Maschine, Partner, Medium, Welt ...

Eine Leitbildgeschichte der Software-Ergonomie

Susanne Maaß

Sind Computer rechnende Maschinen? Sind sie hilfreiche Assistenten? Oder Werkzeuge? Koordinationsmedien? Virtuelle Welten? Es kommt darauf an, wie man sie gestaltet und gebraucht.

In der Geschichte der Computer- und Software-Entwicklung hat es verschiedene Ideen und Ansätze gegeben, dem Computer eine dieser Rollen zuzuschreiben. Eine solche Zuschreibung geschieht nicht immer bewußt und unter Ansehung aller stillen Konsequenzen, sie manifestiert sich aber deutlich im metaphernreichen Sprachgebrauch von Entwicklern und Benutzern. Insbesondere mit Blick auf die benutzergerechte Gestaltung von Mensch-Computer-Schnittstellen wird versucht, den Effekt von Metaphern, weniger bekannte Gegenstandsbereiche durch Anspielung auf vertrautere Bereiche zu beschreiben, zu nutzen.

Die Vielfalt solcher Metaphern war ein Auslöser für eine systematische Untersuchung und Entfaltung der impliziten Leitbilder, die im Mittelpunkt dieses Beitrages stehen sollen. Im Rahmen der Softwareentwicklung spricht man bislang selten von Leitbildern, sondern eher von Perspektiven, unter denen entwickelt wird. Mit dem Begriff Perspektive wird betont, daß verschiedene Menschen oder auch Menschen in verschiedenen Rollen unterschiedliche Sichten auf die Welt haben. Durch die Benennung von Leitbildern wird eher das Gemeinsame an diesen Sichten betont und eine Orientierung für die Zukunft gegeben. Die hier zu behandelnden Leitbilder der Softwareentwicklung wurden im nachherein analytisch aufgedeckt und werden nicht als strikt handlungsleitend eingeschätzt.

Der folgende erste Abschnitt behandelt Softwareentwicklung unter dem besonderen Aspekt benutzergerechter Gestaltung, auch Software-Ergonomie genannt. Die historische Entwicklung dieses Gebietes wird dargestellt. Der Standpunkt der Software-Ergonomie hilft, die hier zu diskutierenden Leitbilder in der Softwareentwicklung zu erkennen und auch zu bewerten. Der zweite Abschnitt benennt sieben verschiedene Leitbilder der Softwareentwicklung und stellt heraus, welche Ziele und Annahmen mit ihnen verbunden sind: Jedes Leitbild setzt Schwerpunkte bei den Gestaltungszielen und impliziert Rollenzuweisungen sowohl an die Benutzer*) als auch an die Softwareentwickler selbst. Die Frage entsteht, ob Informatiker als Softwareentwickler überhaupt den Rollen gerecht werden können, in denen sie sich zunehmend sehen. Reicht das gegenwärtige Informatik-Curriculum aus, um Absolventinnen und Absolventen die nötigen Qualifikationen zu vermitteln? Dies wird in Abschnitt 3 diskutiert.

1. Software-Ergonomie

Software-Ergonomie beschäftigt sich mit der benutzergerechten Gestaltung Computer-unterstützter Arbeit, insbesondere mit der angemessenen Gestaltung der Mensch-Computer-Schnittstelle. Das Forschungsgebiet hat sich seit etwa 10 Jahren etabliert.

---

*) Als Zugeständnis an Lesegewohnheiten und aus Gründen der Schreibökonomie werden im folgenden meist nur männliche Funktionsbezeichnungen benutzt. Frauen sind mal wieder mitgemeint.

Bis in die 70er-Jahre hinein standen bei der Softwareentwicklung *technische Aspekte* im Vordergrund. Primäre Ziele bei der Gestaltung von *Hardware und Algorithmen* waren Effektivität und Korrektheit, Zuverlässigkeit, Platz- und Zeiteffizienz. Benutzt wurden Computer überwiegend von Spezialisten mit naturwissenschaftlich-technischem Hintergrund.

Mit dem Aufkommen interaktiver Systeme und ihrem Einsatz für kommerzielle Anwendungen wandelte sich rasch die Benutzerschaft vom DV-Experten zum sog. naiven Benutzer, und es zeigte sich, daß die *Handhabung* ein Problem war. Ende der 70er-Jahre entstand die Forderung nach "*Benutzerfreundlichkeit*". Die Handhabung sollte einfach, wenn möglich sogar "narrensicher" sein, man erhoffte sich viel von natürlich-sprachlichen Interaktionsformen, forderte Benutzerführung, verständliche Fehlermeldungen und Hilfesysteme. "Der Benutzer" mit seinen geistigen und körperlichen Fähigkeiten und Begrenzungen wurde entdeckt. Kognitive Psychologen nahmen sich der Problematik an, wie die Wahrnehmungs- und Lernfähigkeiten, verbale und kommunikative Fähigkeiten der Menschen genutzt bzw. bei der Gestaltung der Systemoberfläche angemessen berücksichtigt werden könnten. Der damalige Stand der Diskussion um Benutzerfreundlichkeit wird dargestellt in Dehning, Essig & Maaß (1978).

In den 80er-Jahren wurde der Begriff "*Software-Ergonomie*" geprägt (Griese, 1982). Es wurde klar, daß gute Handhabung allein nicht ausreicht, um Systeme benutzergerecht zu machen. Erst die zusätzliche *Aufgabenangemessenheit* der Funktionalität macht Systeme akzeptabel und nützlich für ihre Benutzer. (Vgl. die entsprechenden Leitthemen der Software-Ergonomie-Tagungen 1987 und 1989, Schönpflug & Wittstock, 1987, Maaß & Oberquelle, 1989.) In den 80er-Jahren entstanden sowohl das sog. IFIP-Modell für Benutzerschnittstellen (Dzida, 1983), als auch die Grundsätze ergonomischer Dialoggestaltung (DIN 66234, Teil 8). Beide sprechen den Aspekt aufgabenangemessener Funktionalität an.

