Irene Maucher

Leitbilder zur Softwareentwicklung. In: Institut Arbeit und Technik: Jahrbuch 1995. Gelsenkirchen, S. 166-179

Einleitung

Die Entwicklung betriebswirtschaftlicher Anwendungssysteme, die zur Be- und Verarbeitung von Informationen und zur Koordination von mehr oder weniger komplexen Abläufen in Produktion und Verwaltung eingesetzt werden, sind eng an die Nutzenvorstellungen der jeweiligen Anwender (betriebliche Entscheider) und Benutzer gebunden. Anforderungen an ein Softwareprodukt leiten sich primär aus den zu bearbeitenden Arbeitsaufgaben ab. Auch bei Informations- und Kommunikationssystemen, die z.B. als Organisationssoftware eine Vielzahl an Verrichtungen und Aufgaben koordinieren, sind die Anforderungen letztlich von der Zweckmäßigkeit der Koordinationsleistungen hinsichtlich der zu bearbeitenden Aufgaben abhängig. Die Orientierung der Softwareentwicklung an diesen Anforderungen ist keine Selbstverständlichkeit. Zunehmend drängendere Probleme der Praxis, wie fehlende Akzeptanz, die als nicht hinreichende Nutzung erscheint, oder gar gänzlich gescheiterte Vernetzungsprojekte und Softwareruinen, verweisen insbesondere auf zweierlei:

1. Eine aufgabenorientierte Softwareentwicklung wurde mit Beginn der kommerziellen Verbreitung nicht durch Methoden, Werkzeuge und Verfahren unterstützt.
2. Im Rahmen der Softwareentwicklung bestanden zunächst keine Vorstellungen über die Funktionszusammenhänge des vorhandenen Informationssystems "Betrieb", die sich in entsprechenden Entwicklungsstrategien und Entwicklungsmethoden niederschlugen.

Aufgrund der Nichtberücksichtigung dieser organisatorischen und sozialen Voraussetzungen bei der Entwicklung und Einführung von Software blieben Produktivitätsfortschritte, die aus einer bedarfs- und anwendungsgerechten Gestaltung und Benutzung von Informations- und Kommunikationssystemen erwachsen, erstmal weitgehend ungenutzt. Zu diesem Ergebnis gelangen empirische Forschungsprojekte, welche die Entwicklung und Einführung von IuK-Systemen untersucht haben (vgl. Gabriel et. al. 1992; Maucher 1995).

Der folgende Beitrag zeigt, daß Erfahrungen mit als unzureichend bewerteten oder fehlgeschlagenen Softwareprojekten den Anstoß für die Entwicklung und Anwendung von Methoden, Verfahren und Werkzeugen zur Softwareentwicklung gaben. Dabei waren im Zeitablauf betrachtet die als dominant wahrgenommen Anwendungsprobleme der Anlaß für die Entwicklung entsprechender Gestaltungsprinzipien. So wurde die zunächst an Hardwareressourcen orientierte Softwareentwicklung durch Software-Engineering-Modelle um betriebswirtschaftliche Kriterien ergänzt. Durch Varianten des Prototypings sollte die Bewertung der Projektfortschritte hinsichtlich ihrer Nützlichkeit für die zu bearbeitenden Aufgaben sichergestellt werden. Erst Konzepte der soziotechnischen Systemgestaltung integrierten schließlich das organisch und sozial gewachsene betriebliche Informationssystem in ein umfassendes Gestaltungskonzept. Entlang dieser kurz skizzierten Zielsetzungen von Softwareentwicklung lassen sich also vier Phasen unterscheiden (vgl. Abb.1), wobei der Entwicklungsprozeß der Softwareerstellung einem reflexiven Prinzip folgt. D.h., die Modifikation des Vorgehens wird insbesondere durch Kritik aus dem Anwendungszusammenhang angestoßen und die entwickelten Lösungsalternativen verfolgen den Zweck vorherrschende Probleme zu lösen.

