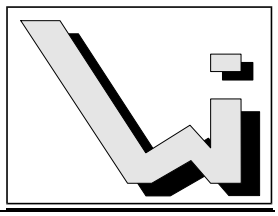
Arbeitsberichte zur Wirtschaftsinformatik



Aufwandschätzung und Produktivität in der Softwareentwicklung

Probleme und Problemlösungsansätze

Michael Bächle
Bernd Jahnke
Achim Kindler

Herausgeber:
Professor Dr. Bernd Jahnke, Universität Tübingen
Abteilung für Betriebswirtschaftslehre, insb. Wirtschaftsinformatik
Melanchthonstr. 30, 72074 Tübingen
Telefon: 07071/29-75423, Telefax: 07071/21229
E-Mail: jahnke@uni-tuebingen.de
WWW: http://www.wiwi.uni-tuebingen.de/lswi/

Band 12 **Tübingen 1995**

Aufwandschätzung und Produktivität in der Softwareentwicklung

Probleme und Problemlösungsansätze

von
Dipl.-Kfm. Michael Bächle,
Prof. Dr. Bernd Jahnke,
Dr. Achim Kindler

Lehrstuhl für Wirtschaftsinformatik,
Universität Tübingen

Zusammenfassung:
Softwareentwicklungsprojekte sind vielfach durch gravierende Termin- und Budgetüberschreitungen gekennzeichnet. Sofern Methoden und Verfahren zur Aufwandschätzung hierfür ausschlaggebend sind, kommen bestehende Meßprobleme des Software Engineering bzw. der Softwaremetrie zum Tragen, die durch verbesserte Interpretation eingeschränkt werden können. Ein solcher Lösungsansatz wird nachfolgend in seinen Grundzügen vorgestellt.

Inhaltsverzeichnis

Seite

Abbildungsverzeichnis

Abkürzungsverzeichnis

CASE	Computer Aided Software Engineering
COCOMO	Constructive Cost Model
DIN	Deutsches Institut für Normung e.V.
DOD	Department of Defense
FP	Function Point
IEEE	Institute of Electrical and Electronics Engineers
LOC	Lines of Code
MRE	Magnitude of Relative Error

1 Motivation

Dem zunehmenden Wettbewerbsdruck in der Softwarebranche können die einzelnen Anbieter nur noch erfolgreich begegnen, wenn entsprechende Maßnahmen zur Verbesserung der unternehmensindividuellen Kostenstruktur durchgeführt werden. Da aus betriebswirtschaftlicher Sicht die Kosten insbesondere bei der Softwareerstellung durch die Produktivität eines Unternehmens entscheidend mitbestimmt werden, ist es erforderlich, die relevanten Einflußgrößen der Produktivität zu identifizieren und geeignet zu quantifizieren. Ausgehend von der Problembeschreibung der Aufwandschätzung ist es Ziel dieses Beitrags, einen gangbaren Lösungsweg für das Projektmanagement vorzustellen. Im Mittelpunkt steht dabei vorrangig die Entwicklung betriebswirtschaftlicher Anwendungssoftware im Kundenauftrag.

2 Probleme der Aufwandschätzung von Softwareentwicklungen

Die Bemühungen zur Entwicklung von Aufwandschätzverfahren führten bisher zu keinen befriedigenden Ergebnissen. Dies liegt nicht zuletzt daran, daß der Aufwandschätzung bislang nur eine sekundäre Bedeutung für den Projekterfolg beigemessen wurde, was letztlich einen verbreiteten Einsatz mehr oder weniger plausibler Daumenregeln und Erfahrungswerte bewirkte[1]. Da diese nicht objektiv und somit kaum nachvollziehbar sind, werden sie nicht weiter untersucht. Modelle der Aufwandschätzung im Sinne dieses Beitrags sind parametrische Modelle. Diese können weiter unterteilt werden in analytisch orientierte und phänomenologisch orientierte Aufwandschätzverfahren.

Kennzeichnend für analytisch orientierte Modelle sind theoretische Annahmen über Prozeß und Produkt, von denen das Schätzergebnis im wesentlichen abhängt. Ein Beispiel für ein analytisch orientiertes Modell sind die Schätzformeln der Software Science, die u. a. auf Annahmen der kognitiven Psychologie beruhen[2]. Phänomenologisch orientierte Modelle werden im wesentlichen aufgrund empirischer Beobachtungen entwickelt und basieren zumeist auf regressionsanalytisch ermittelten Zusammenhängen. Beispiele für Modelle dieser Art sind die Schätzformeln von Walston und Felix[3] sowie das von Boehm entwickelte Verfahren COCOMO[4].

---

1 Zur Kritik der mangelnden Berücksichtigung von Managementaspekten in der Softwareentwicklung siehe insbesondere Elzer,P.: Management, 1989, sowie Thayer, R.H.; Pyster, A.B.; Wood, R.C.: Major Issues, 1981.

2 Vgl. Halstead, M.H.: Software Science, 1977; Coulter, N.S.: Cognitive Psychology, 1983.

3 Vgl. Walston, C.E.; Felix, C.P.: A Method, 1977.

4 Vgl. Boehm, B.W.: Software Engineering Economics, 1981 bzw. die deutsche Übersetzung Boehm, B.W.: Software-Produktion, 1986.

2.1 Einfluß des Entwicklungsumfeldes

Die Brauchbarkeit eines Aufwandschätzverfahrens kann nur dann ermittelt werden, wenn es anhand entsprechender Gütekriterien überprüfbar ist. Eine solche Untersuchung führte 1987 Kemerer[5] durch. Als Gütekriterium wurde dabei der von Conte, Dunsmore und Shen vorgeschlagene Absolutwert des relativen Fehlers ("Magnitude of Relative Error") MRE herangezogen. Der MRE gibt die relative Abweichung des Schätzwertes Y_{est} vom tatsächlichen Wert Y_{act} an, bezogen auf den tatsächlichen Wert[6]

$$MRE = \left| \frac{Y_{act} - Y_{est}}{Y_{act}} \right|$$

Ein abnehmender MRE entspricht einer größeren Schätzgenauigkeit. Werden, wie bei Kemerer, n Projekte mit ihren geschätzten und tatsächlichen Werten untersucht, so ist es sinnvoll, aus den n Schätzungen den Mittelwert MRE für ein Aufwandschätzverfahren zu bilden.[7]

Kennzeichnend für die zugrundeliegende Datenbasis von Kemerer ist, daß sie aus Projekten eines Softwarehauses stammt, welches sich vorrangig mit der Entwicklung kommerzieller Anwendungssoftware beschäftigt. Dementsprechend wurden 12 der 15 Projekte ausschließlich in der Programmiersprache COBOL realisiert.

Die Ergebnisse der vergleichenden Untersuchung von Kemerer sind in Abb. 1 für die 4 untersuchten Aufwandschätzverfahren (SLIM, detaillierte Version von COCOMO, Function Point Methode und ESTIMACS) bzgl. des Gütekriteriums MRE dargestellt[8].