Das IFIP-Modell unterscheidet genauer zwischen verschiedenen Schnittstellenaspekten: Ein-/Ausgabeschnittstelle und Dialogschnittstelle bestimmen die Handhabung, während die sog. Werkzeugschnittstelle (heute würde man eher Anwendungsschnittstelle sagen) die zugreifbare Funktionalität widerspiegelt. Zusätzlich wird schon die Organisationsschnittstelle benannt, die die Einbettung der Computer-gestützten Arbeit in organisatorische Zusammenhänge herstellt. Allerdings wird dieser Aspekt in der Folgezeit noch nicht weiter behandelt.

Die DIN 66234 Teil 8 legt fünf Gestaltungsgrundsätze fest. An erster Stelle wird die Aufgabenangemessenheit gefordert. Die vier anderen Prinzipien (Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlerrobustheit) werden meist primär auf die Handhabung bezogen. In dieser Zeit, als die Abstimmung der Funktionalität auf die Aufgaben der Benutzer als wichtiger Aspekt Software-ergonomischer Gestaltung erkannt wurde, interessierten sich zunehmend Arbeitspsychologen für das Gebiet.

Bei der Software-Ergonomie-Tagung 1993 schlug der Arbeitspsychologe Walter Volpert in seinem Hauptvortrag vor, das gemeinsame Gebiet zwischen Informatik, Psychologie und Arbeitswissenschaft zukünftig als "*Arbeitsinformatik*" zu bezeichnen, entsprechend anderen interdisziplinären Bereichen wie Arbeitsmedizin, Arbeitspsychologie, Arbeitsrecht (Volpert, 1993). Er sieht die Arbeitsinformatik beschäftigt mit der Analyse, Gestaltung und Bewertung von Arbeitsaufgaben, Arbeitsmitteln, Arbeitsorganisation und Qualifizierungsprozessen. Die Frage der Unterstützung kooperativer Arbeit und damit der Einbettung computergestützter Arbeit in *organisatorische Zusammenhänge* wurde schon Anfang der 90er-Jahre für die Software-Ergonomie formuliert (Oberquelle, 1991) und prägt die gegenwärtige Phase der Forschung.

Zusammengefaßt muß Software-ergonomische Gestaltung individuelle, aufgabenspezifische, technische und gesamtorganisatorische Gegebenheiten beachten. Abb. 1, unten, führt diese vier Faktoren in ihrer wechselseitigen Abhängigkeit vor Augen.

Innerhalb der Software-Ergonomie haben sich drei parallele Strömungen ausgebildet, die die Frage der angemessenen Softwareentwicklung und -gestaltung aus verschiedenen Blickwinkeln betrachten (genaueres hierzu in Maaß, 1993).

Technisch orientierte Forschung legt ihren Schwerpunkt auf die Optimierung der technischen Komponente und versucht, durch technische Weiter- und Neuentwicklung Fortschritte in der Mensch-Computer-Interaktion zu erzielen. Dabei macht sie implizit Annahmen über die Benutzer, die sie nicht weiter problematisiert oder verifiziert. *Kognitiv-psychologische* Forschung betrachtet Systemgestaltung primär analytisch unter den Gesichtspunkten der menschlichen Wahrnehmung, des Denkens, Problemlösens und Lernens. Sie strebt damit eine Abstimmung der Technik auf allgemein erwiesene und partiell erforschte Fähigkeiten der Benutzer an. Dabei gibt es einen internen Streit darüber, inwieweit ihre Ergebnisse in realistischen Arbeitskontexten relevant bleiben. Die *arbeitswissenschaftliche* Software-Ergonomie-Forschung schließlich sieht Technikeinsatz im größeren Kontext von organisierter Arbeit. Hier werden also die Faktoren Aufgabe und Organisation in den Vordergrund gerückt, und technische Gestaltungsmaßnahmen werden hierzu in Beziehung gesetzt.

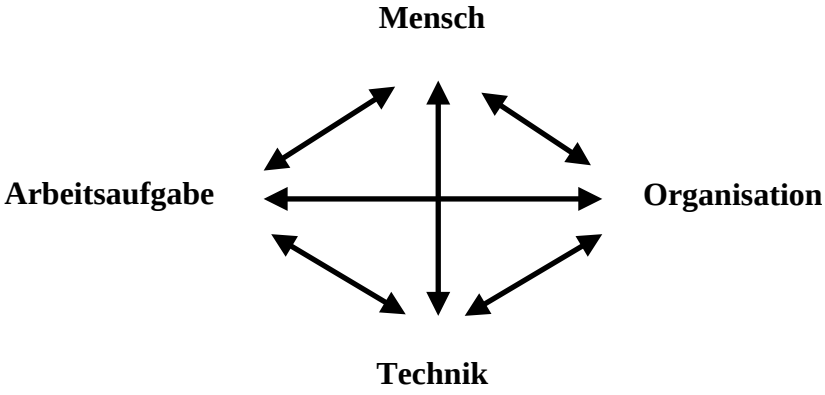

Abb. 1 Faktoren benutzergerechter Softwaregestaltung

Das Aufzeigen dieser vier Faktoren benutzergerechter Softwaregestaltung und damit die Einbettung der technischen Gestaltungsdiskussion in den breiteren Kontext technikunterstützter Arbeitsgestaltung ist ein wichtiges Ergebnis der Software-ergonomischen Forschung (vgl. Oberquelle, 1991). In Diskussionen um die angemessene Gestaltung der Mensch-Computer-Interaktion werden Metaphern aus allen vier Bereichen verwendet. Im folgenden Abschnitt sollen verschiedene dieser Metaphern als Leitbilder analysiert werden.

