Peter Gorny

Design software-ergonomischer Benutzungsoberflächen

Tutorial

1 Vorbemerkung

Ein Tutorial zum Design von Benutzungsoberflächen[1] nach dem Stand der Software-Ergonomie ist heute immer noch ein gewagtes Unterfangen: es gibt noch keine durchgehenden gesicherten Methoden, es gibt kaum Werkzeuge und doch überholen die Normen-Gestalter die Forschung und Entwicklung. Es gibt inzwischen DIN-Normen (DIN 66234) und ISO-Normen (ISO 9241) zu dem Thema und sogar eine rechtskräftige Europäische Richtlinie (90/270), in denen Kriterien für die Oberflächen-Gestaltung festgelegt werden, obwohl es erst teilweise Meßmethoden gibt, mit denen man (fertige) Oberflächen bewerten kann, und nur rudimentäre Erkenntnisse, wie man die Kriterien bei der Konstruktion von

(neuen) Oberflächen berücksichtigen kann. Die Anwendung software-ergonomischer Methoden auf Systeme zur Unterstützung kooperativer Arbeit macht die Designentscheidungen noch anspruchsvoller, weil für die Werkzeugunterstützung von Kooperation kaum Forschungsergebnisse vorliegen: die Charakteristika der Aufgaben und die erforderlichen Kenntnisse und Fähigkeiten der Benutzer werden von den Arbeitswissenschaften erst gerade kategorisiert.

Die Software-Ergonomie will die Aufmerksamkeit der Designer auf die Benutzbarkeit[2] von Software-Systemen lenken. Es ist offenbar, daß man Benutzbarkeit nur messen (oder erzeugen) kann, wenn man den Kontext betrachtet, in dem ein Programm als Werkzeug benutzt werden soll. Konkret bedeutet das die Berücksichtigung der Aufgabenstellungen, der Eigenschaften der Benutzer und gewisse Merkmale der Werkzeugbenutzung. Vor einer Gestaltung oder Bewertung einer Benutzungsoberfläche nach den Gütekriterien der Software-Ergonomie muß man also eine Arbeitsorganisations- und Aufgabenanalyse vorgenommen und die Benutzereigenschaften festgestellt haben.

2 Gütekriterien der Software-Ergonomie und des Software-Engineering

2.1 Brauchbarkeit

Die Europäische Richtlinie 90/270 über die Mindestvorschriften bezüglich der Sicherheit und des Gesundheitsschutzes bei der Arbeit an Bildschirmgeräten ist zwar rechtlich bindend, jedoch nur für den deutschen Gesetzgeber. Ein entsprechendes deutsches Arbeitsschutzgesetz und die darauf bauenden Richtlinien der Berufsgenossenschaften sind noch in Arbeit. Deshalb nimmt das Tutorial nur Bezug auf die ISO-Norm 9241 "Ergonomic requirements for office work with visual displays" (Tabelle 1). Diese ISO-Norm hat die DIN 66234 zum Vorbild. Von den für die Softwaregestaltung wesentlichen Teilen sind erst zwei relativ weit entwickelt: Teil 2 "Guidance on task requirements" ist verabschiedet und Teil 10 "Dialogue principles" steht kurz davor. Die Entwürfe für die übrigen Teile mit Software-Ergonomie-Bezug dürften noch einigen Änderungen in den internationalen Standardisierungsgremien unterworfen werden.

Part 1	General introduction
Part 2	**Guidance on task requirements** *(First edition 1992-06-01)*
Part 3	Visual display requirements
Part 4	Keyboard requirements
Part 5	Workstation layout and postural requirements
Part 6	Environmental requirements
Part 7	Display requirements with reflections
Part 8	Requirements for displayed colours
Part 9	Requirements for non-keyboad input devices
Part 10	**Dialogue principles** *(Draft voting terminates 1994-03-16)*
Part 11	**Guidance on specifying and measuring usability** *(Working paper v 8.8, 7 May 93)*
Part 12	Presentation of information
Part 13	User guidance
Part 14	Menu dialogues
Part 15	Command dialogues
Part 16	Direct manipulation dialogues
Part17	Form filling dialogues

Abb. 1: ISO 9241 "Ergonomic requirements for office work with visual displays (VDTs)"

In den Teilen 2 und 11 wird der Zusammenhang hergestellt zwischen der Brauchbarkeit (usability) des Gesamtsystems und den Benutzern, den Zielen der Aufgaben einschließlich der Aufgabencharakteristika, den zur Verfügung stehenden Werkzeugen und der Arbeitsumgebung. Außerdem wird der Begriff "usability" aufgebrochen in die Teilkriterien "effectiveness", "efficiency" und "satisfaction".

