in: W. Coy, P. Gorny, I. Kopp & C. Skarpelis (1993, Hrsg.) Menschengerechte Software als Wettbewerbsfaktor. German Chapter of the ACM Berichte Nr. 40 (pp. 592-599), Stuttgart: Teubner

Anforderungen an die Prozessgestaltung der Softwareentwicklung.

Matthias Rauterberg

ETH-Zürich

1. Bestandsaufnahme der Software-Entwicklung

Analysiert man aktuelle Softwareentwicklungsprozeße, so erkennt man eine Reihe von Problemen und Schwachstellen. Die Ursachen hierfür sind sowohl in den verwendeten theoretischen Konzepten und den traditionellen Vorgehensweisen (insbesondere Projektmanagement), als auch unzureichenden Kostenrechnungsmodellen begründet. Aufgrund verschiedener Ansätze in der konkreten Praxis der Softwareentwicklung liegt bereits eine zahlreiche Sammlung von Lösungsmöglichkeiten in der Literatur vor, welche auf die Bedeutung der Partizipation aller betroffenen Gruppen hinweist. Bei der Analyse dieser Fallbeispiele lassen sich drei wesentliche Barrieren erkennen: die Spezifikations-Barriere, die Kommunikations-Barriere und die Optimierungs-Barriere (Rauterberg 1991f).

Eines der wichtigsten Probleme besteht ganz allgemein darin, ein gemeinsames Verständnis aller betroffenen Personengruppen über den zu automatisierenden Anteil im betroffenen Arbeitssystem herzustellen (Weltz, Lullies & Ortmann 1991), also gemeinsam verbindliche Antworten auf die Fragen "ob", "wo" und "wie" des geplanten Technikeinsatzes zu finden. Hierzu gehört insbesondere die Bestimmung aller Eigenschaften des neu zu gestaltenden Arbeitssystems. Jedes Arbeitssystem besteht aus einem sozialen und einem technischen Teilsystem (Ulich 1991:151f). es gilt nun beide Teilsysteme gemeinsam in ein optimales Gesamtsystem zu integrieren. Um die gemeinsame Optimierung jedoch durchführen zu können, bedarf es unter anderem valider Kriterien für die Mensch-Maschine-Funktionsteilung (Ziegler 1988, Hoyos 1990, Beck & Janssen 1993).

Für die optimale Gestaltung des gesamten Arbeitssystems kommt es vorrangig darauf an, das soziale Teilsystem als ein mit für dieses Teilsystem spezifischen Eigenschaften und Bedingungen ausgestattetes System zu betrachten, welches es in seiner Verkoppelung mit dem technischen Teilsystem zu optimieren gilt: "human resources are at the core of future growth and Europe´s innovation capability" (CEC 1989:2). Die verschiedenen Arten der Verkoppelungen lassen sich zB. mittels des Kontingenzmodells von Cummings und Blumberg (1987) bestimmen (siehe hierzu Ulich 1989). Dabei kommt der Arbeitsaufgabe als 'Schnittpunkt' zwischen der Organisation und dem Individuum eine zentrale Rolle zu (Volpert 1987:14).

2. Perspektiven der Softwareentwicklung

2.1. Mensch-Maschine-Funktionsteilung

Eine sehr verbreitete Strategie zur Funktionsverteilung zwischen Mensch und Maschine ist die "maximale Automatisierung". Will man jedoch den spezifischen Fähigkeiten des Menschen Rechnung tragen, so wird ein Vergleich zwischen Mensch und Maschine angestellt (Hoyos 1990). Es lassen sich nach Bailey (1989:189) die folgenden fünf Strategien für eine Mensch-Maschine-Funktionsteilung (MMF) unterscheiden:

(1) *comparison allocation:* die MMF wird gemäß den MABA-MABA-Listen[1] (Hoyos 1990) vorgenommen; diese Strategie setzt jedoch voraus, daß sich der Mensch mit der Maschine vergleichen läßt (siehe auch das KABA-Projekt; Zölch & Dunckel 1991).

(2) *leftover allocation:* diese – im technischen Kontext sehr beliebte – Strategie ist auf maximale Automatisierung ausgerichtet und beläßt lediglich die nicht automatisierbaren Funktionen in Menschenhand.

(3) *economic allocation:* bei dieser Strategie wird versucht, eine gemeinsame Optimierung der technischen und menschlichen Ressourcen durch die Realisierung der insgesamt preiswertesten Lösung zu erreichen.

