Software Architekturen für Mensch-Computer-Schnittstellen - dargestellt am Beispiel eines multilingualen Textsystems

Klaus-Peter Fähnrich, Michael Kärcher
Fraunhofer-Institut für
Arbeitswirtschaft und Organisation (IAO), Stuttgart

ZUSAMMENFASSUNG

Es wird eine geschichtete Softwarearchitektur für Benutzerschnittstellen vorgestellt. Sie wurde auf der Basis eines geschichteten Mensch-Computer-Interaktionsmodelles entwickelt. Die Architektur wurde für ein multilinguales Textsystem implementiert.

ZIELSETZUNG

Das vorgestellte Vorhaben hatte es sich entsprechend der Themenstellung zum Ziel gesetzt:

- Ausprägungen der Mulitilingualität zu definieren und weitgehend zu realisieren,
- ein einsatzfähiges, effizient implementiertes multilinguales Textsystem zu realisieren,
- die Mensch-Computer-Schnittstelle entsprechend einer geschichteten Architektur zu implementieren.

Dabei sollte unter anderem geprüft werden, ob die Forderung einer effizienten Implementation zu einem gewissen Grade eine konzeptionell sauber gestaltete Schnittstelle erschwert.

VORGEHENSWEISE

Das Forschungs- und Entwicklungsprogramm wurde in zwei Teilvorhaben abgewickelt, die insgesammt einen Zeitraum von fast 3 Jahren beanspruchten:

- einem empirisch angelegten Teil zur Definition eines multilingualen Tastaturlayouts, das den vollen Teletex-Zeichensatz (ISO 6937) auf einer Tastatur realisiert,
- und einem zweiten Teilvorhaben, in dem das vollwertige Textsystem spezifiziert, implementiert und einem Langzeitpilottest unterzogen wurde.

Dieser Beitrag beruht auf Arbeiten im Rahmen der von der Kommission der Europäischen Gemeinschaften geförderten Vorhaben "Study into the Definition of a Multilingual Keyboard" und "Definition and Implementation of a Multilingual Word Processor". Beim empirischen Teil der Arbeiten hat R. Ilg, bei der Implementation haben A. Gräble und A. Schulz aus unserem Institut mitgewirkt.

Innerhalb des ersten Teilvorhabens wurden die wesentlichen Grundlagen der zu realisierenden Multilingualität gelegt, indem:

- mehrere Varianten für das entsprechende Tastaturlayout spezifiziert,
- ein flexibles Hard- und Softwaresystem zur Durchführung entsprechender Feldtests entwickelt,
- Feldtests bei der Kommission der Europäischen Gemeinschaften in Brüssel durchgeführt, und
- die Ergebnisse in die internationalen Normungsgremien eingebracht wurden.

Die Ergebnisse sind in /1/, /2/ und /3/ dokumentiert. Innerhalb des zweiten Teilvorhabens wurde nun ein System für den Echteinsatz realisiert, indem:

- eine Schichtenarchitektur für Mensch-Computer-Schnittstellen und eine Vorgehensweise zur softwareergonomischen Gestaltung von interaktiven Systemen entwickelt,
- Multilingualität definiert, und das entsprechende Textverarbeitungssystem spezifiziert und entsprechend der Schichtenarchitektur implementiert,
- und letztlich das System bei der Kommission der Europäischen Gemeinschaften in Brüssel in einem Feldtest mit 3 Systemen à 3 Arbeitsplätzen zum Einsatz gebracht wurde.