Die Kunst des Individualprogrammierens

Softwareprogramme wurden in der Anfangsphase in enger Parallelität zur Hardware entwickelt. Dieses Vorgehen wurde einerseits durch die begrenzten Unterstützungsmöglichkeiten

bestimmt, welche die vorhandenen Sprachen, Werkzeuge und Verfahren zur Programmentwicklung boten. Andererseits spiegelte es zugleich eine gedankliche Vorstellung der Systementwickler wider, deren dominantes Leitbild sich an der "Kunst des Individualprogrammierens" orientierte. Softwaresysteme wurden nicht als Objekte der Systemgestaltung begriffen, zu deren Entwicklung es allgemeiner Prinzipien bedarf, die im Zusammenhang mit dem Anwendungsfeld zu konkretisieren sind. Es existierte kein Begriff über ein Informationssystem "als ein nach organischen, technischen und organisatorischen Prinzipien zusammengesetztes Ganzes von Informationsbeziehungen zwischen Informationseinheiten" (Zilahi-Szabo 1993 S. 24) und keine Vorstellung über die Funktionszusammenhänge des soziotechnischen Systems "Betrieb".

Mit der rasanten Entwicklung der Hardware wurde die Schere zwischen dem Leistungsvermögen der Hardware und dem Stand der Softwareentwicklung zunehmend breiter. Durch die unsystematische Programmentwicklung, bei der im Ergebnis häufig schwer nachvollziehbare, ineffektive und fehlerhafte Programme produziert wurden, und durch wachsende Ansprüche an die Komplexität der Programme wurde die Herstellung und Wartung immer kostenintensiver. Die zentrale Kritik der Anwender zielte deshalb vor allem auf die fehlende Kostentransparenz und Qualität der Applikationen ab. Eine Reaktion darauf ist die Entwicklung des Teilgebiets Software Engineering (SW-E) innerhalb der Angewandten Informatik, das gegen Ende der 60er Jahre entstand (vgl. Knittel 1995).

Software-Engineering-Modelle - noch kein Ausweg aus der Softwarekrise

SW-E kann definiert werden als „das Aufstellen und Benutzen fundierter, ingenieurmäßiger Prinzipien, um auf ökonomische Weise Software zu erstellen, die zuverlässig ist und effizient auf realen Maschinen läuft" (Pressmann 1989 S. 12). Die wesentlichen Komponenten des SW-E bilden Methoden, Werkzeuge und Verfahren. Methoden umfassen ausgehend von der Projektplanung, der Entwicklung von Kostenvoranschlägen bis zur Wartung einer Applikation den gesamten „Lebenszyklus" einer Software. Durch Werkzeuge werden die einzelnen Methoden in automatisierter oder halbautomatisierter Form unterstützt. Verfahren verbinden schließlich Werkzeuge und Methoden. Durch sie wird festgelegt, in welcher Reihenfolge die Methoden angewendet werden und durch die Einhaltung welcher Meilensteine der Projekterfolg beurteilt werden kann. Durch die Art und Weise, wie Methoden, Werkzeuge und Verfahren kombiniert werden, lassen sich unterschiedliche Modelle des SW-E generieren. Unabhängig vom jeweiligen SW-E-Modell umfaßt der Prozeß des SW-E generell die drei Phasen Definition, Entwicklung und Wartung.

Die unterschiedlichen Modelle des SW-E weisen als Gemeinsamkeit die Gestaltung von softwaretechnischen Systemen auf. Das dominante Gestaltungsleitbild orientiert sich an dem Vorgehen und den Erfahrungen der Ingenieursdisziplinen bei der Konstruktion von komplexen technischen Systemen. In erster Linie soll der Aufbau eines effizienten Projektmanagements und die Sicherung eines hohen Qualitätsstandards der Produkte gewährleistet werden (vgl. Knittel/Schwolo 1993; Zilahi-Szabo 1993).