Nicht unerwartet ist das schlechte Ergebnis der Modelle, deren Konstruktionsprozeß keine Datenbasis mit ausschließlich kommerzieller Anwendungssoftware zugrundelag (SLIM, COCOMO). Am deutlichsten wird dies bei SLIM, welches von Putnam für das DoD (Department of Defense der USA) entwickelt wurde. Da das DoD hauptsächlich systemnahe bzw. technisch-wissenschaftliche Softwaresysteme entwickelt, kann das schlechte Schätzergebnis von SLIM auf das völlig andersartige Entwicklungsumfeld zurückgeführt werden. Ähnliches gilt auch für die Endversion von COCOMO, welches das genaueste Modell der drei COCOMO-Versionen darstellt. Bei genauerer Betrachtung der 3 COCOMO-Versionen (Basis-, Zwischen- und Endmodell) fällt auf, daß mit zunehmendem Detaillierungsgrad des Modells die Schätzgenauigkeit, trotz angeblich wachsender Berücksichtigung der relevanten Produktivitätseinflußgrößen, nicht zunimmt[9]. Ursache dafür ist, daß die Endversion von

---

5 Vgl. Kemerer, Ch.: An Empirical Validation, 1987.

6 Vgl. Conte, S.D.; Dunsmore, H.E.; Shen, V.Y.: Metrics and Models, 1986, p. 172.

7 Conte, S.D.; Dunsmore, H.E.; Shen, V.Y.: Metrics and Models, p. 172ff, halten einen Wert von kleiner 0,25 für MRE als ausreichend für die Schätzgenauigkeit eines Aufwandschätzverfahrens.

8 Vgl. Kemerer, Ch.: An Empirical Validation, 1987, pp. 422-425.

9 Vgl. Jahnke, B.; Bächle, M.: Produktivität, 1992, S. 6ff.

COCOMO zwar mehr Informationen bzgl. der berücksichtigten Produktivitätseinflußgrößen enthält, diese Informationen aber im untersuchten Entwicklungsumfeld nicht entscheidungsrelevant sind und deshalb keinen nennenswerten Beitrag zur Verbesserung der Schätzgenauigkeit leisten können.

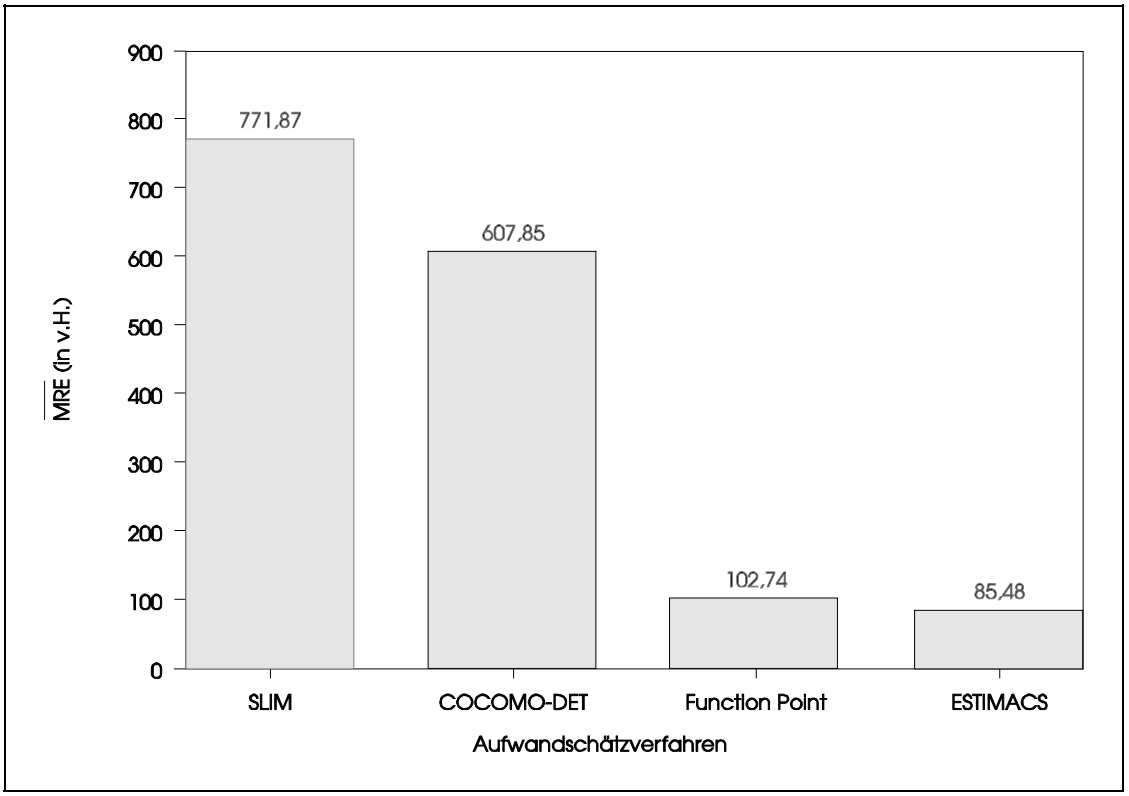


Abb. 1 Vergleich verschiedener Aufwandschätzverfahren anhand des MRE (COCOMO-DET - detaillierte Version von COCOMO)

Als Fazit kann somit festgehalten werden[10]:

- Alle untersuchten Aufwandschätzverfahren wurden in verschiedenen Entwicklungsumfeldern konstruiert. Dies impliziert, daß die einzelnen Modelle nicht unkalibriert in anderen Entwicklungsumfeldern eingesetzt werden dürfen. Eine entsprechende Anpassung der Quantifizierung einzelner Produktivitätseinflußgrößen muß in jedem Fall vorgenommen werden.
- Darüberhinaus gilt:Selbst bei Durchführung einer Kalibrierung der Aufwandschätzverfahren kann nicht vermieden werden, daß einzelne, für das eigene Entwicklungsumfeld relevante Produktivitätseinflußgrößen nicht identifiziert und damit auch nicht berücksichtigt werden.

[10] Vgl. Jahnke, B.; Bächle, M.: Produktivität, 1992, S. 10.

2.2 Diskussion ausgewählter Einflußgrößen

2.2.1 Erfahrung mit der Programmiersprache

Trotz der abnehmenden Bedeutung der Codierphase innerhalb des gesamten Softwareentwicklungsprozesses, z. B. aufgrund des zunehmenden Einsatzes von CASE-Werkzeugen, bleibt die Kenntnis und Erfahrung der eingesetzten Programmiersprachen von mitentscheidender Bedeutung für die Produktivität des Softwareentwicklungsprozesses.

Ein Vergleich der Produktivitätsspannen (siehe Abb. 2) der Produktivitätseinflußgröße „Erfahrung mit der Programmiersprache" für einige veröffentlichte Aufwandschätzverfahren zeigt, daß die Wirkungsrichtung zwar einheitlich, die Stärke des Einflusses aber sehr unterschiedlich beurteilt wird.[11] So kann beispielsweise im Aufwandschätzverfahren COCOMO der Schätzer im Extremfall der Programmiersprachenerfahrung eine Bedeutung beimessen, die um den Faktor 1,20 über der minimal möglichen Produktivitätswirkung dieser Einflußgröße im Aufwandschätzverfahren liegt. Bei Walston/Felix beträgt der vergleichbare Wert für die Produktivitätsspanne 3,16.

Autoren	**Produktivitätsspanne**
Walston, Felix	+ 3,16
Boehm	+ 1,20
Saalfrank, Schelle, Schnopp	+ 1,48

Abb. 2: Produktivitätsspanne der Produktivitätseinflußgröße „Erfahrung mit der Programmiersprache" (Vorzeichen steht für positiven Einfluß)

Aufgrund der verschiedenartigen Entwicklungsumfelder, welche den drei Untersuchungen zugrundelagen, überrascht dieses Ergebnis keineswegs, sondern macht umso deutlicher, wie wichtig eine umfeldspezifische Quantifizierung der Einflußgrößen ist.