2. Leitbilder in der Softwareentwicklung

Arbeitsgegenstände müssen benannt werden, einmal von den Arbeitenden für sich selbst und erst recht in der Kommunikation mit anderen. Softwareentwickler haben also schon immer Sprechweisen finden müssen, um die Zwecke ihrer Software und ihre besonderen Eigenschaften im Gespräch mit Fachkollegen oder auch gegenüber den späteren Verwendern ihrer Produkte zu bezeichnen. Vielleicht ist es gerade die besondere Abstraktheit ihres Arbeitsgegenstandes und seine gleichzeitige, völlig offene Gestaltbarkeit, die sie zu so vielfältigen Bildern für den Computer greifen läßt, wie sie im folgenden besprochen werden sollen. Die Bilder charakterisieren verschiedene Rollen, die der Computer für seine Benutzer*) annehmen kann. Hier soll allerdings nicht die Weite des Spektrums an verwendeten Bildern thematisiert werden, sondern die Tatsache, daß mit ihnen implizit Ziele und

---

*) Bereits der Begriff "Benutzer" ist, wie Grudin (1990) zutreffend ausführt, bildhaft und spiegelt eine ingenieurshafte Sicht wider. In der folgenden Diskussion von Leitbildern bemühe ich mich, eine Vermischung der Perspektiven durch nachlässige Wortwahl zu vermeiden. Um die Verwendung der Bezeichnung "Benutzer" komme ich in Ermangelung eines besseren allgemeinen Begriffs nicht herum.

Annahmen verbunden sind: Jedes Leitbild bedeutet eine Schwerpunktsetzung bei der Gestaltung der Software; gleichzeitig werden Softwarebenutzern und -entwicklern gewisse Rollen zugewiesen und ihr Verhältnis untereinander wird festgelegt (Abb. 2 gibt einen Überblick).
Die folgende Diskussion der verschiedenen Leitbilder wird keine ausführlichen Beispiele für den Gebrauch entsprechender Metaphern anführen. Diese sind bei Interesse in Maaß & Oberquelle (1992) nachzulesen.

Das in den frühen Jahren der Computerentwicklung gängigste Leitbild war das *Maschi-*

Leitbild	**Computer-Gestaltungsziele**	**Rolle des Software-Benutzers**	**Rolle des Software-Entwicklers**
Maschine	technisch optimal, maximale Automatisierung	Bediener, potentieller Störfaktor	Ingenieur, Maschineneinrichter
Mensch-Maschine-System	effizienter Informationsaustausch und -verarbeitung	Systemkomponente, Zulieferer und Abnehmer	Systemanalytiker und -planer, Kontrolleur
Dialogpartner (KI)	menschenähnliches Sprachverhalten	Hilfebedürftiger	unsichtbarer Modellierer
Dialogpartner (SE)	durchschaubares formales Komm.verhalten, angepaßt an Benutzerkonventionen	Fachkraft, durchschaut Konventionen, paßt sich bewußt an	Verantwortlicher für Computerverhalten, Ansprechpartner für B.
Werkstatt	praktische Arbeitsumgebung: Raum, Objekte, Werkzeuge	kompetenter Werker, richtet sich ein	technischer Experte, unterstützt qualifizierte Arbeit
Kooperationsmedium	optimale Unterstützung menschlicher Kommunikation und Kooperation	Mitglied einer kooperierenden Gruppe	technischer Experte, Moderator, Helfer
Alltagsanreicherung	C. unsichtbar, eingebettet in reale Welt	in der realen Welt Agierender	technischer "Alltagsverbesserer"
Künstliche Welt (VR)	zusätzliche neue Realität	in VR Agierender	Modellierer neuer Welten

Abb. 2 Leitbilder in der Softwareentwicklung

nen-Leitbild. Computer wurden als extrem schnelle Rechenmaschinen entwickelt. Beim Softwareentwurf wurde in erster Linie auf Effektivität und Effizienz geachtet. Maximale Automatisierung und damit Ersatz menschlicher Arbeit sind generell das Ziel. Der mit der Maschine arbeitende Mensch wird zum potentiellen Störfaktor, der möglichst weitgehend zu eliminieren ist. Soweit seine Mitarbeit noch nötig ist, wird er zum Maschinenbediener degradiert, der die Vorgänge, die er anstößt, nicht durchschaut. Die Entwickler sehen sich als Ingenieure und wirken gleichzeitig als Maschineneinrichter für die Bediener, die an der vorgefundenen Bedienungsschnittstelle nichts verändern können und sollen.

Eng mit dem ersten Leitbild verbunden ist das *System-Leitbild*. Hier werden Mensch und Maschine als zwei quasi gleichartige Komponenten eines (Mensch-Maschine-)Systems gesehen, in dem Informationen hin- und herfließen und verarbeitet werden. Die maschinelle Komponente wird dahingehend optimiert, daß Informationsaustausch und -verarbeitung möglichst effizient vonstatten gehen können. Der Mensch ist notwendiger Zulieferer und Abnehmer von Informationen. Damit er in diesem Sinne gut funktionieren kann, müssen ihm die Ausgaben verständlich präsentiert werden und die Eingaben in effektiver und effizienter Weise abgenommen werden. Man hat hier bereits an interaktive, allerdings noch

nicht besonders leicht bedienbare Systeme zu denken. Als Mensch-Maschine-Systeme dieser Art sind z.B. die großen Versicherungs- oder Buchungssysteme mit ihren Masken und Transaktionscodes zu sehen, die vor 10-15 Jahren installiert wurden und heute immer noch auf diese Weise bedient werden. Die Entwickler befinden sich in der Rolle der Systemanalytiker und -planer, die maschinelle und menschliche Tätigkeiten neu planen, aufeinander abstimmen und kontrollieren müssen.

