunden werden. Das Problem liegt darin, daß die Gegenüberstellung der vom Kunden erlaubten Kosten mit den geschätzten Kosten der Entwickler eine Vergleichsbasis erfordert, die aufgrund der Immaterialität von Software nur durch entsprechende Transformationen herzustellen ist.

Ansätze des Requirement Engineering

Ähnlich wie das Qualitätsmanagement liefert das Requirement Engineering Ansätze für eine frühzeitige Marktorientierung im Kostenmanagement. Das

---

78 Vgl. Wildemann, H. (2000c), S. 76f.
79 Vgl. Akao, Y. (1992); King, B. (1994); Ehrlenspiel, K. (1995a), S. 185
80 Vgl. Seidenschwarz, W. (1991), S.178ff.; Niemand, S. (1996), S. 42

Requirement Engineering ist noch eine sehr junge Disziplin und befaßt sich mit der Umsetzung der Kundenanforderungen in eine technische Spezfikation.[81] Dabei werden die durch das Management erhobenen Kundenanforderungen in eine technische Systemsicht überführt, um konkrete Anforderungsspezifikationen für einzelne Softwareprodukte bzw. Versionen abzuleiten.[82] Die Umsetzung der Anforderungen wird durch Beschreibungstechniken unterstützt.[83] Graphische Beschreibungstechniken bieten dabei den Vorteil, Anforderungsvariationen einfacher beschreiben zu können.[84] Um die Vielzahl von Anforderungen entsprechend den Kundenanforderungen in eine Umsetzungsreihenfolge zu bringen, müssen die Anforderungen priorisiert werden. Eine geeignete Vorgehensweise zur Priorisierung von Anforderngen ist die Portfoliotechnik, mit deren Hilfe eine differenzierte Analyse anhand der Kriterien Dringlichkeit der Umsetzung (time to market), Marktbedeutung, technische Wertigkeit und Aufwand/Nutzen-Relation erfolgen kann.[85] Das Requirement Engineering liefert somit Ansätze, die technische Sicht der Entwickler und die fachliche Sicht der Kunden gegenüberzustellen. Diese Gegenüberstellung kann dann für eine Softwareproduktgestaltung als Diskussionsgrundlage dienen. Ein Defizit besteht jedoch in der nur indirekten Berücksichtigung der Kosten als Diskussionsbasis zur Planung von Softwareprodukten. Des weiteren können Vorgaben zur kostenorientierten Steuerung nicht abgeleitet werden.

Der Beitrag der bestehenden Ansätze im Hinblick auf die aufgestellten Anforderungen ist in Abb. 1-2 im Überblick dargestellt. Es wird deutlich, daß verschiedene Teilkonzepte für ein Kostenmanagement bei der Softwareentwicklung bestehen.

---

81 Vgl. Broy, M. / Schneider, W. (1998), S. 29 sowie Deifel B. (1998)

82 Vgl. Partsch, H. (1998), S. 25

83 Für einen Überblick und eine Beurteilung praxisnaher Beschreibungstechniken vgl. Broy, M. / Schneider, W. (2000), S. 43 sowie die dort angegebene Literatur.

84 Vgl. Broy, M. / Schneider, W. (1999), S. 38f. sowie für die Darstellung einer graphischen Beschreibungstechnik Deifel, B. (1999).

85 Vgl. Broy, M. / Schneider, W. (2000), S. 40f.; Wildemann, H. (2000c), S. 76

Anforderungen an ein Kostenmanagement bei der Softwareentwicklung / Ansätze zum Kostenmanagement		Ansätze der SW-Aufwandschätzung	Allgemeine Ansätze der Kostenplanung	Ansätze der Kostenrechnung	Ansätze des Kostenmanagements	Ansätze des Projektmanagements	Ansätze des Qualitätsmanagements	Ansätze des Requirement Engineering
Allgemeine Anforderungen	Anwendbarkeit in der Softwareentwicklung	●	◒	◒	◒	◒	●	●
	Berücksichtigung softwarespezifischer Kosteneinflußgrößen	●	◒	◒	○	◒	○	○
	Durchgängigkeit bei der Kostenplanung und Kostenverfolgung	◒	◒	◒	◒	◒	○	○
Anforderungen der Kostenplanung	Methoden zur Kostenplanung	●	●	○	●	○	◒	○
	Marktorientierung zur SW-Produktgestaltung	○	○	○	◒	○	◒	◒
	Leitlinien zur kosten-orientierten Planung	○	○	○	◒	○	○	○
Anforderungen der Kostenverfolgung	Methoden zur Kostenverfolgung	○	○	●	●	●	○	○
	Meßgrößen zur kosten-orientierten Steuerung	◒	○	◒	◒	◒	○	○
	Kostenorientiertes Berichtswesen	○	○	○	●	●	○	○
	Leitlinien zur kosten-orientierten Steuerung	○	○	○	◒	◒	○	○

