Die Konstruktion von Benutzerschnittstellen: Ein neuer Ansatz für Systemarchitekturen und Werkzeuge

Thomas Greutmann, Zürich

Zusammenfassung

Es wird ein Ansatz zur Strukturierung von User Interface Management Systems (UIMS) vorgestellt, welcher auf einer "mehrdimensionalen" Architektur beruht. Diese Architektur beruht auf einem objektorientierten Ansatz, welcher durch zusätzliche Gliederungen ergänzt wurde. Die Architektur erlaubt die Konstruktion von erweiterbaren und in Bezug auf die Art der Benutzeroberfläche portablen UIMS. Die auf der Basis von solchen UIMS entwickelten Anwendungssysteme sind ebenfalls portabel und durch die Endbenutzer individualisierbar. Die zu einem UIMS notwendigen Entwurfswerkzeuge werden ebenfalls besprochen.

1. Einleitung

Um die Entwicklung und Implementation von ergonomisch gut gestalteten Benutzerschnittstellen für Anwendungssysteme zu vereinfachen und ökonomischer zu gestalten, wird seit Kasik (1982) der Ansatz der "User Interface Management Systems" (UIMS) verfolgt. Diese basieren auf einer möglichst vollständigen Trennung der Programmteile der Benutzerschnittstelle und der eigentlichen Anwendungs-Funktionalität, wobei Benutzerschnittstellen-Funktionen den Entwicklern als vorgefertigte Bausteine zur Verfügung stehen. Häufig werden UIMS durch spezielle Werkzeuge (interaktive Editoren oder Generatoren) unterstützt, in diesem Fall ist die Bezeichnung "User Interface Development System" (UIDS, Hill 1986, Myers 1989) üblich. UIMS bezeichnen dann die Benutzerschnittstellen-Laufzeitkomponenten. Diese Terminologie wird auch im folgenden verwendet. Uebersichten über existierende UIMS bzw. UIDS finden sich bei Keller (1989, S.168ff), Myers (1989) oder Brown und Cunningham (1989, S.207ff).

Die Verwendung von UIMS bzw. UIDS sollen den Entwicklungsaufwand durch "Fertigbauelemente" reduzieren und die Qualität der Benutzerschnittstelle durch bewährte Bausteine verbesseren. Der reduzierte Entwicklungsaufwand soll ausserdem ein iteratives Vorgehen (unter Einbezug der Benutzer) und Rapid Prototyping vereinfachen (Hayes, Szekely & Lerner 1985, Schulert, Rogers & Hamilton 1985, Burgstaller, Grollmann & Kapsner 1989).

Die meisten der existierenden UIMS/UIDS sind als mächtige Implementierungs-Werkzeuge für Software-Entwickler konzipiert. Im folgenden wird ein anderer Ansatz vorgestellt, der die Bedürfnisse der zukünftigen Anwendungssystem-Benutzer und der Benutzer-Entwickler-Kooperation in den Vordergrund stellt. Im Gegensatz zu den meisten UIMS/UIDS wird ein besonderes Gewicht auf die Portabilität des Werkzeugs gelegt, damit das Werkzeug für existierende Oberflächentypen und -standards geeignet

ist. Ziel dieser Arbeit ist es, technische Voraussetzungen und Hilfsmittel für den Entwurf und die Implementierung von benutzergerechten Schnittstellen für Anwendungssysteme anzubieten.

