Software, Ergonomie, Gestaltung?

Falk Höhn

Fachbereich Kunst & Design, FH-Hannover

Zusammenfassung

Der folgende Beitrag stellt den Versuch eines Formgestalters[1] dar, sich in die vielschichtigen Diskussionen von Software-Entwicklern einzumischen und seine Sicht auf das Problemfeld der Software-Entwicklung, besonders ihrer Ergonomie und Gestaltung, darzustellen. Gestalter verfügen über die Erfahrungen, mit komplexen, ihnen mehr oder weniger geläufigen Aufgabenstellungen umzugehen und diese zu lösen. Die gegenwärtige Gestaltungsdiskussion im Kontext der Software-Entwicklung zeigt, daß es Zeit für die Gestalter ist, in diese Diskussion einzugreifen und ihren spezifischen Beitrag einzubringen.

1. Formgestaltung

Die Formgestaltung (das Industriedesign) befaßt sich allgemein mit der Optimierung industriell gefertigter Erzeugnisse hinsichtlich

- utilitärer- (...auf ihre Nutzung gerichteten...)
- faktibilitärer- (...auf ihre Herstellung gerichteten...)
- operationaler- (...mit ihrer Bedienung zusammenhängenden...)
- ökologischer- (...auf ihre Umweltverträglichkeit gerichteten...) und
- ökonomischer- (...auf ihre Verwertung gerichteten...)

Eigenschaften [1]. Dabei ist allerdings in den letzten Jahren ein Trend spürbar, der von der ausschließlichen Gestaltung technischer Produkte wegführt, hin zu einer stärkeren Orientierung auf die Gestaltung von komplexeren Strukturen (z. B. Informations- oder Verkehrssysteme). Der Gestalter übernimmt dabei zunehmend die Rolle des Organisators und Managers solcher Produktentwicklungen. Sein unschätzbarer Vorteil liegt insbesondere darin, daß er unvoreingenommen und ganzheitlich (alle Aspekte berücksichtigend und gegeneinander abwägend) an die Lösung der Probleme herangehen kann, ja muß. Die eigentlichen Fachleute (Konstrukteur, Technologe, Ergonom, Ökonom usw.) sind oftmals schon zu spezialisiert, um noch die Gesamtheit zu überblicken.
Da jedes zu gestaltende Produkt/System eine „Schnittstelle" zum Menschen besitzt, spielt die Ergonomie eine zentrale Rolle für die gestalterische Arbeit und liefert oftmals wesentliche Lösungsansätze bei der Verbesserung von Produkten, da es im Kern meistens um die Verbesserung der „Schnittstellen" geht. Bei der Betrachtung von Systemstrukturen spielt dann die Gestaltung von Arbeitsabläufen oder -prozessen eine wesentliche Rolle. Ergonomie und Gestaltung sollte man trotzdem nicht gleichsetzen!

Seit einiger Zeit rückt nun die Produktklasse „Software" ins Blickfeld der Gestalter. Dafür gibt es zwei wesentliche Gründe:

---

[1] **Formgestalter** - nicht **Designer**, weil der Terminus des „Design" im Zusammenhang mit „Software" zu Irritationen führen kann.

1. „Klassische“ technische Produkte verschwinden immer mehr, da sich in rasanter Geschwindigkeit die Mensch-Maschine-Schnittstellen von „mechanischen“ zu „elektronischen“ Schnittstellen verwandeln. Ein Charakteristikum dabei ist, daß die Schnittstelle von der Maschine räumlich abkoppelbar wird. Zunehmend geht es also primär um die Gestalt der Schnittstelle (Hardware/Software) und nur noch sekundär um die Gestalt der Maschine bzw. des Systems.
2. In immer stärkerem Maße werden komplexe Arbeitsabläufe (, wie man sie beispielsweise im Büro oder in Schaltwarten vorfindet,) in Softwaresystemen nachgebildet. Diese Systeme werden dann eingesetzt, um die Effizienz der „klassischen“ Arbeitsabläufe zu erhöhen bzw. zu rationalisieren. Software-Entwickler stoßen bei der Gestaltung aber sehr schnell an Grenzen, da ihr Repertoire an ergonomischen und besonders an gestalterischen Mitteln und ihr Vermögen, diese bei der Gestaltung der Schnittstelle und ihrer Software einzusetzen, begrenzt sind. Die Schlußfolgerung: Erkenntnisse der Ergonomie und Gestaltung müssen angewendet werden. [2]