Das Leitbild vom Computer als *Dialogpartner* wird sowohl von Anhängern der Künstlichen Intelligenz als auch von Software-Ergonomen vertreten. Der Umgang mit dem Computer basiert auf Sprache. Ein- und Ausgaben sollen so gestaltet werden, daß sie den Kommunikationsgewohnheiten des menschlichen Dialogpartners entgegenkommen. Insbesondere müssen Formen der Metakommunikation gegeben sein.
Das KI-Verständnis dieses Leitbildes zielt darauf ab, dem Computer ein möglichst menschenähnliches Sprachverhalten zu geben. Er soll natürlich-sprachliche Eingaben verstehen bzw. soweit verarbeiten können, daß er durch Rückfragen weitere verwertbare Angaben erlangen kann. Der Computer soll dabei freundlich und partnerschaftlich-hilfsbereit erscheinen. Dem menschlichen Dialogpartner wird die Rolle des Hilfebedürftigen zugewiesen, von dem ausschließlich angenommen wird, daß er der natürlichen Sprache einigermaßen mächtig ist. Er muß sich nicht darüber klar werden, wie weit die sprachlichen Fähigkeiten des Computers wirklich gehen, sondern darauf vertrauen, daß der Computer ihn richtig versteht oder zumindest lange genug nachfragt. Der Softwareentwickler wird in zweifacher Weise zum Modellierer: Zunächst muß er sich ein Modell machen von den Absichten und dem zu erwartenden sprachlichen Verhalten des menschlichen Partners. Dieses Modell geht in die Modellierung des künstlichen Partners ein. Der Computer erhält im Dialog eine scheinbare Eigenständigkeit, die den Entwickler weitgehend in den Hintergrund treten läßt. Hier liegt einer der Hauptunterschiede zur zweiten Variante des Dialogpartner-Leitbildes.
Im Software-ergonomischen Kontext ist das Gestaltungsziel nicht die Menschenähnlichkeit. Dem am Computer arbeitenden Menschen soll bewußt gemacht werden, daß der Computer vom Softwareentwickler ein sehr eingeschränktes und formales Kommunikationsverhalten "einprogrammiert" bekommen hat. Dieses Verhalten sollte möglichst bekannten oder durchschaubaren Konventionen folgen, an die ein Mensch sich leicht anpassen kann. Die Fachsprache des menschlichen Dialogpartners soll verwendbar sein. Der Softwareentwickler bleibt bei dieser Sicht als Verantwortlicher für das Computerverhalten deutlich im Blick. Er ist der eigentliche Partner, mit dem sich der am Computer Arbeitende bei Verständigungsproblemen auseinandersetzen muß. Der Computer erscheint zwar in der Interaktion als virtueller Kommunikationspartner, aber seine Beschränkungen sollen erkannt und nicht versteckt werden. (Genauer zu diesem Leitbild, vgl. Maaß, 1984.)

Das Leitbild des Werkzeugs, oder später der *Werkstatt*, kam im Zusammenhang mit der Entwicklung graphischer Benutzungsoberflächen auf, als die Interaktion mit dem Computer nicht mehr allein durch Tastatureingabe sprachlicher Strings, sondern zunehmend durch "direkte Manipulation" von graphischen Objekten stattfand. Nunmehr geht es darum, im Computer eine praktische Arbeitsumgebung für Spezialisten einzurichten. Der Computer stellt Raum, Objekte und Werkzeuge zu ihrer Bearbeitung möglichst anschaulich und praktisch handhabbar zur Verfügung. Der Arbeitende werkt damit kompetent und eigenständig, er kennt sich aus und richtet sich ein wie ein Handwerker in seiner Werkstatt. Unter diesem Leitbild wird von einer selbstverständlichen Qualifikation des Werkstattbenutzers ausgegangen. Der Softwareentwickler ist der technische Experte, der bei seiner Arbeit stark angewiesen ist auf die Zusammenarbeit mit den späteren Benutzern seiner Produkte, um deren qualifizierte Arbeit angemessen technisch unterstützen zu können. (Die Werkzeug/Material-Metapher wird ausführlich behandelt in Budde & Züllighoven, 1990.)

Ein Leitbild, das erst seit wenigen Jahren Beachtung findet, ist das des *Kooperationsmediums*. (Zu einer sehr frühen Formulierung dieses Leitbildes durch C.A. Petri, vgl. Schelhowe, 1994, in diesem Band.) Hier wird der Computer nicht mehr zur Unterstützung von

Einzelarbeit, sondern von vernetzter Arbeit konzipiert. Er soll menschliche Kommunikation und Kooperation erleichtern. Der gruppenweite Zugriff auf gemeinsame Dokumente, die gezielte Verteilung von Dokumenten werden bequem möglich; durch Protokoll-, Abgleichs- und Erinnerungsfunktionen werden Aktivitäten festgehalten, angestoßen und koordiniert. Der Softwareentwickler wird zum Moderator, Organisator und Helfer ("facilitator") und ist in dieser Rolle angewiesen auf die Vorgaben und die Mitarbeit der Gruppe.

