Zanger, Cornelia

Software als Gegenstand betriebswirtschaftlicher Betrachtungen

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 271

Nr. 271

Cornelia Zanger

Software als Gegenstand betriebswirtschaftlicher Betrachtungen

Kiel, Mai 1991

Dozentin Dr. C. Zanger, TH Dresden

Die Arbeit entstand als Nachtrag zu einem Aufenthalt als Gastdozentin am Institut für betriebswirtschaftliche Innovationsforschung der Universität Kiel. Die Autorin möchte Herrn Prof. Dr. Klaus Brockhoff und Herrn Prof. Dr. Jürgen Hauschildt ganz herzlich für die wissenschaftliche Förderung dieses Aufsatzes danken.

Zusammenfassung

Im Kontext moderner Informationstechnik wird Software zur Determinante des strategischen Unternehmenserfolgs.
Software

- hat Produktcharakter, wird in einem Leistungsprozeß erstellt, muß vermarktet und geschützt werden,
- ist ein Investitionsobjekt, Zahlungsströme müssen bestimmt und prognostiziert werden,
- ist ein innovatives Produkt und
- ist ein Objekt betriebswirtschaftlicher Bewertung.

Der Beitrag liefert einige Ansatzpunkte für die betriebswirtschaftliche Bestimmung von Software in diesem Prozeß.

Gliederung

1. Anliegen

Als Element moderner Informationstechnik hat Software bis zum Beginn der 90er Jahre ein stetig wachsendes Gewicht erlangt.[1] Dies ist zum einen durch die Erfüllung wachsender Ansprüche an die Softwarequalität bedingt, die sich in steigendem Funktions- und Leistungsumfang, größerer Anwenderfreundlichkeit, höherer Zuverlässigkeit und verbesserter Wartbarkeit äußert.[2] Zum anderen ist dafür das wachsende wirtschaftliche Gewicht des Softwareaufwandes maßgebend. Die seinerzeit von Boehm[3] vorhergesagte Verlagerung des Aufwandsschwerpunktes von der Hardware zur Software kann als bestätigt angesehen werden. Der Anteil des Softwareaufwandes am Gesamtaufwand für Informationstechnik im Unternehmen hat sich von ca. 15 % am Ende der 50er Jahre auf ca. 50 % im Jahre 1990 erhöht.[4] Die Gründe liegen einerseits im rapiden Rückgang der Hardwarepreise und der damit verbundenen Verbesserung des Preis-/Leistungs-Verhältnisses der Hardware.[5] Die Fortsetzung dieses Trends wird prognostiziert.[6] Als weitere Ursachen sind die zunehmende Integration von Software in alle Bereiche der Tätigkeit des Unternehmens,[7] die bereits eingangs genannte Erfüllung der Forderung nach hoher Softwarequalität und die mit diesen Prozessen verbundene steigende Komplexität der Software anzusehen.

Verbunden mit dieser Tendenz zur Ausweitung des Softwareeinsatzes im Rahmen von Informations- und Kommunikationssystemen ist die Tatsache, daß moderne Software immer mehr zum strategischen Erfolgsfaktor für Unternehmen wird. In dem für die 90er Jahre erwarteten "Wettbewerb der Informationssysteme" wird die Software die Wettbewerbsfähigkeit von Unternehmen und ihre Erfolgschancen wesentlich beeinflußen.[8]

Software wird zu einer Determinante des strategischen Unternehmenserfolges. Teilt die Betriebswirtschaftslehre diese Auffassung? Es scheint uns, daß die Software eher noch ein Dasein im Spezialistendenken der (Wirtschafts-)Informatiker führt. Hier ist eine Neubesinnung angebracht:

---

1 Krallmann (1990)
2 Griese et al. (1987), S. 523
3 Boehm (1973), S. 5
4 Die Rolle der Software Anfang der 90er Jahre (1987), S. 24.1.02
5 ebenda, S. 24.1.03
6 Folberth (1990), S. 71 ff.
7 Scheer (1988), S. 3. Der Softwareeinsatz ist selbstverständlich auch in den außerbetrieblichen Bereichen tendenziell steigend. Diese Prozesse sollen jedoch nicht Gegenstand der Betrachtung sein.
8 Österle (1990), S. 12

- Software ist ein *Produkt eines Leistungsprozesses*. Sie muß erstellt, vermarktet, geschützt werden.

- Software ist ein *Investitionsobjekt*. Ihre Zahlungsströme müssen bestimmt und prognostiziert werden.

- Software ist ein *innovatives Produkt*. Sie muß entwickelt und diffundiert werden.

- Software ist ein *Objekt betriebswirtschaftlicher Bewertung*. Ihre Beiträge zur Zielerfüllung müssen bestimmt und quantifiziert werden.

Der folgende Beitrag soll einige Ansatzpunkte für einen derartigen Prozeß der Neubestimmung liefern.

2. Softwaremarkt und Softwaretypen

2.1. Marktcharakteristika

Die Software nimmt gegenwärtig am Informatik-Markt nach der Informationsverarbeitungstechnik und der Kommunikationstechnik bezüglich des Umsatzes die dritte Stelle ein (vgl. Abb. 1).

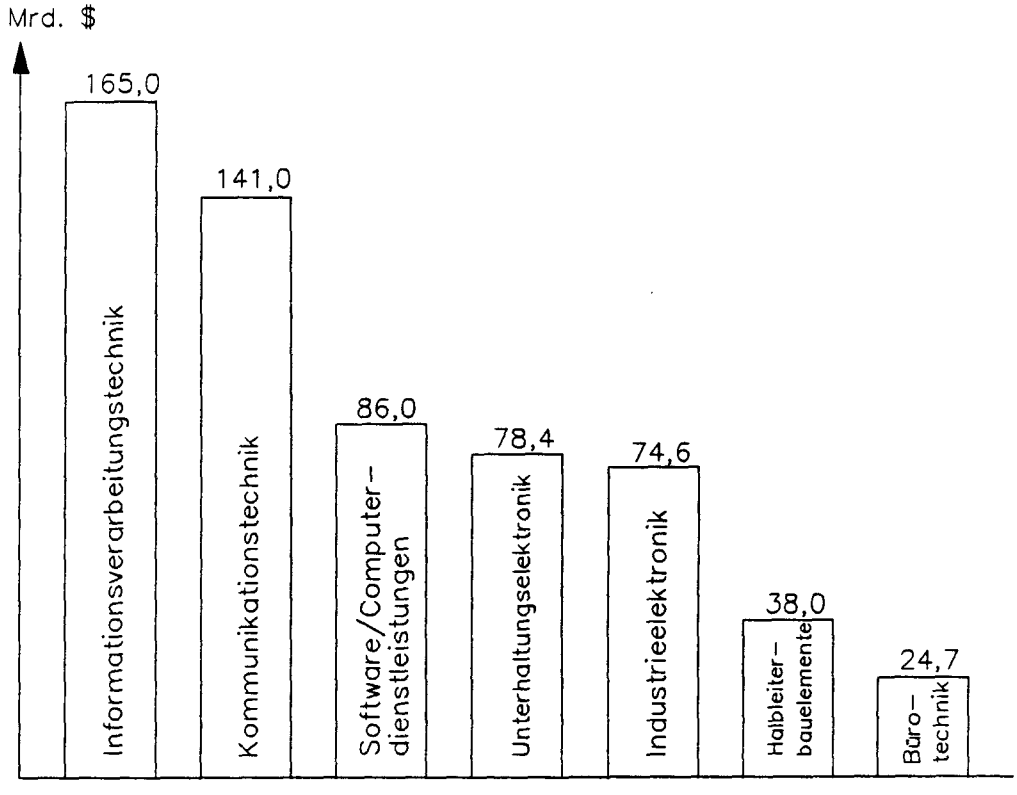


Abb. 1 Stellung der Software am Informatik-Markt 1987
(Quelle: Berechnet nach Zukunftskonzept Informationsverarbeitung, BMFT 1988, S. 18)

Die Umsatzzahlen verdeutlichen, daß sich ein eigenständiger Software-Markt herausgebildet hat, der jährliche Steigerungsraten von ca. 20 %[1] aufweist. Für den Software-Markt wird bis zum Jahre 2000 ein stetig wachsendes Volumen prognostiziert.[2]

Im Unterschied zu anderen Teilmärkten des Informatik-Marktes (z.B. Halbleiterbauelemente, Informationsverarbeitungstechnik, Unterhaltungselektronik), die von japanischen Konzernen

---

1 Tüschen (1989), S. 72
2 Zukunftskonzept Informationsverarbeitung, BMFT (1989), S. 19

beherrscht werden, wird der Software-Markt gegenwärtig zu fast 60 % von USA-Firmen kontrolliert.[1]

Das spiegelt sich ebenfalls in der Liste der erfolgreichsten Anbieter-Firmen für PC-Software 1989 (vgl. Abb. 2) wider. Es handelt sich um Firmen, die vom amerikanischen Markt ausgehend, weltweit operieren.[2]

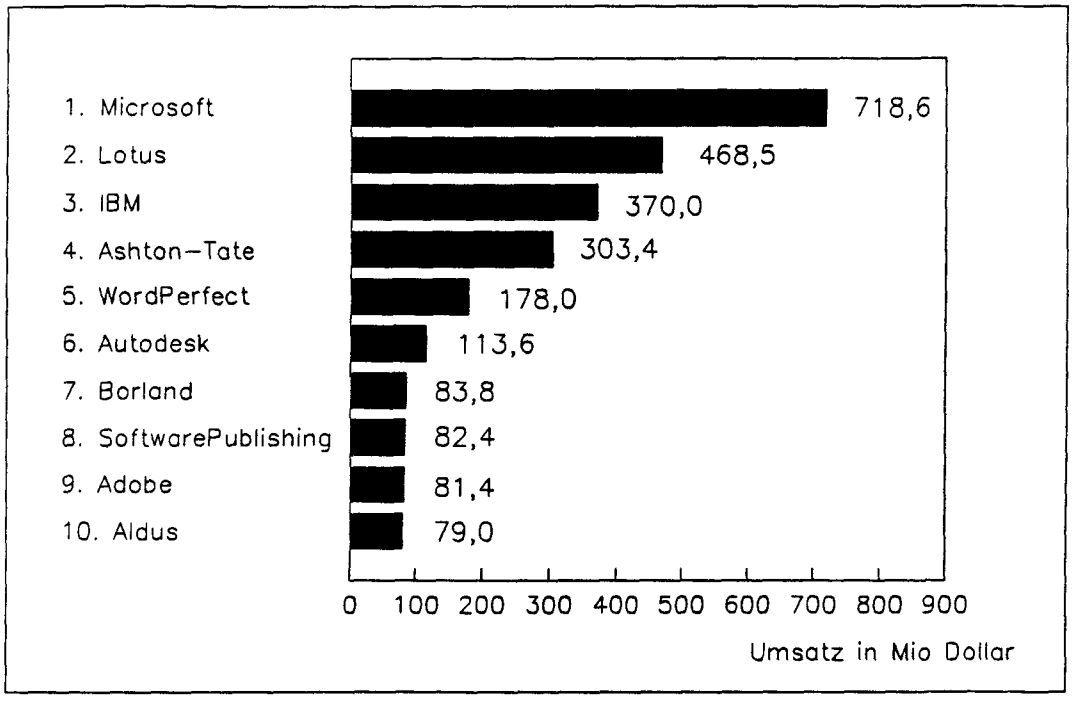


Abb. 2: Die erfolgreichsten Anbieter von PC-Software 1989
(Quelle: Computer abc 1990, S. 726)

Es fällt auf, daß unter den führenden Unternehmen sowohl Hardwarehersteller als auch reine Softwarehäuser sind. Die Marktaufteilung wird mit ca. 50 % Marktanteil der Software- und Systemhäuser, ca. 30 % softwareverkaufende Hardwarehersteller und ca. 20 % softwareverkaufende Anwender (als Nebenprodukt) angegeben.[3]

---

1 Brandt, R.; Schwartz, E.J.; Goss, N. (1991), S. 62

2 Computer abc (1991), S. 724 f. Eine Ausnahme bildet lediglich die Firma Software Publisching, die bisher ausschließlich auf dem amerikanischen Markt präsent ist.

3 Hansen (1986), S. 323

Die angestellte globale Betrachtung zur Entwicklung des Software-Marktes unterstreicht die strategische Bedeutung der Software, genügt aber in ihrer Differenziertheit betriebswirtschaftlichen Zwecken bei weitem nicht.