2. Eine Architektur für flexible Dialoghandler

2.1 Bestehende Ansätze zur Strukturierung von UIMS

Eine Reihe von UIMS/UIDS sind als geschlossene Systeme mit fest vorgegebenen Dialogelementen ohne sichtbare interne Struktur konzipiert (vgl. Hayes u.a. 1985, Olsen 1986, Hix 1989, Keller 1989). Diese Systeme sind für die Entwicklung von realen Anwendungssystemen jedoch zu starr, da sie von Anwendungsentwicklern nicht oder nur sehr umständlich erweitert werden können. Hill und Herrmann (1989) merken an, dass dies auch für das vielzitierte "Seeheim-Modell" für UIMS (Green 1985) gilt.
Um UIMS/UIDS flexibel, offen ("open-ended") und erweiterbar zu gestalten, müssen diese Systeme modular aufgebaut werden (Cardelli 1988, Burgstaller u.a. 1989). Alle Eigenschaften eines einzelnen Dialogelements (Dialogkontrolle, Darstellung etc.) müssen in einem einzigen Programmbaustein zusammengefasst werden - ein objektorientierter Ansatz drängt sich auf ("vertikale" Gliederung des UIMS).
Die vertikale Gliederung sollte aber zusätzlich durch eine "horizontale" ergänzt werden: die einzelnen Dialogelemente sollten aus verschiedenen Schichten mit wohldefinierten Schnittstellen bestehen (Burgstaller u.a. 1989). Dadurch wird u.a. die Portabilität von UIMS/UIDS verbessert, indem benutzeroberflächen-nahe Schichten ausgetauscht werden können, wenn das UIMS/UIDS auf einen neuen Oberflächentyp übertragen wird.

2.2 Die Architektur von HIDE

Eine horizontale und vertikale Gliederung trägt allerdings der Unterscheidung zwischen UIMS (der Benutzerschnittstellen-Laufzeitkomponente) und UIDS (ein Implementations- oder Entwurfswerkzeug) nicht genügend Rechnung: das UIMS (Laufzeitsystem) verwaltet in erster Linie die Interaktion mit dem Benutzer, mit einem UIDS (Editor) werden die einzelnen Dialogelemente definiert (Layout etc.). Dies sind völlig unterschiedliche Sichten auf ein Dialogelement. Bei der Entwicklung und Implementation des User Interface Management System HIDE (Handler for Interface Description) wurde daher eine andere Architektur gewählt, welche sowohl auf Ideen des objektorientierten Ansatzes als auch des Schichtenmodells für UIMS/UIDS aufbaut und als "mehrdimensional" bezeichnet werden kann (vgl. Abb. 1). Wie der objektorientierte Ansatz für UIMS basiert der für HIDE gewählte Ansatz auf gleichartigen Dialogobjekttypen, welche die Grundbausteine des ganzen Dialoghandlers bilden.

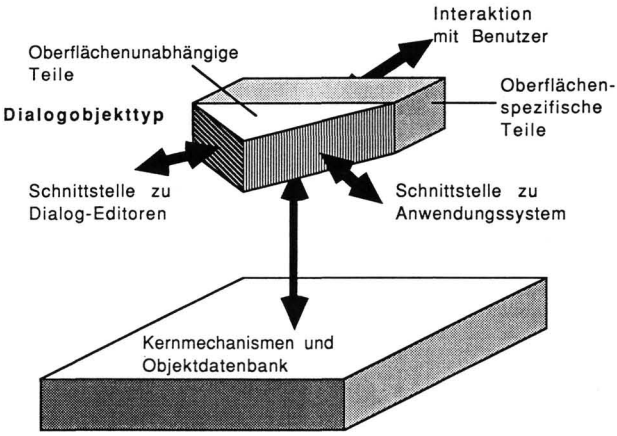


Abb. 1: Schema eine Dialogobjekttyps, der sich aus oberflächenspezifschen und oberflächenunabhängigen Teilen sowie Interaktions-, Editor- und Anwendungsschnittstelle zusammensetzt und auf Kernmechanismen aufbaut.

Wie im Schichtenmodell ist ein Dialogobjekttyp in oberflächenspezifische und oberflächenunabhängige Teile unterteilt, die oberflächenspezifischen Teile enthalten die Interaktionsschnittstelle, welche die Darstellung auf dem Bildschirm sowie die Eingaben der Benutzer bearbeitet. Die oberflächenunabhängigen Teile besitzen jedoch im Gegensatz zum Schichtenmodell zwei klar verschiedene Schnittstellen: eine Editor- und eine Anwendungsschnittstelle (siehe unten). Ein Dialogobjekttyp wird mit Hilfe einer Programmiersprache (Modula-2, Wirth 1983) beschrieben, die einzelnen Objekte dieses Typs werden dagegen in einer Objektdatenbank verwaltet, welche zusammen mit einigen für alle Dialogobjekttypen benötigten Kernmechanismen die Basis des Dialoghandlers bildet.