Die beiden allerneuesten Leitbilder heißen Alltagsanreicherung ("computer-augmented environments") und künstliche Welt (VR, "virtual reality").
Als *Alltagsanreicherung* wird der Computer in Objekte der realen Welt eingebettet und wird damit selbst weitgehend unsichtbar. Die Interaktion mit ihm läuft nicht mehr explizit über eine Mensch-Computer-Schnittstelle, sondern implizit im Umgang mit den Objekten. Neuartige Wandtafeln speichern z.B. ihre Beschriftung elektronisch und machen sie damit weiterverwendbar. Dabei werden sie weitgehend in gewohnter Weise mit einem Stift beschriftet. Elektronische Erkennungskarten ("active badges"), die von allen Angestellten eines Unternehmens zu tragen sind, entriegeln ihnen automatisch die Türen alle Räume, für die sie zugangsberechtigt sind. Sie bewirken eine automatische Umleitung von eingehenden Telefongesprächen für den Betreffenden auf das jeweils nächstliegende Telefon. Licht und Klimaanlage reagieren auf die (fehlende) Anwesenheit von Personen (bzw. Erkennungskarten) in Räumen durch Ab- oder Anschalten von Anlagen.
Der Mensch bleibt ein in der realen Welt Agierender, der im Umgang mit vertrauten realen Objekten auch solche zusätzlichen elektronischen Funktionen anstößt oder steuert. Die Rolle des Softwareentwicklers ist schwer zu benennen. Er ist technischer "Alltagsverbesserer", Zauberer im Hintergrund. Ein systematischer Entwicklungsprozeß unter Beteiligung

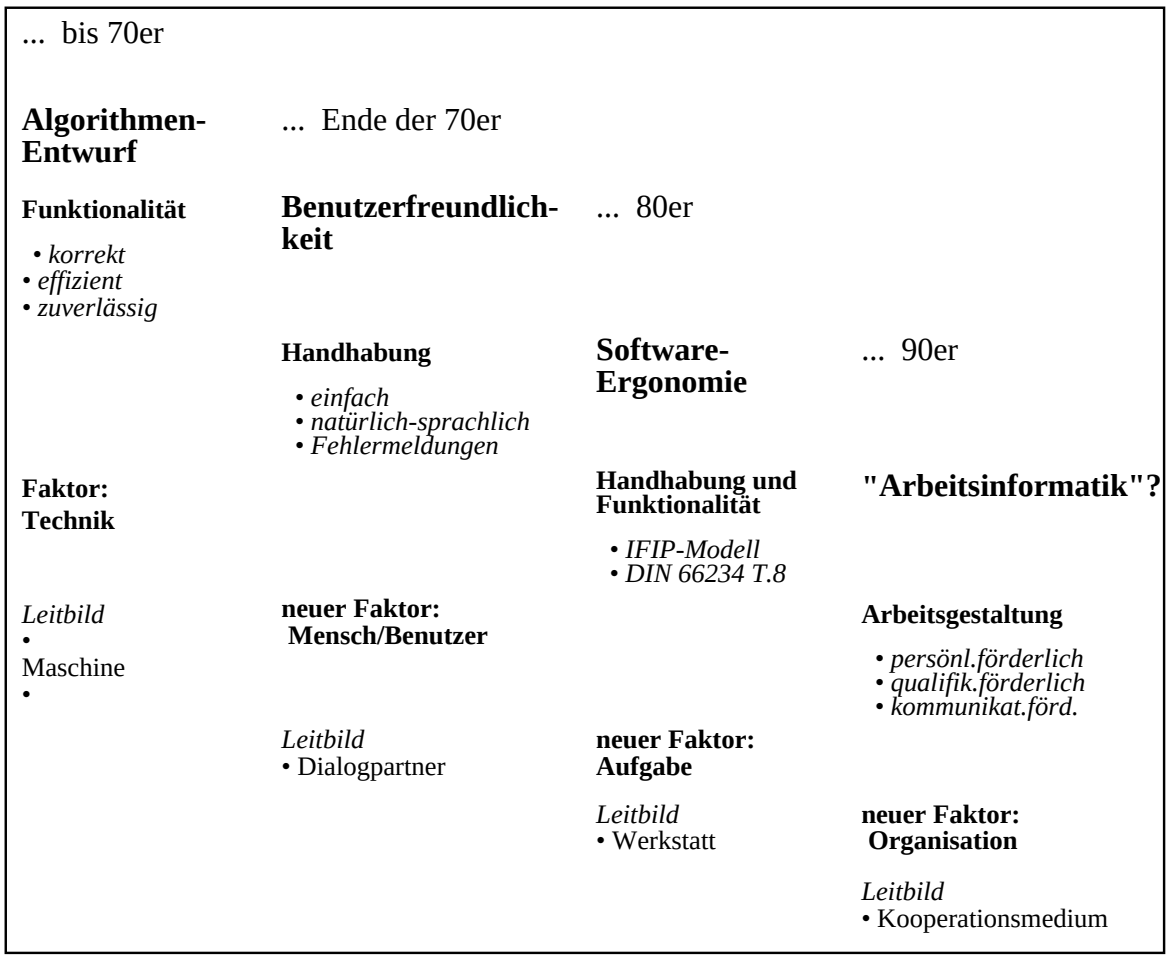

Abb. 3 Entwicklung der Softwareergonomie und zugehörige Leitbilder

von Betroffenen ist unter diesem Leitbild zwar vorstellbar, allerdings wird die Technik derzeit noch ausschließlich in Forschungslabors entwickelt und erprobt. (Vgl. Themenheft der Communications of the ACM, Juli 1993.)