2.2.2 Teamgröße

Ebenfalls von Bedeutung ist die organisationsbezogene Produktivitätseinflußgröße „Teamgröße". So weisen verschiedene Studien darauf hin, daß die Produktivität des Softwareentwicklungsprozesses ab einem bestimmten Punkt mit zunehmender Anzahl an Mitarbeitern in einem Entwicklungsprojekt abnimmt.

Brooks führt diesen Zusammenhang zwischen Teamgröße und Produktivität im wesentlichen auf zwei Gründe zurück:[12]

---

[11] Vgl. Walston, C.E.; Felix, C.P.: A Method, 1977, pp. 64; Boehm, B.W.: Software-Produktion, 1986, , S. 569; Saalfrank, R.; Schelle, H.; Schnopp, R.: Produktivitätseffekte, 1987, S. 102.

[12] Vgl. Brooks, F.P.: The Mythical Man-Month, 1974.

- Die zunehmende Teamgröße bedeutet letztlich einen überproportional zunehmenden Kommunikationsbedarf zur Koordination der einzelnen Tätigkeiten. Dadurch wird die mögliche Anzahl produzierbarer Function Points bzw. Lines of Code je Zeiteinheit verringert.
- Neu hinzukommende Projektmitarbeiter müssen sich zunächst mit den Gegebenheiten des Projekts (wie Pflichtenheft) vertraut machen, bevor sie produktiv im Projekt mitarbeiten können.

Für die meisten Aufwandschätzverfahren ist bemerkenswert, daß sie diese Produktivitätseinflußgröße, trotz ihrer überragenden Bedeutung, nicht explizit berücksichtigen. In fast allen Verfahren wird die Teamgröße als Einflußgröße der Produktivität nur implizit über andere, korrelierende Produktivitätseinflußgrößen (wie Umfang des Softwaresystems) erfaßt.

2.2.3 Einsatz wiederverwendbarer Software

Durch die Wiederverwendbarkeit von (Teil-) Projektergebnissen können gravierende Produktivitätsfortschritte erzielt werden[13] Trotz des prinzipiell erzielbaren Vorteils wird Wiederverwendung vielfach auf Programme und Programmteile beschränkt. Im Gegensatz hierzu basiert das Verständnis der Wiederverwendung nach Basili und Rombach auf der Annahme, "**All experience can be reused** ..."[14] und der Tatsache, "Without reuse everything must be relearned and re-created; progress in an economical fashion is unlikely."[15]

Aus dem weit definierten Verständnis der Wiederverwendung nach Basili und Rombach läßt sich u. a. die Forderung nach einer Projektdatenbank ableiten, da herkömmliche Instrumente wie Baustein- oder Programmbibliotheken für eine projektübergreifende Zielsetzung als unzureichend erachtet werden müssen.

2.2.4 Umfang des Softwaresystems

Die Ausbringungsmenge eines Softwaresystems stellt eine weitere Einflußgröße auf die Produktivität dar. In parametrischen Aufwandschätzverfahren wird sie explizit in der Schätzung der Lines of Code oder der Function Points berücksichtigt. Ähnlich der Teamgröße, wird auch für diese Einflußgröße in den meisten empirischen Untersuchungen mit steigendem Umfang eine sinkende Produktivität festgestellt[16].

---

[13] Die durch den Einsatz wiederverwendbarer Software in der Codierungsphase erzielbaren Kosteneinsparungen beziffert Boehm, B.W.: Improving, 1987, p. 54 mit 50 Porzent. Zur analytischen Berechnung der Wiederverwendungskosten vgl. Endres, A.: Software-Wiederverwendung, 1988, S. 86f.

[14] Basili, V.R.; Rombach, H.D.: Reuse, 1991, p. 6.

[15] Basili, V.R.; Rombach, H.D.: Reuse, 1991, p. 2.

[16] Zu einem anderen Ergebnis bei seinen Untersuchungen kommt allerdings Flaherty, M.J.: Productivity Measurement System, 1985, p. 172.

2.3 Kritik der Lines of Code als Produktivitätsmaß

In der Betriebswirtschaftslehre wird Produktivität ganz allgemein als das Verhältnis der Ausbringungsmenge zu den Einsatzmengen der Produktionsfaktoren definiert. Für die praktische Produktivitätsmessung werden mehrere „Teilproduktivitäten" als Kenngrößen ermittelt, um die Produktivität der einzelnen Produktionsfaktoren beurteilen zu können. Bei der Produktivitätsmessung des Softwareentwicklungsprozesses gestaltet sich diese Bildung geeigneter Kenngrößen, nicht zuletzt aufgrund der Immaterialität eines Softwaresystems, als äußerst schwierig.

Eine solche Kenngröße stellt für die Ausbringungsmenge die Anzahl der Lines of Code dar. Dies erscheint zunächst als durchaus plausibel, da durch Lines of Code das Problem der Immaterialität umgegangen werden kann. Die unbedachte Verwendung dieser Kenngröße führt jedoch zu der Paradoxie, daß mit zunehmendem Sprachniveau einer Programmiersprache die in Lines of Code pro Zeiteinheit gemessene Produktivität abnimmt, obgleich der Einsatz höherer Programmiersprachen die Produktivität der Programmierer bzw. des Softwareentwicklungsprozesses erhöht.[17] Zusätzlich zum Sprachniveau wird die Produktivitätsmessung durch weitere Probleme bei der Zählung der Lines of Code erschwert.

[17] Vgl. Jones, C.: Programming Productivity, 1986, p. 7; Jones, T.C.: Measuring, 1978, p. 51ff.

3 Ansätze zur Verbesserung der Aufwandschätzung

Wie unter 2.1 dargestellt, weisen bekannte Methoden und Verfahren zur Aufwandschätzung erhebliche Schwächen auf. Die methoden- und verfahrensspezifischen Defizite sind u. a. mit Softwaremetriken verbunden.[18]. Weitere Probleme bestehen auch in der z. T. implizit vorherrschenden Erwartung, alle Produktivitätseinflußfaktoren mit einem Verfahren abbilden zu können sowie in der eingeschränkten Anpaßbarkeit phänomenologisch orientierter Verfahren.

3.1 Verbesserung der Interpretation von Softwaremetriken

Für die Definition, Operationalisierung, Erfassung und Validierung von Softwaremetriken kann die nachfolgend in Abb. 3 dargestellte Vorgehensweise nach Basili und Rombach angewandt werden. Dabei werden Ziele (G1, G2) durch geeignete Fragen (Q1-Q4) operationalisiert und anhand von Metriken (M1-M6) quantifiziert.