(4) *humanized task approach:* bei dieser Strategie sollen durch die gefundene Lösung primär die menschlichen Fähigkeiten gefördert werden; der Technikeinsatz dient lediglich der Unterstützung und Kompensation menschlicher Schwächen (siehe auch Ulich 1991).

(5) *flexibel allocation:* hierbei kann der Benutzer weitgehend frei entscheiden, wie und mit welchen Mitteln, bzw. Werkzeugen er seine Aufgaben erledigt; durch dies Strategie wird dem differentiell-dynamischen Gestaltungsprinzip nach Ulich (1978) optimal Rechnung getragen.

---

1 "man-are-better-at, machine-are-better-at" (Lanc 1975)

Tabelle 1 Vergleich der Fähigkeiten von Menschen und Maschinen (MABA-MABA - Liste; Hoyos, 1990:14)

Funktionen, die der Mensch besser bewältigt als die Maschine	Funktionen, die eine Maschine besser bewältigen kann als ein Mensch
1. Detektion energetisch sehr schwacher Signale und deren Verstärkung;	1. Lösung einfacher arithmetischer Aufgaben mit großer Geschwindigkeit, Fähigkeit zu sehr schnellen Reaktionen (10^{-6} - 10^{7} s);
2. Flexibilität und Improvisation (schnelles Finden einer Alternativlösung);	2. Differenzierung, d. h. Durchführung der mathematischen Operation d/dt;
3. Wechsel von einer bestimmten Strategie zu einer anderen (Übergang zu einer anderen Lösung);	4. Integrierung, d. h. Durchführung des Integrals einer Funktion;
4. Langfristiges Behalten von großen Informationsmengen (2,8 x 10^{20} bit, nach Neumann) und schnellerer Suchvorgang;	4. Einsatz großer Kraft oder Leistung bei großer Präzision und genau definiertem Ablauf (bei der Maschine ist die Leistung im Praxisbezug unbeschränkt);
5. Räumliche Wahrnehmung (Wahrnehmung von Raumtiefe und Formen);	5. Exakte Wiederholung bestimmter Prozesse nach einem vorgegebenen Programm über einen beliebigen Zeitraum;
6. Interpolation (Bestimmung der Werte zwischen fixen Punkten bzw. Werten);	6. Langfristige Wachsamkeit, keine Ermüdungserscheinungen;
7. Prädiktion und Antizipation (Vorhersage weiterer Entwicklung in logisch schwer definierbaren Bedingungen);	7. Kurzzeitiges Behalten einer Information, kurzzeitige Speicherung;
8. Induktive Urteilsprozeße (Verallgemeinerung) bzw. Bildung einer Ansicht;	8. Durchführung von komplexen simultanen Funktionen mit großer Geschwindigkeit bzw. nach genauer zeitlicher Abfolge;
9. Realisierung homöostatischer Prozeße (Beibehalten einer stabilen Lage bei Änderung der äußeren Bedingungen);	9. Deduktive Urteilsprozesse;
10. Adaptation und Lernen;	10. Einfache Entscheidungen von dem Typ ja-nein mit großer Geschwindigkeit (allerdings mit weniger Möglichkeit, die Ergebnisse zu korrigieren);
11. Durchführung komplexer Entscheidungen; Lösung komplizierter unvollkommen definierter Situationen bzw. unvorhergesehener Situationen;	11. Detektion von Signalen, deren Qualität mit den menschlichen Sinnesorganen nicht wahrgenommen werden kann, mit wesentlich größerer Genauigkeit, als dies der Mensch in seinem Bereich kann.

2.2. Analyse des Arbeitssystems

Um eine Optimierung der Human-Ressourcen in einem Arbeitssystem zu ermöglichen, muß ein Paradigmenwechsel beim Management erfolgen (Klotz 1993). Nicht mehr die primäre Optimierung des Technikeinsatzes ist gefragt, sondern eine verstärkte Hinwendung zu der Organisation der Arbeit ist notwendig. Daraus folgt unmittelbar, daß die Gestaltung von Software – als Mittel der Arbeitsgestaltung – zunächst sich an der Analyse, Bewertung und Gestaltung der Organisationsformen des Arbeitssystems auszurichten hat. Die fehlende Effizienz einer schlecht gestalteten Organisation ist auch durch eine noch so ausgereifte Software nicht auszugleichen; es sei denn, die Softwaregestaltung wird zur impliziten Arbeitsgestaltung (Weltz & Ortmann 1992: 112ff.)! Die Organisationsentwicklung und der dabei erreichbare Reifegrad des Unternehmens ist von ausschlaggebender Bedeutung für die Optimierung eines Arbeitssystems (Humphrey 1989).