2.2. Klassifizierung der Software

Eine inhaltliche Abgrenzung des Softwarebegriffs wurde von der Informatik vorgenommen. In Bezug auf die Aufgabenstellung ist Software der "Sammelbegriff für die Systemprogramme und die Anwendungsprogramme von Rechnern."[1] Die zur Komplettierung der Programme erforderlichen Dokumentationen sind ebenfalls in den Softwarebegriff einzuschließen.

Hilfreich für eine differenziertere Betrachtung ist eine weitere inhaltliche Untergliederung (vgl. Abb. 3).

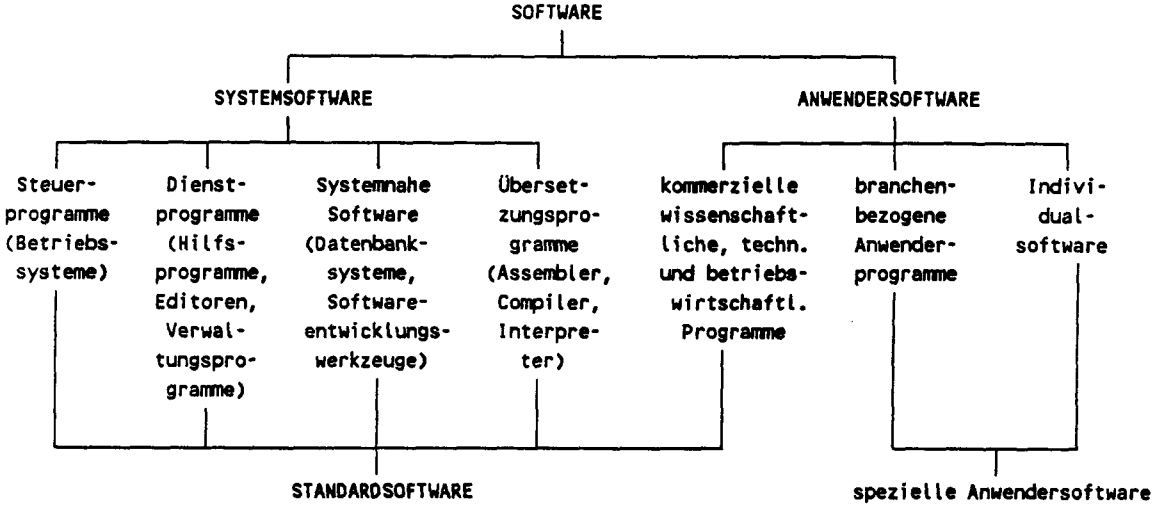


Abb. 3: Inhaltliche Gliederung der Software (unter Verwendung von Hansen (1986), S. 324 sowie Stahlknecht (1989), S. 93)

Allgemein anerkannt ist die Klassifizierung in die beiden großen Gruppen Systemsoftware und Anwendersoftware.[2] Alternativ dürfte die Unterscheidung nach dem Standardisierungsgrad in die Kategorien Standardsoftware[3] und spezielle Anwendersoftware nützlich sein. Standardsoftware ist "konfektionierte" Software, d.h. Software, die nicht für den Eigenbedarf entwickelt wird, sondern direkt für den Verkauf am anonymen Markt.

---

1 Hansen (1986), S. 323

2 Von dieser Differenzierung gehen die meisten Autoren aus, beispielhaft seien genannt: Herrmann (1983), S. 19, Hansen (1986), S. 324, Becker et al. (1986), S. 44, Stahlknecht (1989), S. 93

3 Cave/Maymon (1984), S. 16 ff.

Zum Verhältnis zwischen den Arten von Software gibt es kaum verläßliche Angaben. Es kann jedoch davon ausgegangen werden, daß mit der wachsenden Verbreitung von Mikrocomputern die Entwicklung und der Vertrieb von Standardsoftware bedeutende Impulse erfahren haben. Angaben, die von 1/3 Standardsoftware und 2/3 spezieller Anwendersoftware ausgehen, sind durchaus als realistisch anzusehen.[1]

Eine systematische betriebswirtschaftliche Untersuchung der Software führt über die bisher erwähnten funktionalen Aspekte hinaus zwangsläufig zur Frage nach dem Marktcharakter der Software. Im allgemeinen wird davon ausgegangen, daß es sich bei der Software um ein immaterielles Produkt oder ein Dienstleistungsprodukt[2] handelt. Erfaßt das die Software in ihrer Gesamtheit? Gehen wir vom Produktbegriff aus, der das Produkt als Kaufobjekt[3] sieht und als gebündelte Menge von Eigenschaften beschreibt, die zum Gegenstand des Tauschs werden soll.[4] Mit dieser Definition erfassen wir den Teil der Software, der als Standardsoftware für einen Kunden entwickelt und auf dem Markt angeboten wird ebenso wie die im Auftrag für Dritte erstellte Softwareleistung, nicht aber die für den eigenen Bedarf im Unternehmen entwickelte Individualsoftware. Software kann also sowohl als Dienstleistungsprodukt (immaterielles Produkt) als auch als innerbetriebliche Dienstleistung in Erscheinung treten (vgl. Abb. 4).

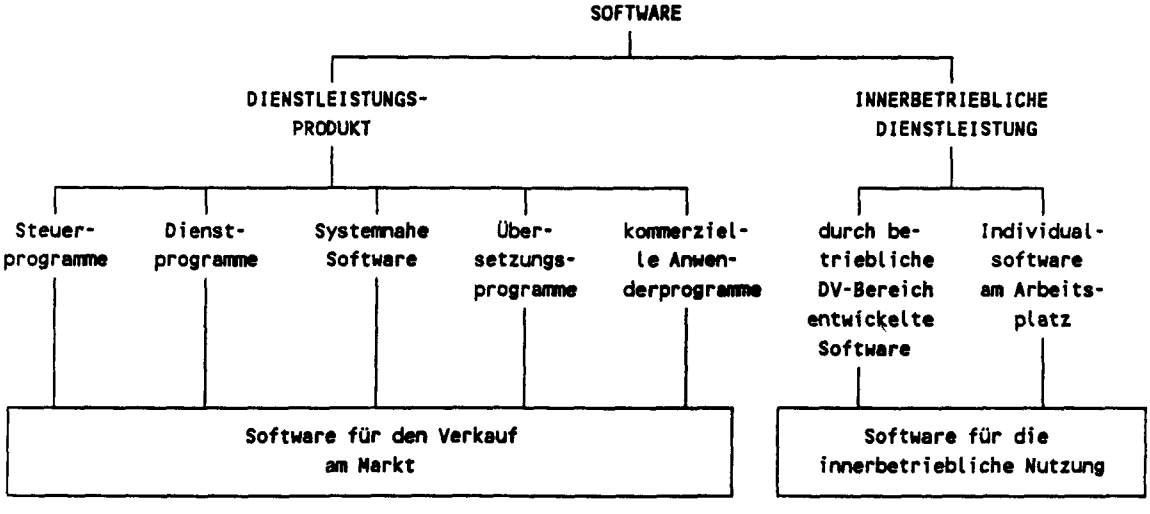


Abb. 4: Die Abgrenzung des Produktbegriffs für Software

---

1 Die Rolle der Software Anfang der 90er Jahre (1987), S. 24.1.03

2 Neugebauer (1986), S. 36. Die Anwendung des Dienstleistungsbegriffs geht auf Gutenberg (1979), S. 2 zurück, der die Bereitstellung von Dienstleistungen als Form der betrieblichen Leistungserstellung explizit hervorhob.

3 Kotler (1989), S. 424

4 Brockhoff (1988), S. 10

Ähnlich wie der Erfolg der innovativen Produktpolitik in hohem Maße von der Zusammenarbeit zwischen Anbietern und "Lead users" abhängen,[1] ist gezeigt worden, daß Software für die innerbetriebliche Nutzung effizient durch Zusammenarbeit von betrieblichen DV-Abteilungen und Fachabteilungen entwickelt werden kann und dies auch von den Unternehmen zunehmen präferiert wird.[2]

Weitere Bestrebungen zur Klassifizierung von Software beziehen sich im wesentlichen auf den Komplexitätsgrad[3] und die Entwickler (Hersteller von EDV-Anlagen, Softwarehäuser, Anwender).[4]

In Abb. 5 wird die Klassifizierung unter Einbeziehung weiterer Faktoren in Form eines morphologischen Kastens differenziert. Aus Sicht einer ganzheitlichen Betrachtung erscheinen z.B. auch solche Faktoren, wie

- der Neuheitsgrad und die Entstehung der Neuerung,
- die Organisation der Entwicklung,
- die Arbeitsteilung im Entwicklungsprozeß,
- die Bindung an die Hardware und
- mögliche Formen der Nutzung wichtig.

Die aufgezeigte Klassifizierung eröffnet vielfältige Möglichkeiten für die Ableitung von betriebswirtschaftlichen Fragestellungen. Sie ist Ausgangspunkt für die folgende Einordnung bisheriger Untersuchungen und die Betrachtungen zum innovativen Charakter der Software.

---

1 v. Hippel, E. (1988)

2) Vgl. dazu die empirischen Untersuchungen von Knolmayer/Disterer (1989), Knolmayer/Disterer (1988), S. 124 ff. und Disterer (1988).

3 Dreckmann (1987), S. 24.02.03.

4 Neugebauer (1987), S. 263.

FAKTOREN	AUSPRÄGUNG DER FAKTOREN			
Bedeutung im Produktprogramm	Software als Hauptprodukt		Software als Nebenprodukt	
Marktbezug	Software für Verkauf		Software für Eigenbedarf	
Standardisierungs-grad	Standardsoftware		nicht standardisierte Software	
Breite der Anwendung	Systemsoftware	weitere Standard-software	branchenbezo-gene Anwender-software	Individual-software
Komplexität	Programmsystem	Programmpaket	Einzelprogramm	Modul
Neuheitsgrad	neues Softwareprodukt		Version	unverändertes Produkt
Entstehung der Neuerung	Innovation	Imitation	Modifikation	Wartung/Fehler-beseitigung
Entwickler	Hersteller von Hardware	Softwarehaus Programmierer-büro Beratungsfirma	Anwender (spezialisiert auf Software-entwicklung)	Anwender für individuellen Bedarf
Organisation beim Softwareentwickler	selbständiger Entwicklungs-betrieb	DV-Fach-abteilung	Entwickler neben eigentl. Arbeit	
Arbeitsteilung	Entwicklungsteam		"Ein-Mann"-Entwicklung	
Bindung an Hardware	Verkauf mit Hardware	Separat-verkauf	Systemverkauf inkl. Hardware, Orgware, Projektierungsleistung	
Ziel der Nutzung	Unterstützung materieller (Produktions-) Prozesse	Unterstützung immaterieller (Management-) Prozesse	Unterstützung von Lern-prozessen (Teachware)	Unterstützung konsumtiver Prozesse (z.B. Computerspiele)

Abb. 5: Morphologischer Kasten zur Klassifizierung von Software aus betriebswirtschaftlicher Sicht

3. Charakteristische betriebswirtschaftliche Fragestellungen zur Software

Die Existenz derartig vieler Softwaretypen signalisieren unterschiedliche betriebswirtschaftliche Probleme, die nach drei Situationen differenziert werden können:

(1) *Nutzungsbedingungen* von nicht selbst entwickelter Software (Standardsoftware, durch Dritte entwickelte Auftragsprojekte) und Vergleich zur Entwicklung

(2) *Entwicklung* von Software für den eigenen Bedarf im Unternehmen oder für den Verkauf am Markt als ein Nebenprodukt

(3) *Industriemässige Produktion* von Software als Hauptprodukt des Unternehmens (Softwarehäuser) oder als wesentliches Geschäftsfeld (Betriebssysteme bei Hardwareherstellern) mit anschließendem Vertrieb (vgl. Abb. 6)

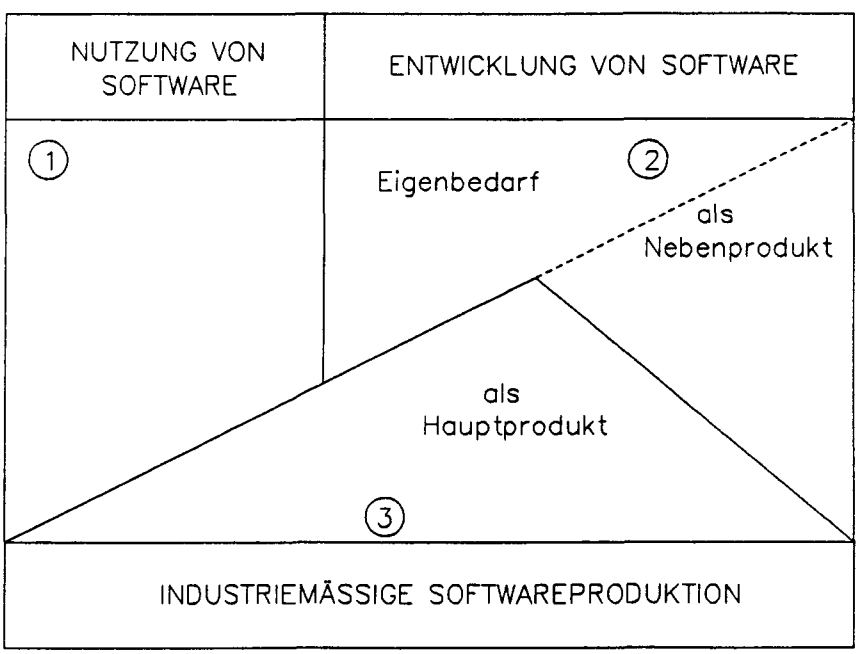


Abb. 6: Mögliche Felder betriebswirtschaftlicher Untersuchungen zur Software

Im folgenden soll ein kurzer Überblick über bisherige betriebswirtschaftliche Untersuchungen zu diesen drei Untersuchungsfeldern gegeben und auf einige Defizite hingewiesen werden.