Der Aufbau von Dialogobjekttypen und deren Schnittstellen soll an einem Beispiel erläutert werden. Eine Eingabemaske kann als Dialogobjekttyp definiert werden, diese kann aus verschiedenen Arten von Eingabefeldern (z.B. Texteingabe, Auswahl aus Alternativen) bestehen, welche wiederum eigene Dialogobjekttypen sind. Der Dialogobjekttyp "Texteingabefeld" z.B. könnte wie folgt aussehen:

- Die **oberflächenspezifischen Teile** enthalten die Darstellung des Feldes sowie des Inhalts, die Interpretation von Benutzereingaben (je nach Art der Oberfläche Tastatur und/oder Maus) sowie damit zusammenhängende Funktionen (Grössen- oder Positionsangaben).
- **Oberflächenunabhängige Teile** sind z.B. das Lesen und Schreiben des Objekts von der Objektdatenbank.
- Die **Editor-Schnittstelle** enthält Funktionen zum Erzeugen, Löschen oder Verändern (Grösse, Name etc.) eines Texteingabefeldes. Diese Funktionen werden zur Laufzeit nicht benötigt, wohl aber in einem Entwurfs- oder Implementations-

werkzeug.

- Die **Anwendungs-Schnittstelle** ermöglicht es dem Anwendungssystem, auf die Eingabedaten zuzugreifen bzw. die Ausgabedaten zu verändern (z.B. Abfrage des Eingabewertes, Setzen eines Default-Wertes etc.). Diese Funktionen werden zur Laufzeit benötigt.

Der gesamte Dialoghandler HIDE ist ein offenes System und setzt sich aus der Menge der einzelnen Dialogobjekttypen zusammen, welche baukastenartig aneinandergereiht werden. Die Interaktions-, die Editor- und die Anwendungsschnittstelle wachsen mit jedem Dialogobjekttyp mit (Abb. 2).

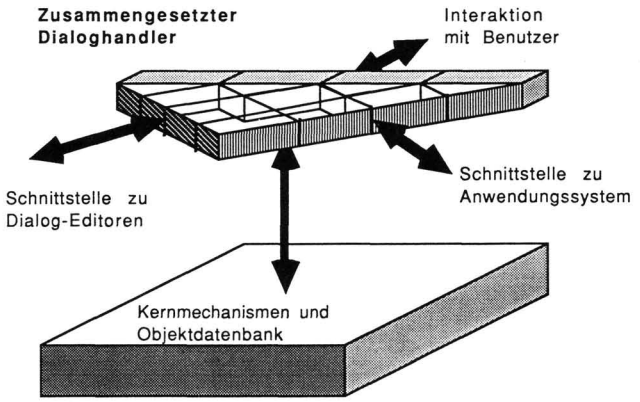


Abb. 2: Der Dialoghandler setzt sich aus den einzelenen Dialogobjekttypen zusammen, die Schnittstellen wachsen mit jedem dazugefügten Dialogobjekttyp mit.

2.3 Die Portiertung des Dialoghandlers HIDE

Bei der Portierung des Dialoghandlers HIDE auf einen anderen Oberflächentyp werden die oberflächenspezifischen Teile jedes Dialogelementtyps durch Teile ersetzt, die für den neuen Oberflächentyp passen (Abb. 3). Die Anwendungs- sowie die Editorschnittstelle sind von einer Portierung nicht betroffen. Durch Auswechslung der oberflächenspezifischen Teile ist es insbesondere möglich, existierende Oberflächentypen und -standards zu unterstützen - dies im Gegensatz zu anderen UIMS/UIDS, welche einen neuen, vorgegebenen Oberflächentyp definieren.