Unter dem letzten hier zu behandelnden Leitbild "*künstliche Welt*" *(VR)* werden mit dem Computer neue zusätzliche Realitäten geschaffen. Im extremen Gegensatz zum Falle der "Alltagsanreicherung" verschwindet für den in VR Agierenden die reale Welt fast völlig. Einem Taucher gleich in einen Anzug gehüllt und von seiner Umwelt weitgehend abgeschlossen durchstreift er künstliche (Bilder-)Welten. Seine Körperbewegungen werden als Eingaben zur Steuerung seiner Bewegungen in der künstlichen Welt interpretiert. Hier wird der Softwareentwickler zum Modellierer von Welten; meist handelt es sich dabei um Nachbildungen der realen Welt. (Die Rolle des Realitätskonstrukteurs kann Softwareentwicklern allerdings generell zugeschrieben werden, vgl. Floyd, 1992.)

Die beschriebenen Leitbilder wurden hier in der ungefähren zeitlichen Reihenfolge ihres Aufkommens behandelt. Ihre Entstehung verlief weitgehend parallel zur Entwicklung der Software-Ergonomie und dem Wandel ihrer Schwerpunkte (vgl. oben, Abb. 3). Die beiden jüngsten Leitbilder sind allerdings nicht in dieser Weise zuzuordnen. Je nach ihrer inhaltlichen Ausrichtung tragen sie zur Arbeitsgestaltung bei oder beeinflussen das tägliche Leben. Allerdings löste nicht ein Leitbild jeweils klar das andere ab, sondern sie existieren alle nebeneinander. Reale Systeme verkörpern heute selten ein einzelnes Leitbild, sondern sie tragen meist Züge verschiedener Leitbilder. So geben z.B. auch direkt manipulative Systeme sprachliche Fehlermeldungen; auch "mediale" Software muß eine Interaktionsschnittstelle zum Benutzer haben, die nach verschiedenen Leitbildern gestaltet sein kann. Nake und Schelhowe (1993) sprechen deshalb vom "instrumentalen Medium" bzw. "medialen Instrument" Computer.

Im folgenden letzten Abschnitt sollen die sich mit den Leitbildern wandelnde Rolle von Softwareentwicklern und die daraus entstehenden Qualifikationsanforderungen diskutiert werden.

3. Vom Anspruch der Leitbilder

Alle genannten Leitbilder wurden im Nachherein formuliert: Am Anfang stand die Technikgenese, danach erst wurde ein verallgemeinernder Begriff für den speziellen Ansatz, die besondere Sicht oder die zugrundeliegende Idee geprägt. Der Begriff "direct manipulation" z.B. wurde erst 1983 von Ben Shneiderman aufgebracht, obwohl die betreffende Interaktionstechnik schon 20 Jahre vorher entwickelt worden war und seit Anfang der 80er-Jahre auf dem Markt zur Verfügung stand. Neuere Leitbilder wie "virtual reality" oder "computer-augmented reality" wurden mit sehr viel geringerem zeitlichen Abstand nach der Technikentwicklung formuliert.

In der Softwareentwicklung haben Leitbilder zunächst eine Bildfunktion (i.S.v. Marz & Dierkes, 1993). Ihre nachträgliche Benennung und Diskussion aktiviert Vorstellungen und Bilder, sie fordert dazu heraus, die Dinge unter verschiedenen Perspektiven zu sehen und damit unterschiedliche Aspekte wahrzunehmen und neu zu gewichten. Dies führt dazu, daß in die benannten Richtungen weitergedacht wird und die sich ergebenden Analogien als Anregungen zu neuer Technikentwicklung genutzt werden. So ergibt sich dann auch eine Leitfunktion für Forschung und Entwicklung; allerdings besteht kein deterministischer Einfluß. Griffige Leitbilder dienen natürlich auch der Vermarktung von Ideen.

Nicht alle hier präsentierten Leitbilder liegen im Interesse benutzergerechter Softwaregestaltung. Dies läßt sich leicht anhand der Rollenzuschreibungen an die Benutzer erkennen. Handlungsspielräume, Kompetenz und Weiterentwicklungsmöglichkeiten, wie sie die Software-Ergonomie fordert, werden Benutzern in der zweiten Variante des Dialogpartner-Leitbildes, im Werkstatt- und Medien-Leitbild und auch im Falle der "Alltagsanreicherung" zugestanden. Maschinen- und System-Leitbild wirken stark einschränkend. Das Leitbild "künstliche Welt" scheint zwar neue Dimensionen zu erschließen, beengt den

Menschen aber m.E. ganz besonders: Die Mensch-Computer-Schnittstelle legt sich gleich einer Haut um den Menschen, so daß keine seiner Bewegungen mehr uninterpretiert bleibt.

Software-ergonomische Leitbilder legen immer einen partizipativen Softwareentwicklungsprozeß nahe, in dem Entwickler und Anwendungsfachleute miteinander und voneinander lernen. Ein angemessener Dialogpartner kann z.B. nur realisiert werden, wenn der Softwareentwickler sich eingehend mit der Begriffswelt und Fachsprache der Softwareverwender beschäftigt. Das Werkstatt-Leitbild erfordert eine gründliche Auseinandersetzung mit den Arbeitsgegenständen und Arbeitsvollzügen der zu unterstützenden Fachkraft. Die Gestaltung des Computers als Kooperationsmedium verlangt vom Entwickler Einsicht in optimale Formen kooperativer Arbeit und eine intensive Beschäftigung mit den eingeschwungenen Arbeitsvollzügen der zu unterstützenden Zielgruppe. Ohne Kontakt zu den Betroffenen läuft der Entwickler als "Alltagsverbesserer" Gefahr, innovative und technisch anspruchsvolle Lösungen anzubieten, die gar nicht im Interesse der Betroffenen sind. Eine Zusammenarbeit mit den späteren Nutzern der Technik ist hier auch deshalb besonders wichtig, weil die weitgehend implizite Interaktion mit dem Computer den Benutzern gar nicht mehr bewußt zu werden braucht und der Computer damit leicht zur versteckten Kontrollinstanz werden kann.