Untersuchungsfeld 1

Schwerpunkte der betriebswirtschaftlichen Betrachtung sind der *Anwendernutzen* sowie die *Entscheidung über die Nutzung von Standardsoftware oder die Entwicklung von Individualsoftware* (entsprechend Abb. 1).

Eine umfangsreiche Dokumentenanalyse zum Nutzen von Informationssystemen (ca. 1800 deutsche und angelsächsische Quellen) legten Mertens/Anselstetter/Eckardt/Nickel[1] vor. Als *positive Wirkungen* für den Anwender ermittelten sie eine Reihe von branchen-unabhängigen Faktoren, wie

- relative Kostenreduktion zwischen 25 % und 60 %[2], wobei jedoch offen bleibt, von welcher Basis diese Reduktion ausgeht
- Zeitersparnis und höhere Produktivität
- Personaleinsparung
- effizienter Einsatz materieller Ressourcen
- Verringerung der Fehlerhäufigkeit in der Verarbeitung von Daten
- höhere Leistungsqualität
- bessere Planung und Disposition
- besseres Informationshandling.[3]

Was dabei jeweils "höher", "besser" oder "effizienter" genau bedeutet, kann nur fallweise quantifiziert werden, so daß auf dieser Basis auch kein Nutzen-Kosten-Vergleich angestellt werden kann.

Branchenspezifische positive Wirkungen konnten bezogen auf den Gesamtbetrieb und seine Funktionsbereiche (z.B. schnellere und bessere Kundenbedienung durch bessere Ablaufsteuerung in Angebotswesen, Beschaffung, Fertigung, Logistik) [4] festgestellt werden.

Im Zusammenhang mit der Nutzenermittlung stellt sich aus Sicht des Anwenders mit jedem neuen Projekt die Frage: Was ist wirtschaftlicher - Eigenentwicklung oder Erwerb von Standardsoftware? Kataloge von Kriterien sollen die Entscheidung unterstützen hinsichtlich der

- Wirtschaftlichkeit der Alternativen
- Qualität der Software
- optimale Integration in vorhandene oder geplante betriebliche Informationssysteme
- Einordnung in die Informatikstrategie des Unternehmens (Präferenzen für bestimmte

---

1 Mertens/Anselstetter/Eckardt/Nickel (1982), S. 135 ff.
2 ebenda, S. 139
3 ebenda, S. 139 ff.
4 ebenda, S. 145 ff.

Hardware- und/oder Softwarehersteller).[1]

Als Defizit ist anzumerken, daß die Autoren den Nachweis der Anwendbarkeit dieser Kriterienkataloge im Entscheidungsprozeß ebenso wie den Nachweis der statistischen Repräsentanz der Kriterien in empirischen Bewertungsrechnungen schuldig bleiben.

Untersuchungsfeld 2

Bei der *Entwicklung von Software*, sei es nun für den Eigenbedarf im Unternehmen, sei es für den Verkauf, stehen aus betriebswirtschaftlicher Perspektive zwei Fragen im Mittelpunkt. Dies ist zum einen die Ermittlung und Beeinflussung des *Aufwandes für die Softwareentwicklung* und zum anderen die *Bestimmung der Produktivität* der mit der Arbeit beauftragten Softwareentwickler.

Die Bestimmung des *Aufwandes für die Softwareentwicklung* ist das am intensivsten bearbeitete Untersuchungsfeld. In der Literatur werden 20 Schätzverfahren[2] zur Vorkalkulation des Softwareaufwandes beschrieben, die einer laufenden Weiterentwicklung mit dem Ziel der Vervollkommnung unterliegen.[3]

Ausgeprägtes Interesse an der Aufwandschätzung zeigen sowohl Informatiker als auch Betriebswirte.[4] Die Ursache dafür muß in dem bisher wissenschaftlich wenig beherrschten Bereich des Softwareengineering gesucht werden. Als Ziel der Schätzung des Entwicklungsaufwandes kann die Erhöhung der Transparenz der Entwicklungsarbeiten angesehen werden und der Wunsch, den Erfolg der Softwareentwicklung beherrschbar zu machen.

Als Hauptprobleme der Aufwandsschätzung werden häufig
- eine ungenügende Zieldefinition
- schwer quantifizierbare Einflüsse
- hoher Innovationsgrad
- Änderung der Randbedingungen genannt.[5]

Hinsichtlich der Zuverlässigkeit der geschätzten Aufwandswerte kommen die Autoren zu unterschiedlichen Auffassungen. So ermittelte Lehmann[6], daß trotz der Anwendung von

---

1 Becker et al. (1986), S. 471 ff., Computer-Praxis abc (1990), S. 6.1-10 ff., Österle (1990), S. 34 ff.
2 Einen umfassenden Überblick vermitteln Noth/Kretschmar (1986)
3 Saalfrank et al. (1987), S. 95 ff.
4 Die ersten Impulse gehen auf die amerikanische Computerindustrie mit dem IBM-Handbuch von 1968 zurück.
5 Platz/Schmelzer (1986), S. 162
6 Lehmann (1979), S. 122

Schätzverfahren 59 % aller Softwareentwicklungen den geplanten Aufwand überschreiten. Noth und Kretzschmar kommen zu dem Schluß, daß nicht die Entwicklung neuer Schätzverfahren, sondern die empirische Fundierung der Verfahren im Unternehmen notwendig ist. Dies erfordert das Sammeln, Systematisieren und Auswerten von umfangreichem Erfahrungswissen.[1]

Bei der starken methodischen Ausrichtung der Arbeiten zur Aufwandsermittlung bleibt ein inhaltliches Problem weitestgehend unbeachtet, nämlich der Einfluß des Innovationsgrades der Softwareentwicklung auf den Aufwand und ggf. auch die Auswahl geeigneter Schätzverfahren.

Neben der Schätzung des Softwareaufwandes zu Beginn der Entwicklungsarbeit beschäftigen sich betriebswirtschaftliche Untersuchungen mit der *Zurechnung des Aufwandes* zum fertigen Softwareprodukt.[2] Dabei stellt sich eine verursachungsgerechte Aufwandszuordnung bei Software vergleichsweise schwierig dar.[3] Besondere Beachtung im Kontext der Aufwandsbetrachtung findet der nach dem Abschluß der Entwicklung im Verlauf der Softwarenutzung entstehenden *Wartungsaufwand*. Im Vergleich zu anderen Produkten erreicht der Wartungsaufwand im Vergleich zum Entwicklungsaufwand ein beachtliches Ausmaß. Für Großrechner wurden in den 70er Jahren Relationen von Entwicklungsaufwand zu Wartungsaufwand von 1 : 1 nachgewiesen.[4] Modularer Softwareaufbau und moderne Softwareentwicklungs-Tools werden sowohl den Entwicklungs- als auch den Wartungsaufwand *in absoluter Höhe* reduzieren können, der *relativ hohe Anteil* des Wartungsaufwandes wird jedoch bestehen bleiben.

Eine kontrovers diskutierte Frage ist die *Messung der Produktivität* der Softwareentwicklung. Die meistens verwendete Kenngröße ist die Einheit LOC/MT (lines of code pro Manntage).[5] Diese, von der amerikanischen Softwareindustrie[6] entwickelte und empfohlene Maßzahl, erwies sich bei Tests als wenig aussagefähig.[7] Dies muß nicht verwundern, da spätestens seit dem Weinberg-Experiment[8] bekannt ist, daß für die Lösung einer Problemstellung so viele unterschiedliche und einmalige Softwarelösungen möglich sind, wie Programmierer beauftragt werden. Daraus folgt: qualitativ gleichwertige Programme können eine ganz unterschiedliche Anzahl von Quellprogrammzeilen aufweisen. Das bedeutet, daß ein guter Pro-

---

1 Noth/Kretschmar (1986), S. 121 ff.
2 Einen umfassenden Überblick gibt Herrmann (1983)
3 ebenda, S. 63 ff.
4 Lientz/Swanson/Topkins (1978), S. 466 ff.
5 Z.B. in Boehm (1984), Platz/Schmelzer (1986), Sneed (1987), Griese et al. (1987). Die Größe "line of code" ist definiert als Programmzeilen des Quellprogramms ohne Leerzeilen und Kommentarzeilen sowie ohne in einem Programmsystem mehrfach vorhandene Zeilengruppen, Griese et al. (1987), S. 517
6 Boehm (1984). Boehm ist Direktor des Software-Hauses Thompson-Ramo-Wooldridge.
7 Noth/Kretzschmar (1984), S. 87 f.
8 Weinberg (1971)

grammierer nicht zwangsläufig sehr viele Programmzeilen pro Zeiteinheit schreiben muß, sondern gerade mit geringem Programmumfang effiziente Lösungen finden kann.

Soll nicht nur die Produktivität von Programmierern, sondern auch von Unternehmen auf diese Weise verglichen werden, so geben Griese et al. mit Recht zu bedenken, das man nur das vergleichen kann, was auch vergleichbar ist.[1] Aus betriebswirtschaftlicher Sicht ist bezüglich der Produktivitätsmessung festzustellen, daß das aus der Informatik kommende Meßmodell der Forderung nach einem eindeutigen Maßstab für die Produktivität von verschiedenen Programmierern oder softwareentwickelnden Unternehmen nicht genügt.

Untersuchungsfeld 3

Es muß verwundern, daß das interessante, weil relativ neu entstandene Feld der industriemäßigen Produktion von Software bisher aus betriebswirtschaftlicher Sich wenig bearbeitet wurde. Drei größere empirische Untersuchungen stellen vor allem ab auf

- die Aufgabenabgrenzung von Softwarehäusern[2]
- die Typisierung von Softwarehäusern[3]
- die Planung von selbständigen Softwareunternehmen[4].

Griese führt eine Längsschnittbetrachtung zu Anzahl, Größe, Leistungsangebot und fachlichen Aufgaben von Softwarehäusern durch. Er kommt bereits 1982 zu dem Schluß, daß sich der Schwerpunkt der Tätigkeit der Softwarehäuser immer mehr von der Unterstützung der Kunden bei der Entwicklung von Software hin zum Angebot von Standardsoftware entwickeln wird. Er prognostiziert ein Anwachsen der Gründung von professionellen Softwarehäusern.[5]

Eine umfangreiche empirische Analyse basierend auf der standardisierten Befragung von 198 Softwarehäusern wurde 1983 hinsichtlich Marketing, Softwareproduktion und Finanzierung durchgeführt. Im Ergebnis einer multivariaten Analyse konnten acht Typen von Softwareunternehmen gefunden werden (vgl. Übersicht 1).

---

1 Griese et al. (1987), S. 516
2 Griese (1982), S. 146 ff.
3 Neugebauer (1986)
4 Tüschen (1989)
5 Griese (1982), S. 149 ff.