Die erste Version von HIDE wurde für eine direktmanipulative, grafische, fensterorientierte Oberfläche implementiert. HIDE wurde anschliessend auf eine zeichenorientierten Oberfläche portiert, wobei lediglich die oberflächenabhängigen Teile ersetzt werden mussten. Diese bestehen in der zeichenorientierten Oberfläche neu aus 16 Modulen mit einer totalen Grösse von 96 KByte (gegenüber 18 Modulen mit 130 KByte in der grafischen Oberfläche), was etwa 30% des Gesamtumfangs von HIDE ausmacht. Die Portierung wurde in knapp zwei Wochen durchgeführt.

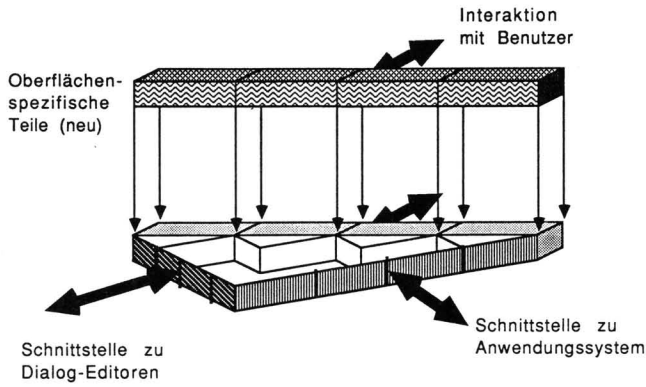
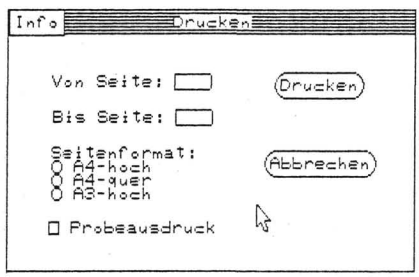


Abb. 3: Die Portierung des Dialoghandlers geschieht durch Auswechslung der oberflächenspezifischen Teile jedes Dialogelementtyps.

Ein Beispiel für dieselbe Eingabemaske für die grafische und die zeichenorientierte Oberfläche von HIDE findet sich in Abb. 4. Da die Anwendungsschnittstelle durch die Portierung nicht verändert wird, sind die beiden Masken aus der Sicht eines Anwendungssystems identisch, die Anwendungsschnittstelle abstrahiert völlig von der visuellen Darstellung der Dialogelemente ("abstract protocol" nach Cardelli 1988).




```
Drucken: Von-Seite:_   Bis-Seite:    Probeausdruck: Yes(No)
         Seitenformat:(A4-hoch)A4-quer A3-hoch
```

Abb. 4: Eine Eingabemaske für Druckparameter einer Anwendung für die grafische Oberfläche (oben) und die zeichenorientierte Oberfläche (unten) von HIDE. In der zeichenorientierten Version sind die Operationsknöpfe (Drucken, Abbrechen und Info) nicht sichtbar, sie sind durch spezielle Tasten (ENTER, ESCAPE und ALT-H) aufrufbar.

2.4 Entwicklung von Anwendungssystemen mit dem Dialoghandler HIDE

Anwendungssysteme werden auf der Anwendungsschnittstelle auf den Dialoghandler HIDE aufgesetzt, die Editor-Schnittstelle wird nicht benötigt Im Normalfall wird die gesamte Interaktion mit dem Benutzer über den Dialoghandler abgewickelt, welcher die Dialogkontrolle übernimmt und die Anwendungsfunktionalität in Form von Prozeduren

aufruft. Für die Behandlung von speziellen Fällen kann jedoch das Anwendungssystem die Dialogkontrolle übernehmen und den Dialoghandler umgehen.
Weil die gesamte Interaktion über den Dialoghandler abgewickelt wird und die Anwendungschnittstelle von der visuellen Darstellung der Dialogelemente abstrahiert, ist die Portierung von Anwendungssystemen von einer HIDE-Version auf eine andere einfach zu bewerkstelligen. Lediglich die Darstellung von Ausgabedaten (grafisch oder textuell) ist abhängig von der Art und den Möglichkeiten des Oberflächentyps. Ausgabedaten werden durch spezielle, anwendungsspezifische Darstellungsprozeduren auf den Bildschirm gebracht. Bei der Portierung eines Anwendungssystem müssen daher lediglich diese Darstellungsprozeduren ausgewechselt werden, der Programmcode mit der eigentlichen Funktionalität kann identisch übernommen werden.
Die Portierung wurde an zwei Beispielen erfolgreich getestet: ein kleines Adressverwaltungsprogramm und das umfangreiche Prototyping-Werkzeug IDEA (siehe unten) wurden von der grafischen auf die zeichenorientierte Version von HIDE übertragen. In beiden Fällen mussten lediglich die Darstellungsprozeduren, welche ca. 20-30% des Objektcodes ausmachen, ausgewechselt bzw. modifiziert werden. Die Portierungen konnten jeweils von einer Person in deutlich weniger als einer Woche durchgeführt werden.

