Software-ergonomische Beratung in der Praxis - ein Beitrag zur Organisationsentwicklung

Marc Hassenzahl, Jochen Prümper, Edmund Buchbinder

0 Zusammenfassung

Mit der internationalen Normenreihe ISO 9241 „Ergonomische Anforderungen für Bürotätigkeit mit Bildschirmgeräten“ und der gesetzlichen Fixierung des Anspruchs auf Gebrauchstauglichkeit eines Softwaresystems im Rahmen des Arbeitsschutzes ist das allgemeine Interesse an Software-Ergonomie deutlich gestiegen.

Bei der näheren Beschäftigung mit den entsprechenden Normen und Gesetzen ist die Verwirrung des Interessierten jedoch oft groß. Software-ergonomische Normen sind wenig konkret und geben kaum Handlungsanweisungen, dafür aber um so mehr unspezifische Empfehlungen. Für eine Organisation (bzw. den Verantwortlichen) stellt sich nun die Frage, wie software-ergonomische Erkenntnisse bei der Gestaltung oder Einführung eines Softwaresystems angemessen berücksichtigt werden können? Eine Antwort darauf ist externe *software-ergonomische Beratung*.

Die vorliegende Arbeit möchte zunächst einen Beitrag zur Klärung von Begriffen und Anforderungen software-ergonomischer Qualitätssicherung leisten. Danach sollen Formen software-ergonomischer Beratung in Abhängigkeit von der „Evolutionsstufe“ der auftraggebenden Organisation beschrieben werden. Dem Leser steht es frei, die „Evolutionsstufe“ *seiner* Organisation selbst einzuschätzen und damit einen Hinweis auf Chancen und Probleme bei der Beschäftigung mit Software-Ergonomie zu bekommen. Abschließend wird unser Verständnis externer software-ergonomischer Beratung als ein Beitrag zur Organisationsentwicklung erläutert.

1 Einleitung

Mit dem Inkrafttreten der Bildschirmarbeitsverordnung (BildscharbV) im Rahmen des Arbeitsschutzgesetzes (ArbSchG) Ende 1996 wurde der Anspruch von Arbeitnehmer auf „benutzerfreundliche Software“ gesetzlich fixiert (siehe 24 für einen Überblick). Doch so mancher Verantwortliche fragt sich noch, was „benutzerfreundliche Software“ überhaupt ist, welche Rolle die Software-Ergonomie dabei spielt und wie man „benutzerfreundliche Software“ herstellen kann. Folglich wird trotz der gesetzlichen Verankerung der Aspekt „Benutzerfreundlichkeit“ in den meisten Software-Entwicklungsprojekten oft grob vernachlässigt.

Ein weiterer Grund für die fehlende Berücksichtigung software-ergonomischen Wissens ist nicht zuletzt die unzureichende universitäre und betriebliche Ausbildung von Software-Entwicklern im Bereich der Software-Ergonomie (vgl. 2). Die für den Software-Entwicklungsprozeß so wichtigen psychologischen, arbeitswissenschaftlichen und juristischen Kompetenzen werden meistens zugunsten einer einseitig technisch-orientierten Sichtweise vernachlässigt. Dabei zeigen Studien, daß gerade diese Kernkompetenzen wichtige Prädiktoren für den Erfolg von Software-Entwicklungsprojekten sind (z.B. 8, 27).

Literaturangabe: Hassenzahl, M., Prümper, J. & Buchbinder, E. (1998). Software-ergonomische Beratung in der Praxis – ein Beitrag zur Organisationsentwicklung. In: A. Clermont und W. Schmeisser (Hrsg.), *Betriebliche Personal- und Sozialpolitik* (S. 551-566). München: Vahlen.

Eine Möglichkeit, die unberücksichtigten Kompetenzen in ein Software-Entwicklungsprojekt einzubringen, ist das Anfordern externer Software-Ergonomie-Beratung. Die Inhalte, Chancen und Probleme software-ergonomischer Beratung sind das Thema des vorliegenden Beitrags.

Zunächst sollen dazu die Begriffe Software-Ergonomie und *Gebrauchstauglichkeit* (Benutzerfreundlichkeit) gemäß ISO 9241 geklärt und ihre gesetzliche Verankerung im Rahmen der *Bildschirmarbeitsverordnung* skizziert werden. Darauf aufbauend sollen Grundelemente software-ergonomischen Gestaltens vorgestellt werden. Ein Ansatz, verschiedene Formen der software-ergonomischen Beratung zu beschreiben, geht von der Evolutionsstufe der auftraggebenden Organisation im Hinblick auf Software-Ergonomie aus. In Abhängigkeit von der Evolutionsstufe sind unterschiedliche Beratungsformen wahrscheinlich. Weiterhin diskutieren wir unser Verständnis der Rolle eines Software-Ergonomie-Beraters in einer Organisation.

2 Software-ergonomische Qualitätssicherung

Software-Ergonomie ist die wissenschaftliche Disziplin, die sich mit der *Gebrauchstauglichkeit* von Software beschäftigt. Praktisch tätige Software-Ergonomen, d.h. Berater, versuchen, die ergonomische Qualität einer Software durch den Einsatz ihres Fachwissens und verschiedenster Methoden zu sichern. Sie betreiben also *software-ergonomische Qualitätssicherung*. Der folgende Abschnitt versucht zu klären, was Gebrauchstauglichkeit überhaupt ist, welche Vorteile man sich davon verspricht und welche gesetzliche Verankerung Gebrauchstauglichkeit hat. Außerdem soll kurz auf grundlegende Techniken der software-ergonomischen Qualitätssicherung eingegangen werden.

2.1 Die Gebrauchstauglichkeit von Software als Qualitätsmerkmal

Die Begriffe „Benutzerfreundlichkeit“, „Benutzungsfreundlichkeit“ oder „Nutzungsfreundlichkeit“ werden synonym zu Gebrauchstauglichkeit verwendet. Obwohl Benutzungsfreundlichkeit oder Benutzerfreundlichkeit die bekannteren Begriffe sind, werden sie in Expertenkreisen nicht gerne verwendet. Norman und Draper (21) bezeichnen beispielsweise den Begriff „user friendly“ (benutzerfreundlich) als abwertend gegenüber dem Benutzer, weil er Inkompetenz andeutet. Benutzer benötigen in der Regel keine besonders freundlichen oder gar „idiotensichere Systeme“ (idiot proof systems), sondern brauchbare Software-Werkzeuge. Außerdem impliziert der Begriff „Freundlichkeit“ eher einen speziellen „Service“, eine Art Zusatzleistung oder Bonus also, und nicht eine unabdingbares Muß für ein Softwaresystem. Aus diesen Gründen wurde im amerikanischen Sprachraum „user friendliness“ durch „usability“ ersetzt, was im Deutschen mit „Gebrauchstauglichkeit“ übersetzt werden kann. Im weiteren soll deshalb ausschließlich der Begriff „Gebrauchstauglichkeit“ verwendet werden.