	Bezeichnung	Charakteristik
Typ I	Anwendungsberater	Schwerpunkt der Tätigkeit ist Schulung und Beratung Softwareentwicklung als ergänzende Leistung
Typ II	Programmier-Helfer	Unterstützung für Softwareentwicklung im Programmiererteam des Kunden
Typ III	System-Anbieter	Hardware-/Software-Anwendungssysteme
Typ IV	Anbieter von Standardsoftware	Standardsoftware ergänzt durch Schulung und Beratung
Typ V	Software-Projekt-nehmer	Übernahme von Großprojekten im Anwenderauftrag
Typ VI	Software-Händler	Vertrieb von Lizenzprodukten mit Beratung und Schulung
Typ VII	Anbieter von höchst qualifizierter Standardsoftware	Überdurchschnittlich qualifizierte Entwickler mit umfassendem Beratungs- und Schulungsangebot
Typ VIII	Branchen-Spezialisten	Auftragsentwicklung von Branchen-Software

Übersicht 1: Empirisch extrahierte Typen von Software-Unternehmen (nach Neugebauer (1986), S. 241 ff.

Die Erfolgsprognose für die einzelnen Typen fällt unterschiedlich aus:[1]

- Unternehmenstyp II hat geringe Erfolgsaussichten auf Grund eines nicht nachfragegerechten Leistungsprogramms.
- Unternehmenstyp V hat Probleme durch eine zu starke Spezialisierung.
- Typ I und III zeigen Inkonsistenzen zwischen Leistungsprogramm und Anforderungen der Kunden.
- Für Typ VI wird infolge ungezielten Marketings trotz kurzfristiger Erfolge langfristig keine günstige Entwicklung prognostiziert.
- Die Typen IV, VII und VIII erweisen sich als besonders aussichtsreich auf eine erfolgreiche Entwicklung durch die Umsetzung ihres Unternehmenskonzepts mit einem überdurchschnittlichen technologischen Potential.

Offen bleibt im Ergebnis der Untersuchung die Beantwortung der Frage, welche strategischen Verhaltensempfehlungen für die einzelnen Typen von Software-Unternehmen abzuleiten sind.

Ausgehend vom Geschäftsfeld Software versucht Tüschen in Auswertung von empirischen

1 Neugebauer (1986), S. 249

Explorationen in 16 Softwarehäusern den Gegenstand, die Organisationsstruktur, die Instrumentarien und das Führungssystem der strategischen und operativen Planung zu bestimmen. Er stellt einen Zusammenhang zwischen der Größe des Softwarehauses und der Planungspraxis her. Für große Softwarehäuser sind Konzepte und Ansätze der Unternehmensplanung in aller Regel ebenso wie in anderen Unternehmen anwendbar. In kleinen Software-Unternehmen bestehen hingegen Probleme bei der Identifizierung und Handhabung von Planungsfragen. In diesen Fällen werden leicht handhabbare Planungsinstrumente empfohlen.[1]

Das Innovationsverhalten von Softwarehäusern sieht Tüschen in Abhängigkeit von der Ressourcenstärke des Unternehmens. Große Softwarehäuser können Innovationsstrategien erfolgreich durchsetzen, während für kleinere und mittlere Unternehmen die typische Strategie darin besteht, kleine Entwicklungsschritte zu gehen und sich zu bemühen, Entwicklungsaufwand und -risiko auf Auftraggeber zu übertragen.[2] Diese Hypothese müßte empirisch getestet werden.

Wertet man zusammenfassend die Untersuchungen zur Software aus betriebswirtschaftlicher Sicht, so fällt auf, daß einzelne Bereiche (wie z.B. Aufwandsschätzung und Produktivität) weitreichend untersucht wurden, währenddererseits eine Reihe von Defiziten erkennbar werden. Vermißt wird insbesondere eine detaillierte Analyse von Marketing und Vertrieb von Software hinsichtlich bestehender spezifischer Probleme sowie eine integrierende Betrachtung verschiedener Teilaspekte. Die Ursache dafür dürfte u.a. in einer ungenügenden Bestimmung der betriebswirtschaftlichen Eigenschaften der Software zu suchen sein.

---

1 Tüschen (1989), S. 331
2 Ebenda, S. 329

4. Spezifische Eigenschaften von Software

Ausgehend von der im Abschnitt 2 dargestellten Klassifizierung von Software sollen in Auswertung der im Abschnitt 3 genannten Forschungsbefunde *spezifische Eigenschaften von Software aus betriebswirtschaftlicher Sicht* abgegrenzt werden. Diese sollen Grundlagen zur Charakterisierung des innovativen Charakters von Software schaffen, wie sie als Zielstellung des Aufsatzes formuliert wurden.

(a) Emanzipation der Software von der Hardware

Zu Beginn der Entwicklungsgeschichte der modernen Rechentechnik in den 50er/60er Jahren war die Software in Form von Ein- und Ausgaberoutinen und Programmroutinen integraler Bestandteil der Hardware und insofern für eine separate Wirtschaftlichekeitsbetrachtung nicht von Interesse. Erst mit der zweiten und dritten Rechnergeneration bekam die Software selbständige Bedeutung. Heute, an der Schwelle zur künstlichen Intelligenz und der fünften Computergeneration ist die Software zu einem selbständigen Produkt geworden. Es hat sich weltweit ein Softwaremarkt entwickelt. Trotzdem ist ein besonderes Verhältnis zwischen den Produkten Hard- und Software bestehen geblieben. Einerseits ist Hardware nicht ohne Software betreibbar, andererseits gäbe es keine Softwareentwicklung ohne die technischen Voraussetzungen der Hardware. Aber noch ein anderer Prozeß von betriebswirtschaftlicher Relevanz ist zu beobachten. Software wird insbesondere im Mikrocomputerbereich zu einem wichtigen, wenn nicht dem entscheidenden Verkaufsargument, da sich mit der Homogenisierungstendenz der Hardware (Industriestandard IBM) die Produktvorteile zunehmend auf die Software konzentrieren.[1]

(b) Identität von Entwicklungs- und Produktionsprozeß

Als spezifisch im Vergleich zu industriellen Produkten stellt sich der Prozeß der Entwicklung und Produktion von Software dar. Ausgangspunkt kann ein Phasenmodell der Leistungserstellung und -verwertung sein, wie es von vielen Autoren[2] vertreten wird:

---

1 Brockhoff (1988), S. 8

2 Beispielhaft für die Vielzahl der Darstellungen seien genannt Boehm (1976), S. 1226 ff., Cave/Maymon (1988), S. 19 ff., Sneed (1987), S. 24 ff., Balzert (1982), S. 17 ff. Die Ansätze zur Modellierung des Softwareentwicklungsprozesses kommen aus der Informatik und sind dort als "software life cycle" bekannt. Die Begriffswahl steht nicht im inhaltlichen Zusammenhang mit dem in der Betriebswirtschaft beschriebenen Lebenszyklus von Produkten.

(1) Spezifikationsphase
Problemanalyse, Definition der Solleigenschaften der Software, Planung

(2) Entwurfsphase
Problemlösung durch logische Gliederung der Funktionen und Daten, Algorithmierung

(3) Implementierung
Transformation der Problemlösung in Programmanweisungen für den Zielrechner, inkl. Testung

(4) Abnahme und Einführung
Nachweis der Vertriebsfähigkeit und Markteinführung

(5) Marketing und Vertrieb (bei Standardsoftware)
Laufender Vertrieb der Software

(6) Nutzung/Wartung und Pflege
Ständige Wartung und Pflege von in Nutzung befindlicher Software, d.h. Änderung der in Nutzung befindlichen Software zur Beseitigung von Fehlern und Anpassung an veränderte Hardwarebedingungen oder Kundenforderungen

Die Untersuchungen zu den Phasen der Softwareerstellung und -verwertung stimmen dahingehend überein, daß es sich beim Softwareentwicklungsprozeß um einen iterativen Prozeß handelt, bei dem jede Iteration auf dem Ergebnis der letzten Iteration aufbaut.[1]

Vermißt wurde in allen Darstellungen jedoch wieder die Phase (5), Marketing und Vertrieb, die für Standardsoftware von besonderer Bedeutung ist und institutionalisiert sein kann (z.B. selbständige Software-Händler[2]).

Wo liegt die Besonderheit im Vergleich zu anderen Produkten? Die Phasen (1) bis (4) umfassen den Entwicklungsprozeß der Software, der mit dem Produktionsprozeß identisch ist (das Vervielfältigen von Disketten und Dokumentationen einmal nicht als separaten "Produktionsprozeß" betrachtet, sondern als Bestandteil des Gesamtprozesses). Daraus folgt, daß die Software nach Ablauf des Entwicklungs-/Produktionsprozesses beliebig oft verfügbar ist und - sofern Kunden vorhanden sind - verkauft werden kann.

---

1 Diese Überlegungen basieren auf dem Wasserfallmodell von Boehm (1976), S. 1226 ff.
2 Neugebauer (1986), S. 243 beschreibt den Software-Händler als Unternehmenstyp von Softwarehäusern.

(c) Drei Dimensionen der Wartung

Der sich parallel zur Softwarenutzung vollziehende Wartungsprozeß weist im Vergleich mit industriellen Produkten Besonderheiten auf. Im Normalfall umfaßt die Software-Wartung (maintenance)

- corrective maintenance (Korrektur von Fehlern)
- adaptive maintenance (Anpassen an veränderte Umgebung)
- perfective maintenance (Verbessern des Leistungsumfangs und der Leistungsqualität)[1].

Neben der Fehlerbeseitigung beinhaltet die Wartung damit Verbesserungen in Form von Weiterentwicklung der Software (update). Es entstehen neue Versionen, Modifikationen oder Ausgaben von Softwareprodukten. Das Interessante daran ist, daß diese gleichzeitig in drei Dimensionen wirkt.

(1) Der *Entwickler* hat im Wartungsprozeß die Software weiterentwickelt. Er kann von diesem neuen Niveau bei weiteren Entwicklungsarbeiten ausgehen.

(2) *Der einzelne Softwarenutzer*, der einen Fehler beanstandet, erhält vom Entwickler eine Korrekturinformation.

(3) Darüberhinaus werden die updates gleichzeitig für *alle Nutzer* verfügbar. Dies kann kostenlos oder mit einem Preis für die Nutzung der updates verbunden sein. Dem liegen ganz offensichtlich unterschiedliche Absatzstrategien der Softwareanbieter zugrunde.

Auf Grund der besonderen Bedeutung der Wartung von Software haben sich spezifische Organisationsformen in den Unternehmen[2] entwickelt.

(d) Divergenz von Wartungs- und Entwicklungsaufwand

Von besonderem Interesse für die betriebswirtschaftliche Betrachtung erweist sich das bereits genannte Verhältnis von Entwicklungs- zu Wartungsaufwand. Empirische Untersuchungen stimmen darin überein, daß lediglich 30 % bis max. 50 % des Gesamtaufwandes Entwicklungsaufwand sind, hingegen mindestens 50 % bis 70 % Wartungsaufwand darstellen.[3] Der hohe Aufwand während der Nutzung resultiert im wesentlichen daraus, daß die ersten Entwicklungsphasen (Spezifikation und Entwurf) nur in geringem Umfang durch technologische Werkzeuge unterstützt werden. Die meisten Werkzeuge unterstützen die Implementierungs-

---

1 Lientz/Swanson (1980)
2 Lehner/Hartl (1990)
3 Sneed (1987), S. 44 ff., Griese et al. (1987), S. 530 ff.

phase, d.h. die dritte und letzte Phase des Entwicklungsprozesses.

Da die Fehlerbeseitigung bekanntlich mit fortschreitendem Entwicklungsprozeß immer aufwendiger wird, ist der bisher hohe Anteil der Wartung am Gesamtaufwand nicht verwunderlich.[1] Die Bestrebungen zur Verbesserung der Wirtschaftlichkeit von Softwareentwicklungen setzen daher vor allem bei der Vervollkommnung der Softwareentwicklungstechnologie in frühen Phasen an (Requirement Engineering).[2]

Für den Entwickler dürfte sich der Übergang vom Entwicklungsprozeß zum Wartungsprozeß allerdings gleitend vollziehen. Praktisch stellt die Entscheidung der Geschäftsleitung über den Zeitpunkt der Einführung des Softwareproduktes am Markt den Übergang zu Phase (5) und (6), d.h. zum Wartungsprozeß dar.