Das Grundmodell softwaretechnischer Systemgestaltung wird alsWasserfallmodell bezeichnet und beinhaltet ein systematisches, phasenorientiertes Vorgehen, bei dem idealerweise streng sequentiell mindestens folgende Abschnitte abgearbeitet werden (vgl. Scheibel 1989; Pietsch 1992; Knittel 1995):

Die zu entwickelnde Applikation wird nach diesem Modell stufenweise erstellt, wobei das Ergebnis (Output) einer Phase jeweils die Grundlage (Input) für die nächste Phase bildet. Rekursivität, definiert als die Rückkehr zum Ausgangspunkt einer Problemstellung, die zur Untergliederung eines komplexen Problems in überschaubare Teilprobleme notwendig ist, ist im Rahmen des Modells idealerweise nur innerhalb der einzelnen Phasen vorgesehen. Die strategische Zielsetzung des Top-down Prinzips richtet sich auf ein effizientes

Projektcontrolling. Auf der Basis einer zeitlichen und inhaltlichen Abgrenzung der einzelnen Entwicklungsphasen sollen letztendlich die Beurteilung des "Baufortschritts" des Produktes ermöglicht und gegebenenfalls Korrekturmaßnahmen durchgeführt werden.

Die strenge Linearität des klassischen Wasserfallmodells stößt in der Praxis an Grenzen. Insbesondere bei komplexen Informationssystemen wird der Kunde Schwierigkeiten haben, sämtliche Anforderungen a priori zu benennen. Die Definition von Systemanforderungen sind auch für den Kunden ein rekursiver Prozeß, bei dem es unterschiedliche Problemsichten (z.B. Organisationssicht und Benutzersicht) zu integrieren gilt. Eine vollständige Beschreibung der Systemanforderungen zu einem frühen Zeitpunkt scheint also kaum erreichbar. Ferner sind die (semi)formalen und abstrakten Modellabbildungen, die Phasenmodelle am Ende eines jeden Entwicklungsabschnittes zur Bewertung vorsehen, für eine tatsächliche Bewertung durch betriebliche Entscheidungsträger und Benutzer wenig geeignet (vgl. Budde/Sylla/Züllinghoven 1987). Eine funktionierende Programmversion ist jedoch erst in einer späten Projektphase verfügbar, was dazu führt, daß mögliche gravierende Fehler erst spät erkannt und korrigiert werden können.

Durch die Systematik streng getrennter Phasenabläufe konnten beabsichtigte Kontrollansprüche nur scheinbar befriedigt werden, da sinnlich wahrnehmbare und somit für Nicht-EDV-Experten verständliche Bewertungsmöglichkeiten fehlten. Zudem stellt sich die Sicherung eines hohen Qualitätsstandards der Produkte entlang von Kriterien wie Wartbarkeit, Änderbarkeit, Zuverlässigkeit und Korrektheit, solange sie auf die Bewältigung technischer Probleme begrenzt bleiben, als nicht ausreichend für Qualität im umfassenden Sinne heraus. Denn im Kern handelt es sich bei einem wesentlichen Teil des Qualitätsproblems, der funktionalen Nützlichkeit im Arbeitsbereich, um ein Erkenntnisproblem aufgrund des Vorgehens. Das Qualitätsmerkmal funktionale Nützlichkeit läßt sich nur subjektiv vom eigentlichen Benutzer als Experten in bezug auf seinen Arbeitsvollzug bewerten. Ca. 80% aller anfallenden Kosten im Software-Lebenszyklus sind deshalb auf Versäumnisse in der Definitionsphase zurückzuführen (vgl. Pietsch 1992). Die Anforderungen des Modells an eine ingenieurmäßig exakte Analyse zum Zeitpunkt des Projektbeginns und die späte Konfrontation der eigentlichen Benutzer mit dem Softwaresystem zum Zeitpunkt der Einführung hatte für das Controlling der Softwareprojekte insbesondere folgende Konsequenzen: Einerseits entfallen nur ein geringer Anteil der Gesamtkosten auf die Phasen "Anforderungsdefinition" und "Entwurf", wodurch andererseits jedoch ein großer Teil der Kosten für Wartung und Fehlerbehebung verursacht wird. Diese Probleme stellten sich in der Praxis als neuralgische Punkte der Systemgestaltung heraus und mündeten in der Forderung zur Evaluation der Projektfortschritte.