MULTILINGUALITÄT BEI TEXTSYSTEMEN

Multilingualität soll im folgenden für Sprachen mit lateinischem Alphabet auf unterschiedliche Komponenten eines Textsystems heruntergebrochen werden:

- **Ein-/ausgabeseitig** muß der Zeichensatz (ISO 6937) auf der Tastatur produzierbar und auf Bildschirm sowie Drucker darstellbar sein. Dabei sollte die Implementation den Zeichensatz für eine weite Klasse von Endgeräten realisieren bzw. konzeptionell ermöglichen. Bei Hardwarerestriktionen ist mit adequaten Ersatzdarstellungen zu arbeiten.
- **Dialogseitig** sollen Menüeinträge, Hilfstexte in Menüs und Formularen, Status- und Fehlermeldungen in der vom Benutzer gewählten Sprache ausgegeben werden.
- **Werkzeugseitig** existiert eine Vielzahl von Effekten: In der zentralen Editierfunktionalität z.B. sprachabhängige Objektanwahl (z.B. unterschiedliche Interpunktionsregeln) oder Silbentrennung und Orthographie- bzw. Stilprüfung (Spelling- bzw. Style-Checking). Für Retrievalfunktionen werden oftmals, stichwortartige oder volltextartige Klassifikationen eingesetzt. Um in diesem Bereich Multilingualität zu erreichen, wären Techniken der künstlichen Intelligenzforschung und Computerlinguistik einzusetzen.

Das folgende Bild fasst die einzelnen Bereiche der Multilingualität noch einmal zusammen:

Bild 1: Multilingualität und entsprechende Phänomene in einer Schichtenarchitektur

Das folgende Bild zeigt den Zeichensatz, der die wesentliche Voraussetzung für die Realisierung der Mulitilingualität des Systems darstellt:

	040	060	100	120	140	160
00		0	@	P		p
01	!	1	A	Q	a	q
02	"	2	B	R	b	r
03	#	3	C	S	c	s
04	¤	4	D	T	d	t
05	%	5	E	U	e	u
06	&	6	F	V	f	v
07	'	7	G	W	g	w
10	(	8	H	X	h	x
11	)	9	I	Y	i	y
12	*	:	J	Z	j	z
13	+	;	K	[	k	{
14	,	<	L	\	l	|
15	-	=	M	]	m	}
16	.	>	N	^	n	~
17	/	?	O	_	o	

	240	260	300	320	340	360
00		°		–	Ω	ĸ
01	¡	±	̀	‘	Æ	æ
02	¢	²	́	’	Đ	đ
03	£	³	̂	‚	ª	ð
04	$	×	̃	“	Ħ	ħ
05	¥	µ	̄	”		ı
06	#	¶	̆		Ĳ	ĳ
07	§	·	̇		Ŀ	ŀ
10	¤	÷	̈		Ł	ł
11	'	'	̈		Ø	ø
12	"	"	°		Œ	œ
13	«	»	̧		º	ß
14	←	¼	_	⅛	Þ	þ
15	↑	½	̋	⅜	Ŧ	ŧ
16	→	¾	̨	⅝	Ŋ	ŋ
17	↓	¿	̌	⅞	ŉ	

Bild 2: Der Zeichensatz aus ISO 6937 (Teletex)

DIE FUNKTIONALITÄT DES TEXTSYSTEMS MULITEX

MuLiTex ist als komfortables, multilinguales Textbearbeitungssystem konzipiert. Die Hauptbereiche der Funktionalität sind hierbei:

- o Basale Eingaben (Schreibmaschinenmodus)
- o Erweiterte, komfortable Editierfunktionalität
- o Formatierungs- (Layout-) Funktionalität
- o Druck- bzw. Ausgabefunktionalität
- o Archivierung bzw. Dateihandling
- o Steuerung der Multilingualität

Besondere Beachtung bei der Spezifikation und Implementation erfuhr der zentrale Editor und Formattierer. Das System wurde entsprechend den WYSIWYG Prinzip (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et) konzipiert. Während der Systemanalyse wurde eine Systematik verwendet /4/, die auf die Definition weniger, generische Operationen abzielt.Dadurch wird die Bedienung weiter Teile des Systems über die entsprechenden Objekte und Operationen konsistent und die Anzahl notwendiger Kommandos gesenkt.

Zur Konsistenz der Benuzterschnittstelle trug außerdem bei, daß für die zentrale Editier- und Formattierungsfunktionalität ein direkt manipulatives Interface /5/, /6/ implementiert wurde. Dabei kam entsprechend der Syntax der direkten Manipulation ein komfortabler Mechanismus zur Objektselektion zum Einsatz.