Fassen wir zusammen: Jedes der genannten Leitbilder setzt seine besonderen Gestaltungsschwerpunkte und birgt gewisse Annahmen über die Benutzer. Jedes Leitbild stellt andere Ansprüche an die Entwickler. Die Informatik-Ausbildung sollte ihre Absolventinnen und Absolventen auf diese vielfältigen Ansprüche vorbereiten. Ist sie dazu in der Lage?

In der Informatik dominieren heute noch technikzentrierte Leitbilder. Die etablierten Verfahren der System- und Anforderungsanalyse und der Softwareentwicklung (Software "Engineering") sind darauf zugeschnitten. Unter Leitbildern, die den Computer als Maschine oder System sehen, grenzen sich die Entwickler klar von den Softwarebenutzern ab. Für eine (i.S. dieser Leitbilder) optimale Softwaregestaltung sind sie nicht auf eine Zusammenarbeit mit ihnen angewiesen. Aber schaffen sie damit benutzergerechte Software? P.J. Denning schreibt in seinem lesenswerten Artikel "Educating a New Engineer": "Many of the skills our students lack are in the areas of communication and collaboration, rather than technologies." (Denning, 1992, S. 88) Dabei müßte ein erfolgreicher Ingenieur nach Dennings Einschätzung "in addition to being competent in engineering, be a skilled listener for concerns of customers or clients, be rigorous in managing commitments and achieving customer or client satisfaction, and be organized for ongoing learning." Und er schließt: "This is not the kind of engineer we normally train." (ebd.)

Detaillierte Analysen von erfolgreichen wie von gescheiterten großen Softwareentwicklungsprojekten geben Denning Recht. Sie kommen übereinstimmend zu dem Ergebnis, daß besonders gute Softwareentwickler sich gerade dadurch auszeichnen, daß sie zuhören, kommunizieren und kooperieren können und immer bereit sind dazuzulernen (deMarco & Lister, 1987; Pressman & Herron, 1993). Softwareentwickler dürfen sich also nicht weiter als unabhängige Experten einschätzen, die aufgrund ihrer technischen Kompetenz alle Gestaltungsentscheidungen frei treffen können. Sie müssen sich von den entsprechenden Leitbildern lösen.

Denning schlägt vor, das Engineering-Curriculum so umzuorganisieren, daß die Studierenden nicht nur technische Inhalte, sondern auch Formen der zielbewußten und verantwortungsvollen Arbeit, insbesondere der partnerschaftlichen Zusammenarbeit, lernen und sich in der verständlichen Ergebnispräsentation für andere üben.

Denning's kritische Einschätzung der Ausbildungssituation US-amerikanischer Ingenieure und der dringend gebotenen Veränderungen im Curriculum läßt sich ohne weiteres auf die Informatik in Deutschland übertragen. Informatiker in der Softwareentwicklung scheitern nicht an zu geringen "Kern-Informatik"-Kenntnissen, sondern an ihrer Unfähigkeit, die bestehenden Arbeitsstrukturen und -abläufe zu erschließen und sich partnerschaftlich mit den Betroffenen zu verständigen, zu deren Unterstützung Software zu entwickeln ist. Sie müssen lernen, die Standpunkte der späteren Softwarebenutzer zu respektieren, ihre Kompe-

tenz und Kooperation zu schätzen und ihr Anliegen zu dem eigenen zu machen.

Wenn die Informatik ihren AbsolventInnen eine praxisgerechte Ausbildung bieten und sie auf eine kooperative, verantwortungsvolle Berufstätigkeit vorbereiten will, muß sie sich von ihrem überwiegend technikzentrierten Selbstverständnis lösen und zu einer Gestaltungswissenschaft entwickeln (vgl. Rolf, 1992). Angemessene Gestaltung von Technik basiert auf einem Verständnis des Gesamtkontextes, in den sie eingepaßt werden soll. Als Softwareentwickler und Arbeitsgestalter müssen Informatiker und Informatikerinnen individuelle, aufgabenspezifische, technische und organisatorische Gegebenheiten beachten. Dazu benötigen sie außer ihrem technischen Spezialwissen zusätzliche Fachkenntnisse anderer Art und müssen offen sein für die Zusammenarbeit mit anderen Fachleuten. Die gemeinsame Arbeit an festumrissenen Projekten muß bereits im Studium trainiert werden.

Nur wer auf eine gleichberechtigte Zusammenarbeit vorbereitet ist und sich ehrlich darauf einläßt, hat die Chance, erfolgreich benutzergerechte Software zu entwickeln. Wenn unsere AbsolventInnen die dafür nötigen Fähigkeiten und Einstellungen erwerben sollen, muß das Informatik-Curriculum entsprechend modifiziert und erweitert werden. Eine kritische Auseinandersetzung mit verschiedenen möglichen Leitbildern in der Softwareentwicklung und den damit verbundenen Annahmen bzgl. der Gestaltungsziele, der Rollen von Benutzern und Entwicklern und der Art ihrer Beziehung untereinander liefert eine Motivation für diese Neuorientierung.

Dank

Etliche der dargestellten Gedanken entstanden in Diskussionen mit meinem Kollegen Horst Oberquelle. Ihm, Ralf Klischewski und Arno Rolf danke ich auch für die kritische Kommentierung früherer Versionen dieses Beitrages.

Literatur

Budde & Züllighoven, 1990
Reinhard Budde, Heinz Züllighoven. Software-Werkzeuge in einer Programmierwerkstatt. Berichte der Gesellschaft für Mathematik und Datenverarbeitung, Nr.182, Oldenbourg, München, 1990.