(e) Charakteristischer Zahlungsstrom im Software-Lebenszyklus

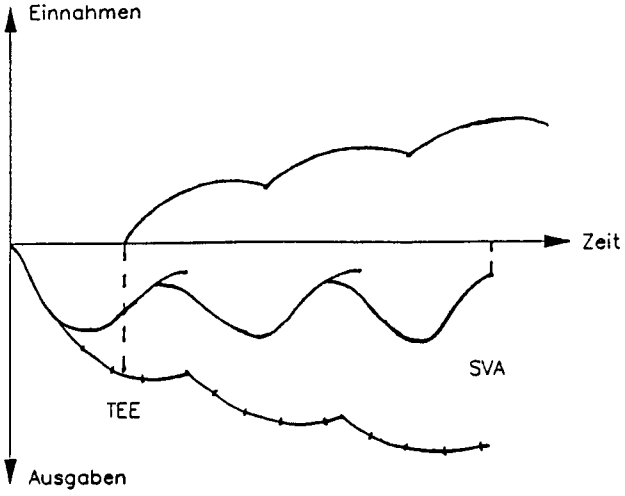
Im Unterschied zum glockenförmigen, wenn auch asymmetrischen Kurvenverlauf des Produktlebenszyklus bei "traditionellen" Produkten, werden für Software andere Verläufe erwartet. Diese Annahme unterstützen auch Cave/Maymon[3]. Sie gehen davon aus, daß für Software die Ausgaben nach dem Ersteinsatz nicht wie bei anderen Forschungs- und Entwicklungsprojekten drastisch zurückgehen, sondern in der gesamten Nutzungszeit mit einem beachtlichen Aufwandsanfall infolge der Wartung zu rechnen ist. Daraus leiten sie einen tendenziellen Kurvenverlauf ab.[4] Es ist jedoch nicht zu erwarten, daß alle Softwaretypen dieser Tendenz entsprechen. Vielmehr sind recht unterschiedliche Kurvenverläufe zu vermuten.

Ausgehend von der Softwareklassifizierung entsprechend Abb. 3 werden in Abb. 7 Ansätze für mögliche typbezogene Verlaufsformen hypothetisch dargestellt. Daraus folgt, daß der von Cave/Maymon beschriebene Verlauf in etwa mit dem Typ III (branchenbezogene Anwendungsprogramme) zu identifizieren ist.

---

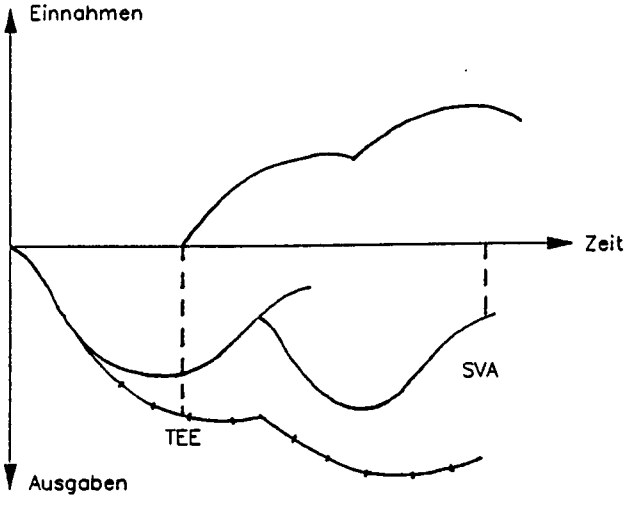
1 Scheer (1988), S. 6 f.
2 ebenda, S. 7
3 Cave/Maymon (1988), S. 18
4 ebenda, S. 20

Zahlungsstrom im Software-Lebenszyklus	Charakteristik



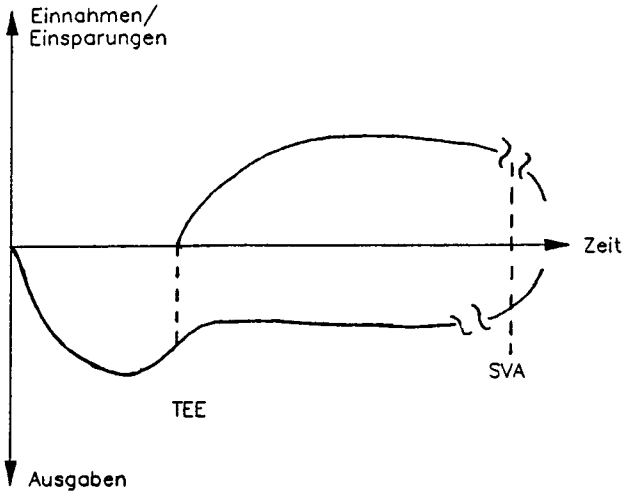
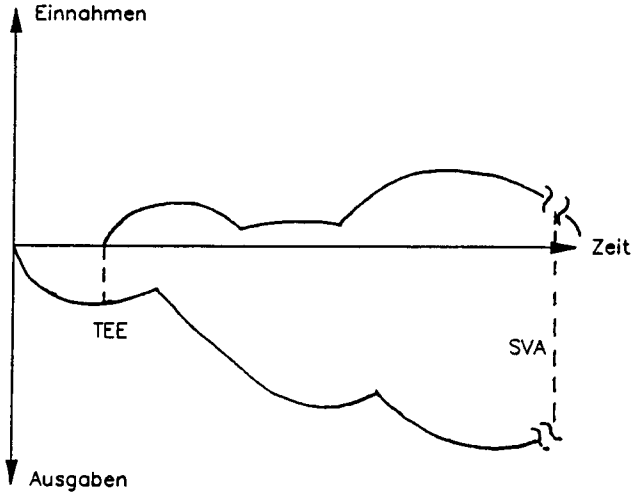
TYP I: Betriebssysteme/
Dienstprogramme

In kurzen Zeitabständen entstehen neue Versionen der Software.
Die Wartung für die obsolet werdenden Versionen wird nach kurzer Zeit eingestellt. Die Zuwachsraten der Einnahmen werden mit steigender Versionsanzahl kleiner, da viele Kunden updates zu günstigen Konditionen erwerben können.



TYP II: Systemnahe Software/
Übersetzungsprogramme/
kommerzielle wissenschaftlich-
technische und betriebs-
wirtschaftliche Programme

Die Einnahmen und Ausgaben zeigen einen tendenziell ähnlichen Verlauf wie für Typ I. Die Zeitintervalle des Erscheinens neuer Versionen sind allerdings länger bemessen.

Zahlungsstrom im Software-Lebenszyklus	Charakteristik
 TYP III: branchenbezogene Anwendungsprogramme (z.B. betriebliche Informationssysteme)	System hat eine lange Nutzungsdauer (10 - 12 Jahre), im gesamten Zeitraum ist Wartung erforderlich und werden stetig Einsparungen/Einahmen erzielt
 TYP IV: Individualsoftware	Ein möglicher Fall. Der Entwickler hofft zunächst mit wenig Ausgaben eine problemgerechte Software zu erstellen. Das bringt nicht den gewünschten Effekt und er muß kräftig investieren, um doch noch eine nutzbare Lösung zu erhalten.

Legende:

——————— Summenkurve des Aufwandes
TEE Tauglichkeit zur Erstanwendung
SVA Softwareveralten

Abb. 7: Typen unterschiedlicher Zahlungsströme im Software-Lebenszyklus

Neben der Art der Software müßten weitere Einflußfaktoren auf die Herausbildung bestimmter Typen untersucht werden, so z.B.

- große, mittlere und kleinere Projekte (wertmäßig und nach Anzahl der Quellprogrammzeilen -LOC-)
- sehr erfolgreiche, durchschnittlich erfolgreiche und nicht erfolgreiche Projekte
- Entwicklersicht und Nutzersicht (so weit nicht identisch).

(f) Multidimensional bestimmte Softwarequalität

Wie die Bestimmung der Produktqualität[1] im allgemeinen, so erweist sich die Bestimmung der Softwarequalität im besonderen als sehr kompliziertes Problem. Während bis zum Beginn der 70er Jahre Softwarequalität vor allem an der Fehlerhäufigkeit des Endproduktes Software gemessen wurde,[2] versucht Boehm als einer der Ersten, Softwarequalität komplex zu messen und mittels einer Baumstruktur einzelne Qualitätsmerkmale sichtbar zu machen.[3] Dieser Gedanke wurde weitergeführt zu den heute allgemein anerkannten Qualitätsmerkmalen von Software:

- Funktionale Vollständigkeit
- Zuverlässigkeit
- Benutzerfreundlichkeit
- Effizienz (im Sinne von optimaler Ausnutzung der Computerressourcen)
- Wartbarkeit
- Erweiterbarkeit (im Sinne von Modularität, Flexibilität, Portabilität)
- Übertragbarkeit.[4]

Die Bildung eines aggregierten Qualitätsurteils wird mit Hilfe eines Scoring-Modells versucht. Die Auswahl der Qualitätsparameter ist technisch bedingt. Bei der Gewichtung muß erwartet werden, daß die Präferenzen bezüglich der einzelnen Qualitätsmerkmale von Softwareentwicklern und Softwarenutzern teilweise unterschiedlich gesetzt werden. So dürfte z.B. aus Sicht der Minimierung späterer Wartungskosten das Interesse des Entwicklers an der Wartbarkeit der Systeme ausgeprägter sein als das des Anwenders, der in der ersten Zeit der Nutzung (in der mit der größten Fehlerhäufigkeit zu rechnen ist) Wartungsleistungen im Sinne von Garantieleistungen fordert.

Betriebswirtschaftliche Fragestellungen ergeben sich aus der Diskussion um die Softwarequalität in zwei Richtungen. Zum einen wäre zu untersuchen, ob außer den beschriebenen Qualitätsmerkmalen Merkmale existieren, die in stärkerem Zusammenhang mit der Wirtschaftlich-

---

1 Brockhoff (1988), S. 27 ff.
2 Griese et al. (1987), S. 522
3 Boehm/Brown (1978), S.
4 Sneed (1987), S. 98 f.

keit der Software stehen. Zum anderen ist die Frage nach einer nutzerbezogenen Gewichtung der Qualitätsparameter als Grundlage von Kaufentscheidungen zu stellen.[1]

(g) Spezifische Rechtslage der Software

Da Software ihrem Wesen nach eine Information ist, kann sie multivalent genutzt werden. Aus dieser Tatsache ergibt sich die spezifische Rechtsproblematik der Software, die darin besteht, daß Software weder vollständig den Rechtsnormen materieller Güter noch der immaterieller Güter entspricht.[2] Herkömmliche Rechtssysteme zum Schutz von Vermögensinteressen, wie das Patent-, Urheber-, Wettbewerbs-, Trade Secret- und Vertragsrecht bereiten Probleme bezüglich ihrer Anwendung für Software.[3] Dabei gibt es im Vergleich von der Bundesrepublik Deutschland zu den USA unterschiedliche Rechtsentscheide.

Patentrecht

In der BRD können Computerprogramme nicht als technische Erfindung angesehen werden und sind damit nicht patentfähig.[4] Das Patentrecht der USA trifft dazu keine negative Aussage.[5]

Urheberrechtsschutz

Für neuartige Computerprogramme wird in der BRD gemäß § 2 des Urheberrechtsschutzgesetzes Urheberrechtsschutz gewährt.[6] In der Rechtspraxis ist die Anwendung auf Quellprogramme bisher unstrittig, für Objektprogramme wird ein Schutz als Copy des Quellprogramms diskutiert.[7] In den USA wird der Urheberrechtsschutz für Quellprogramme auf Grund der Originalität ebenfalls bejaht, strittig ist die Frage für Objektprogramme.[8]
Diese nicht eindeutig geklärte Rechtslage hat für den Softwareentwickler vor allem wirtschaftliche Folgen: Der Gesamtaufwand für die Herstellung eines Softwareproduktes ist im wesentlichen einmaliger Aufwand (Forschungs- und Entwicklungsaufwand, Wartungsaufwand). Der anfallende laufende Aufwand für Datenträger und Dokumentation sowie der Aufwand für Marketing und Vertrieb pro verkauften Softwareprodukt sind im Vergleich zur Hardware

---

1 Dies scheint insbesondere wichtig, da das Qualitätsbewertungssystem auch von der Industrie empfohlen wird (z.B. durch SIEMENS vgl. Sneed (1987), S. 98)
2 Beckurts/Schuchmann (1986), S. 203, Lehner/Friedwagner/Pernsteiner (1989), S. 2
3 Kullmann (1986), S. 163
4 ebenda, S. 57
5 ebenda, S. 112
6 ebenda, S. 84 ff., Hoeren (1990), S. 22
7 Kullmann (1986), S. 84 ff.
8 ebenda, S. 132 ff.

gering. Eine optimale Verwertung der eingesetzten Mittel ist folglich nur über möglichst hohe Verkaufszahlen erreichbar. Die unentgeltliche Weitergabemöglichkeit von Software wirkt dieser Entwicklerstrategie entgegen. Jeder Nutzer einer "Raubkopie" ist ein Kunde weniger. Das Problem verschärft sich bei Veräußerung von Software.