○ nicht erfüllt ◒ bedingt erfüllt ● erfüllt

Abb. 1-2 : Übersicht und Bewertung bestehender Ansätze hinsichtlich der Anforderungen an ein Kostenmanagement bei der Softwareentwicklung

Die softwarespezifischen Ansätze zur Kostenschätzung sind nur bedingt geeignet für eine Kostenplanung. Die allgemeinen Ansätze der Kostenplanung, der Kostenrechnung, des Kostenmanagements und des Projektmanagements zeigen zwar weite Verbreitung in den klassischen Anwendungsgebieten der Hardwareentwicklung, eine softwarespezifische Anwendung der zugrunde liegenden Methoden ist aufgrund der fehlenden Anpassung und Erweiterung hinsichtlich der Spezifika des Produktes Software nur lückenhaft vorhanden. Bei den Ansätzen des Qualitätsmanagements und des Requirement Engineering ist zu bemängeln, daß eine Verknüpfung mit einer kostenorientierten Softwareentwicklung nicht existiert. Somit unterstützt kein Ansatz alle Anforderungen an ein Kostenmanagement bei der Softwareentwicklung. Eine systematische Einbeziehung der für die Kosten verantwortlichen Einflußgrößen im Softwareentwicklungsprozeß in die Überlegungen zum Kostenmanagement fehlt in den traditionellen Ansätzen nahezu komplett. Ein Beeinflussung der Kosten in der Softwareentwicklung ist aber nur über eine Beeinflussung der Kostentreiber mit einem durchgängigen, softwarespezifischen Kostenmanagement möglich. Defizite, die mit dieser Arbeit geschlossen werden sollen, liegen daher in den offenen Fragestellungen zur Ausgestaltung eines Kostenmanagement, das auf eine kundenorientierte Entwicklung von Softwareprodukten abzielt.

1.3 Zielsetzung und Vorgehensweise

Zielsetzung der vorliegenden Arbeit ist die Erarbeitung eines Gesamtkonzeptes für ein Kostenmanagement bei der Softwareentwicklung, um die Kosten eines Softwareentwicklungsprozesses prognostizierbar, meßbar und regulierbar zu machen. In diesem Zusammenhang sind Notwendigkeit und Gestaltungsparameter eines Kostenmanagements zur Unterstützung einer marktorientierten Softwareproduktgestaltung zu analysieren. Dies beinhaltet alle Maßnahmen zur Gestaltung der Kosten, die der frühzeitigen und antizipativen Beeinflussung von Kostenstruktur und Kostenverhalten sowie der Senkung des Kostenniveaus dienen. Gleichzeitig sollen Bedeutung und Anwendungsmöglichkeiten eines Kosten-Controlling aufgezeigt werden.