Marketingstrategen großer (und auch kleinerer) Softwarehäuser greifen in letzter Zeit die oben aufgeführten Begriffe wieder verstärkt auf, um sie ihrem jeweiligen Produkt als Qualitätsmerkmal zuzuweisen. Obwohl sich vielleicht jeder unter dem Begriff Gebrauchstauglichkeit zunächst etwas vorstellen kann, wird doch einige Verwirrung deutlich, wenn man sich nach den genauen Inhalten des Konzepts erkundigt. Hier steckt der Teufel im Detail: Es stellt sich die Frage, wann ist ein Softwaresystem gebrauchstauglich und wann nicht?

Eine erste Antwort auf diese Frage gibt Teil 11 der internationalen Normenreihe ISO 9241 „Ergonomische Anforderungen für Bürotätigkeit mit Bildschirmgeräten“ (14, siehe auch Kasten 1). Hier wird Gebrauchstauglichkeit definiert als das Ausmaß, mit dem ein Softwaresystem *effektiv* und *effizient* die Erledigung der Arbeitsaufgabe des Benutzers unterstützt. Effektivität ist dabei definiert als die Genauigkeit und Vollständigkeit der Zielerreichung. Effizienz bezeichnet das Verhältnis von Aufwand zu erreichter Effektivität. Ebenso muß die subjektive *Zufriedenstellung* des Benutzers durch die Software gewährleistet werden.

Kasten 1: Übersicht der für die Gestaltung gebrauchstauglicher Software direkt relevanten Teile der ISO 9241

Für die Gestaltung gebrauchstauglicher Software direkt relevante Teile der ISO 9241:
Teil 10: Grundsätze der Dialoggestaltung
Teil 11: Richtlinien zur Gebrauchstauglichkeit
Teil 12: Informationsdarstellung
Teil 13: Benutzerführung
Teil 14: Dialogführung mittels Menüs
Teil 15: Dialogführung mittels Kommandosprachen
Teil 16: Dialogführung mittels direkter Manipulation
Teil 17: Dialogführung mittels Bildschirmformularen

Nach Dzida (5) wird die Effizienz durch Faktoren beeinflußt, die in Teil 10 der Norm als die zentralen *Grundsätze der Dialoggestaltung* definiert sind: Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Individualisierbarkeit und Lernförderlichkeit (13). Die übrigen Normteile dienen dann der weiteren Konkretisierung der Anforderungen an eine gebrauchstaugliche Software, z.B. Grundsätze der Informationsdarstellung (Teil 12) oder Empfehlungen zur Gestaltung von Menüs (Teil 14).

Ein interessantes Merkmal der Normreihe ISO 9241 wird schon durch einige der Formulierungen oben deutlich: Sie besteht in großen Teilen aus eher abstrakten Grundsätzen, Prinzipien und Empfehlungen. Dies unterscheidet sie in starkem Maße von üblichen technischen Normen, in denen klare Grenzwerte und wohldefinierte Meßmethoden zur Ermittlung der Normkonformität den zentralen Inhalt bilden.

Dieser Charakter ist gleichzeitig Fluch und Segen: Segen, weil die Norm durch das Etablieren abstrakter Prinzipien mit den schnellen und oft radikalen Veränderungen im Bereich Software Schritt halten kann. Ein Grundprinzip wie *Gebrauchstauglichkeit* kann in der aktuellen Definition auch noch nach Jahren seine Gültigkeit haben, wohingegen konkrete Empfehlungen - wie beispielsweise zur Gestaltung von Menüs - durch neuere aktuelle Entwicklungen sehr schnell veralten können.

Fluch ist eine solche Norm, weil das Verstehen und Anwenden abstrakter Grundprinzipien oft mehr erfordert als das bloße Lesen der Norm. Hier ist *Interpretation*

gefragt, und wo interpretiert wird, entstehen immer auch Mißverständnisse, Fehlinterpretationen oder sogar gerechtfertigte, unterschiedliche Auslegungen.

Eines dieser Mißverständnisse ist es, Gebrauchstauglichkeit einer Software als ein reines *Produktmerkmal* zu verstehen. So wird „Benutzerfreundlichkeit“ (Gebrauchstauglichkeit) in der *Software-Qualitätssicherung* als eines von acht Software-Qualitätsmerkmalen angesehen. Sie steht dabei gleichberechtigt neben Merkmalen wie Zuverlässigkeit, Verfügbarkeit, Sicherheit, Funktionserfüllung, Leistung, Wartungsfreundlichkeit und Übertragbarkeit (z.B. 28, S.26ff). Auf die einzelnen Begriffe soll hier nicht weiter eingegangen werden. Eines wird allerdings deutlich: Merkmale wie Funktionserfüllung sind reine *Produktmerkmale*, d.h. sind allein an das jeweilige Produkt gebunden. Ihre Erfüllung kann einzig und allein durch Betrachtung der Software bestimmt werden.

Dies allerdings auf Gebrauchstauglichkeit zu übertragen (wie es mit dem Merkmal „Benutzerfreundlichkeit“ angedeutet wird) wäre falsch und würde der Definition im Teil 11 der ISO 9241 nicht gerecht werden. Hier wird ausdrücklich festgehalten, daß die Gebrauchstauglichkeit einer Software nur in ihrem *Nutzungskontext* festgestellt werden kann (1, 14). Der Nutzungskontext besteht aus Merkmalen der zu erledigenden *Aufgabe*, der *Benutzer*, sowie aus *organisationalen, sozialen und physischen Randbedingungen*. Gebrauchstauglichkeit ist also ein Merkmal, das nur bedingt in einem Labor festgestellt werden kann. Sie entsteht erst im Zusammenspiel des Softwaresystems mit dem Nutzungskontext (siehe Kasten 2 für eine Analogie).

Kasten 2: Eine Analogie zur Gebrauchstauglichkeit

„Schuhe und Software“: eine Analogie

Besitzen Sie Sandalen?

Fragen Sie sich einmal, ob diese Sandalen „gebrauchstauglich“ sind. Was wäre eine wahrscheinliche Antwort? Ein zögerliches „Es kommt darauf an“?

Zuerst einmal ist die Bewertung der „Gebrauchstauglichkeit“ der Sandalen natürlich von Ihnen abhängig (dem **Benutzer**). Stellen Sie sich vor, Sie könnten keine Schleife binden (Sie sind offensichtlich ein Neuling im Umgang mit Schuhen). Dann würden Sie Sandalen mit ihren einfachen Schnallen und Riemen sicher als sehr nützlich ansehen.