ARCHITEKTURKONZEPTE FÜR SOFTWARESYSTEME

Bisherige Architekturen für Softwaresysteme zeigen keine Trennung von Funktionalität und Interaktion; d.h. die Interaktion mit der Funktionalität ist direkt innerhalb der eigentlichen Anwendungsprogramme implementiert. Dieses hatte aus software-ergonomischer Sicht folgende Implikationen:

- o Einerseits kann damit für eine spezielle Applikation die optimale Interaktionsform gewählt werden.
- o Andererseits besteht die Gefahr, daß inkonsistente Benutzerschnittstellen enstehen. Dies gilt in besonderem Maße für Systeme, bei denen ein Benutzer mit einer Vielzahl von Applikationen interagiert.

Aufgrund der Erkenntnis, daß einer der wesentlichen Engpässe im Bereich der Interaktion des Benutzers mit der Applikation liegt, wurden Überlegungen zu neuartigen Architekturen für Softwaresysteme angestellt. Im folgenden wird dem klassischen Ansatz ein geschichtetes Konzept gegenübergestellt. Die theoretische Fundierung dieses Ansatzes liegt einem geschichteten Modell der Mensch-Computer Interaktion; vgl. /4/, /5/, /6/. Dieses Modell beinhaltet (1) die konzeptionelle Ebene der Aufgabenrepräsentation (2) die semantische Ebene der Funktionaliät und Werkzeuge, auf der Objekte und Operationen repräsentiert sind, (3) die syntaktische Ebene mit der Definition möglicher Dialoginteraktionen und (4) die Interaktionsebene, die vorwiegend die physikalische Verbindung zwischen Benutzer und Rechner darstellt. Bei Implementationen entsprechend diesem Modell werden folgende Effekte erwartet:

- o Die Applikation kann mit geringerem Aufwand als bisher an verschiedene Ein-/Ausgabegeräte angepasst werden. Dies kommt durch die wachsende Vielzahl von unterschliedlichen Ein-/Ausgabesystemen zum Tragen.

- Die alternative aber konsistente Verwendung unterschiedlicher Dialogtechniken (Menuesysteme, Kommandosprachen) z.B. zur Realisierung von Einlern- und Expertenmodi wird erleichtert.

- Stärker als bisher werden "generische" Objekte und Funktionen /4/ verwendet, die einen verbesserten Lerntransfer zwischen verschiedenen Teilen des Systems und damit eine verbesserte Erlernbarkeit sicherstellen sollen.

Verändern soll sich damit auch der Prozeß der Spezifikation, Implementation und Weiterentwicklung der Software:

- Duplizierung von Entwicklungsaufwand bei der Implementation der Benutzerschnittstelle kann vermieden werden. Dabei freigewordene Kapazität kann für eine Verbesserung der software-ergonomischen Qualität eingesetzt werden. Dies gilt in besonderem Maße, wenn mächtige Werkzeuge zur Implementationsunterstützung geschaffen werden.

- Die Implementation der Benutzerschnittstelle erfolgt getrennt von der Implementation der Anwendungssysteme. Falls geeignete Werkzeuge existieren, kann die Implementation der Benutzerschnittstelle von einem Spezialisten für Software-Ergonomie vorgenommen werden.

- Die Implementation der Benutzerschnittstelle neuer Applikationen unter Einsatz der existierenden Schnittstellensoftware wird wesentlich vereinfacht.

- Die Einbindung neuer Applikationen, die erweiterte Anforderungen an die Benuzterschnittstelle stellen, erfordert beträchtlich höheren Aufwand, daher sollte:

- die Spezifikation der Benutzerschnittstelle vor Beginn der Implementation möglichst vollständig (d.h. auch auf spätere Erweiterungen ausgelegt) vorliegen.

In der Praxis ist die überwiegende Mehrzahl der bisherigen Applikationen nicht entsprechend der hier vorgestellten Architektur implementiert. Die Integration dieser Software in eine geschichtete Schnittstelle bereitet besondere Probleme, die bisher nur unzulänglich gelöst sind.