Iteration und Evaluation durch Prototyping

Prototyping ist eine Variante des SW-E, bei der auf der Grundlage von provisorischen Anforderungsdefinitionen Modelle für zu realisierende Anwendungen erstellt werden. Als frühzeitig lauffähige Lösungen werden entweder einzelne Funktionalitäten der gewünschten Software implementiert oder ein voll funktionsfähiges Programm, das in einem neuen Entwicklungsschritt verbessert wird. Anwender und Benutzer können somit frühzeitig Änderungswünsche äußern und Anforderungen detaillieren. Die Bewertung der Produktqualität wird subjektiv meßbar.

Ob es sich bei Prototyping um eine alternative oder komplementäre Entwicklungsstrategie zum Phasenmodell handelt, ist von der Prototyping Variante abhängig. Exploratives und experimentelles Prototyping sind komplementäre Entwicklungsstrategien zum Phasenmodell. „Exploratives Prototyping ist eine Technik zur Unterstützung der Problemanalyse und der Systemspezifikation" (Knittel 1995 S. 38). Das Ziel ist, mit Hilfe einer frühzeitig lauffähigen Implementation, die auf den groben Vorstellungen der Benutzer und Anwender über die Funktionen des geplanten Systems basiert, zu einem tieferen Problemverständnis zu

gelangen. Experimentelles Prototyping unterstützt die Entwurfsphase konventioneller Entwicklungsmodelle (vgl. Koslowski 1988). Es zielt darauf ab, die Benutzer bereits am Design des Softwaresystems zu beteiligen. Die Entwickler haben die Rolle eines Software-Prototypen realisierenden Beraters, während die Benutzer im iterative trial-and-error mode das System entwerfen (vgl. Blaser/Kleppel 1978). Exploratives und experimentelles Prototyping generieren entweder unvollständige oder als unbrauchbar bewertete Lösungen.

Beim evolutionären Prototyping kann man von einem fließenden Übergang vom Prototypen zur Endversion sprechen. Diese auch "Versioning" (Gabler 1988) genannte Ausprägung des Prototyping beinhaltet eine schrittweise Gestaltung mehrerer Systemversionen. "Bei konsequenter Weiterverfolgung dieses Gedankens verschmilzt hier Prototyping als Verfahren zur Unterstützung der Systemgestaltung mit einem generellen Konzept als Systemgestaltungsstrategie" (Knittel 1995, S. 40).

Für alle der skizzierten Varianten gilt, daß sie Ansprüche an die Evaluation der Projektfortschritte erfüllen. Detaillierte Problemsichten und Bewertungskriterien wie

- Nützlichkeit der Dialogstruktur und Abbildung des Sachproblems im Anwendungskontext sowie
- Nutzbarkeit der Software im Sinne einer ergonomischen Gestaltung der Benutzungsoberflächen

können durch Prototyping in den Entwicklungsprozeß integriert werden. Die Verbindung einer phasenspezifischen Systematik, wie sie im Rahmen des SW-E entwickelt wurde, mit der Chance zur Evaluation, war ein Ansatz zur Lösung von Akzeptabiltätsproblemen, vorausgesetzt die Verwendbarkeit eines Softwareproduktes wird verbessert.