An Software kann kein Eigentumsrecht, sondern nur ein Nutzungsrecht, im Sinne eines immateriellen Produktes, oft in Form eines Lizenzvertrages, vergeben werden. Dabei ist die juristische Kontrolle über die Einhaltung des Rechtszustandes, d.h. keine unbefugte Weitergabe an Dritte, auf Grund der Multivalenz-Eigenschaft stark eingeschränkt. Von diesem Problem zu trennen sind strategische Entscheidungen einzelner Softwareanbieter, die mit einer aggressiven Produktpolitik versuchen, Konkurrenten aus Marktsegmenten zu verdrängen oder fern zu halten. Im Falle einer solchen Marktstrategie ist es denkbar, daß in einer bestimmten Zeit Nutzungsrechte freizügig vergeben werden, z.B. selbständiges Kopieren von Software und Nutzung auf mehreren Anlagen durch einen Kunden.

Neben der unentgeltlichen Nachnutzung von Software bietet sich neuartige, innovative Software aber in besonderem Maße auch zur Imitation an. Wie die Clones auf dem Hardwaremarkt, so ist auch der Softwaremarkt voll von erfolgreichen Imitationen. Unter Umständen kann es gerade die Imitationsstrategie sein, die ein Zurückgewinnen gefährdeter Geschäftsfelder ermöglicht, wie dies mit der Übernahme der Benutzerphilosophie von Apple Macintosh in das Softwareprodukt "Windows" gelungen ist.[1]

Die Möglichkeit der unentgeltlichen Nachnutzung von Software führt jedoch beim Entwickler ganz offensichtlich auch zu positiven Effekten. In seinem Bestreben, dem wirtschaftlichen Verlust durch die "Raubkopierer" entgegenzuwirken, geht er im wesentlichen zwei Wege. Zum einen wird versucht, durch wirksameren Kopierschutz - also auf technischem Wege - Nachnutzungs- und Imitationsbarrieren zu errichten. Zum anderen ist insbesondere den Imitatoren seitens der Entwickler nur durch eine ständige Verbesserung und Erneuerung der Software - also auf *innovativem Wege* - durch die Schaffung von zeitlichem Vorsprung vor der Konkurrenz zu begegnen.

Die weitere Aufhellung des Zusammenhangs von spezifischer Rechtssituation der Software und betriebswirtschaftlichen Konsequenzen ist ein Untersuchungsfeld, das bisher kaum Beachtung fand.

---

1 o.V., Apple: New Team, New Strategie (1990), S. 43

5. Der innovative Charakter von Software

5.1. Bestimmungsfaktoren des Softwareinnovationsprozesses

Software wird im allgemeinen als besonders innovativ charakterisiert.[1] Was bedeutet aber in diesem Zusammenhang "besonders innovativ"?

Hauschildt definiert *Innovationen* als "qualitativ neuartige Produkte oder Verfahren, die ein bestimmtes System (z.B. eine Unternehmung) erstmalig in den Markt oder in den Betrieb (in Produktion oder Administration) einführt."[2]
Ansatzpunkte für eine Kategorisierung von Software-Innovationen untersucht Brockhoff.[3] Für einen speziellen Fall zeigt er, daß die Softwareinnovation nach Softwareleistungen und Bezugssystem differenziert werden kann (vgl. Abb. 8).

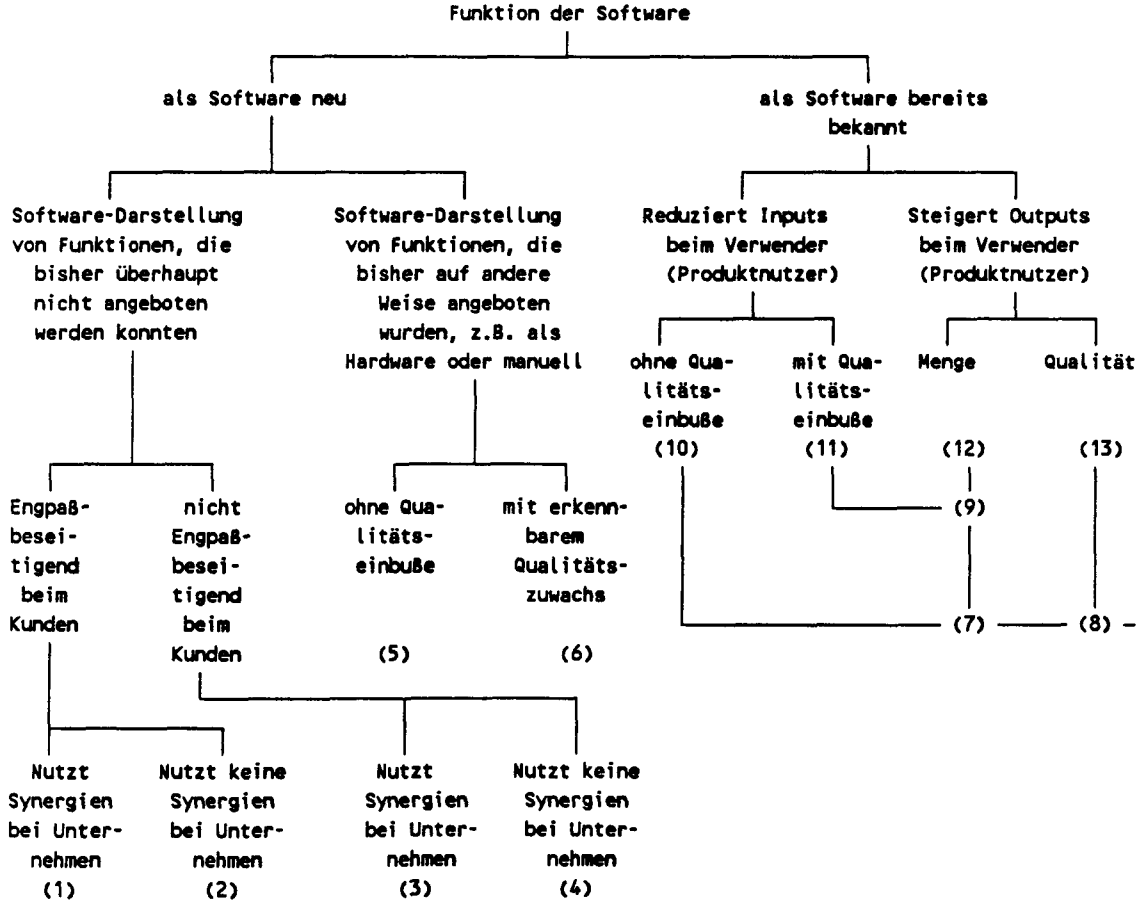


Abb. 8: Beurteilung des Innovationsgrades von Software nach Brockhoff (1989)

---

1 Dreckmann (1987), S. 24.02.03
2 Hauschildt (1990), S. 263
3 Brockhoff (1989)

Zur weiteren Bestimmung des innovativen Charakters der Software soll eine detaillierte Betrachtung von Faktoren

- des Umfeldes (externe Faktoren des Softwareinnovationsprozesses) und
- des Innovationsprozesses selbst (interne Faktoren des Softwareinnovationsprozesses).

Die Faktoren werden hinsichtlich ihrer innovationsfördernden oder -hemmenden Wirkung differenziert (vgl. Abb. 9).

	Faktoren	
	extern - Umfeld -	intern - Innovationsprozeß -
innovations-fördernde Wirkung	* verschärfter Wettbewerb * dynamischer Nachfrage-überhang * echte Marktnischen * weltweit operierende Firmen * strategische Allianzen * Beschleunigung der Innovationszyklen * Imitatoren als "Diffusionshilfe"	* leistungsfähige Innovationspotentiale, *modernes Know-how* * gute Kapitalausstattung * innovative Software-Strategie * Innovationsschub durch neue Standardsoftware oder Hardware * innovative Momente der Wartung und der Softwaretechnologie
innovations-hemmende Wirkung	* Kopieren und Konservieren von Entwicklungsniveaus infolge Imitation * Industriestandards als Innovationsbarriere * geringe Markttransparenz, starke Zersplitterung * Marktfluktuation von Unternehmen/Know-how-Verluste	* ungenügende Personal- und Kapitalausstattung * ungenügende oder falsche strategische Orientierung * Widerspruch zwischen technological push und Wirtschaftlichkeit * Zeitkonflikte bei Fehlerbeseitigung und Bestimmung von Nutzungsdauern

Abb. 9: Bestimmungsfaktoren des Softwareinnovationsprozesses

5.2. Die externen Faktoren des Software-Innovationsprozesses

Wie im Abschnitt 2 dargestellt, ist das Umfeld der Softwareentwicklung durch die Existenz eines stetig wachsenden Software-Marktes gekennzeichnet. Es ergibt sich danach die Frage,ob Anhaltspunkte für erfolgreiches Innovationsverhalten von Unternehmen auf diesem Markt erkennbar sind.

Akzeptiert man einmal den Umsatz als Erfolgskriterium, so lassen sich für die in Abb. 2 ausgewiesenen erfolgreichsten Anbieter von PC-Software 1989 typische Elemente einer Innovationsstrategie festmachen:

(a) starke Konzentration auf eine Betriebssystemlinie, Durchsetzen dieser Linie als Industriestandard (MS DOS) selbst auf die Gefahr hin, eigenen weiterentwickelten Produkten (OS 2) eine Akzeptanzbarriere beim Nutzer entgegen zu seztzen,[1]

(b) Spezialisierung auf eine oder wenige Produktlinien (z.B. Lotus, Ashton-Tate auf dBase, WordPerfekt), mit dem Ziel von höchstem technologischen Know how,

(c) Hervorbringen von innovativen Produkten auf der speziellen Produktlinie mit hohem Einsatz an FuE-Mitteln, bei hoher Qualität der Software,

(d) Sicherung der Aufwärtskompatibilität und Protabilität der angebotenen Software (z.B. dBase),

(e) intensive Marktarbeit, insbesondere lang anhaltende Werbekampagnen unter Ausnutzung der oft geringen Markttransparenz,

(f) Komplettierung der Software durch benutzerfreundliche Dokumentationen, Handbücher, Serviceleistungen bei Softwarefehlern und preisgünstige Vergabe von aktualisierten Versionen (updates), d.h. Full-Service.[2]

Für die betriebswirtschaftliche Behandlung sind zwei Frage ableitbar:

- Gibt es Hinweise, daß einzelne Optionen besonders wirkungsvoll für den Innovationserfolg sind?[3]
- Existieren weitere Elemente der Strategie?

Neben dem Wachstum des Software-Marktes sind weitere externe Faktoren erkennbar, die den Prozeß der Software-Innovation fördern.

- Der Software-Markt ist durch *harten Konkurrenzkampf* zwischen den Wettbewerbern

---

1 Hauschildt/Leker (1990), S. 971

2 Wesentliche Elemente dieser Strategie (Produktorientierung, Spezialisierung, Qualität, Know-How-Vorteil) bestätigt die empirische Untersuchung von Neugebauer in Bezug auf ihren Beitrag zum Unternehmenserfolg, Neugebauer (1986), S. 238

3 Die Untersuchung von Neugebauer bestätigt dies bezüglich des technologischen Vorteils, vgl. Neugebauer (1986), S. 249

gekennzeichnet, der kein Verharren auf Erfolgspositionen zuläßt. Die führenden amerikanischen Software-Firmen werden zunehmend von Konkurrenten aus Japan und Europa bedrängt. Ständige innovative Softwareentwicklung und hohe Qualitätsnormen erweisen sich zunehmend als ausschlaggebend für den strategischen Wettbewerbserfolg.[1]

- Ein weiterer innovationsfördernder Faktor dürfte der bestehende *Nachfrageüberhang* in bestimmten Teilsegmenten des Marktes (z.B. branchenbezogene Automatisierungssoftware) sein, der für kleine Anbieter erfolgreiche Innovationen in einer *Marktnische* ermöglicht.