Die Probleme bestehen nun darin,

- daß die Gestalter zur Software-Entwicklung recht schwer einen Zugang finden, da ihnen oftmals die sehr speziellen technologischen Grundlagen fehlen;
- daß die Gestalter mit den entwickelten und vorhandenen Methoden wenig anfangen können, weil sie für ihre Zwecke entweder zu allgemein (z. B. KADS) [3;4;5;6] oder zu speziell und eng (z. B. MIKE/KARL) [7; 8] sind;
- daß man sie zu spät zur Gestaltung eines Softwareproduktes hinzuzieht;
- daß den Gestaltern keine adäquaten Werkzeuge zur Verfügung stehen, um ihre Ideen in allen Phasen der Entwicklung einbringen zu können;

Die Software-Entwickler haben inzwischen ein reiches methodisches und technologisches Instrumentarium für sich selbst entwickelt, es jedoch versäumt, eine die Gestalter integrierende Methodik zu entwickeln (; die Gestalter haben bisher aber auch nicht entschieden genug danach verlangt). Verfolgt man die Diskussionen, kann man aber leicht den Eindruck gewinnen, daß die Software-Entwickler alles (Ergonomie und Gestaltung) selbst machen wollen bzw. können, vorausgesetzt man gibt ihnen die entsprechenden Guide-Lines. Das wird so aber nicht funktionieren!

2. Produktentwicklungsprozeß

Auf diesen Problemkreis soll an dieser Stelle nur in der für diesen Beitrag gebotenen Kürze eingegangen werden.
Der Autor geht grundsätzlich davon aus, daß es bei der Entwicklung eines technischen Produktes einen unerläßlichen „Kern“-Entwicklungsprozeß gibt, ohne den eine technische Entwicklung nicht möglich wäre. Daneben gibt es, diesen „Kern“ umklammernde Entwicklungsprozesse, die vor allem die Qualität und die Effizienz der Entwicklung beeinflussen, aber für eine Produktentwicklung (scheinbar!) nicht unbedingt erforderlich sind.

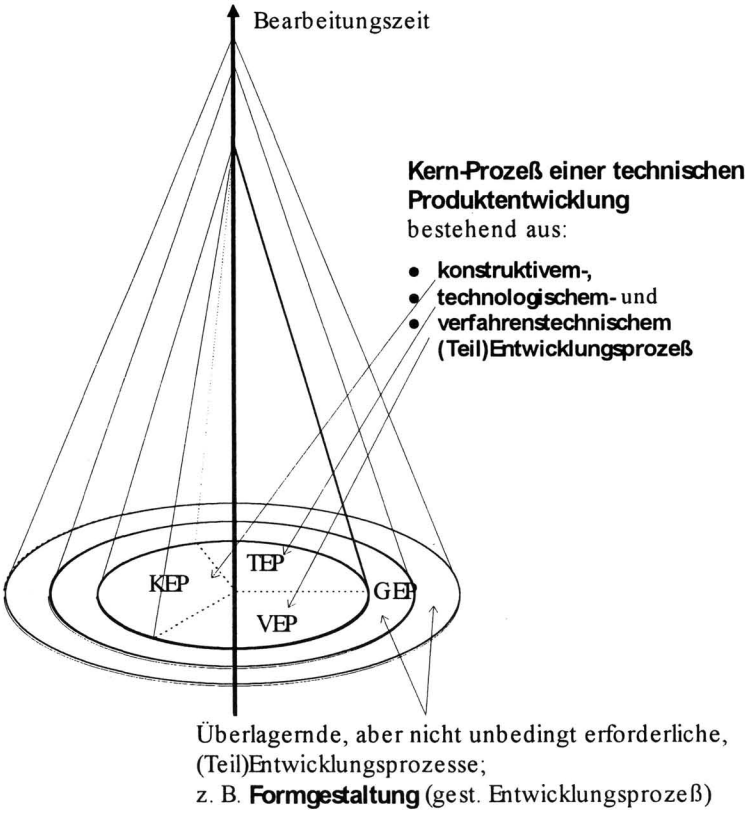

Bild 1 Am Produktentwicklungsprozeß Beteiligte

Bei der Entwicklung „klassischer" Produkte hat sich gezeigt, daß ohne die Integration des gestalterischen Entwicklungsprozesses, sinnvollerweise keine Produktentwicklung stattfinden sollte. Die Integration konnte aber nur gelingen, weil die grundsätzlichen Methoden von Konstrukteuren, Technologen und Gestaltern sehr ähnlich sind und in der Ausbildung der Gestalter entsprechende Inhalte und das notwendige Kontaktwissen vermittelt werden. Nach Meinung des Autors stehen heute die Softwareentwickler vor der Grundsatzentscheidung, Gestalter in ihren Entwicklungsprozeß *sinnvoll* zu integrieren. Bei der Ausbildung von Gestaltern muß dem Rechnung getragen werden. Die Parallelen zur „klassischen" Technik liegen dabei auf der Hand.