Ein interatives Vorgehen, das die Evaluation der Projektforttschritte durch die eigentlichen Benutzer ermöglicht, kann letztendlich die funktionale Zweckmäßigkeit des Softwarewerkzeuges gewährleisten und somit die Effektivität von Software verbessern. Die zunehmende Integration komplexer technischer Informations- und Kommunikationssysteme führte durch seine Eingriffe in das organisch, sozial und organisatorisch gewachsene Informationssystem eines Unternehmens zu einer neuen Form von nutzungsbezogenen Akzeptanzproblemen, die durch stofflich akzeptabel gestaltete Systeme nicht zu regulieren waren. Bei diesen im Grunde sozial verursachten Konflikten handelt es sich um die Verletzung von Interessensphären verschiedenster Stellenträger und weniger um funktionale, auf die Nützlichkeit und Nutzbarkeit bezogene Probleme. Obgleich Prototyping eine beteiligungs- und betroffenenorientierte Systementwicklung unterstützt, bleibt das erkenntnisleitende Prinzip der Evaluation auf Versuch und Irrtum beschränkt. Eine umfassende methodische Unterstützung der Softwareentwicklung zur Integration des soziotechnischen Systems "Betrieb" mit seinen organisatorischen, nutzungs- personalbezogenen und programmtechnisch gleichermaßen wichtigen Komponenten gewährleistet Prototyping noch nicht. Für die drängenden Nutzungsprobleme aufgrund fehlender Akzeptanz bietet Prototyping noch keine hinreichende Lösung.

Soziotechnische Systemgestaltung als umfassendes Gestaltungskonzept

Gestaltungskonzepte für Arbeitssysteme als soziotechnische Systeme wurden bereits in den 50er Jahren entwickelt, als an eine Verbreitung von Software als Arbeitsmittel noch niemand dachte (vgl. Emery 1959). Für die Softwareentwicklung wurde der soziotechnische Gestaltungsansatz Ende der 70er Jahre nutzbar gemacht. (Vgl. Davis 1979) Seine Grundlagen basieren auf der Vorstellung, daß Softwaresysteme lediglich Werkzeuge des Menschen sind, die ihn bei der Organisation und Ausführung seiner Arbeitstätigkeit unterstützen und zur Verbesserung seiner Arbeitszufriedenheit beitragen sollen. Damit rückt der Mensch mit seinem individuellen Arbeitsstil und seinem Bedarf nach Technikunterstützung in den Mittelpunkt des

Gestaltungsprozesses. Systemgestaltung unter diesen Vorzeichen kann sich nicht mehr auf rein ingenieurmäßige und technische Kriterien zur Modellierung von Softwaresystemen begrenzen, sondern wird um ein Zielsystem zur Wirksamkeit von Informationssystemen ergänzt.

Voraussetzung für die Wirksamkeit von Informations- und Kommunikationssystemen ist, daß im Rahmen eines umfassenden Gestaltungskonzeptes sowohl eine Lösung des Sach- und des Interaktionsproblems (funktionale Nützlichkeit und Nutzbarkeit) als auch des Einsatzproblems fokussiert wird.

- Die Anwendung eines Softwaresystems ist dann nützlich, wenn sie den Benutzer bei der Erledigung seiner Arbeitsaufgabe sinnvoll unterstützt. Voraussetzungen dafür sind, daß die Arbeitsaufgabe entweder in ihren typischen Merkmalen repräsentiert ist oder das Softwarewerkzeug flexibel angepaßt und angewendet werden kann.
- Interaktionsprobleme mit einer computergestützten Applikation resultieren aus Benutzungsoberflächen und Dialogschnittstellen, die den Benutzer unnötig belasten. Die Nutzbarkeit von Softwaresystemen bemißt sich in der Berücksichtigung von Kriterien zu ergonomisch gestalteten Softwaresystemen, die als Qualitätskriterien die stoffliche Seite eines Produktes klassifizieren.
- Einsatzprobleme verursachen Softwaresysteme durch Eingriffe in das bestehende Informationssystem „Betrieb". Organisch gewachsene, organisatorische und technische Kriterien
  von Informationsbeziehungen bilden zugleich die soziale und machtpolitische Struktur eines Unternehmens ab. Eingriffe in dieses fragile Beziehungsgefüge verursachen deshalb häufig Konflikte, die in der Praxis zu Akzeptanzproblemen und im Extremfall zum Scheitern von Gestaltungsprojekten führen. Wie empirische Ergebnisse zeigen, besteht durch Transparenz in der Informationspolitik, fortlaufende Vereinbarungen von Entwicklungszielen im Gestaltungsprozeß mit allen Betroffenen und die Schaffung von Rahmenbedingungen für kontinuierliche Lernprozesse von Entwickler, Anwender und Benutzer die Chance, Konflikte möglichst frühzeitig zu regulieren (vgl. Maucher/Knittel 1992; Gabriel et. al. 1995) Zudem verbessert eine partizipative Systemgestaltung die Qualität des Zielsystems.