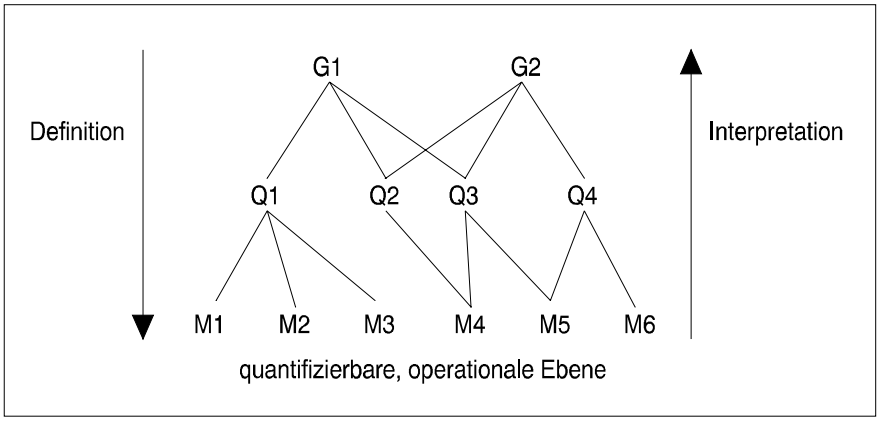


Abb. 3 Goal Question Metric Paradigma[19]

Die bestehenden und kurzfristig nur bedingt lösbaren Meßprobleme sind durch die Interpretation in Abb. 3 erkennbar.[20] Rombach und Ulery betonen diese Interpretation in der Definition einer Softwaremetrik: "A metric defines the data to be collected together with the **context for its interpretation**."[21].Die quantifizierbare, operationale Ebene kann im Sinne der begrenzten Parameteranzahl pro Verfahren zur Aufwandschätzung verstanden werden, wobei sich die Parameteranzahl aus primären Schätzparametern und Einflußgrößen zusammensetzt.[22] Um das bestehende Problem der Schätzgenauigkeit durch Bewertung der Schätzergebnisse einzu-

---

18 Vgl. Baumann, P.; Richter, L.: Software-Metriken, 1992, S. 624ff.

19 Entnommen aus: Rombach, H.D.: Reusability Metrics, 1992, p. 41.

20 Vgl. hierzu Zuse, H.: Software Metrics, 1992, p. 223.

21 Rombach, H.D.; Ulery, B.T.: Software Maintenance, 1989, p. 583 (Hervorhebung von den Verfassern).

22 Vgl. Garmus, D. (Ed.): IPFUG, 1992, p. 4f ; IBM Deutschland GmbH (Hrsg.): Die Function Point Methode, 1985, S. 7ff.

dämmen, besteht eine Möglichkeit darin, der Interpretationsfunktion eine erhöhte Priorität beizumessen. Dies ist u. a. durch Beschreibungsmodelle erreichbar.

Das konzeptionelle Grundprinzip liegt darin, die Messung von Softwaremetriken an den Schätzparametern und Einflußfaktoren formaler Methoden und Verfahren auszurichten und durch eine Vielzahl projektspezifischer Zusatzinformationen zu ergänzen. Ein solches Beschreibungsmodell soll nachfolgend in seinen Grundzügen skizziert werden, wobei die Meßbarkeit infolge der Kombination verschiedener Modellarten nicht bei allen Modellelementen gegeben sein muß.

Die Vorteile von umfassenderen Beschreibungsmodellen bestehen v. a. darin, daß:[23]

- die Interpretationsbasis zur Bewertung von Schätzergebnissen vergrößert wird,
- die Analogiezulässigkeit zugrundegelegter Projekte objektiviert wird,
- quantitative und klassifikatorische Beschreibungs-(teil-)modelle kombiniert werden können,
- die Informationsbasis der isolierten Interpretationskomponente zur Kalibrierung von Methoden und Verfahren (Modell- und Methodenbank) verwendet werden kann.

Die Entwicklung des Beschreibungsmodells kann und muß anhand verschiedener Klassifikationsansätze erfolgen. Vergleichbare Ansätze hierzu liefern z. B.:

- Basili: Resource Data, Change/Defect Data, Process Data, Product Data,[24]
- Noth: Ergebnisse, Ressourcen, Projektumwelt,[25]
- Zuse: Produkte, Prozesse, Ressourcen.[26]

Um Alternativen aufzuzeigen, die in Zusammenhang zu einem mehrjährigen Forschungsprojekt stehen und sich u. a. dadurch begründen lassen, daß ein solches Konzept infolge des nicht unerheblichen Implementierungsaufwands[27] mehrere Funktionen, z. B. im Rahmen der erfahrungsabhängigen Personaleinsatzplanung, erfüllen muß, wird keiner der o. g. Ansätze übernommen. Das in seinen Grundzügen skizzierte Beschreibungsmodell basiert auf der Differenzierung zwischen betriebswirtschaftlicher Anwendungssoftware, Projektorganisation und technischer Umgebung. Schätz- und Projektrisiken, die nicht ausgeschlossen werden können, ergänzen die genannten Beschreibungs-(teil-)modelle.

---

[23] Vgl. Kindler, A.: Wirtschaftlichkeit, 1995, S. 191ff.
[24] Vgl. Basili, V.R.: Experience Factory, 1992, VRB 36.
[25] Vgl. Noth, T.: Erfahrungsdatenbank, 1987, S. 161ff.
[26] Vgl. Zuse, H.: Software Metrics, 1992, p. 37; Fenton, N.E.: Software Metrics, 1991, p. 14.
[27] Vgl. Jones, C.: Software Measurement, 1991, p. 35; Rombach, H.D.; Basili, V.R.: Software-Qualitätssicherung, 1987, S. 155.

3.1.1 Betriebswirtschaftliche Anwendungssoftware

Die zu entwickelnde betriebswirtschaftliche Anwendungssoftware kann auf Basis verschiedener Klassifikationen charakterisiert werden. Ansätze, die weiter zu verfeinern und zu konkretisieren sind, können auf der Unterscheidung nach (1) Betriebsarten/Nutzungsformen, (2) branchenunabhängigem Verwendungszweck, (3) betriebswirtschaftlicher Grundfunktion und (4) branchenspezifischem Einsatzgebiet aufbauen. Wie die weitergehende Differenzierung und Konkretisierung der Ansätze (1) bis (4) erfolgen kann, zeigt das Beispiel eines dialogorientierten Informationssystems für die Führung eines Versicherungsunternehmens[28]

Der Klassifikationsansatz nach (1) ermöglicht die Zuordnung und Identifikation verschiedener Produktmetriken sowie die Analyse und Interpretation von Produktivitätsunterschieden. (2) und (3) konkretisieren die Anwendungssoftware aus betriebswirtschaftlicher Sicht und fördern in Zusammenhang mit (4) die Identifikation von system-, funktions- und kundenerfahrenen Projektmitarbeitern. Aus (4) lassen sich außerdem marktspezifische Analysen, z. B. in Form der Kundenstrukturentwicklung, ableiten.

3.1.2 Projektablauf- und Projektaufbauorganisation

Ohne auf einzelne Projektablauf-, Vorgehens- und Prozeßmodelle näher einzugehen, ist auf ihre zentrale Funktion zur Prozeßstrukturierung, z. B. als Grundlage der Mitarbeitereinsatzplanung und Arbeitszeiterfassung, zu verweisen.[29]

Die Aufbauorganisation eines Projektes stellt vielfach, v. a. aufgrund der vertraglich fixierten Terminrestriktion, einen zum Teil variablen Aktionsparameter dar. Wesentliche Elemente der Projektaufbauorganisation sind die am Projekt beteiligten Projektmitarbeiter. Projektmitarbeiter können anhand ihrer Ausbildung und Projekterfahrung, meßbar in Jahren[30], und anhand von Anforderungs- oder Leistungsprofilen (Systemanalytiker, Programmierer, usw.) unterschieden werden. Um die in Zusammenhang mit dem Projektdefinitionselement der Einmaligkeit zu sehende und aus der unterschiedlichen Projektteamzusammensetzung resultierende Variabilität bzw. Instabilität der Projektaufbauorganisation zu erfassen, empfiehlt sich eine weitergehende Klassifikation derselben. Hierzu eignen sich u. a. strukturelle Organisationsmerkmale[31], wie beispielsweise vertikale Spannen. Bezugnehmend auf die unter 2.2.1 und 2.2.2 ausgesprochene Kritik ermöglicht der skizzierte Ansatz die Interpretation qualifikations- und projektstrukturabhängiger Produktivitätskennzahlen.