Zum gestalterischen Entwicklungsprozeß:

Vielleicht können einige grundsätzliche Ausführungen zur Struktur des gestalterischen Entwicklungsprozesses helfen, den Software-Entwicklungsprozeß in Hinsicht auf die Gestaltung effektiver zu erweitern:

Inzwischen gibt es zahlreiche Vorschläge, wie der gestalterische Entwicklungsprozeß modellhaft zu strukturieren ist. Darauf soll an dieser Stelle nicht näher eingegangen werden.

Man kann sich an der von FRICK [9] vorgeschlagenen Struktur des gestalterischen Entwicklungsprozesses orientieren:

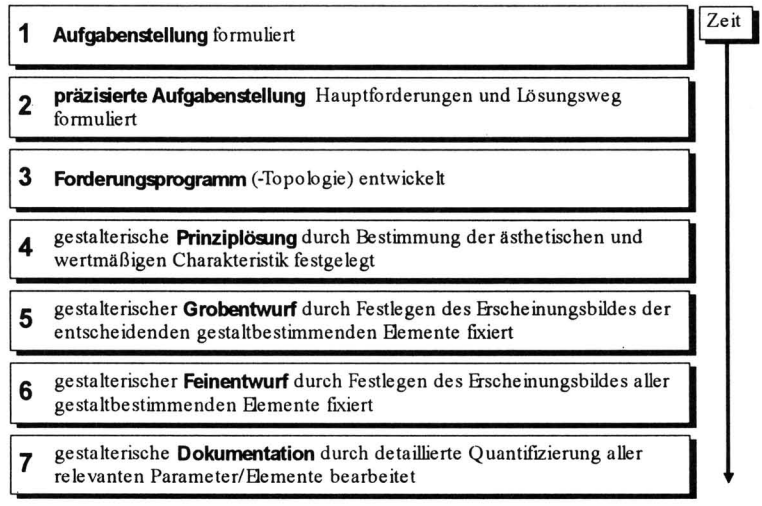

Bild 2: Grundstruktur des gestalterischen Entwicklungsprozesses

Der Gestalter wird mit unterschiedlichen Aufgabenklassen und mit unterschiedlichen Aufgabentypen konfrontiert. Die Aufgabenklassen ergeben sich aus der „Formgebundenheit" in Abhängigkeit von den technischen Freiheitsgraden

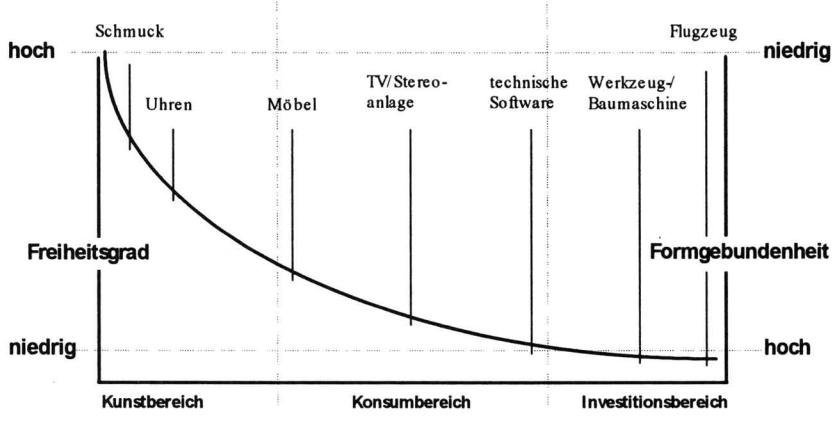

Bild 3: **Freiheitsgrad** in Abhängigkeit von der technisch bedingten **Formgebundenheit** (nach FRICK)

Folgt man diesem Gedanken, könnte man sich eine ähnliche Klassifizierung für Software-Produkte vorstellen:

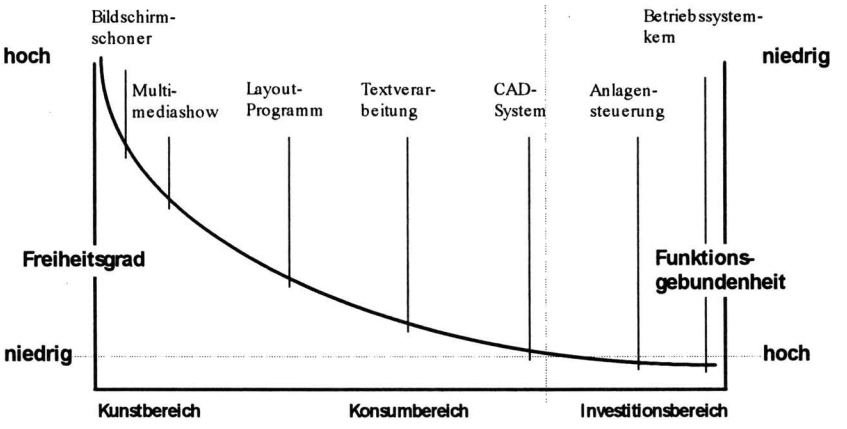

Bild 4: **Freiheitsgrad** in Abhängigkeit von der technisch bedingten **Funktionsgebundenheit**

Je nach Form-/Funktionsgebundenheit ist der für das jeweilige Produkt zu erbringende gestalterische Beitrag innerhalb des Produktentwicklungsprozesses unterschiedlich umfangreich. Aber auch die Geläufigkeit der Aufgabenstellung spielt hinsichtlich des Ergebnisses der gestalterischen Tätigkeit eine nicht zu unterschätzende Rolle. Im Gestaltungsbereich finden wir prinzipiell drei Aufgabentypen [10]:

1. Die **gestalterische Überarbeitung**

   Gesucht wird die Variante bekannter Lösungen.
   Fast alle relevanten Gestaltungsparameter sind bekannt; der Gestalter kennt sich in der Materie aus, seine Vorstellung von der Lösung - das Leitbild - und seine Strategie dazu sind sehr konkret; die zu erwartende Innovation der Lösung ist gering...

2. Die **gestalterische Bearbeitung**

   Gesucht wird eine innovative Lösung mit bekannter Technik (Erfahrungen liegen vor). Der Gestalter hat die Freiheit, grundsätzlich die Struktur, das Wirkprinzip etc. neu zu bearbeiten und mit neue Materialien ins Kalkül zu ziehen; die Aufgabe fordert den Gestalter heraus; die Innovation der Lösung hängt vom Können des Gestalters und seiner Fähigkeit, sich vom Leitbild zu lösen, ab...

3. Die **gestalterische Erfindung**

   Gesucht wird eine völlig neuartige Lösung.
   Der Gestalter entwirft in den vorhandenen objektiven Grenzen ein bis dahin nicht existierendes Objekt. Die Innovation ist garantiert - aber nicht die Optimalität der Gestalt.

   Dieser Aufgabentyp kommt sehr selten vor. Wirklich neue Produkte durchlaufen nach ihrer Erfindung schnelle Zyklen der Variantenbildung und Bearbeitung, bis sich eine optimierte Gestalt/Form, die dann zum Leitbild wird, herausgebildet hat.

Die gesamte Produktklasse „Software" ist im Sinne von (3) neu. Die gegenwärtige Phase der Diskussion dreht sich daher vor allem um die Optimierung der neuen Produkte, und zwar hinsichtlich Nutzungsfreundlichkeit, Komfort und Aufgabenangemessenheit (Ergonomie)

sowie - im kommerziell interessanten Bereich - hinsichtlich Zielgruppenbezug, Differenzierung gegenüber den Konkurrenzprodukten und Ästhetik (Gestaltung).

Eine Ursache dafür, daß die Software-Entwickler beim Problem der Gestaltung der Software nicht vorankommen, liegt aus Sicht des Autors darin begründet, daß (Software)**Gestaltung** und (Software)**Ergonomie** in hohem Maße gleichgesetzt werden. Genau das führt aber letztendlich dazu, daß man eigentlich nicht über Gestaltung diskutiert.