Teil 10 detailliert die Kriterien für die Beurteilung von Dialogen:

- Suitability for the task (Aufgabenangemessenheit)
- Selfdescriptiveness (Selbstbeschreibungsfähigkeit)
- Controllability (Steuerbarkeit)
- Conformity with user expectations (Erwartungskonformität)
- Error tolerance (Fehlertoleranz)
- Suitability for individualization (Individualierbarkeit)
- Suitability for learning (Erlernbarkeit)

Im Tutorial sollen diese Kriterien nicht nur für die Bewertung fertiger Benutzungsoberflächen, sondern für das Design neuer Systeme und ihrer Oberflächen genutzt werden. Dabei müssen sich die Designer bewußt sein, daß mit der Beachtung der Kriterien nicht gewährleistet ist, daß die Oberflächen ästhetisch ansprechend ("schön") oder originell sind. Hier bleibt Intuition und Kreativität gefordert.

2.2 Regeln für die Gestaltung von Benutzungsoberflächen

Neben diesen sehr allgemeinen Kriterien zur Beurteilung der Brauchbarkeit gibt es eine Vielzahl von einzelnen Regeln, die aufgrund von qualitativen oder quantitativen Untersuchungen von Kognitions- oder Arbeitspsychologen aufgestellt wurden. Die größte Sammlung dazu ist von SMITH und MOSIER [18] aufgestellt worden, die inzwischen auch als HyperCard Stack (HyperSAM [11]) zur Verfügung stehen. Die Verknüpfung der Regeln mit bestimmten Benutzer-, Aufgaben- oder Werkzeugbenutzungsmerkmalen wird von den Autoren allerdings nicht vorgenommen.

2.3 Style Guides

Viele Softwarehäuser oder Hersteller haben ihren Mitarbeitern Richtlinien für die Gestaltung von Benutzungsoberflächen an die Hand gegeben. Die bekanntesten sind der IBM-CUA und die Style Guides von Apple (für den Macintosh), Microsoft (für Windows und NT) und von der OSF (für Motif). In Deutschland haben die umfangreichsten Style Guides die Firmen SNI und SAP entwickelt.

Die Style Guides sollen ein spezifisches "look and feel" für einen Benutzungsoberflächen-Typ oder für die Software-Produkte eines Unternehmens (corporate image) sicherstellen. Damit wird das ISO-Kriterium "Erwartungskonformität" wenigstens teilweise berücksichtigt. Aus dieser Zielsetzung heraus ist es erklärlich, daß andere ergonomische Aspekte wie die Merkmale der Benutzer, der Aufgaben oder der Werkzeugbenutzung kaum eine Rolle spielen. Ebenso geben Style Guides keine Hinweise zu ästhetischen Eigenschaften der Oberfläche, auch wenn sie einige "Gesetze" der Gestaltpsychologie aufnehmen.