Soziotechnische Systemgestaltung mündet in Ergänzung zu SW-E Modellen in folgenden erweiterten Prämissen:

- Die Funktionalität einer computergestützten Applikation hat vor allem Handlungsfreiräume bei der Aufgabenerledigung zu schaffen.
- Softwareprodukte sollten bei minimaler Belastung des Benutzers „optimal" beansprucht werden können.
- Technikgestützte Informationssysteme sind organisch in die soziale Arbeits- und Handlungsorganisation eines Benutzers einzupassen (vgl. Knittel 1995).

Ein differenzierter Methodenrahmen zur Entwicklung soziotechnischer Systeme findet sich z.B. in STEPS (Softwaretechnik für evolutionäre, partizipative Systementwicklung) (vgl. Reisin/Schmidt 1989). Kern dieses umfassenden Konzeptes ist die Organisation und Unterstützung des Kommunikations- und Lernprozesses zwischen allen an der Systementwicklung direkt und indirekt Beteiligten und Betroffenen. Die methodischen Säulen Benutzerbeteiligung, technische Entwicklung und Systemnutzung werden hierzu in einem umfassenden Vorbereitungs-, Entwicklungs- und Einsatzprozeß mehrfach zyklisch durchlaufen (vgl. Eason 1982). Jeder Gestaltungskreislauf generiert eine einsatzfähige Version des Systems, die im Falle notwendiger Anforderungsrevisionen entsprechend verändert wird. Das Versionenmodell ist also anwendungsabhängig zu konkretisieren. Folgende methodische Bausteine unterstützen diesen Prozeß:

- zyklisches Vorgehensschema zur Anpassung an die Veränderungen von Entwicklungs- und Einsatzanforderungen,
- Projekt-Etablierung zur Vorab-Festlegung der produkt- und prozeßbezogenen Aktivitäten und
- Referenzlinien zur situationsspezifischen Vereinbarung von Entwicklungszuständen und entsprechenden Eingriffsmöglichkeiten.

Die Integration des soziotechnischen Systems "Betrieb", mit seinen vielfältigen Problemsichten und Interessen unterschiedlichster Stellenträger, wird also erst durch eine soziotechnisch orientierte Systemgestaltung erreicht. In Ergänzung zur softwaretechnischen Systemgestaltung, bei der ingenieurwissenschaftliche Denkmuster Pate standen, ist eine soziotechnisch orientierte Systemgestaltung zudem durch sozialwissenschaftliche Denkstrukturen beeinflußt. Die produkt- und konstruktionsbezogenen Beschreibungsprozesse des SW-E werden nicht generell verworfen, sondern um sozialwissenschaftliche Ansätze ergänzt. Hierzu wird das phasenorientierte Vorgehensmodell des SW-E anhand von Zyklen aufgelöst, innerhalb deren soziale, nutzungsbezogene und technische Kriterien integriert werden. Das Nutzungsproblem wird durch eine benutzerorientierte Systementwicklung, kooperative Entscheidungsprozesse und Qualifizierungsprozesse für Benutzer und Entwickler reguliert.

Die im Rahmen der softwaretechnischen Systemgestaltung verfolgte Controllingstrategie, bei der Effizienz durch das Verhältnis von Leistungs- und Kostenkontrolle zu erreichen versucht wird, wird im Rahmen der soziotechnischen Systemgestaltung durch Effektivitätskriterien, definiert als tatsächliche Wirksamkeit, ergänzt. Nützlichkeit, Nutzbarkeit und tatsächliche Nutzung sind Kriterien für die Effektivität von IuK-Systemen.