Die (Software)Ergonomie ist für die (Software)Gestaltung unerläßlich, kann aber aus gestalterischer Sicht im Prinzip nur dabei helfen, die gröbsten Schnitzer zu verhindern. Folgt man all ihren Empfehlungen, hat man garantiert nichts falsch gemacht, aber ob das (Software)Produkt auch gut gestaltet ist, steht auf einem anderen Blatt. Genau darin liegt das Problem z. B. der Styleguids.

Daraus resultiert die Uniformität von Software-Oberflächen, die sich ihrerseits wieder aus der geringen Anzahl von Entwicklungswerkzeugen bzw. deren gestalterischer Beschränktheit ergibt. Das ist einerseits unter dem Aspekt der konsistenten Bedienung von Softwaresystemen innerhalb einer Softwareumgebung (angeblich) erwünscht. Andererseits verschenkt man bei der Gestaltung aber die Differenzierung/Optimierung der Gestalt, weil man einem engen Leitbild folgt (oder folgen muß). Aber würde es uns denn im täglichen Leben noch gefallen, wenn plötzlich alle Trinkbehälter (Tassen/ Gläser...) wie Schnabeltassen aussehen, bloß weil diese der Erfüllung der Trinkfunktion am nächsten kommen und für den Mitteleuropäer intuitiv bedienbar sind?

Nun hat sich inzwischen natürlich die Erkenntnis durchgesetzt, daß Software-Produkte auch„gestaltet" werden sollten. Da die Schnittstelle zum Menschen/Nutzer in fast 100 % der Fälle den Bildschirm benutzt und dieser ein grafisches Abbild unterstützt, lag es für die Software-Entwickler selbstverständlich nahe, mit der Gestaltung dieser Schnittstelle Grafik-Designer zu betrauen. Und genau an dieser Stelle liegt ein entscheidender Denkfehler: Grafik-Designer lernen (und tun dies dann auch), Informationen für den visuellen Konsum aufzubereiten. Deshalb liegen neben der Typografie und dem Layout ihre unbestrittenen Stärken z. B. in der Gestaltung von Informationssystemen oder der Informationsaufbereitung in Multimedia-Anwendungen. Sie lernen nicht, (technische) Systeme zu analysieren, nach Ansätzen für deren Verbesserung zu suchen und diese Ideen als Lösungsvorschlag für ein verbessertes technisches Produkt zu synthetisieren.
D. h., der grundlegende Ansatz des Formgestalters ist (auch bei der Gestaltung von Softwareprodukten) völlig anders als der eines Grafik-Designers. Das hat heute zur Folge, daß sich vermeintlich gut gestaltete Software im täglichen Nutzungsprozeß nicht bewährt oder immer noch als unhandlich erscheint. (Man denke nur an CAD-Systeme, für deren Benutzung ein „gestandener" Konstrukteur monatelang geschult werden muß, damit er dannach - zweifellos besser, schneller, schöner - das Gleiche machen kann wie vorher, nämlich Konstruieren; oder an Textverarbeitungssysteme aus dem useability-Labor, deren Funktionalität im täglichen Gebrauch überhaupt nur zu ca. 10 % genutzt - sprich: gebraucht - wird...)

3. Schlußfolgerungen

Die Qualität einer Software wird zukünftig wesentlich durch ihre Gestaltung bestimmt. Die Gestaltung muß deshalb möglichst frühzeitig in den Software-Entwicklungsprozeß einbezogen werden. Dazu kann man auf ein Erfahrungspotential der Gestalter zurückgreifen. Das anzuwendende Prinzip sollte dabei in Abhängigkeit vom Aufgabentyp gewählt werden. Da man

davon ausgehen kann, daß viele der Software-Entwicklungen gegenwärtig noch ein hohes Maß an „Erfindung" darstellen, sollten die gestalterischen Spielräume weit gefaßt sein. Es empfiehlt sich, zeitlich davor gelagerte Gestaltungsstudien erstellen zu lassen, diese zu diskutieren und, sofern sie positiv beurteilt wurden, als Entwicklungsgrundlage im Sinne eines gestalterischen Leitbildes zu benutzen. Nach diesem sich bisher bewährenden Prinzip wird gegenwärtig am ERBUS-Projekt [11] gearbeitet.