Würden sie aber deswegen den Mount Everest mit Sandalen besteigen? Oder mit Bergschuhen einen Marathonlauf absolvieren? In beiden Fällen sind die Schuhe nicht gebrauchstauglich, weil sie der **Aufgabe** nicht angemessen sind. Direkt ansehen (**Produktmerkmale**) kann man das den jeweiligen Schuhen nicht. Und es bedeutet auch nicht, daß die jeweiligen Schuhsorten in allen Fällen nicht gebrauchstauglich sind.

Zurück zu den Sandalen. Stellen Sie sich einen schönen Sandstrand auf den kanarischen Inseln vor. Es ist heiß. Ein optimales **Klima** für Ihre Sandalen. Würden Sie Sandalen jetzt weniger nützlich finden, nur weil ein Norweger in den Wintermonaten eher auf Sandalen verzichtet?

Sie wandern über den Strand. Wunderbar! Leider wirbeln Sie mit jedem Schritt eine Menge Sand auf. Das begeistert die anderen Sonnenbadenden nicht besonders. Zie-

hen Sie Ihre Sandalen lieber aus, bevor Sie das sorgsam ausbalancierte **organisationale (soziale)** Gefüge der Sandstrandgemeinde noch ganz durcheinanderbringen!

Sie haben den Badetag überlebt und sind abends zu einer Hochzeit eingeladen. Halten Sie Sandalen hier für angemessen (**soziale Konventionen**)?

Diese Analogie ließe sich wahrscheinlich endlos fortsetzten. Uns erscheint sie direkt einsichtig. Es bleibt die Frage offen, warum bei der Bewertung der Gebrauchstauglichkeit von Software (im Gegensatz zu Schuhen) von einem Experten oft erwartet wird, daß er ohne Berücksichtigung des Nutzungskontexts zu einem sinnvollen Ergebnis gelangt.

Erst wenn deutlich ist, daß eine Software zum Benutzer paßt, er mit ihr optimal seine Arbeitsaufgabe erledigen kann und Merkmale der anwendenden Organisation (z.B. Organisationsformen, Organisationskultur), soziale Merkmale (z.B. Kommunikationsformen, Gruppenarbeit) und physische Merkmale (z.B. eingesetzte Hardware, Umgebungsbedingungen wie Lärm, Beleuchtung etc.) angemessen berücksichtigt worden sind, kann sie als *gebrauchstauglich* gelten.

2.2 Warum gebrauchstaugliche Software?

Gebrauchstaugliche Software kann sowohl für einzelne *Benutzer* und *anwendende Unternehmen* (kurz *Anwender*) als auch für *Softwarehersteller* entscheidende Vorteile bieten (vgl. 1):

Für die Benutzer und die anwendenden Unternehmen liegen die Vorteile u.a. darin, daß gebrauchstaugliche Software...

(1) ... die *Produktivität* und *Arbeitszufriedenheit* der Benutzer steigert, indem beispielsweise die Anzahl oder die negativen Auswirkungen von Fehlern bei der Bedienung reduziert werden (7). Damit verbindet sich auch die Hoffnung, Fehlzeiten und hohe Fluktuation des Personals zu verringern.

(2) ... die *Motivation* der Benutzer erhöht, sich intensiv mit den Möglichkeiten des Softwaresystems auseinanderzusetzen und es so optimal einzusetzen.

(3) ... den *Lernaufwand* verringert und damit Kosten für Benutzerunterstützungsmaßnahmen (z.B. Training, „Hotline“) reduziert.

Für Softwarehersteller stellt Gebrauchstauglichkeit eine Eigenschaft dar, die es ihnen ermöglicht, sich von anderen, ähnlichen Produkten auf dem Markt positiv abzuheben (9). Nielsen (18, S. 9f.) berichtet beispielsweise von einer unveröffentlichten Studie, in der 70 Besprechungen von Softwaresystemen in verschiedenen Computerzeitschriften analysiert wurden. In diesen Besprechungen finden sich 784 Kommentare, die in Zusammenhang mit der Gebrauchstauglichkeit des besprochenen Softwaresystems stehen.

Eine Aufzählung intendierter oder konkreter Vorteile von gebrauchstauglicher Software ließe sich noch ausgiebig weiterführen. Zusammenfassend kann man sagen, daß gebrauchstaugliche Software auf der einen Seite einen Beitrag zur *Humanisierung von Bildschirmarbeit* darstellt. Auf der anderen Seite ist Gebrauchstauglichkeit ein *Qualitätsmerkmal von Software*, das auf einem kompetitiven Markt als Verkaufsargument zunehmend wichtiger wird.

Die allgemeine Akzeptanz der Notwendigkeit einer Berücksichtigung softwareergonomischer Erkenntnisse bei der Arbeitsgestaltung spiegelt sich auch in den gesetzlichen Regelungen wieder.

2.3 Ein Recht auf Ergonomie: Die Bildschirmarbeitsverordnung

Im Rahmen des Arbeitsschutzes nimmt Gebrauchstauglichkeit eine bereits gesetzlich fixierte Stellung ein (siehe Bildschirmarbeitsverordnung, BildscharbV, des deutschen Bundestages „Verordnung über Sicherheit und Gesundheitsschutz bei der Arbeit an Bildschirmgeräten“, die als „Gesetz über die Durchführung von Maßnahmen des Arbeitsschutzes zur Verbesserung der Sicherheit und des Gesundheitsschutzes der Beschäftigten bei der Arbeit“, Arbeitsschutzgesetz - ArbSchG, am 21. August 1996 in Kraft gesetzt wurde).

Kasten 3: Der für die Software-Ergonomie relevante Anhang der BildscharbV „Zusammenwirken Mensch-Arbeitsmittel“ (24)

Zusammenwirken Mensch-Arbeitsmittel

20. Die Grundsätze der Ergonomie sind insbesondere auf die Verarbeitung von Informationen durch den Menschen anzuwenden.

21. Bei Entwicklung, Auswahl, Erwerb und Änderung von Software sowie bei der Gestaltung der Tätigkeit an Bildschirmgeräten hat der Arbeitgeber den folgenden Grundsätzen insbesondere im Hinblick auf die Benutzerfreundlichkeit Rechnung zu tragen:

21.1 Die Software muß an die auszuführende Aufgabe angepaßt sein.

21.2 Die Systeme müssen den Benutzern Angaben über die jeweiligen Dialogabläufe unmittelbar oder auf Verlangen machen.

21.3 Die Systeme müssen den Benutzern die Beeinflussung der jeweiligen Dialogabläufe ermöglichen sowie eventuelle Fehler bei der Handhabung beschreiben und deren Beseitigung mit begrenztem Arbeitsaufwand erlauben.