- Den Software-Markt kennzeichnet die Möglichkeit zu schnellem Transfer von innovativem Know-how. So operieren Marktführer wie IBM oder Borland über *Tochterfirmen* auf dem europäischen oder japanischen Markt[2], was eine schnelle Diffusion von Innovationen begünstigt.

- Vorteilhafte Bedingungen für erfolgreiche Innovationen garantieren auch kooperative Wettbewerbsstrategien s.g. *strategische Allianzen von Hardware und Softwareherstellern*[3] (z.B. IBM und Microsoft bezüglich des Betriebssystems MS DOS), die eine gemeinsame, abgestimmte Entwicklungsstrategie und eine offensive Marktarbeit ermöglichen.
  Das ist besonders bemerkenswert, wenn man bedenkt, daß Anfang der 80er Jahre die selbständige Herstellung von Software in Softwarehäusern als die "zweite Kraft" angesehen wurde, die sich in einer Wettbewerbssituation zu den Hardwareherstellern befindet.[4]

- Eine innovationsfördernde Wirkung weist ebenfalls die für den Bereich der Mikrocomputer festgestellte Tendenz zur *Beschleunigung der Innovationszyklen.*[5] Das zwingt die Entwickler, sich auf immer neue Anforderungen einzustellen und die Systeme insbesondere im Bereich der Standardsoftware im Prinzip jährlich durch innovative Komponenten zu verbessern.

- Last but not least sind die *Imitatoren* zu nennen, die neben einer innovationshemmenden Wirkung auch den Diffusionsprozeß von Softwareinnovationen unterstützen können; ihre Wirkung ist also ambivalent.

---

1 Brandt/Schwartz/Gross (1991), S. 65
2 ebenda, S. 66
3 Backhaus/Piltz (1990), S. 1 ff.
4 Griese (1982), S. 151
5 Sneed (1987), S. 23

Neben innovationsfördernden bestehen im externen Umfeld des Softwareinnovationsprozesses eine Anzahl von *innovationshemmenden Faktoren*:

- Die spezifische Rechtslage der Software (vgl. Pkt. 4 (g)) begünstigt in starkem Maße das Verfolgen von *Imitationsstrategien* mit allen damit verbundenen Nachteilen für den Innovationsprozeß. Diese resultieren zum einen aus der Konservierung eines bereits allgemein bekannten Erkenntnisstandes in der eigenen Entwicklungsarbeit und zum anderen aus der Wirkung auf das Subjekt des Innovationsprozesses, den Entwickler. Über lange Zeit betriebene Imitationsstrategie führt zum Versiegen kreativer Ideen bei den Softwareentwicklern.

- Mit der Durchsetzung bestimmter *Industriestandards* für Hardware und/oder Software (z.B. Kompatibilität zu IBM-PC und MS DOS) wird eine bestimmte "Entwicklungsphilosophie" zum Standard erhoben, der das Durchsetzen von kreativen Alternativkonzepten (z.B. auf UNIX-Basis) behindert.

- Der Software-Markt ist in seiner *Anbieterstruktur stark zersplittert*. Umsatzstarke Hardwarehersteller stehen neben kleinen Anbietern aus Programmiererbüros und Softwarehäusern. Dies führt zu einer auf bestimmten Segmenten *geringen Markttransparenz*, was die Diffusion von innovativen Softwareprodukten erschwert.

- Der Software-Markt ist durch hohe *Marktfluktuation* von Firmen aus wirtschaftlichen Gründen gekennzeichnet. Das bringt Know-how-Verluste für den Entwicklungsprozeß mit sich. Kapitalmangel kann sich trotz einiger positiver Beispiele (Borland International Inc. mit Turbo-Pascal)[1] als starke Innovationsbarriere erweisen.

5.3. Die internen Faktoren des Softwareinnovationsprozesses

Ausgehend von den eingangs geschilderten Spezifika der Software lassen sich interne Faktoren isolieren, die den Innovationsprozeß von Software fördern.

- Die Innovationsfähigkeit eines Unternehmens bezüglich der Software wird, wie empirische Untersuchungen[2] belegen, vom *technisch-technologischen Know-How der Entwickler*, ihrer Qualifikation, den materiellen und finanziellen Möglichkeiten des Unternehmens *Innovationspotentiale* zu schaffen, bestimmt. Leistungsfähige Potentiale sind dem Innovationsprozeß förderlich.

---

1 Brandt/Schwartz/Gross (1991), S. 62
2 Neugebauer (1986)

- Eine *Softwareentwicklungsstrategie*, die auf Innovation nicht ausschließlich auf Imitation ausgerichtet ist, schafft wesentliche interne Voraussetzungen für die Entstehung von innovativer Software.

- Software unterliegt nicht wie materielle Produkte einem natürlichen Abnutzungsprozeß,[1] sondern sie *veraltet* mit dem Erscheinen neuer, leistungsfähigerer Softwareprodukte auf dem Markt. Diese Wirkung tritt insbesondere mit dem Vertriebsbeginn von neuer Standardsoftware ein. Für die auf den obsolet werdenden Standardlösungen aufbauende Anwendersoftware ergeben sich Möglichkeiten der Weiterentwicklung, die einen innovativen Schub mit dem Erscheinen neuer Softwaregenerationen bewirken können.

  Es zeichnen sich Phasen des Zusammenhangs der Innovationsprozesse von Standard- und Anwendersoftware ähnlich dem inneren Zusammenhang von Produkt- und Prozeßinnovationen[2] ab.

  Phase 1: Die Leistungsfähigkeit der Anwendersoftware liegt auf niedrigem Niveau. Mit dem Erscheinen von neuer, leistungsfähiger Standardsoftware ergeben sich Optionen für eine Erhöhung der Leistungsfähigkeit der Anwendersoftware. Es beginnt ein induzierter Innovationsprozeß.

  Phase 2: Neue Versionen von Standardsoftware führen zu weiteren Verbesserungen der Anwendersoftware.

  Phase 3: Die der Anwendersoftware innewohnenden innovativen Möglichkeiten werden mit dem Effekt der Leistungssteigerung wirksam, ohne daß weitere Impulse durch weiterentwickelte Standardsoftware erfolgen.

  Nach Ausnutzung aller Möglichkeiten zur Leistungssteigerung im Rahmen der im Unternehmen verfügbaren Standardsoftware in Phase 3 beginnt der Prozeß von neuem, wenn innovative Standardsoftware am Markt erscheint. Der gleiche Zusammenhang läßt sich für das Erscheinen neuer Hardwaregenerationen bezüglich der Standardsoftware beschreiben.

- Innovationsfördernd wirken ebenfalls die bereits beschriebenen Weiterentwicklungsarbeiten im Rahmen der *Produktwartung* sowie der *Einsatz von modernen Softwareent-*

---

1 Merrmann (1983), S. 32.
2 Utterback/Abernathy (1975), S. 645

wicklungswerkzeugen insbesondere im Prozeß des Requirement Engineering aus.

Aus den genannten Faktoren können i.d.R. auch *interne Innovationshemmnisse* erwachsen:

- Eine *ungenügende Personal- (Quantität und Qualifikation) und/oder Kapitalausstattung* von softwareentwickelnden Unternehmen mindert die Möglichkeiten zur Innovation. Oft gelingt es auf Grund fehlender Möglichkeiten für zielgerichtetes Marketing nicht, kreative Softwareideen effizient umzusetzen.

- Falsche oder *ungenügende strategische Orientierung* erschwert das zielgerichtete Entwickeln von innovativen Softwareprodukten.

- Der *Zwang zur reaktiven Anpassung*, der für die Softwarehersteller mit der Durchsetzung bestimmter Industriestandards für Software und/oder Hardware entsteht (z.B. Kompatibilität zu IBM-PC und MS DOS), wird in einer durch besondere Aktivität gekennzeichneten Phase des innovativen Prozesses zunächst Impulse für die Softwareentwicklung auf den über dem Betriebssystem liegenden Ebenen der Standardsoftware und der Anwendersoftware auslösen. Gelingt es jedoch mit einem Betriebssystem zu einem bestimmten Zeitpunkt nicht mehr, notwendige Entwicklungsanforderungen zu realisieren, so wird ein *Widerspruch zwischen Technik (technological push) und Wirtschaftlichkeit* unvermeidbar sein. Die Umstellung aller Anwendungslösungen auf ein neues Betriebssystem wird unter Umständen wesentlich höheren Aufwand verursachen als der durch die Innovation erzielbare Nutzeffekt. Diese Tendenz wird sich mit der steigenden Komplexität betrieblicher Informationssysteme verschärfen.

- Im Zusammenhang mit diesem Wirtschaftlichkeitskonflikt steht auch ein *Zeitkonflikt*, der aus der unterschiedlichen Lebensdauer verschiedener Softwaresysteme resultiert. Während für betriebliche Anwendungslösungen eine durchschnittliche Lebensdauer von 10 bis 12 Jahren zu erwarten ist, beträgt diese für Mikrocomputersysteme inkl. der Standardsoftware maximal vier Jahre.[1] Aus diesem zeitlichen Widerspruch ist ein bestimmtes Beharrungsvermögen auf Anwenderebene zu erwarten, welches die schnelle Diffusion von Standardsoftware und die Entstehung von innovativer Anwendersoftware in bestimmten Grenzen behindert.

- Schließlich: Wartung hemmt weitere Innovation - die Beseitigung von Fehlern bindet Entwicklungskapazität.

---

1 Becker et al. (1990), S. 247

5.4. Der Innovationsgrad von Software

Sind die externen und internen innovationsfördernden oder -hemmenden Faktoren bestimmt, so ist zu der eingangs gestellten Frage zurückzukommen: Wann ist Software innovativ?

Als Meßgröße ist ein *"Innovationsgrad"* zu bestimmen. Da kein objektiver Maßstab bekannt ist, müssen intersubjektiv akzeptierte Orientierungshilfen, z.B. in Form von Fragenkatalogen[1] gefunden werden. Für Software können sich diese beispielsweise auf Unternehmen oder Softwareklassen (z.B. Datenbanken, Textsysteme) beziehen.

In Anlehnung an einen von Hauschildt[2] für materielle Produkte entwickelten Fragenkatalog könnte für Software der in Überblick 2 vorgeschlagene Katalog empirischer Untersuchungen zur Bestimmung des Innovationsgrades zugrundegelegt werden (vgl. Übersicht 2).

---

1. Gegenstand der Innovation
2. Bietet die neue Software
   - 2.1. neue Funktionen, die qualitativ über bisherige Software hinausgehen
   - 2.2. eine wesentliche Verbesserung der Benutzerfreundlichkeit, ggf. neue Benutzeroberfläche
   - 2.3. die gleichen Funktionen wie bisherige Software, aber mit wesentlich höherer Zuverlässigkeit
   - 2.4. Möglichkeiten des Rechtsschutzes
3. Erfordert oder erlaubt der Absatz der Software
   - 3.1. Erreichen bisher unbekannter Kundengruppen
   - 3.2. völlig neues Marketing-Konzept
   - 3.3. Auseinandersetzen mit neuer Konkurrenz
   - 3.4. neue Typen von Nutzungsverträgen
4. Verlangt die Entwicklung und Herstellung der Software
   - 4.1. völlig neue Hardware/Standardsoftware (z.B. Betriebssystem)
   - 4.2. externe Partner für die Entwicklung
   - 4.3. neues technologisches Know-how
   - 4.4. neue Projektorganisation und -controlling
5. Bietet die neue Software
   - 5.1. effizientere Lösungen/höhere Flexibilität
   - 5.2. Einsparung von Entwicklungs- und Wartungsaufwand
   - 5.3. Zeitersparnis

Übersicht 2: Fragenkatalog zur Beurteilung des Innovationsgrades von Software

---

1 Hauschildt (1990), S. 264
2 Hauschildt (1990), S. 264

6. Zusammenfassung und Ausblick auf die weitere Forschungsarbeit

Ausgehend von der Klassifizierung der Software und den vorhandenen Untersuchungen aus betriebswirtschaftlicher Sicht wurde versucht, spezifische *Eigenschaften der Software* herauszuarbeiten. Dabei wurden folgende Tendenzen ermittelt:

- Emanzipation der Software von der Hardware
- Identität von Entwicklungs- und Produktionsprozeß
- drei Dimensionen der Softwarewartung
- Divergenz von Wartungs- und Entwicklungsaufwand
- charakteristische Zahlungsströme im Softwarelebenszyklus
- multidimensional bestimmte Softwarequalität
- innovativer Charakter von Software
- spezifische Rechtslage der Software

Es wurde eine Reihe von *Defiziten* aufgezeigt, die für die weitere Forschungstätigkeit relevant sind. Als betriebswirtschaftlich besonders wesentliches Feld mit Integrationsfunktion erwies sich der *innovative Charakter der Software* und seine Messung mittels des Innovationsgrades. Für eine über Detailuntersuchungen hinausgehende betriebswirtschaftliche Betrachtung der Software wird zielgerichtete Arbeit in den eingangs fixierten Untersuchungsfeldern erforderlich werden (vgl. Abb. 11).