Es ist erforderlich, (Software)Gestaltungswerkzeuge zu entwickeln, die einerseits den gestalterischen Spielraum nicht beschränken, andererseits aber auch kompatibel zu den Werkzeugen der Software-Entwickler sind. Damit könnte die Effektivität bei der Entwicklung insgesamt deutlich gesteigert werden. [12]
Notwendig sind aber auch Werkzeuge, mit denen Gestalter „Software-Dummys" generieren und flexibel und schnell bearbeiten können (Software-Modellbau). Zu diesem Zweck werden heute meist Systeme wie Toolbook oder Macromedia-Director benutzt, die dazu eigentlich nur als bedingt geeignet anzusehen sind.
Es fehlt an integrierten Werkzeugen zur Beobachtung des Nutzerverhaltens an funktionierender Software, um Rückschlüsse auf spezifische Eigenheiten der Nutzer ziehen zu können usw.

Weiterhin fehlt gestalterisches Methodenwissen über die Analyse von Softwarestrukturen mit dem Ziel, diese Strukturen - falls erforderlich - gestalterisch an den Nutzer und seinen Arbeitsprozeß anzupassen. Es fehlt an handhabbaren Methoden der Erprobung von Software-Prototypen unter realistischen Einsatzbedingungen.

Es besteht erhöhter Bedarf, die Arbeitsteilung im Software-Entwicklungsprozeß unter Berücksichtigung der Integration von Gestaltern zu untersuchen. Ein solcher Ansatz wird bei dem vom BMFT geförderten Forschungsprojekt WORKS/ERBUS verfolgt [13].

In der Folge müssen neue Ausbildungsinhalte für die Gestalter etabliert werden, um den Bedürfnissen bei der Gestaltung dieser neuen Produktklasse nachkommen zu können.

Literatur

[1] **Oelke, H.; Frick, R.:**
Bestimmung der Funktion von industriell hergestellten Produkten. Halle (1977), F/E-Bericht, HiF.

[2] **Rödiger, K.-H.:**
Anwendungsbereiche lernen voneinander ... und woraus lernen wir? In:Software-Ergonomie '95, Hrsg.: H.-D. Böcker, S. 45 ff, B. G. Teubner, Stuttgart.

[3] **Georges, M. et al.:**
KADS-II. An advanced and comprehensive methodology for integrated KBS development. Report ESPRIT Project Nr 5248. 1992.

[4] **Porter, D.:**
Overview of differences between KADS and CommonKADS. Report ESPRIT Project P5248 KADS-II. 1 June 1992.

[5] **Wielinga, B. J.; Schreiber, A. Th.; Breuker, J. A.:**
KADS: A modelling approach to knowledge engineering.
Report ESPRIT Project P5248 KADS-II, Amsterdam, University of Amsterdam, 1991.

[6] **WIELINGA, B. J.; SCHREIBER, A. TH.; BREUKER, J. A.:**
KADS: A modelling approach to knowledge engineering. Knowledge Acquisition. 4 (1992) 5-53.

[7] **ANGELE, J.; FENSEL, D. and STUDER, R.:**
The Knowledge Acquisition and Representation Language KARL. Research report no. 316, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, May 1995.

[8] **FICHTNER, W.; LANDES, D.; RENTZ O.; RUCH, M.; SPENGLER, T. und STUDER, R.:**
Der MIKE-Ansatz zur Modellierung von Expertenwissen im Umweltbereich - dargestellt am Beispiel des Bauschuttrecyclings. In Proceedings of the 9th International Symposium on Computer Science for Environment Protection (Berlin, September 27-29), 1995.

[9] **FRICK, R.:**
Designmethodik - Eine Einführung für Studierende -. Halle (1982), Lehrmaterial an der HIF-Halle.

[10] **HÖHN, F.:**
Informationstheoretische Untersuchungen zur Rationalisierung des gestalterischen Entwicklungsprozesses durch faktografische Informationssysteme und der daraus resultierende rechnerunterstützte Ergonomie-Sachverhaltspeicher ERGOFAKT. TH Ilmenau, Fakultät für Technische Wissenschaften, Diss. A, 1988.

[11/13] **WORKS/ERBUS** - Verbundprojekt - Methodik der arbeitsorientierten Gestaltung von Wissenssystemen am Beispiel eines Wissenssystems für die Formgestaltung von Investitions- und Konsumgütern. BMBF-Projekt Nr. 01H5014

[12] **SCHWIGON, H.:**
MAKEMENU - Ein Werkzeug für den Programmierer. Halle (1987), Forschungsbericht, HiF Halle.

Adresse des Autors

Prof. Dr.-Ing. Falk Höhn
FH Hannover, Fachbereich Kunst & Design
30419 Hannover
Herrenhäuser Str. 8
e-mail: falk@kd.fh-hannover.de