21.4 Die Software muß entsprechend den Kenntnissen und Erfahrungen der Benutzer im Hinblick auf die auszuführende Aufgabe angepaßt werden können.

22. Ohne Wissen der Benutzer darf keine Vorrichtung zur qualitativen oder quantitativen Kontrolle verwendet werden.

Diese *Bildschirmarbeitsverordnung* stellt die Umsetzung der europäischen Bildschirmrichtlinie 90/270/EWG in nationales Recht dar.

Im Anhang der BildscharbV heißt es unter der Überschrift „Zusammenwirken Mensch-Arbeitsmittel“ (siehe auch Kasten 3) neben anderem explizit: „Bei der Entwicklung, Auswahl, Erwerb und Änderung von Software [...] hat der Arbeitgeber [...] Grundsätzen, insbesondere im Hinblick auf die Benutzerfreundlichkeit, Rechnung zu tragen“ (zitiert nach 24, S. 234).

Problematisch an der Formulierung der BildscharbV ist die nur unzureichende Konkretisierung, was die zu befolgenden Grundsätze der Ergonomie sind. Unter Experten gibt es allerdings den Konsens, daß die internationale Normenreihe ISO 9241 zur weiteren Konkretisierung und als Interpretationshilfe herangezogen wird.

Auf dieser Art werden also Inhalte der ISO 9241 - zwar rechtlich nicht unbedingt einwandfrei, aber allgemein akzeptiert - zu einem gesetzlich fixierten Anspruch.

2.4 Grundtechniken der software-ergonomomischen Qualitätssicherung

Bislang haben wir uns damit beschäftigt, was Gebrauchstauglichkeit von Software überhaupt ist, was man sich von ihr verspricht und in welchem Maße sie gesetzlich fixiert ist. Die Frage, die sich nun stellt, ist, wie man gebrauchstaugliche Software überhaupt erstellen kann? Oder genauer gesagt: mit welchen Methoden arbeitet man in der *software-ergonomischen Qualitätssicherung*?

Zuerst einmal muß man feststellen, daß es ein wirklich reiches Repertoire unterschiedlichster Methoden gibt (vgl. 10). Eine detaillierte Vorstellung würde den vorliegenden Beitrag sprengen. Eine wichtige grundsätzliche Unterscheidung ist das reine *Bewerten* der ergonomischen Qualität eines bereits vorhandenen Softwaresystems und das *Gestalten* eines Softwaresystems unter Berücksichtigung ergonomischen Wissens. Im ersten Fall wird ein Experte um ein unabhängiges Gutachten gebeten. Dieses Gutachten dient meistens einer Kaufentscheidung oder einer Überprüfung der Vertragserfüllung von Software-Entwicklern im Hinblick auf die ergonomische Qualität. Diese Art der Software-Ergonomie ist für einen konstruktiv-gestalterisch arbeitenden Berater von eher begrenztem Interesse und wird aus diesem Grund in der vorliegenden Arbeit nicht weiter thematisiert. Dies soll allerdings kein Abwerten der bloße Bewertung andeuten, da sie für eine Organisation durchaus den Einstieg in das Thema Software-Ergonomie erleichtern kann.

Eine weitaus interessantere Beratungsaufgabe als das Bewerten ist das *Gestalten* von Softwaresystemen. Hier kann ein Berater durch das Anwenden angemessener Methoden direkt zur Qualitätssteigerung der Software beitragen.

Obwohl im konkreten Fall unterschiedlichste Methoden zum Einsatz kommen können, gibt es doch *Grundtechniken* der software-ergonomischen Qualitätssicherung, die im folgenden kurz beleuchtet werden sollen.

Diese Grundtechniken oder Postulate sind:

(1) Benutzerbeteiligung und Interdisziplinarität

(2) Prototyping und iterative Gestaltung

Obwohl es immer wieder Beiträge gibt, die einzelne Postulate wie beispielsweise Benutzerzentrierung oder Benutzerbeteiligung kritisch reflektieren (z.B. 12, 29), haben sie nicht nur in der Literatur zur Mensch-Computer Interaktion ihren festen Platz eingenommen (z.B. 18, 21, 23, S. 45ff.). Vielmehr haben sie unter der Bezeichnung „benutzer-zentrierte Softwarentwicklung bzw. Gestaltung“ auch ihren Niederschlag in einem noch rudimentären Entwurf zur ISO 13407 „human-centred design“ (15) gefunden. Im folgenden sollen diese beiden Grundtechniken kurz vorgestellt werden.

2.5 Benutzerbeteiligung und Interdisziplinarität

Unter *Benutzerbeteiligung* (z.B. 17) versteht man das Beteiligen zukünftiger Benutzer am Software-Entwicklungsprozeß. Solche „sozialen Gestaltungsansätze“ gehen davon aus, daß fehlende Einbindung von Betroffenen in die Gestaltung die Effizienz

des gesamten Entwicklungsprozesses und des resultierenden Produktes in Frage stellt.

Leela Damodaran (4) formuliert es beispielsweise so (Übersetzung der Autoren):

> „Es ist die Grundannahme der meisten, wenn nicht sogar aller sozialen Gestaltungsansätze [...], daß die implementierende Organisation ohne effektive Benutzerbeteiligung in allen Phasen der Planung und Gestaltung eines Softwaresystems ihre Probleme einfach nur verschiebt. Treten die Probleme dann nach der Implementierung und Einführung des Systems auf, ist es wahrscheinlich, daß sie ernsthafter und veränderungsresistenter sind, da Veränderungen des Systems durch die 'Verhärtung' des Gestaltungsprozeß zunehmend teurer werden" (S. 365).

In ihrem Artikel faßt sie einige Vorteile zusammen, die man sich von Benutzerbeteiligung verspricht. Die wichtigsten sind:

(1) Erhöhte Gebrauchstauglichkeit durch genauer formulierte Benutzeranforderungen (wirkt auf die Aufgabenangemessenheit des Systems, siehe 13).

(2) Das Vermeiden teurer Funktionalitäten, die Benutzer nicht wollen oder nicht nutzen können.

(3) Erhöhte Akzeptanz des Systems durch die Benutzer.

Etwas allgemeiner ausgedrückt erhofft man sich von Benutzerbeteiligung Vorteile durch mehr Information über die Arbeitsaufgabe bzw. Arbeitssituation und die Berücksichtigung von Wertvorstellungen und Interessen der zukünftigen Benutzer.

Mit dem Akzeptieren der Sinnhaftigkeit von Benutzerbeteiligung wird indirekt noch eine weitaus grundlegendere Tatsache akzeptiert, nämlich die Notwendigkeit von *Interdisziplinarität*. Software-Entwicklung ist mittlerweile so komplex, daß weder eine Person alleine noch eine einzelne Personengruppe (z.B. Software-Entwickler, Benutzer, Projektmanager, Software-Ergonomen) alle Kompetenzen in sich vereinen kann.