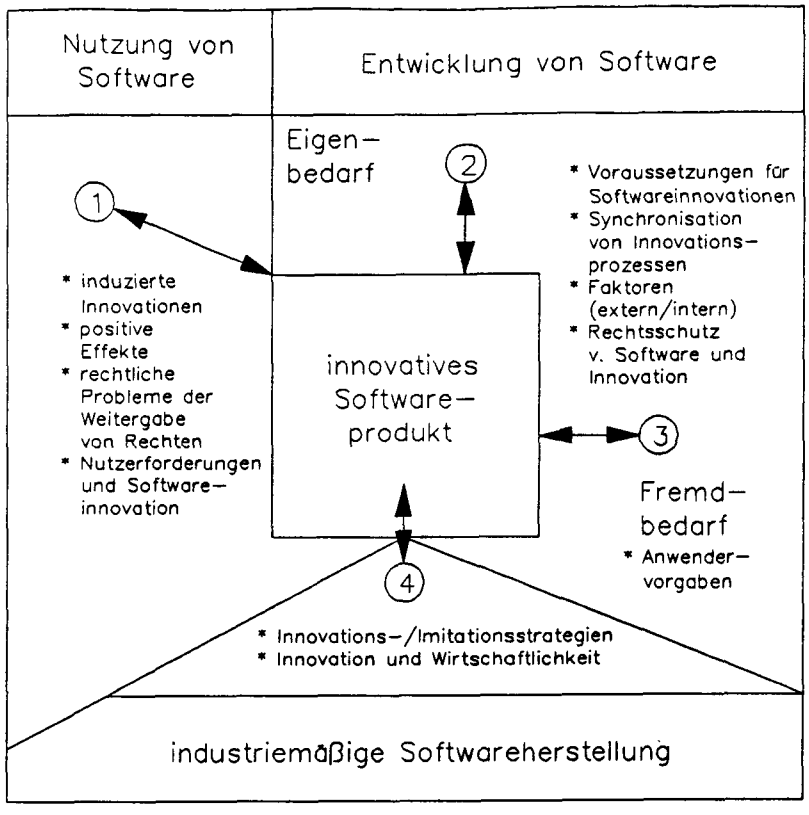


Abb. 11: Leitlinien der weiteren Forschungsarbeit auf dem Softwaresektor

Im Mittelpunkt weiterer Betrachtungen wird der Produktbegriff stehen. Für seine weitere betriebswirtschaftliche Erfassung ergeben sich vier Dimensionen:

(1) Software als innovatives Produkt oder als Produktimitation

(2) Software als in Entwicklung befindliches Produkt

(3) Software als marktsuchendes Produkt

(4) Software als marktbehauptendes Produkt,

die Ausgangspunkt für die Bestimmung von Handlungsabläufen in Softwareentwickelnden und -nutzenden Unternehmen sein können.

Literaturverzeichnis

Anselstetter, R. (1984), Betriebswirtschaftliche Nutzeffekte der DV. Springer-Verlag, Berlin et al., 1984

Backhaus, K., Piltz, K. (1990), Strategische Allianzen - eine neue Form kooperativen Wettbewerbs?. In: ZfbF-Sonderheft 1990/27, S. 1-10

Balzert, H. (1982), Die Entwicklung von Softwaresystemen. Mannheim et al., 1982

Becker, M., Haberfellner, R., Liebetrau, G. (1986), EDV-Wissen für Anwender. 6. überarb. Auflage, Verlag Industrielle Organisation, Zürich, 1986

Beckurts, K.-H., Schuchmann, H.-R. (1986), Wirtschaftsfaktor Informationstechnik - Chancen, Aufgaben, Trends. In: ZfbF 38 (1986) 3, S. 195-206

Boehm, B.W. (1984), Software Engineering Economics. In: Transaction on SE, Vol. 10, Nr. 1, January 1984

Boehm, B.W. (1976), Software Engineering. IEEE-Transaction on Computers, C-25 (1976), S. 1226-1241

Boehm, B.W. (1975), The High Cost of Software. In: Horowitz, E. (Hrsg.): Practical Strategies for Developing large Software Systems, Reading, 1975

Brandt, R., Schwartz, E.I., Gross, N. (1991), Can the U.S. Stay ahead in Software. In: Business Week, March 11, 1991, S. 62-67

Brockhoff, K. (1989), Kriterien für Software-Innovation. Auftragsarbeit (unveröffentlicht), Kiel, 1989

Brockhoff, K. (1988), Produktpolitik. Gustav Fischer Verlag, Stuttgart, 1988

Brooks (1979, The Mythical-Man Month, Essays on Software Engineering. 3. Aufl., Reading, 1979

Cave, W.C., Maymon, G.W. (1988), Leitfaden des Software-Projektmanagements. Forkel-Verlag, Wiesbaden, 1988

Computer-Praxis abc (1990), WRS-Verlag, München, 1990

Disterer, G. (1988), Anwendungsentwicklung durch Fachabteilungsmitarbeiter - Ausmaß und Ausprägungen. Dissertation, Kiel, 1988

Dreckmann, K.-H. (1987), Softwareentwicklungsstrategien und -trends. ONLINE-Kongress 1987, Velpert, 1987

Folberth, O.G. (1990), Fortschritt und kein Ende?. In: ONLINE (1990), Heft 10, S. 71-75

Griese, J. et al. (1987), Ergebnisse des Arbeitskreises Wirtschaftlichkeit der Informationsverarbeitung. In: ZfbF 39 (1987) 7, S. 515-551

Griese, J. (1982), Zur geschichtlichen Entwicklung der Softwarehäuser. In: Angewandte Informatik (1982), Heft 2, S. 146-151

Gutenberg, E. (1961), Grundlagen der Betriebswirtschaftslehre. 6. Auflage, Bd. 1, Die Produktion, Springer-Verlag, Berlin et al., 1961

Hansen, H.R. (1986), Wirtschaftsinformatik. Bd. 1, 5. Auflage, Gustav Fischer Verlag Stuttgart, 1986

Hauschildt, J. (1990), Innovationsmanagement. In: Schuster, J. (Hrsg.), Handbuch des Wissenschaftstransfers, Berlin et al., 1990, S. 263-282

Hauschildt, J., Leker, J. (1990), Flexibilisierung als Strategie von Anbietern und Nachfragern innovativer Güter. In: ZfbF 42 (1990) 11, S. 963-975

Hermann, O. (1983), Kalkulation von Softwareentwicklungen. Oldenbourg Verlag, München und Wien, 1983

v. Hippel, E. (1988), The Sources of Innovation. Oxford University Press, Oxford, 1988

Hoeren, T. (1990), Softwareweitergabe an Dritte. In: PC-Woche 5 (1990) 51/52, S. 22

IBM-Deutschland (Hrsg.) (1968), Handbuch für die EDV-Organisation. IBM Form 71 523-2, Sindelfingen, 1968

Kargl, H. (1989), Fachentwurf für DV-Anwendungssysteme. Oldenbourg Verlag, München und Wien, 1989

Knolmayer, G., Disterer, G. (1989), Anwendungsentwicklung durch Fachabteilungs-Mitarbeiter. In: Manuskripte aus dem Institut für Betriebswirtschaftslehre der Universität Kiel, Nr. 231, 1989

Knolmayer, G., Disterer, G. (1988), User Developed Applications: State of the Art in German Companies. In: Bullinger, H.-J. et al (Hrsg.), EUROINFO '88, Proc. of the First European Conference on Information Technology for Organisational Systems in Athen, May 1988, Amsterdamm et al., North Holland, 1988, S. 124-129

Kotler, P. (1989), Marketing-Management. 4. völlig neu bearbeitete Aufl., Pöschel Verlag, Stuttgart, 1989

Krallmann, H. (1990), Innovative Anwendung der Information- und Kommunikationstechnologie in den 90er Jahren. Oldenbourg Verlag, München und Wien, 1990

Kullmann, W. (1986), Der Schutz von Computerprogrammen und -chips in der Bundesrepublik Deutschland und in den USA. Duncker & Humblot, Berlin, 1986

Lehmann, J. (1979), How Softwareprojects are really managed. In: Datamation, Jan. 1979, S. 119-129

Lehner, F., Hartl, H. (1990), Organisation der Softwarewartung. Institut für Wirtschaftsinformatik und Organisationsforschung, Nr. 90.05, Linz, 1990

Lehner, F., Friedwagner, P., Pernsteiner, H. (1989), Bilanzierung von Software. Institut für Wirtschaftsinformatik und Organisationsforschung, Nr. 89.05, Linz, 1989.

Lientz, B.P., Swanson, E.B. (1980), Software Maintenance Management. Adison Wesley, Reading, 1980

Lientz, B.P., Swanson, E.B., Topkins, G.E. (1978), Characteristics of Applikation Software Maintenance. Communication of the ACM 1978/21, S. 466-471

Mertens, P. et al. (1982), Betriebswirtschaftliche Nutzeffekte und Schäden der EDV - Ergebnisse des NSI-Projektes. In: ZfB 52 (1982) 2, S. 135-153

Nagel, K. (1990), Nutzen der Informationsverarbeitung. Oldenbourg Verlag, München und Wien, 1990

Neugebauer, U. (1986), Das Software-Unternehmen. Empirische Untersuchung des Unternehmensverhaltens und der Faktoren des Unternehmenserfolgs. Oldenbourg Verlag, München und Wien, 1986

Noth, T., Kretzschmar, M. (1986), Aufwandschätzungen von DV-Projekten. 2. Auflage, Springer-Verlag, Berlin et al., 1986

Österle, H. (1990), Unternehmensstrategie und Standardsoftware - Schlüsselentscheidungen für die 90er Jahre. In: Österle, H. (Hrsg.), Integrierte Standardsoftware: Entscheidungshilfen für den Einsatz von Softwarepaketen, Bd. 1, AIT-Verlag, München, 1990

o.V. (1990), Apple: New Team, New Strategy. In: Business Week, 1990, Oct. 15, S. 40-46

o.V. (1987), die Rolle der Software Anfang der 90er Jahre. ONLINE-Kongreß, Velpert, 1987

Overlack, J. (1988), Wettbewerbsvorteile durch Informationstechnologie. Peter Lang, Frankfurt am Main et al., 1988

Platz, J., Schmelzer, H.J. (1986), Projektmanagement in der industriellen Forschung und Entwicklung. Einführung anhand von Beispielen aus der Informationstechnik. Springer Verlag, Berlin et al., 1986

Reschke, H., Schelle, H., Schnopp, R. (Hrsg.) (1989), Handbuch Projekt-Management. Verlag TÜV Rheinland, Köln, 1989

Saalfrank, R. et al. (1987), Produktivitätseffekte von Aufwandseinflußgrößen bei der Softwareentwicklung. In: Angewandte Informatik 1987/3, S. 95-103

Scheer, A.-W. (1988), Wirtschaftsinformatik - Informationssysteme im Industriebetrieb. Springer-Verlag, Berlin et al., 1988

Sneed, H.M. (1987), Software-Management. Rudolf-Müller-online-DV-Praxis, Köln, 1987

Stahlknecht, R. (1989), Einführung in die Wirtschaftsinformatik. Springer-Verlag, Berlin et al., 1989

Tüschen, N. (1989), Unternehmensplanung in Softwarehäusern, Eul-Verlag, Bergisch-Gladbach, 1989

Utterback, J.M., Abernathy, W.J. (1975), A Dynamic Model of Process and Product Innovation. Omega, Vol. 3, 1975, S. 639-682

Weinberg, G.M. (1971), The Psychology of Computer Programming. New York, 1971

Zukunftskonzept Informationsverarbeitung. BMFT, Bonn, 1988