2.3. Ablaufmodell für den Softwareentwicklungsprozeß

Ein wesentliches Problem der adäquaten Verzahnung der verschiedenen Optimierungs-Zyklen im Quadranten-Modell iterativer Softwareentwicklung besteht in der *Synchronisation* dieser Zyklen. Sind an verschiedenen Stellen im partizipativen Softwareentwicklungs-Konzept (Rauterberg 1991; Hesse, Merbeth & Frölich 1992; Rauterberg, Mollenhauer & Spinas 1993) gleichzeitig mehrere Optimierungs-Zyklen aktiv, so müssen diese adäquat synchronisiert werden. Dieser Aspekt ist deshalb besonders wichtig, weil nur so das Ausmaß an Inkonsistenzen innerhalb des gesamten Entwicklungsprozesses minimiert werden kann (–> "simultaneous engineering"). Wenn z.B. parallel zur Implementationsphase weitere Rücksprachen und Anforderungsanalysen mit dem Anwender stattfinden, passiert es leicht, daß die Entwickler gemäß der stets veralteten Spezifikationen oftmals für den "Papierkorb" programmieren. Die Ursache hierfür ist bei fehlender oder mangelnder Synchronisation dieser beiden Optimierungs-Zyklen in ihrer unterschiedlichen Zyklus-Dauer zu sehen (Rauterberg 1991). Einen wesentlichen Einfluß auf die Qualität des Softwareproduktes hat die Art und Weise der Kommunikation zwischen Entwickler und Anwender, sowie der Entwickler untereinander (Kraut & Streeter 1990; Brodbeck & Sonnentag 1993).

2.4. Aufbaumodell für den Softwareentwicklungsprozeß

Software-Projekte sind durch komplexe, innovative und zeitkritische Problemstellungen gekennzeichnet. Das Projektmangement dient dazu, diese schwierige Situation zu bewältigen und sollte daher als ganzheitliches System verstanden werden. Eine ganzheitliche Organisationsform umfaßt die für ein Projekt notwendigen organisatorischen, planenden, überwachenden, steuernden, methodischen und personalbezogenen Aktivitäten. Als Erfolgskriterien sind ergebnis-bezogen die Leistung und Qualität des Systems, der Kosten- und Zeitaufwand für dessen Ent-wicklung sowie prozeßbezogen die Zufriedenheit aller Projektbeteiligten und -betroffenen mit dem Projektverlauf und dessen Ergebnis von Relevanz.(Rauterberg 1991; Weltz & Ortmann 1992; Spinas et al. 1993). Man kann feststellen (Weltz & Ortmann 1992), daß traditionelle Organisationsmodelle bei der Softwareentwicklung mit Problemen verbunden sind, welche Zweifel daran aufkommen lassen (Ulich 1991), ob in solchen Projektstrukturen in effizienter Weise benutzer- und aufgabenangemessene Software entstehen kann (Spinas et al. 1993).

2.5. Qualitätssicherung im Software-Produktionsprozeß

Zuverlässige Methoden, Verfahren und Werkzeuge der Qualitätssicherung im Software-Produktionsprozeß sind eine wesentliche Voraussetzung für den Einsatz informationstechnischer Systeme in Industrie und Dienstleistung (Wallmüller 1990). Qualitätssicherungsmaßnahmen müssen in den Software-Produktionsprozeß eingebettet sein. Sie können i.a. nicht nach einem starren Raster erfolgen, sondern sollten an den Bedarf eines Projektes angepaßt werden. Die Planung des Qualitätssicherungsprozesses kann selbst wieder als Teil des Software-Produktionsprozesses vorgesehen werden. Qualitätssicherungsmaßnahmen können entweder selbst Bestandteil des Softwareentwicklungsprozesses sein oder es kann sich um selbständige Aktivitäten handeln. Neben allgemeinen Maßnahmen zur Qualitätssicherung müssen für einzelne Klassen von Softwaresystemen detaillierte Definitionen des Qualitätsbegriffs und Heuristiken zur Planung und Anwendung von Techniken entwickelt werden. In wie weit eine Normung (z.B. ISO 9000) hilfreich ist, läßt sich vorerst nur schwer abschätzen.