Ein jeder aus diesen Personengruppen kann nur auf einem Gebiet Experte sein: der *Benutzer* als Experte für die zugrundeliegende Arbeitsaufgabe und ihre Umsetzung in die Software, der *Software-Entwickler* als Experte für die Umsetzung der Wünsche und Vorstellungen in die Technologie und deren Grenzen, der *Projektmanager* für das Projektbudget, Zeitmanagement und die Projektpolitik und der *Software-Ergonom* für software-ergonomische Fragen und die Durchführung von Benutzerbeteiligung.

Hat man erst einmal akzeptiert, daß kein „Spieler" in einem Software-Entwicklungsprojekt alleine ein qualitativ hochwertiges Produkt entwickeln kann, wird die interdisziplinäre Zusammenarbeit natürlich und unverzichtbar, auch wenn sie meist nicht ohne Schwierigkeiten verläuft.

2.6 Prototyping und iteratives Gestalten

„Prototyping" als eine Technik der Software-Entwicklung wird in der neueren Literatur häufig beschrieben (für einen Überblick siehe 22). Es existiert zu diesem Thema umfangreiche Literatur, die sich beispielsweise mit verschiedenen Arten des Prototyping und entsprechenden Werkzeugen (Programmiersprachen) auseinandersetzt.

Die Grundidee des Prototypings ist allerdings immer gleich und recht einfach. Dem Benutzer (bzw. dem Auftraggeber) soll Aussehen und Verhalten des System „be-

greifbar“ gemacht werden, bevor das System vollständig implementiert ist und damit nur noch schwer verändert werden kann. Je früher beispielsweise ein Fehler in der Anforderungsdefinition gefunden werden kann, desto einfacher ist es, eine veränderte Anforderung angemessen zu berücksichtigen. Ist das System erst einmal fertiggestellt, läßt sich außer kosmetischen Korrekturen oft nichts mehr verändern. Es steigt die Gefahr, an den Benutzern „vorbei zu entwickeln“. Warum ist es schwer, Fehler in den schriftlich dokumentierten Anforderungen zu entdecken? Dies liegt in der Natur dieser Anforderungen, die abstrakt und immateriell sind. Prototyping stellt also eine Möglichkeit zur Konkretisierung des zu entwickelnden Softwaresystems dar.

Ausgehend von dieser Funktion des Prototypings wird unmittelbar deutlich, daß es so früh wie möglich im Entwicklungsprozeß stattfinden muß. Schon in sehr frühen Phasen lohnt es sich beispielsweise, auf Papierprototypen („mock ups“) oder nicht funktionale Oberflächenprototypen zurückzugreifen (z.B. 18, S. 93ff).

Durch das „Begreifbar“-machen der Software mittels Prototypen kann ein Rückkoppelungsprozeß zwischen Benutzern (Auftraggebern) und Software-Entwicklern in Gang gesetzt werden. Dabei präsentieren Entwickler einen ersten Prototypen. Dieser wird dann von Benutzern mehr oder weniger formal kommentiert und bewertet. Die Kommentare und die Bewertung bilden dann die Grundlage für eine Verbesserung des Softwaresystems. Dieser Rückkopplungsprozeß sollte so oft wie möglich wiederholt werden. Ein solches schrittweises Verbessern einer Software nennt man auch „iteratives Gestalten“ (z.B. 19). Mit dieser Technik reagiert man auf den Umstand, daß der Entwurf einer Software selbst einem Experten nicht auf Anhieb gelingt. Ab einem gewissen Komplexitätsgrad können einfach nicht mehr alle Aspekte und Probleme antizipiert und berücksichtigt werden.

Jede software-ergonomische Qualitätssicherungsmaßnahme wird in irgendeiner Weise auf diese Grundtechniken zurückgreifen. Ob und wie sie angewendet werden, hängt nicht zuletzt von der auftraggebenden Organisation ab. Wie die konkrete Form software-ergonomischer Beratung von der Organisation abhängen kann, macht der folgende Abschnitt deutlich.

3 Formen software-ergonomischer Beratung

Welche Methoden ein Software-Ergonomie-Berater anwenden kann, hängt in starkem Maße von der „Evolutionsstufe“ des Auftraggebers im Hinblick auf Software-Ergonomie ab. In diesem Abschnitt sollen solche Evolutionsstufen und ihre Konsequenzen für die Form der Beratung vorgestellt werden. Danach soll aus unserer Sicht die Rolle des Beraters in der Organisation beschrieben werden.

3.1 (R)evolution!

Nielsen (20) unterscheidet acht Evolutionsstufen einer Organisation im Hinblick auf Software-Ergonomie und die Notwendigkeit, software-ergonomische Qualitätssicherung durchzuführen (siehe Kasten 4).

Auf Stufe eins und zwei spielt software-ergonomische Beratung noch keine Rolle. Die Organisation befindet sich auf der *Stufe der Skepsis*. Sie meint entweder kein Problem zu haben, oder es aber leicht mit eigenen Ressourcen lösen zu können.

Erst ab Stufe drei und vier werden überhaupt externe Berater aktiv um Rat gefragt. Diese *Stufe der Neugierde* ist für den Berater besonders problematisch, weil er hier nur kurzfristig, punktuell und sehr spät zum Einsatz kommt. Da auf dieser Stufe innerhalb der Organisation (noch) keine Bereitschaft besteht, sich auf längerfristige Prozesse einzulassen, können Grundtechniken der software-ergonomischen Qualitätssicherung gar nicht oder kaum angewendet werden. Eine software-ergonomische Beratung unter solchen Bedingungen kann inhaltlich nicht mehr als eine Scheinberatung sein. Trotzdem ist diese Stufe oft unerläßlich in ihrer Funktion als „Eisbrecher". Hat sich der Software-Ergonomie-Berater trotz der widrigen Umstände durchgesetzt und die Qualität des Produktes verbessert, wird die Wahrscheinlichkeit steigen, sich innerhalb der Organisation intensiver mit Software-Ergonomie auseinanderzusetzen. Aber auch der umgekehrte Fall sollte berücksichtigt werden: Das Hinzuziehen eines Software-Ergonomen kann auch eine Art Alibifunktion haben, ohne daß echtes Interesse an einer Zusammenarbeit mit dem Berater besteht. Scheitert die Beratung, kann dem Auftraggeber oft kaum plausibel gemacht werden, welche Gründen dafür verantwortlich gemacht werden können. In diesem Fall wird die Wahrscheinlichkeit einer zukünftigen Berücksichtigung von Software-Ergonomie drastisch sinken.