3 Systematisches Vorgehen beim Design von Benutzungsoberflächen

3.1 Einordnung des Oberflächen-Designs in den Software-Entwicklungsprozeß

Wir gehen davon aus, daß die Gestaltung der Benutzungsoberfläche eines Software-Systems von so grundlegender Bedeutung ist, daß sie prinzipiell in eine der frühen Entwicklungsphasen gehört, bei denen die Anforderungen an das Gesamtsystem zusammengestellt werden. Der hier verwendete Begriff der Phase soll nicht implizieren, daß wir das klassische Phasenmodell des Software-Engineering bevorzugen. Im Gegenteil will ich betonen, daß die hier hilfreichsten Vorgehensmodelle den Benutzer frühzeitig beteiligen und das Prototyping erfordern, z.B. die evolutionären Vorgehensmodelle. Ein *Rapid* Prototyping von Oberflächen eignet sich dazu nicht, weil hierbei die Gefahr besteht, daß bloß schnell einige Masken herunterprogrammiert und die Ergebnisse sorgfältiger Analysen des Anwendungskontexts nicht berücksichtigt werden. Stattdessen ist ein *Slow and Principled* Prototyping zu fordern [9]. Die Zuordnung des Oberflächendesigns zur Phase der Anforderungsdefinition würde bedeuten, daß damit der Prototyp der Oberfläche Bestandteil dieser Definition würde.

Damit wird nicht ausgeschlossen, daß auch in späteren Phasen oder Iterationen der Software-Entwicklung noch Entscheidungen zur Oberfläche getroffen werden können und müssen; es wird jedoch klargestellt, daß diese Entscheidungen an den Kriterien der ISO gemessen - also konkret auf die Merkmale der Benutzer, der Aufgaben und der Werkzeugbenutzung bezogen werden müssen.

3.2 MUSE – Eine Methode zum Entwurf von Benutzungsoberflächen

Um das systematische Design von Benutzungsoberflächen zu ermöglichen, wurde in Oldenburg ein Vorgehensmodell entwickelt, das im folgenden skizziert werden soll. Wie immer ist damit beabsichtigt, die Entscheidungskomplexität zu reduzieren und Einzelkriterien zu geeigneten Augenblicken während des Designprozesses bewußt zu machen. Als Gütemaß werden dabei die ISO-Dialogprinzipien herangezogen, allerdings jetzt als Maß für den Aufwand, der bei der Entwicklung der Software zu ihrer Erfüllung betrieben werden muß. Man kann dieses Vorgehen auch als "inkrementelle Evaluation" bezeichnen.

Wenn z.B. ein "fachlich kompetenter" Benutzer eine Software-Funktion "häufig" benutzt, so muß ein relativ hoher Aufwand betrieben werden, um das Kriterium "Erwartungskonformität" zu erfüllen, während für die "Erlernbarkeit" ein geringerer Aufwand erforderlich ist. Dagegen müßte an Arbeitsplätzen mit häufig wechselndem "ungeschultem" Personal die "Erlernbarkeit" sehr viel höher gewichtet werden.

MUSE (Method for User Interface Engineering [9]) gliedert die Design-Entscheidungen in vier Sichten, die beliebig oft eingenommen werden können.

Konzeptuelle Sicht

Bei der Konzeptuellen Sicht kommt es darauf an, daß die Analyseergebnisse des Arbeitssystems umgesetzt werden in Designentscheidungen über die erforderlichen Werkzeugfunktionen, die dabei in vier Kategorien (Anwendungs-, Steuer-, Adaptier- und Metafunktionen) geordnet werden. Die Entscheidungen werden sich dabei einerseits auf die Erfahrungen mit ähnlichen Systemen (application domain knowledge) stützen und andererseits die Auswahl der Funktionen abhängig machen von den konkreten Merkmalen der zu erledigenden Aufgaben, der Benutzer (von ihren Individualeigenschaften in "Rollen" abstrahiert) und der Werkzeugbenutzung. Dieses Tripel {Aufgabe, Rolle, Werkzeugbenutzung} bestimmt somit die software-ergonomischen Anforderungen an die Oberfläche.

Es hat sich hier als hilfreich erwiesen, für die Analyse des Arbeitssystems das Verfahren der Objektorientierten Analyse (OOA) ([4]; [5]; [16]) zu verwenden. Dabei stehen Arbeitsgegenstände ("objects") im Zentrum der Aufmerksamkeit, denen jeweils Bearbeitungsverfahren ("methods") zugeordnet werden (Beispiel: Arbeitsgegenstand "Fahrauftragsformular", Bearbeitungsverfahren: "Formular ausfüllen"). Wenn nun das "Formular" im Rechnersystem als Maske realisiert wird, so benötigt man statt Bleistift und Radiergummi eine Reihe von "Werkzeugfunktionen" im Rechner.