IMPLEMENTATION ENTSPRECHEND DEM GESCHICHTETEM ARCHITEKTURKONZEPT

Entsprechend dem vorgestellten Architekturkonzept wurde die Implementation durchgeführt. In /7/ werden werden Begriffe wie Tool-Management-System, Dialog-Management-System, E/A-Management-System eingeführt, die auf den jeweiligen Schichten des Modelles die Funktionalität der Schnittstellensoftware repräsentieren.

In MuLiTex wurde ein umfangreiches E/A-Management-System und ein für die Aufgabenstellung vollständiges Dialog-Management-System implementiert. Auf die Implementation eines eigenständigen Tool-Management-Systems wurde verzichtet, da diese Funtkionaliät

bisher adäquat vom Dialog-Management-System mit übernommen werden kann. Die gesamte Applikation läuft unter Kontrolle der Dialogschicht und nicht unter Kontrolle der Werkzeuge. Sämtliche Werkzeuge sind vom E/A-System entkoppelt - die Werkzeuge operieren auf einer Datenstruktur (Editfile), die die interne Repräsentation des bearbeiteten Dokumentes darstellt und um für die Dialogwerkzeuge relevante Information (z.B. Selektionsmarken und -markierierungen) ergänzt wird. Diese interne Repräsentation wird zyklisch oder ereignisgesteuert von der Ausgabeseite des Dialog-Management-System interpretiert und dem E/A-Management-System zur physikalischen Ausgabe übergeben.

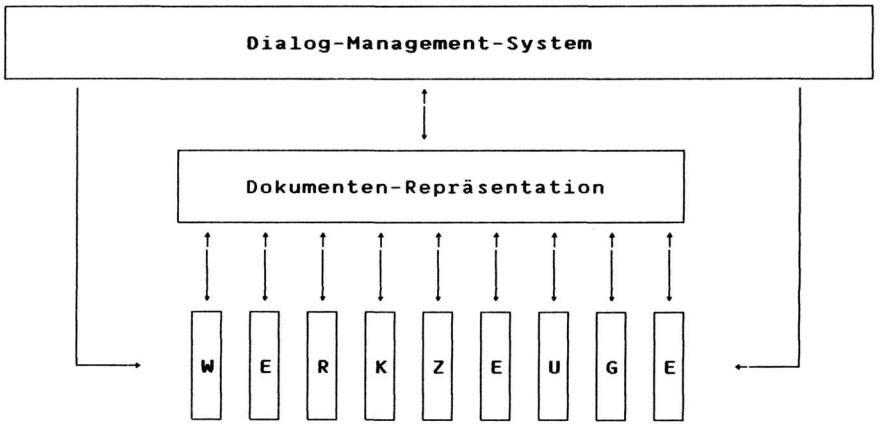

Bild 3: Zusammenspiel von Werkzeugen, Dokumenten-Repräsentation und Dialog-Management-System

Diese Architektur erlaubt auf der einen Seite die komfortable Erweiterbarkeit des Systems, sofern die entsprechenden Werkzeuge sich an die Konventionen des Dialog-Management-Systems halten. Die konsistente Integration in diesem Sinne nicht angepasster Software (hier z.B. System-Dienstprogramme) bereitet jedoch Schwierigkeiten, die zu Inkonsistenzen des Systems führen. Leichter integrierbar sind in jedem Falle Werkzeuge, die darauf verzichten, Dialogfunktionalität im Werkzeug zu implementieren. Dies ist z.B. für eine weite Klasse von UNIX Werkzeugen der Fall.

E/A-Management-System: Die Ein-/Ausgabeschicht hat die primäre Aufgabe, der Dialogschicht ein weitgehend hardwareunabhängiges (und betriebssystemunabhängiges) Interface zu liefern. Weiterhin überführt sie den Eingabestrom in kleinste für die Dialogschicht Bedeutung tragenden Einheiten (Morpheme). Ausgabeseitig löst sie diese Morpheme wiederum für die entsprechende Ausgabegeräte in elementare Ausgabeeinheiten auf.