Organisationen mit dem Reifegrad der Phasen fünf und sechs nehmen den software-ergonomischen Berater erstmals ernst. Dies ist die *Stufe der Akzeptanz*. Erst auf dieser Stufe ist eine inhaltlich sinnvolle und produktive Beratung möglich. Sie sollte dabei immer den Charakter einer Personal- bzw. Organisationsentwicklungsmaßnahme haben (siehe auch folgenden Abschnitt 3.2). Organisationsentwicklung wird hier im Sinne der „Gesellschaft für Organisationsentwicklung" verstanden, nämlich als „längerfristig angelegter organisationsumfassender Entwicklungs- und Veränderungsprozeß von Organisationen und der in ihr tätigen Menschen. Der Prozeß beruht auf Lernen aller Betroffenen durch direkte Mitwirkung und praktische Erfahrung.

Kasten 4: Software-ergonomische Evolutionsstufen und die resultierende Rolle des Beraters (in Anlehnung an 6, 20)

	Beschreibung der Evolutionsstufe	Rolle des Beraters
Skepsis		
1	Die Gebrauchstauglichkeit von Software spielt überhaupt keine Rolle. Software wird ausschließlich unter Berücksichtigung von Leistungsaspekten (Performanz) entwickelt.	**keine** Beratung
2	Die Bedeutung von Gebrauchstauglichkeit wird zwar erkannt, man ist innerhalb der Organisation allerdings der festen Überzeugung, daß das „normale“ Software-Entwicklerteam auch in dieser Hinsicht qualitativ hochwertige Produkte entwickeln kann.	**keine** Beratung
Neugier		
3	Die Verantwortlichen innerhalb der Organisation halten es für notwendig, die Software einem Berater für Software-Ergonomie zu zeigen. Von diesem wird dann meistens erwartet, daß er als eine Art Zauberer aus dem „Frosch“ (der aktuellen Benutzungsoberfläche) einen „Prinzen“ macht. Der von der Organisation erwartete Umfang (und die Kosten) der Beratung ist häufig auch nicht mehr als ein flüchtiger „Kuß“ (um beim Bild des Froschprinzen zu bleiben).	**Zauberer**
4	Bei den Verantwortlichen innerhalb der Organisation herrscht „Kopflosigkeit“ im Hinblick auf Software-Ergonomie. Diese entsteht aus dem Gefühl heraus, plötzlich und umfassend über software-ergonomische Gestaltung informiert sein zu müssen (und sie zu praktizieren). Ein Beispiel hierfür ist die Umstellung von zeichen-orientierten auf grafisch-orientierte Systeme. Der Hauptunterschied zur vorhergehenden Stufe ist eine etwas frühere und umfassendere Beteiligung des Beraters.	**Feuer-wehr-mann**
Akzeptanz		
5	In der Organisation kommen gelegentlich einfachere software-ergonomische Methoden (wie z.B. Benutzerbefragungen) zum Einsatz. Die Methoden werden in der Regel auch hier noch viel zu spät eingesetzt. Allerdings kann der software-ergonomische Berater in diesen Organisationen bereits darauf hoffen, Projektleiter (oder andere Teammitglieder) anzutreffen, die schon erste Erfahrungen mit Software-Ergonomie in früheren Projekten gesammelt haben. Auf dieser Stufe können von der Organisation bereits realistische Beratungsaufträge formuliert werden.	**Entwick-lungs-helfer**
6	In der Organisation kommen bereits in frühen Entwicklungsstufen eines Softwaresystems einfachere software-ergonomische Methoden systematisch zum Einsatz. Innerhalb solcher Organisationen kann ein Berater bereits mit eher einfachen Methoden beachtliche Erfolge erzielen.	**Katalysator**
Partnerschaft		
7	Innerhalb der Organisation gibt es Benutzergruppen und/oder Abteilungen für Software-Ergonomie. Software-ergonomische Qualitätssicherung ist hier ein fester Bestandteil des Entwicklungsprozesses. Hier begegnet der Berater ausgebildeten Fachkräften, die sich in einem Teil ihrer Arbeitszeit ausschließlich mit software-ergonomischen Fragen beschäftigen. Der Berater wird hier meistens bei Spezialfragen um Rat gebeten, oder es werden größere Projekte in seine Verantwortlichkeit gelegt.	**Mentor**
8	Innerhalb der Organisation ist Gebrauchstauglichkeit ein sehr wichtiges Qualitätsmerkmal und bestimmt maßgeblich den Software-Entwicklungsprozeß. Diese letzte Reifestufe wird nur selten erreicht, da selbst größere Unternehmen mit aktiven Benutzergruppen und Abteilungen für software-ergonomische Fragen in den meisten Fällen über keine ausreichend starken software-ergonomische Teams verfügen. Dadurch können nicht alle wünschenswerten Methoden eingesetzt werden. Der software-ergonomische Berater ist in diesen Organisationen ein gern gesehener Diskussionspartner, Moderator und Entwickler, der mit einem „frischen Blick“ noch bestehende Probleme im Software-Entwicklungsprozeß erkennen kann.	**Partner**

Ein Ziel besteht in der gleichzeitigen Verbesserung der Leistungsfähigkeit der Organisation (Effektivität) und der Qualität des Arbeitslebens (Humanität)“ (zitiert nach 25, S. 26). Nur so können den verantwortlichen Software-Entwicklern, beteiligten Benutzern und verantwortlichen Projektmanagern die Prinzipien softwareergonomischen Gestaltens und der Gebrauchstauglichkeit vermittelt werden. Dies ist unbedingt notwendig, da nur diese Grundtechniken und Grundsätze eine selbständige Gestaltung und Bewertung von gebrauchstauglicher Software möglich machen.

Erst ab der Phase sieben begegnet der Software-Ergonomie-Berater einem fachlich kompetenten Gegenüber. Die Organisation befindet sich damit auf der *Stufe der Partnerschaft*. Diese Stufe unterscheidet sich wesentlich von der vorangegangen Stufe. Der Berater für Software-Ergonomie tritt hier in seiner Rolle als „Macher“ („Entwicklungshelfer“ oder „Katalysator“) zurück und wird zum Berater bei besonderen Fragen, Kommentator und Diskussionspartner. Diese Rolle betont eher die Erfahrung des Beraters als Problemlöser und Wissenschaftler denn als Praktiker. Diese Stufe wird von Organisationen eher selten erreicht und spielt damit auch für den Berater eine eher unwesentliche Rolle.