3. Bezug zum Programm "Arbeit und Technik"

Eines der Hauptprobleme traditioneller Softwareentwicklung liegt darin, daß alle bisher primär an Softwareentwicklungen beteiligten Personengruppen nicht wahrhaben wollen, daß Softwareentwicklung in den meisten Fällen primär Arbeits- und/oder Organisationsgestaltung ist. Um mit dieser Perspektive an Softwareentwicklung heranzugehen, hieße, von vornherein Experten für Arbeits- und Organisationsgestaltungsmaßnahmen mit in den Prozeß der Softwareentwicklung einzuplanen. Dies erfordert jedoch notwendiger Weise eine interdisziplinäre Zusammenarbeit zwischen Arbeits- und Organisations-Experten einerseits und Softwareentwicklungs-Experten andererseits (Waeber 1991). Wegen der umfangreichen Qualifikation in dem jeweiligen Fachgebiet ist es nur sehr begrenzt möglich, auf eine interdisziplinäre Zusammenarbeit zu verzichten.

In einer Reihe von Projekten (z.B. MBQ, BOSS, PROTOS) im Rahmen des Förderprogrammes des BMFT "Arbeit und Technik" wurden verschiedene Ansätze für den Einsatz von Methoden zur Benutzerbeteiligung und iterativ-zyklischem Vorgehen entwickelt und im Einzelfall erprobt. "In den meisten dieser Projekte ist es zwar gelungen, Möglichkeiten und die Produktivität von Nutzerbeteiligung aufzuweisen, fraglich erscheint allerdings, wie stabil sich solche Beteiligungsexperimente außerhalb des schützenden Rahmens subventionierter Projekte erweisen und welche Breitenwirkung sie auf die normale Praxis haben" (Weltz & Ortmann 1992:72).

Zur Zeit ist das Förderprogramm des BMFT "Arbeit und Technik" das einzige Förderprogramm auf nationaler Ebene im Gegensatz zu den EG-Förderprogrammen, in dem auch Projekte mit interdisziplinärem Charakter möglich sind, welche auf Anwenderbedürfnisse eingehen, einen ganzheitlichen Gestaltungsansatz verfolgen und damit auf eine Optimierung des ganzen Arbeitssystems ausgerichtet sind (siehe hierzu auch Bullinger 1993).

4. Forschungs- und Entwicklungsbedarf

Es werden Methoden zum "Management der sozialen Prozesse im Arbeitssystem" benötigt, welche zur Zeit von Unternehmensberatern mit einer leider ausschließlich organisatorischen Perspektive eingesetzt werden. Es fehlen Methoden, welche die Analyse, Bewertung und Gestaltung von Unternehmen ermöglichen, bei denen von Anfang an systematisch der Einsatz von Technik mitgedacht und geplant wird. Hierzu ist es notwendig, daß speziell qualifizierte Arbeits- und Organisationsgestalter mit fundierten Kenntnissen über formale Spezifikationsmethoden eingesetzt werden.

Es werden Methoden zum "Management der sozialen Prozesse im Softwareentwicklungsprozeß" benötigt, welche primär auf die Gestaltung der sozialen und kommunikativen Aspekte ausgerichtet sind. Diese Methoden können über entsprechend qualifizierte Projektmanager zum Einsatz gelangen und weiterentwickelt werden.

Es fehlen grundlegende Methodenvergleichsstudien. Die bisher bekannten Studien sind in ihrer Aussagekraft oftmals viel zu begrenzt, als daß sich daraus zwingende Schlußfolgerungen ableiten ließen. Die folgende Liste stellt eine Positiv-Auswahl der wenigen bekannten Methodenvergleichsstudien dar: Boehm, Gray & Seewaldt 1981; Müller-Holz auf der Heide et al. 1991; Kirsch 1991; Jeffries et al. 1991; Jeffries & Desurvire 1992.

Es bedarf verstärkt der "Hilfe zur Selbsthilfe". Projekte, bei denen die Aufgabe der Begleitforscher auf die Rolle des Beraters begrenzt ist, sind unterrepräsentiert. Für diesen Projekttyp sind Experten für Organisations- und Technikgestaltung unabdingbar.

Literaturangaben

Bailey, R.W. (1989). Human Performance Engineering. New Jersey: Prentice.

Beck, A. & Ilg, R. (1991) Aufgabenorientierte Analyse und Gestaltung mit TASK. In: Frese, M., Kasten, C., Skarpelis, C. & Zang-Scheucher, B. (eds.) Software für die Arbeit von morgen (S. 95-106). Berlin Heidelberg New York: Springer.