Gerade bei einer kooperativen Organisationsform bietet dieser Zugang Vorteile gegenüber der im Software Engineering häufig verwendeten funktionalen Beschreibungsmethode. Wenn zum Beispiel in einem "virtuellen Unternehmen" die Just-in-Time-Logistik kooperativ zwischen den Disponenten der verschiedenen realen Unternehmen betrieben wird, so fällt es leicht, den Zugriff auf die gemeinsamen Objekte "Touren-Plan" und "Fahrauftrag" zu entwerfen. Wegen der in diesem Beispiel angenommenen Kooperation wird die Erfüllung der Kriterien "Steuerbarkeit", "Aufgabenangemessenheit" und "Erwartungs-

konformität" besondere Bedeutung haben. Da möglicherweise die Benutzeranalyse ergeben hat, daß die Rolle des Disponenten von selten wechselnden, fachlich erfahrenen Benutzern ausgeübt wird, ist "Erlernbarkeit" und "Selbstbeschreibungsfähigkeit" von geringerem Gewicht.

Als Ergebnis der konzeptuellen Sicht liegt eine Liste von Werkzeugfunktionen vor, die bei dem genannten Beispiel für die Bearbeitung eines "Fahrauftrags" erforderlich sind.

Strukturelle Sicht

In der zweiten Sicht werden Entscheidungen über Dialogformen und die Dialogstruktur (mit ihren temporalen Bedingungen) getroffen. Als Dialogformen stehen die Grundformen Dateneingabedialog, Kommandodialog, Auswahldialog und Direkte Manipulation sowie Kombinationsformen wie z.B. Maskendialog zur Verfügung. Man könnte die Werkzeugfunktionen zum Eintragen von Text in ein bestimmtes Feld statt über einen Kommandodialog durch einen Auswahldialog aktivieren. Es könnte jetzt zum Beispiel auch festgelegt werden, daß sie bei simultaner Arbeitsweise jeweils nur einem der aktiven Benutzer zur Verfügung stehen, das heißt bei den anderen Benutzern dann inaktiv geschaltet werden.

Die oben genannten temporalen Bedingungen in der Dialogstruktur erlauben Entscheidungen über die erlaubten Reihenfolgen von Arbeitsschritten während der Aufgabenerledigung, der Anwendung von Bearbeitungsverfahren und des Aufrufs von Werkzeugfunktionen. (Es ist z.B. unsinnig, einen Text in einem noch nicht gefüllten Textfeld markieren zu wollen, um ihn anschließend zu kopieren: durch eine sichtbar gemachte Deaktivierung der Werkzeugfunktionen "cut" and "copy" würde ein Benutzer einen entsprechenden Hinweis erhalten können.)

Ergebnis der Strukturellen Sicht sind Dialogformen für die Werkzeugfunktionen und jeweils die Vorbedingungen für ihre Anwendbarkeit auf ein Objekt bzw. auf einen Teil eines Objekts.

Konkretisierungssicht

Ziel dieser Sicht ist die Festlegung konkreter Erscheinungsformen für die Dialogformen. So stehen zum Beispiel für die Dialogform "Auswahldialog" die Interaktionsarten (und Widgets) "Menü", "Listboxes", "Radio-Buttons", "Checkboxes" und Funktionstasten zur Verfügung, zum Teil noch in unterschiedlichen Ausprägungen (z.B. Piktogramm-Menü oder Text-Menü, pop-up, pull-down

oder statisch). Um hier für das Oberflächendesign ergonomische Grundlagen zu finden, sind etwa arbeits- oder kognitionspsychologische Forschungsergebnisse wie in den Regeln von SMITH und MOSIER heranzuziehen. (Beispiel: Ein Menü sollte nicht mehr als 10 Items haben. Stattdessen könnte man eine hierarchisch gestufte Menüfolge oder eine Liste verwenden oder - falls jeweils nur einige wenige Items in einem Arbeitskontext ausgewählt werden - die häufig verwendeten Items zusätzlich an den Kopf des Menüs kopieren.)