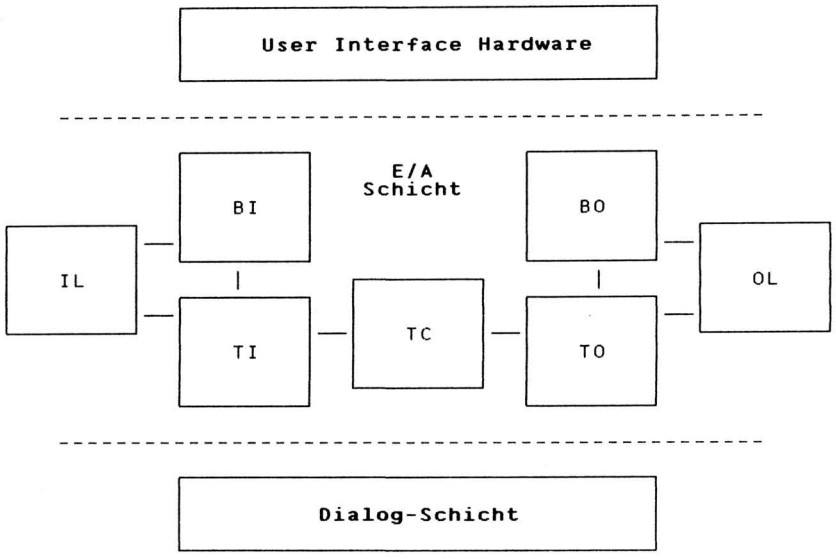

Bild 4: Architektur der **E**in-/**A**usgabe-Schicht

Die wesentlichen Module dieser Schicht sind BI (**B**uffered **I**nput), BO (**B**uffered **O**utput), TI (**T**erminal **I**nput), TO (**T**erminal **O**utput), TC (**T**erminal **C**ommon) sowie IL (**I**nput **L**ogger) und OL (**O**utput **L**ogger). BI und TI realisieren dabei die Eingabeseite der E/A-Schicht, BO und TO deren Ausgabeseite. TC enthält zentrale Funktionalität der E/A-Schicht, die ein- und ausgabeseitig gemeinsam genutzt werden. IL und OL werden benötigt, wenn benutzerorientierte Testmethoden im System realisiert sind.

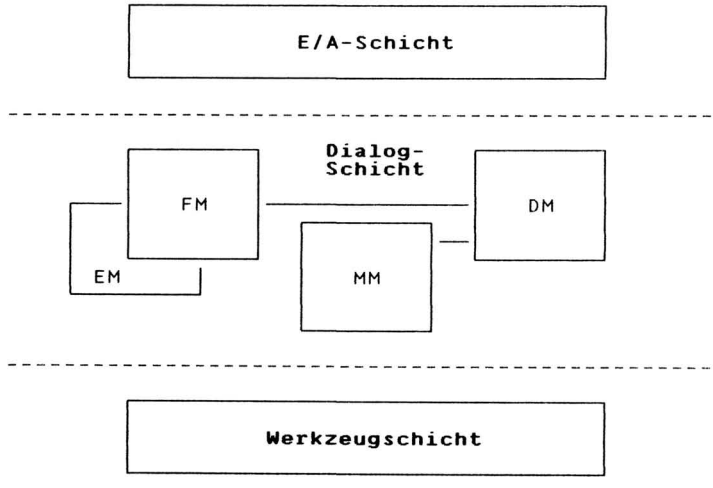

Bild 5: Architektur der Dialogschicht

Dialog-Management-System: Die Dialogschicht hat die Aufgabe, den Werkzeugen ein weitgehend dialogunabhängiges Interface zu liefern. Weiterhin realisiert sie eingabeseitig die verschiedenen Dialogtechniken und bedient ausgabeseitig den Bildschirm auf der Basis semantisch höherwertiger Objekte.