Software-ergonomische Beratung kann also je nach vorgefundener Situation sehr unterschiedlich zu gestalten sein. Tritt der software-ergonomische Berater *früh* in den Entwicklungsprozeß ein (was auf der *Stufe der Akzeptanz* und der *Stufe der Partnerschaft* wahrscheinlicher ist), hat er die Möglichkeit, systematisch Methoden zur software-ergonomischen Qualitätssicherung anzuwenden. Da der Berater in dieser Stufe an Planungsprozessen beteiligt ist, bleibt bei der Anwendung dieser Methoden Raum für Kompromisse. Tritt der Berater *spät* in den Entwicklungsprozeß ein (*Stufe der Neugier*), wird er oft vor vollendete Tatsachen gestellt. Sein Handlungsspielraum ist dadurch extrem eingeschränkt. Nimmt ein Software-Ergonom einen solchen Auftrag an, muß er im Rahmen der software-ergonomischen Qualitätssicherung das Anwenden kompensierender Methoden auch gegen den Willen beteiligter Software-Entwickler zur Bedingung seiner Mitarbeit machen. Ein Beispiel dafür ist die *Re-Analyse und Bewertung* bereits festgelegter Anforderungen. Eine solche Re-Analyse kann zu erheblichen Problemen führen. Zentral ist hier der Umstand, daß sich der Entwicklungsprozeß beim späten Eintreten des Software-Ergonomie-Beraters oft schon in der Entwurfs- oder sogar Implementierungsphase befindet. Die Re-Analyse stellt dann aus der Sicht der Entwickler einen Rückschritt dar, der die in Entwicklungsprojekten oft vorherrschende Zeitknappheit noch verschärfen kann. Trotzdem ist eine erneute Analyse der Aufgaben und Bewertung der Anforderungen für den Software-Ergonomie-Berater unerläßlich. Ein weiteres Beispiel ist das Bestehen auf ein iteratives Vorgehen und die angemessene Beteiligung zukünftiger Benutzer.

Die Metapher der Evolutionsstufe soll zwei Dinge deutlich machen. Zum einen ist die Form software-ergonomischer Beratung stark von der auftraggebenden Organisation abhängig. Nicht jede Herangehensweise ist in jeder Organisation gleich möglich oder erfolgversprechend und die Inhalte und Vorgehensweisen softwareergonomischer Qualitätssicherungsmaßnahmen sollten immer in Verbindung mit der Evolutionsstufe der auftraggebenden Organisation bewertet werden.

Zum anderen steht hinter dem Konzept der Evolutionsstufen auch die Notwendigkeit der Entwicklung, in diesem Fall der *Organisationsentwicklung*. Software-Ergonomie muß zu einem Teil der organisationalen Praktiken, d.h. der Organisationskultur, werden. Dies gilt für anwendende Unternehmen gleichermaßen wie für

Softwarehäuser. Ob dabei alle Stufen durchlaufen werden müssen, wie es das Konzept nahelegt, sei einmal dahingestellt.

3.2 Die Rolle des Beraters auf der Stufe der Akzeptanz

Das Rollenverständnis des software-ergonomischen Beraters ist von zentraler Bedeutung für die Art und Weise der Beratung. Dabei ist die Beratung auf der *Stufe der Neugier* oder der *Stufe der Partnerschaft* einer Organisation aus einer inhaltlichen Perspektive nicht bedeutsam. Auf der Stufe der Neugier besteht die Beratung hauptsächlich aus oberflächlicher Begutachtung und Schadensbegrenzung. Der Gestaltungsspielraum des Beraters geht gegen Null. Auf der Stufe der Partnerschaft drehen sich die Beratungsinhalte eher um Spezialprobleme, die zwar interessant, aber wenig repräsentativ sind (vgl. Kasten 4).

Interessant ist das Rollenverständnis des Beraters auf der *Stufe der Akzeptanz*. Hier kann der Berater viel bewirken, wenn er die entsprechende Herangehensweise wählt.

Wir verstehen in Anlehnung an Lippit, Watson und Wesley (16) den Berater als einen externen „change agent", der als Experte Forschung und Veränderung unter Beteiligung der Organisationsmitglieder (Auftraggeber, Entwickler, Benutzer) bei der Planung, Durchführung, Auswertung und Interpretation betreibt. Professionelle, unabhängige externe Beratung ist somit aus unserer Sicht sowohl ein Forschungs- als auch ein Lern- und Veränderungsprozeß unter Berücksichtigung betrieblicher Erfordernisse. Der Berater ist kein bloßer „Wissensträger" oder „Prozeßspezialist", sondern muß vielmehr im Laufe seiner Beratungstätigkeit eine hohe Rollenflexibilität an den Tag legen: Er muß zwischen Beteiligten verhandeln, er muß lehren, Fakten klären, an verschiedenen Phasen der Problemlösung mitarbeiten und die Ereignisse reflektieren (vgl. 26).

Wir betrachten Software-Ergonomie also mehr als *Organisationsentwicklungsaufgabe*, denn als reine Prozeß- oder gar Benutzungsoberflächenoptimierung. Software-Ergonomie ist in diesem Sinne Ausdruck einer Organisationskultur, die sich um *Humanisierung der Arbeit* bemüht und eine humanzentrierte Sichtweise einer technikzentrierten vorzieht. Allerdings bedeutet dies nicht, daß Effizienz und Produktivität hier keine Rolle spielen. Wenn möglich, werden beide Ziele berücksichtigt. Daß dies im Prinzip möglich ist, zeigen software-ergonomische Variablen wie *Fehlerbewältigungszeit*. So kann ein Ziel software-ergonomischer Gestaltung, *die Reduktion des Anteils von Fehlerbewältigungszeit an der gesamten Arbeitszeit*, die Produktivität von Arbeitnehmern steigern und damit *ökonomische Kosten* reduzieren. Gleichzeitig können aber auch potentiell *psychologische Kosten*, wie negative emotionale Reaktionen (3) oder subjektive mentale Beanspruchung (11) reduziert werden.

4 Schlußfolgerung

Die vorliegende Arbeit möchte Einblicke in Probleme der Beratungspraxis „externer" Software-Ergonomen geben. Wie so oft in der Praxis, muß der Experte mit eingeschränkter Gestaltungsfreiheit umgehen können und ganz unterschiedliche Vorstellungen und Erwartungen von Auftraggebern, beteiligten Benutzern und Software-Entwicklern berücksichtigen. Dies ist nur möglich, wenn die Beratungspraxis durch verständliche und klare Konzepte geleitet wird. Probleme, wie z.B. das zu

späte Eintreten in einen Software-Entwicklungsprozeß müssen angesprochen und die sich daraus ergebenen Konsequenzen dargelegt werden. Auch Punkte wie das eigene Rollenverständnis des Beraters müssen deutlich gemacht werden. Nur wenn der Berater sein Konzept deutlich macht, begründet, auf welche Elemente seines Konzeptes nicht verzichtet werden kann und diesen „Beratungsstil“ konsequent durchhält, kann es zu einer fruchtbaren Kooperation aller Beteiligten kommen.