---

28 Vgl. Jahnke, B.: Konzeption, 1991.

29 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 12; Kurbel, K.; Pietsch, W.: Projektmanagementebenen, 1989, S. 262.

30 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 20.

31 Vgl. Kieser, A.; Kubicek, H.: Organisation, 1992, S. 67ff, S. 167ff.

3.1.3 Technische Umgebung

Die technische Umgebung umfaßt (1) Hardwareumgebung, (2) Softwareproduktionsumgebung und (3) Basissysteme. Zur ersten Charakterisierung von (1) können Entwicklungs- und Zielrechner[32] herangezogen werden. Kemper zeigt beispielsweise einen weitergehenden, umfassenden Ansatz (Rechnerverbundsystem)[33], auf dessen Darstellung hier verzichtet werden muß. Unabhängig vom gewählten Ansatz - oder Kombinationen derselben - ist die Ergänzung von Hardwarekomponenten um Hersteller- und Produktangaben denkbar. Dies gilt auch für die nachfolgenden Klassifikationen von (2) und (3). Ohne auf die im Zusammenhang mit Computer Aided Software Engineering (CASE) verbundenen Abgrenzungs- und Definitionsprobleme eingehen zu können,[34] läßt sich (2) u. a. durch Methoden (z. B. Strukturierte Analyse) und Werkzeuge (z. B. Programmgeneratoren) charakterisieren.[35]

Der in Anlehnung an Denert[36] gewählte und in engem Zusammenhang zur virtuellen Maschine nach Boehm[37] stehende Ansatz (3) umfaßt die über (1) und (2) hinausgehende technische Umgebung. Diese läßt sich durch die innerhalb des Entwicklungsprojektes eingesetzten Programmiersprachen, Datenbanksysteme, Betriebssysteme, Standardsoftware (-produkte) und Schnittstellen charakterisieren.

3.1.4 Schätz- und Projektrisiken

Da Projekte durch Einmaligkeit gekennzeichnet sind, ist die ökonomische und technische Zielerreichung mit Risiken behaftet[38] Diese sind daher - ergänzend zum betriebswirtschaftlichen Projektergebnis (z. B. Personalkosten) und ergänzend zu den bisher aufgezeigten Partialmodellen - zu berücksichtigen. Hierbei ist eine konsequente Differenzierung zwischen Schätzrisken und Projektrisiken zu empfehlen[39]. Für Schätzrisiken, sowie deren nicht unproblematische Analyse[40], empfiehlt sich aus produktions- und kostentheoretischer Sicht eine Differenzierung zwischen Mengen- und Werteinheiten. Bei den zahlreichen Projektrisiken ist eine möglichst umfassende Klassifikation anzustreben. Diese kann beispielsweise zwischen organisatorischen, personellen, technischen und vertraglichen Risiken unterscheiden.

---

32 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): ANIS/IEEE Std 729-1983, 1987, p. 20, p. 35.

33 Vgl. Kemper, H.-G.: Anwendungsentwicklung, 1991, S. 112. Zu heterogenen Hardware- und Softwareumgebungen vgl. Heisterberg, W.: System-Architekturen, 1992, S. 157ff.

34 Vgl. Chroust, G.: Software-Entwicklungsumgebungen, 1992, S. 170; Ludewig, J.: Software Engineering und CASE, 1991, S. 112ff; Österle, H.: Computer aided software engineering, 1990, S. 350ff.

35 Vgl. Endres, A.; Uhl, J.: Software-Entwicklung, 1992, S. 256f; Hildebrand, K.: Klassifizierung, 1991, S. 16ff; Raasch, J.: Systementwicklung, 1993, S. 82ff; Wirtz, K.W.: Methoden und Werkzeuge, 1990, S. 326ff.

36 Vgl. Denert, E.: Sofware-Engineering, 1992, S. 7ff.

37 Vgl. Boehm, B.W.: Software Engineering Economics, 1981, p. 118.

38 Vgl. Boehm, B.W. (Ed.): Software Risk Management, 1989; Lichtenberg, G.: Risiko-Management, 1992.

39 Vgl. Kindler, A.: Wirtschaftlichkeit, 1995, S. 85ff.

40 Vgl. Knöll, H.-D.; Busse, J.: Aufwandschätzung, 1991, S. 95.

3.2 Standardisierte Messung der Softwareentwicklungsproduktivität

Die Entwicklungsproduktivität von Softwareentwicklungsprojekten ist trotz ihrer zentralen Bedeutung bislang nicht einheitlich definiert.[41] Diese Definitionsvielfalt führt in Verbindung mit weiteren Faktoren[42] zu Meß- und Schätzproblemen. Deshalb sind Standards, wie der IEEE Standard for Software Productivity Metrics (IEEE Std 1045-1992), zu begrüßen.

3.2.1 Übersicht über den IEEE Std 1045-1992

Der 1993 veröffentlichte IEEE Std 1045-1992 kann u.U. dazu beitragen, die genannten Definitions- und Meßprobleme durch einen Rahmen zur Produktivitätsmessung und -protokollierung einzudämmen. Zentrale Inhalte sind:

- Output primitives: Leistungsergebnis
- Input primitives: Faktoreinsatz
- Relationships: Produktivitätsrelationen
- Characteristics: (Projekt-) Merkmale

Beim mengenmäßigen Leistungsergebnis differenziert der Std 1045-1992 zwischen source statements, function points und documents.[43] Bezugnehmend auf die unter 2.3 angesprochenen Probleme der LOC-Zählung ist anzumerken, daß zwischen programmiersprachenspezifischen Logical Source Statements (z. B. Datendeklarationen) und Physical Source Statements (z. B. Kommentarzeilen) unterschieden wird. Die Messung von Function Points wird vom Std 1045-1992 nicht explizit gefordert. Vor dem Hintergrund bestehender Relationen zwischen Source Statements und Function Points[44] wird sie jedoch zum besseren Verständnis produktionstheoretischer Zusammenhänge empfohlen.[45] Dokumente und Bildschirmoberflächen (-masken) werden ganzzahlig auf Basis von Seiten gemessen.[46]

Die Messung des Faktoreinsatzes konzentriert sich auf den primären Kosteneinflußfaktor: Projektmitarbeiter. Dieser Faktoreinsatz, gemessen in Arbeitsstunden, soll getrennt nach direkter und indirekter Projektarbeitszeit erfaßt werden, wobei unter indirekter Projektarbeitszeit Unterstützungstätigkeiten zu verstehen sind. Die hierzu erforderliche Arbeitszeiterfassung muß in jedem Fall auf das unternehmensspezifische Software Development- oder Software

---

41 Vgl. Produktivitätsarten, in: Jones, C.: Programming Productivity, 1986, p. 21; Conte, S.D.; Dunsmore, H.E.; Shen, V.Y.: Metrics and Models, 1986, p. 238; Fenton, N.E.: Software Metrics, 1991, p. 42ff; Shepperd, M.: Evaluation, 1988, p. 177; Wallmüller, E.: Software-Qualitätssicherung, 1990, S. 57ff.

42 Beispiel: "The normal reaction to a measurement program by both project management and staff is apprehension, ...". Jones, C.: Software Measurement, 1991, p. 2.

43 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 3ff.