Für den oben erwähnten kooperativen, u.U. simultanen Arbeitsprozeß der Tourendisposition tritt die Frage auf, wie man den Benutzern verdeutlicht, daß eine von ihnen gerade bestimmte Werkzeugfunktion oder ein Objekt "blokkiert". Um dieses auf der Oberfläche zu präsentieren, könnten die Funktionen entweder nicht mehr im Auswahlangebot erscheinen oder "graugeschaltet" sein. Das software-ergonomische Wissen muß dazu ausgewertet werden, um festzustellen, welche der beiden Präsentationsformen für die gegebenen Benutzer und Aufgaben geeigneter ist. Die gleiche Präsentationsform könnte auch für die wegen anderer temporaler Bedingungen nicht aktivierbaren Werkzeugfunktionen verwendet werden, falls nicht auch der Grund der Nichtaktivierbarkeit durch die Präsentationsform signalisiert werden soll.

Ergebnis der dritten Sicht ist somit die Festlegung konkreter Erscheinungsformen für die in der zweiten Sicht festgelegten Dialogformen.

Realisierungssicht

In der vierten Sicht soll ein Prototyp der Benutzungsoberfläche erstellt werden. Dazu müssen die bisher ausgewählten Elemente der Benutzungsschnittstelle vollständig beschrieben und positioniert werden. Ein User Interface Prototyping Tool oder ein User Interface Management System ist dazu gut geeignet.

Eine Anbindung der Schnittstelle an das Anwendungssystem kann in einer frühen Phase des Entwicklungsprozesses natürlich nicht erfolgen, stattdessen müssen die Zugriffe auf die Daten und Funktionen der Applikation als "Mock-up" implementiert werden. Dieser Prototyp wird anschließend mit Anwendern und Benutzern evaluiert, um Grundlage für eine neue Version zu schaffen.

Während der Realisierungssicht müssen zum Beispiel die Größe, Form, Farbe und Plazierung von Fenstern, Listboxes, Dialogboxes usw. ebenso festgelegt werden, wie die Wortwahl für deren Beschriftungen und Menüs. Dazu gibt es eine Vielzahl von Regeln, die aus der Gestaltpsychologie qualitativ abgeleitet oder aufgrund von Experimenten entwickelt wurden und sich sowohl bei SMITH und MOSIER wie auch in den verschiedenen Style Guides wiederfinden. Allge-

meine Kriterien werden auch im Teil 12 der ISO9241 aufgeführt werden. Die Gestaltung der Piktogramme macht besondere Schwierigkeiten, weil sie sowohl leicht erkennbar wie einen Bezug zu der realen Arbeitsumgebung der jeweiligen Benutzer haben sollen, um den Benutzern einen längeren Schulungsprozeß ersparen zu können.

3.3 Verallgemeinerung der unter MUSE entstehenden Oberflächen

Theoretisch müßte bei einem Vorgehen nach MUSE für ein Anwendungssystem bei jedem Tripel {Rolle, Aufgabe, Werkzeugnutzung} wegen der Unterschiede in den Merkmalen eine spezielle Benutzungsoberfläche entstehen. Das ist weder ökonomisch zu vertreten noch in einem Unternehmen im praktischen Einsatz der Software durchzuführen. Aus diesem Grunde wird der Designer schon früh im Designprozeß ähnliche Anforderungen zusammenfassen und so die Anzahl unterschiedlicher Oberflächen reduzieren. Die Vereinigungsmenge der Merkmalsmengen mehrerer Tripel führt zu Wertebereichen für jedes einzelne Merkmal: wenn zum Beispiel für mehrere Benutzer die Werkzeugnutzung *häufig bis selten* auftreten kann, so ist entsprechend die gemeinsame Oberfläche so zu entwerfen, daß beide Benutzungshäufigkeiten entsprechend unterstützt werden (etwa wie üblich, für die häufige Nutzung sogenannte Shortcuts durch Kommandokürzel, für die seltene eine Menüführung). Der Extremfall "eine Benutzungsoberfläche für alle Benutzer und Aufgaben" führt dann zur Standardsoftware. Es sei jedoch angemerkt, daß dieser Extremfall eigentlich nie eintritt, denn auch die Entwickler von Standardsoftware machen sich - oft unbewußt - ein Bild von den Benutzern ihres Programms und den Aufgaben, die damit bearbeitet werden sollen.