Boehm, B W., Gray, T. & Seewaldt, T (1981) Prototyping versus specifying: a multiproject experiment. IEEE Transactions on Software Engineering, SE-10(3):224-236

Brodbeck, F. & Sonnentag, S. (1993) Arbeitsanforderungen und soziale Prozesse in der Software-Entwicklung. (in diesem Band).

Bullinger, H. (1993) Benutzergerechte Gestaltung von Software - eine Herausforderung an den Industriestandort Bundesrepublik Deutschland. (in diesem Band).

C.E.C. Commission of the European Communities (1989) Science, Technology and Societies: European Priorities. Results and Recommendations of the FAST II Programme. Summary Report. Directorate-General Science, Research and Development, Brussels.

Cummings, T., Blumberg, M. (1987) Advanced manufacturing technology and work design. in: Wall, T., Clegg, C. & Kemp, N. (eds.) The Human Side of Advanced Manufacturing Technology (pp. 37-60). Chichester: Wiley.

Dunn, R. & Ullmann, R. (1982) Quality assurance for computer software. New York: McGraw-Hill.

Greif, S. & Hamborg, K. (1991) Aufgabenorientierte Softwaregestaltung und Funktionalität. In: Frese, M., Kasten, C., Skarpelis, C. & Zang-Scheucher, B. (eds.) Software für die Arbeit von morgen (S. 107-119). Berlin Heidelberg New York: Springer.

Hesse, W., Merbeth, G. & Frölich, R. (1992) Software-Entwicklung. München Wien: Oldenbourg.

Hoyos, C. (1990). Menschliches Handeln in technischen Systemen. In: C. Hoyos & B. Zimolong (Hrsg.) Enzyklopädie der Psychologie, Band D III 2, Ingenieurpsychologie (S. 1-30). Göttingen: Hogrefe.

Hoyos, C., Holz auf der Heide, B. & Ortlieb, S. (1993) Eine iterative Software-Entwicklungsstrategie mit gezielter Benutzerbeteiliogung und systematischer Evaluation der Benutzerfreundlichkeit. (in diesem Band).

Humphrey, W. (1989) Managing the software process. Amsterdam: Addison Wesley.

Jeffries, R., Miller, J., Wharton, C. & Uyeda, K. (1991) User interface evaluation in the real world: a comparison of four techniques. In: S. Robertson, G. Olson & J. Olson (eds.) Human Factors in Computing Systems: Reaching through technology CHI'91 (pp.119-124). New York: ACM.

Jeffries, R. & Desurvire, H. (1992) Usability testing vs. heuristic evaluation: was there a contest? SIGCHI Bulletin, 24(4):39-41.

Kirsch, C. (1991) Benutzerbeteiligung bei der Datenmodellierung. Softwaretechnik-Trends. 11(3):93-103.

Klotz, U. (1993) Software als Wettbewerbsfaktor - Perspektiven von Technologiepolitik und Informatikindustrie vor dem Hintergrund der aktuellen Standort-Diskussion. (in diesem Band).

Kraut R. & Streeter L. (1990) Satisfying the need to know: interpersonal information access. In: D. Diaper et al. (eds.) Human-Computer Interaction – INTERACT'90 (pp. 909-915). Amsterdam: Elsevier.

Lanc, O. (1975) Ergonomie. (Urban Taschenbücher, Band 197). Stuttgart: Kohlhammer.

Lullies, V., Weltz, F. & Bollinger, H. (1990) Konfliktfeld Informationstechnik. Frankfurt: Campus.

Martin, C F. (1988) User-Centered Requirements Analysis. Englewood Cliffs: Prentice Hall.

Müller-Holz auf der Heide, B., Aschersleben, G., Hacker, S. & Bartsch, T. (1991) Methoden zur empirischen Bewertung der Benutzerfreundlichkeit von Bürosoftware im Rahmen von Prototyping. In: Frese, M., Kasten, C., Skarpelis, C. & Zang-Scheucher, B. (eds.) Software für die Arbeit von morgen (S. 409-420). Berlin Heidelberg New York: Springer.

Rauterberg, M. (1991) Partizipative Konzepte, Methoden und Techniken zur Optimierung der Softwareentwicklung. In: P. Brödner, G. Simonis & H. Paul (Hrsg.), Arbeitsgestaltung und partizipative Systementwicklung (S. 95-125). Opladen: Leske & Budrich.