44 Vgl. Jones, C.: Software Measurement, 1991, p. 76.

45 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 9.

46 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 9ff.

Life Cycle-Modell (siehe 3.1.2) und die hiermit zusammenhängende Work Breakdown Structure abgestimmt werden.[47]

Die zahlreichen, z. B. zwischen verschiedenen Teilprozessen differenzierenden Produktivitätsrelationen weisen folgende allgemeine Form auf:[48]

$$\text{Productivity}_a = \frac{O_a}{E_a}$$

mit

Productivity_a = Produktivität bei der Entwicklung des Produktes a

O_a = Leistungsergebnis von Produkt a

E_a = Entwicklungsaufwand (Effort) für Produkt a (Faktoreinsatz)

und sind durch sämtliche Kombinationen aus Leistungsergebnis und Faktoreinsatz ableitbar.

Ergänzend zu den bisherigen, im Überblick dargestellten Inhalten, umfassen die Projekt-, Management- und Produktmerkmale die Entwicklungsproduktivität beeinflussende Faktoren.[49] Sie dienen somit vergleichbaren Zielen wie die skizzierten Teilmodelle, die - wie unter 3.1 erläutert - als Ergänzung zu methoden- und verfahrensspezifischen Parametern (operationale Ebene) zu verstehen sind. Abgesehen von den Merkmalen, die durch Methoden und Verfahren zur Aufwandschätzung abgebildet sind,[50] können die unter 3.1 in ihren Grundzügen skizzierten Teilmodelle als Konkretisierung einzelner Merkmale verstanden werden (z. B. Produktbeschreibung durch 3.1.1). Andererseits bilden sie Erweiterungen, z. B. durch Hersteller- und Produktangaben (3.1.3) oder Schätz- und Projektrisiken (3.1.4).

3.2.2 Problemorientierte Anwendung des IEEE Std 1045-1992

Beim Leistungsergebnis wird der Dokumentation ein hoher Stellenwert beigemessen. Dieser Stellenwert kommt durch die Berücksichtigung von Seitengröße, Seiteninhalt, Dokumententyp, Dokumentenursprung und Dokumentenverwendung[51] zum Ausdruck.

Der Dokumentation ist, abgesehen vom absoluten Projektaufwandsanteil[52], ein im Projektverlauf unterschiedlicher Stellenwert beizumessen.[53] Die zu Projektbeginn dominierenden, nicht standardisierten Erhebungstechniken (z. B. Interviewtechnik) und Präsentationen (z. B. Ergebnisse der Ist-Analyse) können auf Grundlage der hierzu notwendigen Projektarbeitszeit

---

47 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 12.

48 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 13.

49 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 19ff.

50 Vgl. Boehm, B. W.: Software Engineering Economics, 1981, p. 114ff.

51 Vgl. Institute of Electrical and Electronics Engineers, Inc. (Ed.): IEEE Std 1045-1992, 1993, p. 9ff.

52 Vgl. Boehm, B.W.; Papaccio, P.N.: Software Costs, 1988, p. 1467; Jones, C.: Programming Productivity, 1986, p. 192.

53 Vgl. Jones, C.: Programming Productivity, 1986, p. 187.

und des erzeugten Dokumentationsvolumens im Sinne einer partiellen Produktivitätsrelation quantifiziert werden. Diese, z. B. im Gegensatz zu COCOMO (siehe 2.1) stehende methodische Schwerpunktverlagerung ist geeignet, die Probleme einer frühzeitigen Schätzung einzudämmen.[54]

[54] Vgl. z.B. DeMarco, T.: Software-Projektmanagement, 1989, S. 44, S. 238.

4 Zusammenfassung und Ausblick

Vorteile, die sich für das Management von Softwareprojekten aus den unter 3.1 dargestellten Partialmodellen ergeben, sind[55]:

- Die mehrdimensionale Klassifikation des Zielsystems und der technischen Umgebung fördert die Identifikation anwendungs- und systemerfahrener Projektmitarbeiter.
- Die Differenzierung zwischen Betriebsarten/Nutzungsformen (z. B. Dialogverarbeitung) erlaubt die projektspezifische Zuordnung und Identifikation singulärer Produktmetriken (z. B. Anzahl Masken).
- Die Interpretation singulärer Produktivitätskennzahlen abgeschlossener oder zukünftiger Projekte kann auf Elementen der Projektaufbauorganisation (z. B. Projekterfahrung) und der technischen Umgebung (z. B. Werkzeugeinsatz) aufbauen.

Die beispielhaft genannten Projektmerkmale können als Ähnlichkeitskriterien innerhalb einer Projektähnlichkeitsmatrix verwendet werden und so die Analogiezulässigkeit zugrundegelegter Projekte objektivieren. Dokumentation, wie sie der IEEE Std 1045-1992 beinhaltet, kann dazu verwendet werden, die Probleme des sogen. Softwaresizing zu reduzieren. Beide Ansätze setzen jedoch voraus, daß die Projektdatenerfassung und -beschreibung von allen Beteiligten dem Verständnis der Wiederverwendung nach Basili und Rombach entspricht (siehe 2.2.3).

Die Standardisierung von Softwareentwicklungsprojekten wird - ebenso wie die Standardisierung von Softwaremetriken - zunehmen, z. B. durch die Vereinheitlichung von Verträgen durch die Normenreihe DIN ISO 900x.[56] Zur Verifikation von Softwaremetriken sind jedoch zahlreiche Unternehmen, v. a. aus Gründen der statistischen Grundgesamtheit, nicht in der Lage. Aus diesem Grund bedarf es der übergreifenden Kooperation zwischen Wissenschaft und Praxis.[57]

In diesem Zusammenhang ist auf eine interessante Analogie aus dem Bereich der strategischen Planung hinzuweisen, die sogen. PIMS-(Profit Impact of Market Strategy)Studie. Das auf Borch (General Electric) zurückgehende und zwischenzeitlich organisatorisch verselbständigte PIMS-Programm, an dem mittlerweile etwa 600 Unternehmen teilnehmen, hat das gleiche Ziel wie die oben angesprochene Kooperation: Ausdehnung der Datenbasis.[58]

---

55 Vgl. Kindler, A.: Wirtschaftlichkeit, 1995, S. 191ff.
56 Vgl. Deutsches Institut für Normung (Hrsg.): DIN ISO 9000, Teil 3, 1992, S. 6.
57 Vgl. Baumann, P.; Richter, L.: Software-Metriken, 1992, S. 628.
58 Zum PIMS-Programm vgl. Neubauer, F.: PIMS-Programm, 1990.

Literaturverzeichnis

Basili, V.R. [Experience Factory, 1992]: The Experience Factory: Packaging Software Experience, Vortrag anläßlich des Symposiums "Metriken '92" vom 29.-31. Januar 1992 in München.

Basili, V.R.; *Rombach*, H.D. [Reuse, 1991]: Support for Comprehensive Reuse in: *Rombach*, H.D: Reusability Metrics, Vortrag anläßlich des Symposiums „Metriken ‘92“ vom 29.-31. Januar in München.

Baumann, P.; *Richter*, L. [Software-Metriken, 1992]: Wie groß ist die Aussagekraft heutiger Software-Metriken?, in: Wirtschaftsinformatik, Schwerpunktthema: Kommunalverwaltung. Mit technischen Innovationen zu Verwaltungsinnovationen, Jg. 34 (1992), Heft 6, S. 624-631.

Boehm, B.W. [Software Engineering Economics, 1981]: Software Engineering Economics, Englewood Cliffs (New Jersey) 1981.