Bei Software, die in verschiedenen Ländern eingesetzt werden soll, tritt ein weiteres Problem auf - auch bei Software mit sehr spezifischen Anwendungsbereichen: die Internationalisierung der Benutzungsoberfläche. Es ist längst bekannt, daß es eine Abstufung von Maßnahmen der Übertragung eines Programmes in eine andere Sprachwelt geben muß, beginnend mit einer reinen Übersetzung der Texte und Bezeichnungen auf der Oberfläche über eine Anpassung an andere Maßsysteme, alphabetische oder ikonische Schriften, Schreibarten (links-rechts/rechts-links), andere Zähl- und Bezeichnungsweisen bis hin zu in anderen Kulturen üblichen Metaphern und Piktogrammen, gar nicht zu

reden von anderen rechtlichen oder technischen Vorschriften im Anwendungsbereich, die sogar eine Änderung der Anwendungsfunktionen erfordern.

Bei einer Analyse des Kontextes, in dem die Software eingesetzt werden soll, müssen also auch die kulturellen Unterschiede berücksichtigt werden, um daraus Gestaltungsmaßnahmen ableiten zu können.

4 Grenzen von MUSE und Ausblick

Die im Tutorial vorgestellte Methode MUSE soll die Entscheidungskomplexität beim Oberflächendesign reduzieren und die Einflußfaktoren auf die Benutzbarkeit eines Anwendungssystems bewußt machen. Bei dem Vorgehen nach MUSE wird vorausgesetzt, daß die an der Gestaltung der Benutzungsoberfläche Beteiligten auch die Kriterien kennen und die Zusammenhänge zwischen den Merkmalswerten und den Designentscheidungen nachvollziehen können. Dieses breite Wissen über die Arbeitswissenschaften, die Arbeits-, Kognitions- und Gestaltpsychologie sowie über das Software-Engineering kann weder beim Anwender noch bei den Software-Entwicklern vorausgesetzt werden. Daher ist die Methode praktisch nicht einsatzfähig, es sei denn, die Software-Entwicklung erfolgt in multidisziplinären Teams.

Um dem Problem abzuhelfen, werden in einigen Forschungsgruppen Werkzeuge entwickelt, die eine wissensbasierte Unterstützung des Designprozesses bieten sollen. Die Projekte TASK [3]/TOOLS [12] in Stuttgart, IDA [15] in Birlinghofen, DIADES II [6] in Darmstadt und JANUS [2] in Bochum bauen dabei auf jeweils spezielle Vorgehensmodelle und decken vorwiegend die Bereiche ab, die den zwei letzten Sichten nach MUSE entsprechen, während das Projekt EXPOSE ([10]; [8]) in Oldenburg und Rostock genau das Vorgehensmodell von MUSE durch alle vier Sichten unterstützt.

Es ist zu erwarten, daß aus diesen Forschungsprojekten in einigen Jahren auch praktische Werkzeuge auf den Markt kommen. Bis dahin sind die Designer gefordert, die software-ergonomische Literatur (siehe [1]; [7]; [13]; [14]; [17]) allein zu erarbeiten.

5 Anmerkungen

1 Ich lege Wert auf den Begriff "Benutzungsoberfläche". Damit ist die Gesamtheit der *Wirkungen* gemeint, die ein bestimmter Teil des Software-Systems, die "Benutzungsschnittstelle" hervorruft. Das Programmstück

Benutzungsschnittstelle stellt dem Benutzer die Funktionalität des Systems zur Verfügung. (Schlicht falsch ist die Eindeutschung von "user interface" als "Benutzeroberfläche". Die besteht normalerweise aus Haut und Haaren, und ihre Gestalter gehören zu den Zünften der Schneider, Friseure und Kosmetiker - oder zu den Schönheitschirurgen.).

2 Der englische Begriff "usability" kann im Deutschen sowohl "Brauchbarkeit" (des ganzen Programms) wie "Benutzbarkeit" (der Benutzungsoberfläche) bedeuten.