Die wesentlichen Module dieser Schicht sind FM (**F**orms **M**anager), EM (**E**dit **M**anager), DM (**D**isplay **M**anager und MM (**M**essage **M**anager). Eingabeseitig kommen zwei Dialogtechniken zur Anwendung: FM realisiert eine Kombination aus Formular- und Menütechnik - EM realisiert Editierfunktionalität des Textbearbeitungssystems unter Einsatz von Funktionstasten ("Keypad Editing"). FM und EM verteilen die anfallenden Aufgaben auf die zur Verfügung stehenden Werkzeuge der Werkzeugschicht. DM realisiert die logische Handhabung des Bildschirms. MM versorgt DM mit textlichen Meldungen.

MULTILINGUALITÄT UND IHRE IMPLEMENTATION

Auf der Basis der im vorigen Kapitel vorgestellten Implementierungsdetails sollen einige Phänomene der Multilingualität in ihrer Implementation näher beleuchtet werden, um daran die Vorteile der Schichtenarchitektur diskutieren zu können. Dies geschieht ausgewählt für drei Bereiche:

- Interner Zeichensatz
- Interpretation der Eingabe der Zeichen
- Ausgabe multilingualer Texte

Entgegen der bisherigen Praxis bei Textsystemen wird der Zeichensatz in 16 Bit codiert - damit werden sowohl Umschaltmechanismen zwischen verschiedenen Zeichensätzen (Buchstaben, Graphik, Steuerzeichen) als auch die Zeichensätze entsprechend der Normung realisiert. Ohne Änderungen der Software können für alle buchstabenorientierten Sprachen (Griechisch, Kyrillisch etc.) die Editierwerkzeuge genutzt werden. Geringe Modifikationen werden für Sprachen notwendig, die nicht die Schreibweise von links nach rechts anwenden. Diese Modifikationen sind ausgabeseitig vor allem auf den Display-Manager zentralisierbar.

Wie bereits diskutiert, übernimmt der Terminal-Input-Handler die Decodierung der Eingaben der Eingabegeräte. Er realisiert die Unabhängigkeit der Dialogschicht von:

- Unterschiedlichen Tastaturen bzw. Eingabegeräten und speziell

- der Anordnung der Belegung einer speziellen Tastatur sowie verschiedener national oder international genormter Varianten.

Dazu bildet er im Falle einer Tastatur physikalische auf virtuelle Tastenpositionen ab. Anschließend findet die Abbildung mittels sog. Keymap-Tables auf (hardwareunabhängige) Tokens statt. Dieser Prozeß beinhaltet speziell das tastaturunabhängige Handling allgemeiner durch Sequenzen von Tastendrucken erzeugter Zeichen und Kommandos. Dies sind z.B. extradiakritische Zeichen (z.B.: á, è, û, ÿ, ñ, ç, å) oder Spezialzeichen (ł, ı, ¿, ĸ, Œ, œ, ¥, ⅜) sowie der Zweitbelegung von Funktionstasten.

Die Keymap-Tables werden von einem speziellen Compiler aus einer formalen Meta-Sprache erzeugt und sind zur Laufzeit des Systems ladbar. Dieser Mechanismus erlaubt vor allen Dingen die leichte Erweiterung bzw. Änderung des Tastaturlayouts sowie der wichtigsten dynamischen Aspekte der Komposition von Tastensequenzen zu Zeichen ohne editieren, compilieren und linken der eigentlichen Software.

Die gesamte Problematik wurde in der E/A-Schicht zentralisiert; Seiteneffekte auf höherliegende Schichten der Software treten nicht auf. Die parallele Problematik auf der Ausgabeseite wird von TO, dem komplementären und wiederum tabellengesteuerten Ausgabemodul gelöst. Durch "Kurzschließen" dieser Module (Weiterleiten der aus TI gewonnenen internen Darstellung an TO) wird ein Testen der E/A-Schicht unabhängig von den höheren Schichten ermöglicht.

Das letzte Beispiel diskutiert die Ausgabe multilingualer Textstrings. MM realisiert, daß im gesamten System für angezeigte Texte, die für den Bildschirm bestimmt sind, symbolische Namen verwendet werden können. Dies ist Grundvoraussetzung für die konsistente Multilingualität an der Benutzerschnittstelle.