Das Konzept der „externen“ Beratung macht besonders im Bereich der Software-Ergonomie Sinn. Hier ist der buchstäbliche „frische“ Blick auf Praktiken und Prozesse gefragt, den oft nur ein Externer liefern kann.

5 Danksagung

Wir möchten Uta Sailer für ihre Idee zur „Schuh“-Analogie danken.

6 Literatur

(1) Bevan, N. & Macleod, M. (1994). Usability measurement in context. *Behaviour & Information Technology*, Vol. 13, **1 & 2**, S. 132-145.

(2) Brennecke, A. & Keil-Slawik, R. (1997). Einsatz elektronischer Lehr- und Lernumgebungen in der Software-Ergonomie-Ausbildung. In R. Liskowsky, B.M. Velichkovsky & W. Wünschmann (Hrsg.), *Software-Ergonomie ’97. Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung*, S. 83-92, Stuttgart: B.G. Teubner.

(3) Brodbeck, F.C. (1991). Fehlerbewältigungszeiten und die Nutzung von Unterstützungsmöglichkeiten. In M. Frese & D. Zapf (Hrsg.), *Fehler bei der Arbeit mit dem Computer - Ergebnisse von Beobachtungen und Befragungen im Bürobereich*, S. 80-94. Bern: Huber.

(4) Damodaran, L. (1996). User involvement in the systems design process - a practical guide for users. *Behaviour & Information Technology*, Vol. 15, **6**, S. 363-377.

(5) Dzida, W. (1995). Standards for user-interfaces. *Computer Standards and Interfaces*, **17**, S. 89-97.

(6) Ehrlich, K. & Rohn, J. (1994). Cost-justification of usability engineering: A vendor's perspective. In R.G. Bias and D.J. Mayhew (Hrsg.), *Cost-Justifying Usability*. Boston, MA: Academic Press.

(7) Frese, M., Irmer, C. & Prümper, J. (1991). Das Konzept Fehlermanagement: Eine Strategie des Umgangs mit Handlungsfehlern in der Mensch-Computer Interaktion. In M. Frese, C. Kasten, C. Skarpelis & B. Zang-Scheucher (Hrsg.), *Software für die Arbeit von morgen*, S. 241-251. Berlin, Heidelberg, New York: Springer.

(8) Frese, M., Prümper, J. & Solzbacher, F. (1994). Eine Fallstudie zu Benutzerbeteiligung und Prototyping. In F.C. Brodbeck & M. Frese (Hrsg.), *Produktivität und Qualität in Software-Projekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung*, S. 135-143. München: Oldenbourg.

(9) Glass, B. (1997). Swept Away in a Sea of Evolution: New Challenges and Opportunities for Usability Professionals. In R. Liskowsky, B.M. Velichkovsky & W. Wünschmann (Hrsg.), *Software-Ergonomie ’97. Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung*, S. 17-26. Stuttgart: B.G. Teubner.

(10) Hampe-Neteler, W. (1994). *Software-ergonomische Bewertung zwischen Arbeitsgestaltung und Softwaretechnik*. Frankfurt am Main: Peter Lang.

(11) Hassenzahl, M., Prümper, J. & Sailer, U. (1997). Die Priorisierung von Problemhinweisen in der software-ergonomischen Qualitätssicherung. In R. Liskowsky, B.M. Velichkovsky & W. Wünschmann (Hrsg.), *Software-Ergonomie ’97. Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung*, S. 191-201. Stuttgart: B.G. Teubner.

(12) Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W. & Brodbeck, F.C. (1996). Don't underestimate the problems of user centredness in software development projects - there are many! *Behaviour & Information Technology*, Vol. 15, **4**, S. 226-236.

(13) ISO (1996a). ISO 9241-10: Ergonomic requirements for office work with visual display terminals. Part 10: Dialogue Principles. *International Organization for Standardization*.

(14) ISO (1996b). ISO 9241-11: Ergonomic requirements for office work with visual display terminals. Part 11: Guidance on usability, comitee draft. *International Organization for Standardization*.

(15) ISO (1996c). ISO CD 13407: Human-centred design, comitee draft. *International Organization for Standardization*.

(16) Lippitt, R., Watson, J. & Wesley, B. (1958). *The dynamics of planned change*. New York: Holt, Rinehart & Winston.

(17) Mumford, E. & Welter, G. (1984). *Benutzerbeteiligung bei der Entwicklung von Computersystemen*. Berlin.

(18) Nielsen, J. (1993a). *Usability Engineering*. Boston, San Diego, New York, London, Sydney, Tokyo, Toronto: Academic Press.

(19) Nielsen, J. (1993b). Iterative user interface design. *IEEE Computer*, **26**, 11, S. 32-41.

(20) Nielsen, J. (1994). Guerrilla HCI: Using discount usability engineering to penetrate the intimidation barrier. In R.G. Bias and D.J. Mayhew (Hrsg.), *Cost-Justifying Usability*, S. 245-272. Boston, MA: Academic Press.

(21) Norman, D.A. & Draper, S.W. (1986). *User-centered system design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

(22) Pomberger, G. & Blaschek, G. (1996). *Software Engineering. Prototypen und objekt-orientierte Software-Entwicklung*, 2. Auflage. München, Wien: Hanser.

(23) Preece, J., Rogers, Y., Benyon, D., Holland, S. & Carey, T. (1994). *Human-Computer Interaction. Harlow*, GB: Addison-Wesley.

(24) Richenhagen, G., Prümper, J. & Wagner, J. (1997). *Handbuch der Bildschirmarbeit*. Neuwied, Kriftel, Berlin: Luchterhand.

(25) Rosentiel, L. v., Einsiedler, H., Streich, R. & Rau, S. (1987). *Motivation durch Mitwirkung*, S. 25-39. Stuttgart: Schäffer-Poeschel.

(26) Rosentiel, L. v., Molt, W. & Rüttinger, B. (1995). *Organisationspsychologie*, 8. Auflage. Stuttgart, Berlin, Köln: Kohlhammer.

(27) Strohm, O. (1991). Arbeitsorganisation, Methodik und Benutzerorientierung bei der Software-Entwicklung. Eine arbeitspsychologische Analyse und Bestandsaufnahme. In M. Frese, C. Kasten, C. Skarpelis und B. Zang-Scheucher (Hrsg.), *Software für die Arbeit von morgen*, S. 431-441. Berlin, Heidelberg, New York: Springer.

(28) Trauboth, H. (1996). *Software-Qualitätssicherung: konstruktive und analytische Maßnahmen*. München, Wien: Oldenbourg.

(29) Webb, B.R. (1996). The role of users in interactive systems design: when computers are theatre, do we want the audience to write the script? *Behaviour & Information Technology*, Vol. 15, **2**, S. 76-83.