Boehm, B.W.: Improving Software Productivity, 1987]: Improving Software Productivity, in: IEEE Computer, Vol. 20 (1987), No. 9, pp. 43-57.

Boehm, B.W. (Ed.): Software Risk Management, Washington et al. 1989.

Boehm, B.W.; *Papaccio*, P.N. [Software Costs, 1988]: Understanding and Controlling Software Costs, in: IEEE Transactions on Software Engineering, Vol. SE-14 (1988), No. 10, p. 1462-1477.

Brooks, F.P. [The Mythical Man-Month, 1974]: The Mythical Man-Month, in: Datamation, Vol. 20 (1974), No. 12, pp. 44-52.

Chroust, G. [Software-Entwicklungsumgebungen, 1992]: Software-Entwicklungsumgebungen - Synthese und Integration, in: Informatik Forsch. Entw., Bd. 7 (1992), Heft 4, S. 165-174.

Conte, S.D.; *Dunsmore*, H.E.; *Shen*, V.Y. [Metrics and Models, 1986]: Software Engineering Metrics and Models, Menlo Park et al. 1986.

Coulter, N.S. [Cognitive Psychology, 1983]: Software Science and Cognitive Psychology, in: IEEE Transactions on Software Engineering, Vol. SE-9 (1983), No. 2, pp. 166-171.

DeMarco, T. [Software-Projektmanagement, 1989]: Software-Projektmanagement: Wie man Kosten, Zeitaufwand und Risiko kalkulierbar plant, München 1989.

Denert, E. [Software-Engineering, 1992]: Software-Engineering: Methodische Projektabwicklung, 1991, korr. Nachdr., Berlin et al. 1992.

Deutsches Institut für Normung e.V. (Hrsg.) [DIN ISO 9000, Teil 3, 1992]: DIN ISO 9000, Teil 3: Qualitätsmanagement- und Qualitätssicherungsnormen, Leitfaden für die Anwendung von ISO 9001 auf die Entwicklung, Lieferung und Wartung von Software (Identisch mit ISO 9000-3: 1991), Berlin 1992.

Elzer, P. [Management, 1989]: Management von Softwareprojekten, in: Informatik-Spektrum, 12 (1989), S. 181-197.

Endres, A. [Software-Wiederverwendung, 1988]: Software-Wiederverwendung: Ziele, Wege und Erfahrungen, in: Informatik-Spektrum, 11 (1988), S. 85-95.

Endres, A.; *Uhl*, J. [Software-Entwicklung, 1992]: Objektorientierte Software-Entwicklung: Eine Herausforderung für die Projektführung, in: Informatik-Spektrum, 15 (1992), S. 255-263.

Fenton, N.E. [Software-Metrics, 1991]: Software-Metrics: A Rigorous Approach, London et al. 1991.

Flaherty, M.J. [Productivity Measurement System, 1985]: Programming Process Productivity Measurement System for System/370, in: IBM Systems Journal, Vol. 24 (1985), No. 2, pp. 168-175.

Garmus, D. (Ed.) [IFPUG, 1992]: IFPUG Function Point Counting Practices Manual, Release 3.4, Software Productivity Research, Inc., Burlington (Massachusetts) 1992.

Halstead, M.H. [Software Science, 1977]: Elements of Software Science, New York 1977.

Heisterberg, W. [System-Architekturen, 1992]: Anwendungs- und System-Architekturen als Leitfaden für das Management heterogener Systeme, in: Bullinger, H.-J. (Hrsg.): Software-Architekturen im Unternehmen: Komponenten, Modelle, Werkzeuge und Methoden, IAO-Forum 13. November 1991, Berlin et al. 1992, S. 151-186.

Hildebrand, K. [Klassifizieung, 1991]: Klassifizierung von Software Tools, in: Wirtschaftsinformatik, Jg. 33 (1991), Heft 1, S. 13-25.

IBM Deutschland GmbH (Hrsg.) [Die Function Point Methode, 1985]: Die Function Point Methode: Eine Schätzmethode für IS-Anwendungsprojekte, IBM-Form GE 12-1618-1, o.O. 1985.

Institute of Electrical and Electronics Engineers, Inc. (Ed.) [ANSI/IEEE Std 729-1983, 1987]: ANSI/IEEE Std 729-1983: IEEE Standard Glossary of Software Engineering Terminology, in: The Institute of Electrical and Electronics Engineers, Inc. (Ed.): Software Engineering Standards, New York 1987.

Institute of Electrical and Electronics Engineers, Inc. (Ed.) [IEEE Std 1045-1992, 1993]: IEEE Std 1045-1992, IEEE Standard for Software Productivity Metrics, New York 1993.

Jahnke, B. [Konzeption, 1991]; Konzeption und Entwicklung eines Führungsinformationssystems, in: Bartmann, D. (Hrsg.): Lösungsansätze der Wirtschaftsinformatik im Lichte der praktischen Bewährung, Berlin, Heidelberg, New York 1991, S. 39-65.

Jahnke, B.; *Bächle*, M. [Produktivität, 1992]: Produktivität im Softwareentwicklungsprozeß. Problematik und Einflußgrößen, Arbeitsberichte zur Wirtschaftsinformatik, Band 7, Tübingen 1992.

Jones, C. [Programming Productivity, 1986]: Programming Productivity, New York et al. 1986.

Jones, C. [Software Measurement, 1991]: Applied Software Measurement: Assuring Productivity and Quality, New York et al. 1991.

Jones, T.C. [Measuring, 1978]: Measuring Programming Quality and Productivity, in: IBM Systems Journal, Vol. 17 (1978), No. 1, pp. 39-63.

Kemerer, Ch. [An Empirical Validation, 1987]: An Empirical Validation of Software Cost Estimation Models, in: Communications of the ACM, Vol. 30 (1987), No. 5, pp. 416-429.

Kemper, H.-G. [Anwendungsentwicklung, 1991]: Dezentrale Anwendungsentwicklung: Entwicklung von betrieblichen Anwendungssystemen durch Fachabteilungen und Endbenutzer in deutschen Großunternehmen, Bergisch Gladbach, Köln 1991, zugl. Dissertation, Universität GH, Essen 1989.

Kieser, A.; *Kubicek*, H. [Organisation, 1992]: Organisation, 3., völlig neub. Aufl., Berlin, New York 1992.

Kindler, A. [Wirtschaftlichkeit, 1995]: Wirtschaftlichkeit von Software-Entwicklungsprojekten. Ansätze zur Verbesserung der Aufwandschätzung, Wiesbaden 1995.

Knöll, H.-D.; *Busse*, J. [Aufwandschätzung, 1991]: Aufwandschätzung von Sofware-Projekten in der Praxis. Methoden, Werkzeugeinsatz, Fallbeispiele, Mannheim, Wien, Zürich 1991.

Kurbel, K.; *Pietsch*, W. [Projektmanagementebenen, 1989]: Projektmanagementebenen bei evolutionärer Softwareentwicklung, in: Kurbel, K.; Mertens, P.; Scheer, A.-W. (Hrsg.): Informations- und Steuerungssysteme, 1989, S. 261-285.

Lichtenberg, G. [Risiko-Management, 1992]: Risiko-Management bei EDV-Projekten: Technische und vertragliche Aspekte, Ehningen bei Böblingen, 1992.

Ludewig, J. [Software Engineering und CASE, 1991]: Software Engineering und CASE - Begriffserklärung und Standortbestimmung, in: Informationstechnik it, Jg. 33 (1991), Heft 3, S. 112-120.