Dazu verfügt das System über einen "Message-Definition-File", in dem den symbolischen Namen die entsprechenden Texte in den verschiedenen realisierten Sprachen zugewiesen sind. Entsprechend den "Keymap-Tables" wurde aus den selben Gründen auch hier ein spezieller Compiler implementiert.

Eingebettet in die Meldungstexte können Variablen-Ersetzungs-Direktiven auftreten, die dann von einem assoziierten Modul behandelt werden. Die hier gewählte Implementierung ist wesentlich leistungsfähiger als entsprechende Referenzimplementationen.

RESUMÉ

Die Implementation entsprechend der Schichtenarchitektur hatte folgende Vorteile:

- Schon zur Spezifikationszeit wurde ein klares konzeptionelles Modell von Funktionalität und Benuzterschnittstelle entwickelt.
- Die Implementation konnte schichtenweise vorgenommen und getestet werden - die Software war frühzeitig stabil.
- Die Prinzipien der Multilingualität und weitere Charakteristika der Benuzterschnittstelle wurden effizient und einfach modifizierbar implementiert.

Zum Schluß sollte noch angemerkt werden, daß das System im Gegensatz zu vergleichbaren Systemen, die in typischen Rapid-Prototyping-Sprachen (z.B. Lisp oder Lisp-Derivaten) sogar in 64 kB Hauptspeicher auf einer PDP-11 im Mehrbenuzterbetrieb lauffähig ist. Damit sollte der Nachweis erbracht werden, daß fortgeschrittene Softwarearchitekturen nicht notwendigerweise mit extensivem Speicherplatzbedarf einhergehen.

LITERATUR

/1/ Hugh McGregor Ross: The Application of Standardised Character Sets to Multilingual Text Communication and Processing; Final Report for the Commission of the European Communities, December 1980.

/2/ K.-P. Fähnrich; R. Ilg; M. Kärcher; M. de Smith: A Unified Keyboard Layout for Multiple Latin Alphabet Languages, Final Report for the Commission of the European Communities, October 1982.

/3/ K.-P. Fähnrich; R. Ilg; M. Kärcher; M. de Smith: MuLiTex - A Multilingual Wordprocessor, Report for the Commission of the European Communities, September 1984.

/4/ Ziegler, J.; Fähnrich, K.-P., Raether, C.: Textsysteme und ihre Benuzterschnittstelle. In: Proc. Offene Multifunktionale Büroarbeitsplätze und Bildschirmtext, TU Berlin 1984.

/5/ Fähnrich, K.-P.; Ziegler, J.: Workstations Using Direct Manipulation as Dialog Principle - Aspects Of Design, Application and Evaluation. In: Shackel, B. (Hrsg.) Proc. 1st Conf. Human-Computer-Interaction, INTERACT 84, London 1984.

/6/ Fähnrich, K.-P.; Ziegler, J.: Direkte Manipulation als Interaktionsform an Arbeitsplatzrechnern. In: Bullinger (Hrsg.) Proc. der Tagung Software-Ergonomie 85 - Mensch-Maschine-Kommunikation, Teubner, Stuttgart 1985.

/7/ Bullinger, H.-J.; Raether, C.; Fähnrich, K.P.; Kärcher, M.: Software-Ergonomie im Produktionsbereich - Dargestellt am Beispiel der Analyse und Gestaltung von Programmiersystemen von Werkzeugmaschinen. In: Bullinger (Hrsg.) Proc. der Tagung Software-Ergonomie 85 - Mensch-Maschine-Kommunikation, Teubner, Stuttgart 1985.

Klaus Peter Fähnrich, Michael Kärcher
Fraunhofer-Institut für Arbeitswirtschaft
und Organisation (IAO), Stuttgart
Holzgartenstr. 17
D-7000 Stuttgart 1

Das vorgelegte Dokument wurde vollständig auf MuLiTex, dem multilingualen Textsystem erstellt.