6 Literatur

[1] Balzert; Hoppe et al.: Einführung in die Software Ergonomie. Reihe: MCK-Grundwissen. Berlin und New York 1988.

[2] Balzert, H.: Das JANUS-System: Automatisierte wissensbasierte Generierung von Mensch-Computer-Schnittstellen. In: Informatik - Forschung und Entwicklung. Heidelberg 1994.

[3] Beck, A.: Methoden und Werkzeuge für die frühen Phasen der Software-Entwicklung. In: Ackermann, D. et al.: Software-Ergonomie '91: Benutzerorientierte Software-Entwicklung. Stuttgart 1991.

[4] Budde, R.; Züllighoven, H.: Programmierwerkzeuge in einer Programmierwerkstatt. München 1990.

[5] Coad, P.; Yourdon, E.: Object-Oriented Analysis. Yourdon Press, Englewood Cliffs 1991.

[6] Dilli, L.; Vogt, G.; Hoffmann, H.-J.: Diades II: Ein objektorientiertes Entwurfswerkzeug für interaktive Benutzungsschnittstellen. Ergonomie & Informatik. In: Mitteilungen des GI-Fachauschusses 2.3, (1993) 19, S. 14-21.

[7] Dix; Finlay; Abowd; Beale: Human-Computer Interaction. Prentice Hall, New York u.a. 1993.

[8] Forbrig, P.; Gorny, P.; Viereck, A.: Unterstützung des Software-Design-Prozesses durch EXPOSE. In: Coy, W.; Gorny, P.; Kopp, I.; Skarpelis, C. (Hrsg.): Menschengerechte Software als Wettbewerbsfaktor. Forschungsansätze und Anwenderergebnisse aus dem Programm: Arbeit und Technik. Stuttgart 1993, S. 463-479.

[9] Gorny, P.; Viereck, A.; Qin, L.; Daldrup, U.: Slow and Principled Prototyping of Usage Surfaces: a Method for User Interface Engineering. In: Züllighoven, H. et al. (Hrsg.): Proceedings RE'93 - Prototyping. Bonn und Stuttgart 1993, S. 125-133.

[10] Gorny, P.; Viereck, A.: Ein Software-Ergonomie-Expertensystem. In: Ackermann, D; Ulich, E. (Hrsg.): Software-Ergonomie '91. Stuttgart 1991, S. 152-161.

[11] Iannella, R.: HyperSAM 2.0 - A hypertext interface to Smith and Mosier. HyperCard Stack. Bond University. Gold Coast QLD, Australia 1993.

[12] Janssen, C.; Weisbecker, A.; Ziegler, J.: Generierung graphischer Benutzungsschnittstellen aus Datenmodellen und Dialognetz-Spezifikationen. In: Züllighoven, H.; Altmann, A.; Doberkat, E. (Hrsg.): Requirements Engineering '93: Prototyping. German Chapter of the ACM Band 41. Stuttgart 1993, S. 335-347.

[13] Koch; Reiterer; Tjoa: Software-Ergonomie. Wien und New York 1991.

[14] Preece et al.: Human Computer Interaction. Addison-Wesley, 1994.

[15] Reiterer, H.: A Human Factors Based User Interface Design. In: Grechenig, T.; Tschelegi, M. (Hrsg.): Human Computer Interaction. Proceedings of the Vienna Conference VCHCI'93. Lecture Notes in Computer Science. Springer, Wien 1993, S. 291-302.

[16] Rumbaugh, J. et al.: Object-Oriented Modelling and Design. Prentice Hall, Englewood Cliffs 1991.

[17] Shneiderman: Designing the User Interface: Strategies for Effective Human-Computer Interaction. 2nd Edition. Addison Wesley, Reading u.a. 1993.

[18] Smith, S.; Mosier, J.: Guidelines for designing user Interface software. The Mitre Corp., Bedford, MA 1986.

[19] Viereck, A.; Gorny, P.: Software-Ergonomische Beratung bei der Benutzungsschnittstellen-Entwicklung. In: Frese, M. et al. (Hrsg.): Software für die Arbeit von morgen. Berlin u.a. 1991, S. 309-408.