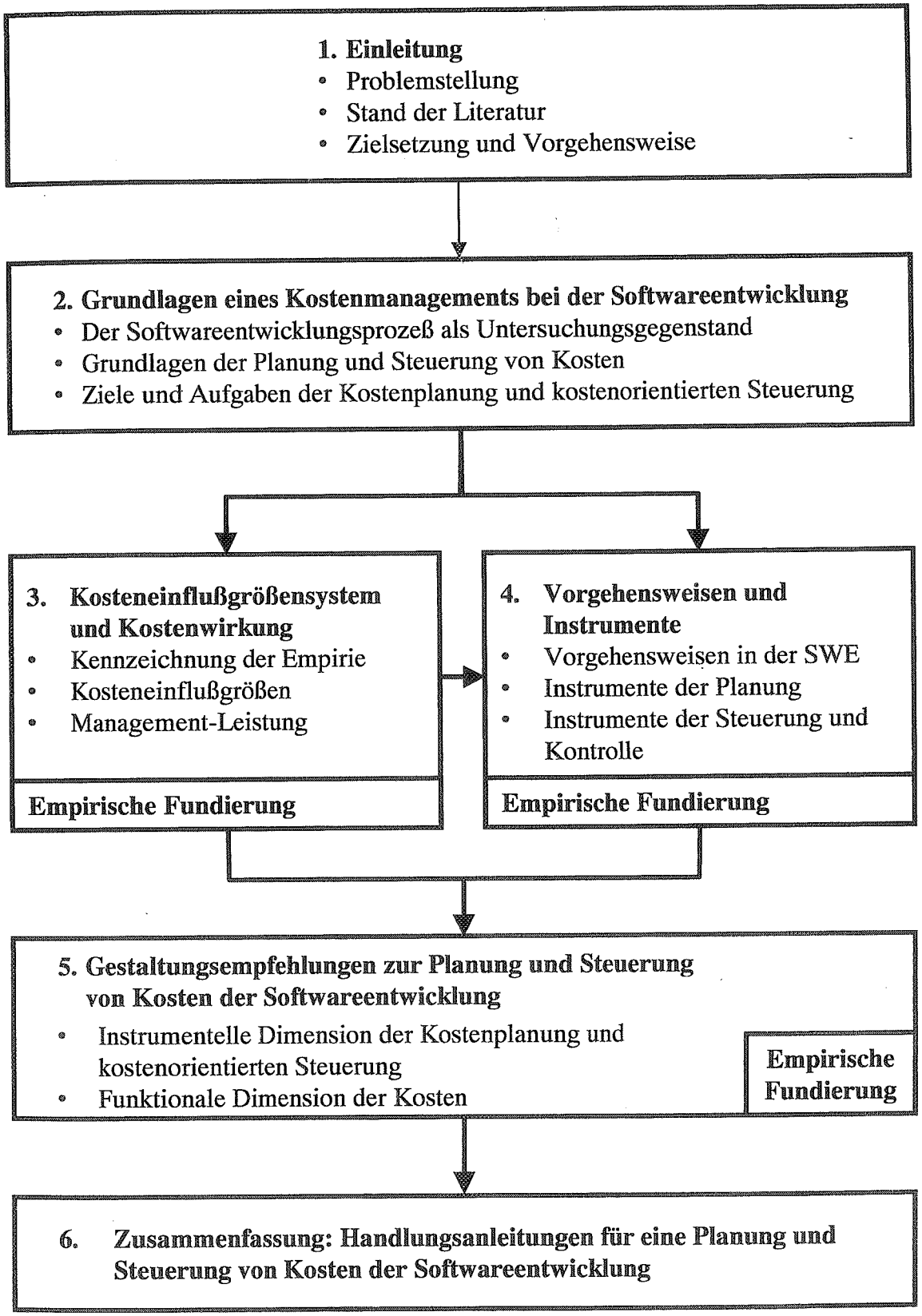

Abb. 1-3: Gang der Untersuchung

Zur Erreichung der dargestellten Ziele wurde die in Abb. 1-3 skizzierte Vorgehensweise gewählt. Nach Erörterung der Problemstellung sowie der bestehenden Literaturansätze, deren Lösungsbeiträge und der verbleibenden Defizite erfolgt im zweiten Kapitel die Schilderung der konzeptionellen Grundlagen für ein Kostenmanagement bei der Softwareentwicklung. Die Darstellung eines Standard-Softwareentwicklungsprozesses zeigt die unterschiedlichen Aktivitäten zur Softwareentwicklung im Zeitverlauf. Des weiteren werden die Grundlagen eines Kostenmanagements beschrieben, indem eine begriffliche Definition eines Kostenmanagements erfolgt sowie die Gestaltungsobjekte Kostenstruktur, Kostenniveau und Kostenverlauf dargestellt werden. Auf Basis dieser Grundlagen und der Aktivitäten im Softwareentwicklungsprozeß werden die Aufgaben und Ziele für ein Kostenmanagement bei der Softwareentwicklung detailliert und Leitlinien erarbeitet, die den Orientierungsrahmen für die Ableitung von Gestaltungsempfehlungen vorgeben. Die Erfüllung der Leitlinien stellt dann ein Kostenmanagement in der Softwareentwicklung sicher.

Gegenstand des dritten Kapitels ist die Beschreibung des Kosteneinflußgrößensystems und die Deskription der Kostenwirkung. Da die Relevanz der Kosteneinflußgrößen anhand der empirischen Untersuchung überprüft wird, erfolgt zu Beginn des Kapitels eine Kennzeichnung der empirischen Untersuchung. Für die Beschreibung des Kosteneinflußgrößensystems werden die produkt- und prozeßspezifischen Kosteneinflußgrößen systematisiert, die Management-Leistung als übergeordnete Kosteneinflußgröße beschrieben und deren Wirkungsweise empirisch überprüft. Die Kosteneinflußgrößen bilden die Grundlage für die Identifikation der unterschiedlichen Arten der Softwareentwicklung und verdeutlichen die Anforderungen an ein Kostenmanagement zur typspezifischen Ausgestaltung. Kapitel vier beschreibt zu Beginn die unterschiedlichen Vorgehensmodelle für die Softwareentwicklung. Die Diskussion der verschiedenen Einsatzgebiete der Vorgehensmodelle hat zum Ziel, die Softwareentwicklungsprozesse zu systematisieren, die im Hinblick auf ein Kostenmanagement bei der Softwareentwicklung zu unterscheiden sind. Die identifizierten Prozeßtypen bilden die Grundlage zur Ableitung von Gestaltungsempfehlungen für den spezifischen Instrumenteneinsatz. Des

weiteren werden die Instrumente für ein Kostenmanagement der Softwareentwicklung sowie die Ergebnisse der empirischen Untersuchung in bezug auf den heutigen Einsatz der Instrumente und deren Erfolg bei der Anwendung näher beschrieben. Im fünften Kapitel werden Gestaltungsaspekte für ein Kostenmanagement in der Softwareentwicklung erarbeitet und die Wirkungsweise der Instrumente anhand von Fallstudien analysiert.