Neubauer, F.F. [PIMS-Programm, 1990]: Das PIMS-Programm und Portfolio-Management, in: Hahn, D.; Taylor, B. (Hrsg.): Strategische Unternehmungsplanung: Strategische Unternehmungsführung, Stand und Entwicklungstendenzen, 5., neub. und erw. Aufl., Heidelberg 1990, S.283-310.

Noth, T. [Erfahrungsdatenbank, 1987]: Unterstützung des Managements von Software-Projekten durch eine Erfahrungsdatenbank, Berlin et al. 1987.

Österle, H. [Computer aided software engineering, 1990]: Computer aided software engineering - Von Programmiersprachen zur Softwareproduktionsumgebung, in: Kurbel, K.; Strunz, H. (Hrsg.): Handbuch Wirtschaftsinformatik, 1990, S. 345-361.

Raasch, J. [Systementwicklung, 1993]: Systementwicklung mit Strukturierten Methoden: Ein Leitfaden für Praxis und Studium, 3., bearb. u. erw. Aufl., München, Wien 1993.

Rombach, H.D. [Reusability Metrics, 1992]: Reusability Metrics, Vortrag anläßlich des Symposiums „Metriken '92“ vom 29.-31. Januar 1992 in München.

Rombach, H.D.; *Basili*, V.R. [Software-Qualitätssicherung, 1987]: Quantitative Software-Qualitätssicherung: Eine Methode zur Definition und Nutzung geeigneter Maße, in: Informatik-Spektrum, 10/1987, S. 145-158.

Rombach, H.D.; *Ulery*, B.T. [Software Maintenance, 1989]: Improving Software Maintenance Through Measurement, in: Proceedings of the IEEE, Vol. 77 (1989), No. 4, pp. 581-595.

Saalfrank, R.; *Schelle*, H.; *Schnopp*, R. [Produktivitätseffekte, 1987]: Produktivitätseffekte von Aufwandseinflußgrößen bei der Softwareentwicklung, in: Angewandte Informatik, 3/1987, S. 95-103.

Shepperd, M. [Evaluation, 1988]: An Evaluation of Software Product Metrics, in: Information and Software Technology, Vol. 30 (1988), No. 3, pp. 177-188.

Thayer, R.; *Pyster*, A.; *Wood*, R. [Major Issues, 1981]: Major Issues in Software Engineering Project Management, in: IEEE Transactions on Software Engineering, Vol. SE-7 (1981), No. 4, pp. 333-342.

Wallmüller, E. [Software-Qualitätssicherung, 1990]: Software-Qualitätssicherung in der Praxis, München, Wien 1990.

Walston, C.E.; *Felix*, C.P. [A Method]: A Method of Programming Measurement and Estimation, in: IBM Systems Journal, Vol. 16 (1977), pp. 54-73.

Wirtz, K.W. [Methoden und Werkzeuge, 1990]: Methoden und Werkzeuge für den Softwareentwurf, in: Kurbel, K.; Strunz, H. (Hrsg.): Handbuch Wirtschaftsinformatik, Stuttgart 1990, S. 323-343.

Zuse, H. [Software Metrics, 1992]: Software Metrics in the Software Life-Cycle (Teil 2), Vortrag anläßlich des Symposiums "Metriken '92" vom 29.-31. Januar 1992 in München.

BISHER ERSCHIENENE ARBEITSBERICHTE

1990

Band 1 *Jahnke*, Bernd: Konzeption und Entwicklung eines Führungsinformationssystems. (Erschienen in: *Bartmann*, Dieter (Hrsg.): Lösungsansätze der Wirtschaftsinformatik im Lichte der praktischen Bewährung, Berlin/Heidelberg/New York 1991, S. 39-65)

Band 2 *Wallau*, Siegfried: Akzeptanz betrieblicher Informationssysteme - eine empirische Untersuchung.

1991

Band 3 *Jahnke*, Bernd: Informationsverarbeitungs-Controlling, Konzepte - Inhalte - Methoden. (Erschienen in: *Huch*, Burkhard/*Behme*, Wolfgang/*Schimmelpfeng*, Katja (Hrsg.): EDV-gestützte Controlling-Praxis: Anwendungen in der Wirtschaft, Frankfurt 1992, S. 119-143,
Vorabveröffentlichung in der FAZ - Blick durch die Wirtschaft, 3. 3. 1992, S. 7)

Band 4 *Fehling*, Georg/*Groffmann*, Hans-Dieter/*Jahnke*, Bernd: Entwicklung der Benutzerschnittstelle eines computergestützten Informationssystems im Rahmen des SAA-CUA Konzepts - Dargestellt am Beispiel eines Führungsinformationssystems für die Württembergische Gebäudebrandversicherung.

1992

Band 5 *Groffmann*, Hans-Dieter: Kennzahlenmodell (KDM) als Grundlage aktiver Führungsinformationssysteme. (Erschienen in: *Rau*, Karl-Heinz/*Stickel*, Eberhard (Hrsg.): Daten- und Funktionsmodellierung. Erfahrungen - Konzepte - Perspektiven, Wiesbaden 1992, S. 1-29)

Band 6 *Jahnke*, Bernd: Einsatzkriterien, kritische Erfolgsfaktoren und Einführungsstrategien für Führungsinformationssysteme. (Erschienen in: *Behme*, Wolfgang/*Schimmelpfeng*, Katja (Hrsg.): Führungsinformationssysteme. Neue Entwicklungstendenzen im EDV-gestützten Berichtswesen, Wiesbaden 1993, S. 29-43)

Band 7 *Jahnke*, Bernd/*Bächle*, Michael: Produktivität im Softwareentwicklungsprozeß, Problematik und Einflußgrößen.

1993

Band 8 *Jahnke*, Bernd: Entscheidungsunterstützung der oberen Führungsebene durch Führungsinformationssysteme. (Erschienen in: *Preßmar*, Dieter B. (Hrsg.): Informati-

onsmanagement, Band 49 der Schriften zur Unternehmensführung, Wiesbaden 1993, S. 123-147)

Band 9 *Jahnke*, Bernd/*Groffmann*, Hans-Dieter: Führungsinformationssysteme zwischen Anspruch und Realisierbarkeit.

1994

Band 10 *Jahnke*, Bernd/*Bächle*, Michael/*Simoneit*, Monika: Methodische Analyse von Vertriebsprozessen zur Zertifizierungsvorbereitung nach ISO 9004.
(In leicht gekürzter Form erschienen in: *Heilmann*, Heidi et al. (Hrsg.): Handbuch der modernen Datenverarbeitung, Heft 175, Januar 1994, S. 50-60.
Eine englische Fassung des Arbeitsberichts mit dem Titel: Modeling Sales Processes as Preparation for ISO 9004 Certification ist erschienen in: International Journal of Quality & Reliability Management, Quality improvements in manufacturing and service industries: recent trends and perspectives, Vol. 12, No. 9 (1995), pp. 76-99)

Band 11 *Jahnke*, Bernd/*Tjiok*, Clifford: Business Process Reengineering and Software Systems Strategy. (Erschienen mit dem Titel: Identifying IS Support Alternatives for Business Process Reengineering in: Knowledge and Process Management, No. 1, Vol. 5, 1998, pp. 41-50)

1995

Band 12 *Bächle*, Michael/*Jahnke*, Bernd/*Kindler*, Achim: Aufwandschätzung und Produktivität in der Softwareentwicklung. Probleme und Problemlösungsansätze.