Effektivität eines Informations- und Kommunikationssystems kann durch Effizienzkriterien, die in ihrer Bewertung von der tatsächlichen Wirksamkeit des Informations- und Kommunikationssytems abstrahieren, nicht erreicht werden. Effektiv wird ein Informations- und Kommunikationssystem Betrieb erst dann, wenn die einzelnen Systemelemente "Technik", "Organisation" und "Personal" zu einem organischen System zusammenwachensen. Dieser Prozeß kann durch partizipative Strukturen der Systemgestaltung, iterative und evolutionäre Vorgehensmodelle zur Systementwicklung und Systemeinführung und eine Aufgaben angemessene Entwicklung der technischen Werkzeuge wirksam unterstützt werden.

Literatur

Blaser, A./Kleppel, E.(1978): Developing Business Application Systems. Can the User do it? In: Hansen, H. R. (Hrsg.): Entwicklungstendenzen in der Systemanalyse. 5. Wirtschaftsinformatik-Symposium d. IBM-Deutschland-GmbH, München 1978, S. 435-464

Budde, R./Sylla, K.-H./Züllinghoven, H. (1987): Prototyping. In: Mertens, P. (Hrsg.): Lexikon der Wirtschaftsinformatik. Berlin 1987, S.279-280

Davis, L. E. (1979): The Coming Crisis for Production Management: Technology and Organization. In: Davis, L. E.; Taylor, J. C. (Hrsg.): Design of Jobs. Santa Monica 1979, S. 94-103

Eason, K. D. (1982): The Process of Introducing Information Technology. In: Behaviour and Information Technology 1. 1982, S. 197-213

Emery, F. E. (1959): Characteristics of Sociotechnical Systems. Tavisdock document no. 527. London 1959

Gabriel, R./Krebs, S./Maucher, I. (1992) Technologieentwicklung / Einführungsstrategien.

Abschlußbericht „Weiterbildungsinformationssystem Mikroelektronik", Band I, Duisburg 1992

Gabriel, R./Knittel, F./Krebs, S./Maucher, I. (1995): Einsatz und Bewertung von Informations- und Kommunikationssystemen aus Anwender- und Benutzersicht. In: Wirtschaftsinformatik. Ergebnisse empirischer Forschung. Heft 1, 1995, S. 24-32

Knittel, F. (1995): Technikgestützte Kommunikation und Kooperation im Büro. Entwicklungshindernisse - Einsatzstrategien - Gestaltungskonzepte. Bochum 1995

Knittel, F./Schwolo, U. (1993): Erfolgsprobleme der Systemgestaltung. Eine kritische Betrachtung über zwei Ebenen. Arbeitsbericht 93-10 des Lehrstuhls für Wirtschaftsinformatik an der Ruhr-Universität Bochum. Bochum 1993

Koslowski, K. (1988): Partizipative Systementwicklung und Software Engineering. Opladen 1988

Maucher, I./Knittel, F. (1992): Innovationsbarrieren im Büro. Forschungsbericht Nr. 3, zum Forschungsprojekt "Weiterbildungsinformationssystem Mikroelektronik". Duisburg 1992

Maucher, I. (1995): Softwareentwicklung als refexiver Prozeß - von der Individualprogrammierung zur soziotechnischen Systemgestaltung. Tagungsbeitrag (unveröffentlichtes Manuskript) zum Abschußworkshop: PPS-Systeme für die Fabrik der Zukunft. Gelsenkirchen 1995

Pietsch, W. (1992): Methodik des betrieblichen Software-Projektmanagements. Berlin, New York 1992

Pressmann, R. (1989): Software Engineering. Hamburg 1989

Reisin, F.-M./Schmidt, G. (1989): STEPS - Ein Ansatz zur evolutionären Systementwicklung. In: Jansen, K.-D.; Schwitalla, U.; Wicke, W. (Hrsg.): Beteiligungsorientierte Systementwicklung. Beiträge zu Methoden der Partizipation bei der Entwicklung computergestützter Arbeitssysteme. Opladen 1989, S. 94-105

Ziliahi-Szabo, M.-G. (1993): Wirtschaftsinformatik. Anwendungsorientierte Einführung. München, Wien 1993

aus: Jahrbuch 1995 des Instituts Arbeit und Technik