Rauterberg, M. (1992a) An iterative-cyclic software process model. In: Proceedings of 4th. International Conference on Software Engineering and Knowledge Engineering held in Capri (Italy), June 17-19, 1992; Los Alamitos: IEEE Computer Society Press; pp. 600-607.

Rauterberg, M. (1992b) Optimisation Cycle: a Concept for Optimal Software Development. In: R. Trappl (ed.), Cybernetics and System Research, vol.1 (pp.279-286). Singapore London: World Scientific.

Rauterberg, M. (1992c) Partizipative Modellbildung zur Optimierung der Softwareentwicklung. In: R. Studer (Hrsg.), Informationssysteme und Künstliche Intelligenz: Modellierung (S. 113-128). Berlin : Springer.

Rauterberg, M., Mollenhauer, R. & Spinas, P. (1993) Phasenmodell ist OUT: Benutzerbeteiligung jetzt auch bei Standardsoftware-Entwicklung. (in diesem Band).

Rauterberg, M. & Strohm, O. (1992) Work Organization and Software Development. In: P. Elzer & V. Haase (eds.), Proceedings of 4th IFAC/IFIP Workshop on "Experience with the Management of Software Projects". Annual Review of Automatic Programming. 16 (2):121-128.

Spinas, P. & Waeber, D. (1991) Benutzerbeteiligung aus der Sicht von Endbenutzern, Softwareentwicklern und Führungskräften. In: D. Ackermann & E. Ulich (Hrsg.), Software-Ergonomie '91. Benutzerorientierte Software-Entwicklung (S. 36-45). Stuttgart: Teubner.

Spinas, P., Rauterberg, M., Strohm, O., Waeber, D. & Ulich, E. (1993, in Druck) Benutzerorientierte Software-Entwicklung. Konzepte, Methoden und Vorgehen zur Benutzerbeteiligung. Schriftenreihe Mensch, Technik, Organisation (Hrsg. E. Ulich), Band 3. Zürich: Verlag der Fachvereine.

Strohm, O. (1991b) Projektmanagement bei der Softwareentwicklung. In: D. Ackermann & E. Ulich (Hrsg.), Software-Ergonomie '91. Benutzerorientierte Software-Entwicklung (S. 46-58). Stuttgart: Teubner.

Ulich, E. (1978) Über das Prinzip der differentiellen Arbeitsgestaltung. Industrielle Organisation, 47, 566-568.

Ulich, E. (1989a) Arbeitspsychologische Konzepte der Aufgabengestaltung. In: S. Maass & H. Oberquelle (Hrsg.), Software-Ergonomie '89: Aufgabenorientierte Systemgestaltung und Funktionalität (S. 51-65). Stuttgart: Teubner.

Ulich, E. (1991) Arbeitspsychologie. Stuttgart: Poeschel-Verlag.

Waeber, D. (1991) Aufgabenanalyse und Anforderungsermittlung in der Softwareentwicklung: Zur Konzeption einer integrierten Entwurfsstratgie. In: M. Frese, C. Karsten, C. Skarpelis & B. Zang-Scheucher (Hrsg.), Software für die Arbeit von morgen. Bilanz und Perspektiven anwendungsorientierter Forschung. Ergänzung zum Tagungsband (S. 35-45). Krefeld: Vennekel & Partner.

Wallmüller, E. (1990) Software-Qualitätssicherung in der Praxis. München: Oldenbourg.

Weisbecker, A. (1993) Unterstützungswerkzeuge zur benutzergerechten Gestaltung der Mensch-Computer-Schnittstelle. (in diesem Band).

Weltz, F., Lullies, V. & Ortmann, R G. (1991) Softwareentwicklung als Prozeß der Arbeitsstrukturierung. In: Ackermann, D. & Ulich, E (Hrsg.) Software-Ergonomie '91 (S. 70-75). (Berichte des German Chapter of the ACM, Vol. 33). Stuttgart: Teubner.

Weltz, F. & Ortmann, R. (1992) Das Softwareprojekt – Projektmanagement in der Praxis. Frankfurt: Campus.

Zölch, M. & Dunckel, H (1991) Erste Ergebnisse des Einsatzes der “Kontrastiven Aufgabenanalyse”. In: D. Ackermann & E. Ulich (Hrsg.) Software-Ergonomie '91 (S. 363-372). (Berichte des German Chapter of the ACM, Vol. 33). Stuttgart: Teubner.