Communications of the ACM, 1993
Computer Augmented Environments: Back to the Real World, Communications of the ACM, 36, 7, July 1993.

Dehning, Essig & Maaß, 1978
Waltraud Dehning, Heidrun Essig, Susanne Maaß. Zur Anpassung virtueller Mensch-Rechner-Schnittstellen an Benutzererfordernisse im Dialog. Fachbereich Informatik, Universität Hamburg, Bericht Nr. 50, 1978.

DeMarco & Lister, 1987
Tom DeMarco, Timothy Lister. Peopleware. Productive Projects and Teams. Dorset, New York, 1987.

Denning, 1992
Peter J. Denning. Educating a New Engineer. Communications of the ACM, 35, 12, 1992, S.83-97.

DIN, 1988
DIN 66 234, Teil 8: Bildschirmarbeitsplätze. Grundsätze ergonomischer Dialoggestaltung. Beuth, Köln, 1988.

Dzida, 1983

Wolfgang Dzida. Das IFIP-Modell für Benutzerschnittstellen. Office Management, Sonderheft 1983, S.6-8.

Floyd, 1992
Christiane Floyd. Software Development as Reality Construction. in: Christiane Floyd, Heinz Züllighoven, Reinhard Budde, Reinhard Keil-Slawik (Eds.). Software Development and Reality Construction. Springer, Berlin, 1992, S.86-100.

Griese, 1982
Joachim Griese. Software-Ergonomie. Das Aktuelle Schlagwort. Informatik-Spektrum, 5, 2, 1982, S.124-125.

Grudin, 1990
Jonathan Grudin. Interface. In: Proc. Conf. on Computer-Supported Cooperative Work, CSCW'90, Los Angeles, 1990, S.269-278.

Maaß, 1984
Susanne Maaß. Mensch-Rechner-Kommunikation - Herkunft und Chancen eines neuen Paradigmas. Bericht FBI-HH-B-104/84, Fachbereich Informatik, Universität Hamburg, 1984.

Maaß, 1993
Susanne Maaß. Software-Ergonomie. Benutzer- und aufgabenorientierte Systemgestaltung. Informatik-Spektrum, 16, 4, 1993, S.191-205.

Maaß & Oberquelle, 1989
Susanne Maaß, Horst Oberquelle (Hrsg.). Software-Ergonomie '89. Aufgabenorientierte Systemgestaltung und Funktionalität. Berichte des German Chapter of the ACM, 32, Teubner, Stuttgart, 1989.

Maaß & Oberquelle, 1992
Susanne Maaß, Horst Oberquelle. Perspectives and Metaphors for Human-Computer Interaction. In: Christiane Floyd, Heinz Züllighoven, Reinhard Budde, Reinhard Keil-Slawik (Eds.). Software Development and Reality Construction. Springer, Berlin, 1992, S.233-251.

Marz & Dierkes, 1993
Lutz Marz, Meinolf Dierkes. Leitbildprägung und Leitbildgestaltung. Zum Beitrag der Technikgenese-Forschung für eine prospektive Technikfolgen-Regulierung. Verbund Sozialwissenschaftliche Technikforschung, Mitteilungen Heft 10, 1993.

Nake & Schelhowe, 1993
Frieder Nake, Heidi Schelhowe. Vom instrumentalen Medium. Kooperation in der Software-Entwicklung unter konfligierenden Leitbildern.Universität Bremen, Forschungszentrum Arbeit und Technik, artec-paper 26, Juli 1993.

Oberquelle, 1991
Horst Oberquelle. MCI - Quo vadis? Perspektiven für die Gestaltung und Entwicklung der Mensch-Computer-Interaktion. In: David Ackermann, Eberhard Ulich (Hrsg.). Software-Ergonomie '91. Benutzerorientierte Software-Entwicklung. Berichte des German Chapter of the ACM, 33, Teubner, Stuttgart, 1991, S.9-24.

Pressman & Herron, 1993
Roger S. Pressman, S.Russell Herron. Software-Schock - Risiko und Chance. Hanser, München, 1993.

Rolf, 1992
Arno Rolf. Sichtwechsel. Informatik als Gestaltungswissenschaft. In: Wolfgang Coy et al. (Hrsg.) Sichtweisen der Informatik. Vieweg, Braunschweig, Wiesbaden, 1992, S.33-47

Schelhowe, 1994
Heidi Schelhowe. Computer als Medium in der Neugestaltung der Kommunikationsverhältnisse: Carl Adam Petri und die Entstehung der Medienperspektive. (In diesem Band)

Schönpflug & Wittstock, 1987
Wolfgang Schönpflug, Marion Wittstock (Hrsg.). Software-Ergonomie '87. Nützen Informationssysteme dem Benutzer? Berichte des German Chapter of the ACM, 29, Teubner, Stuttgart, 1987.

Shneiderman, 1983
Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. IEEE Computer, August 1983, S.57-69.

Volpert, 1993
Walter Volpert. Von der Software-Ergonomie zur Arbeitsinformatik. In: Karl-Heinz Rödiger (Hrsg.). Software-Ergonomie ´93. Von der Benutzungsoberfläche zur Arbeitsgestaltung. Berichte des German Chapter of the ACM, 39, Teubner, Stuttgart, 1993, S.51-65.

Erschienen in: Hellige, H.D. (Hrsg.): Leitbilder der Informatik- und Computer-Entwicklung, Tagung der GI-Fachgruppe 'Historische Aspekte von Informatik und Gesellschaft' und des Deutschen Museums, München, 4.-6.10.1993, Tagungsband, artec paper 33, Bremen, 1994

Prof. Dr. Susanne Maaß
seit 1998 am Fachbereich Mathematik/Informatik

Universität Bremen

e-mail: maass@informatik.uni-bremen.de