3. Ein Entwurfswerkzeug für Benutzerschnittstellen

In dem hier vorgestellten Ansatz wird nicht davon ausgegangen, dass Werkzeuge allein das Problem der Gestaltung "guter" Benutzerschnittstellen lösen können, sondern lediglich unterstützend wirken. Dazu müssen diese Werkzeuge in eine Entwicklungsmethodologie eingebettet und auf die spezifischen Bedürfnisse der Methodologie abgestimmt werden. Dabei stehen im hier vorgestellten Ansatz folgende Ueberlegungen im Vordergrund:

- Der Entwurf der Benutzerschnittstelle ist ein sehr zentraler Punkt in der ganzen Software-Entwicklung. Dazu gehören nicht nur Fragen der Befehlsnamengebung, Bildschirmgestaltung oder des Dialogablaufs, sondern auch die Festlegung der Funktionalität des Anwendungssystems und Fragen der Mensch-Maschine-Funktionsteilung. Bildschirmgestaltung und Festlegung der Anwendungsfunktionalität sind stark miteinander verknüpft und - obwohl oft verlangt - nicht ohne weiteres zu trennen.
- Für den Entwurf guter Benutzerschnittstellen und guter Anwendungssysteme ist es notwendig, dass Benutzer aktiv in den Systementwurf einbezogen werden.

Ein UIDS muss daher in erster Linie für den Entwurf von Anwendungssystem-Benutzerschnittstellen (insbesondere Festlegung der Funktionalität und der Mensch-Maschine-Funktionsteilung) unter Einbezug der Benutzer geeignet sein. Es soll als Kommunikationsmedium zwischen Entwickler und Benutzer dienen, welches es erlauben soll, Entwürfe von Benutzerschnittstellen schnell zu erstellen, zu diskutieren und zu verändern. Für letzteres sind insbesondere Simulationsmöglichkeiten (Rapid Prototyping) sinnvoll. Zwischen Simulation und Aendern soll ein schneller Wechsel möglich sein, damit eine gemeinsame Arbeit (Erstellen/Testen/Verändern) von Benutzer und

Entwickler am Bildschirm möglich wird. Die Möglichkeit eines schnellen Wechsels zwischen Simulation und Verändern ist bei existierenden UIDS kaum anzutreffen, höchstens bei speziellen Simulationswerkzeugen (z.B. Trillium, Henderson 1986), welche aber nicht zur Konstruktion von lauffähigen Anwendungssystemen gedacht sind.
Im Rahmen dieser Anforderungen wurde für den Dialoghandler HIDE eine UIDS namens IDEA (Interactive Design and Evaluation of Applications) entwickelt, welche das interaktive Entwerfen, Erstellen, Simulieren und Verändern von Benutzerschnittstellen für Anwendungssysteme erlaubt. IDEA ist selber ein Anwendungssystem, welches auf der Basis von HIDE entwickelt wurde und sowohl auf der Editor- als auch auf der Anwendungsschnittstelle aufsetzt. Die interaktiv erstellten Prototypen können nach Tests mit Benutzern mit Hilfe des Dialoghandlers HIDE zu Anwendungssystemen ausgebaut werden, die Implementation von Benutzerschnittstellen wird also gleichermassen unterstützt. Erste experimentelle Studien (Krähenmann, Kupferschmid & Mayer 1990), in denen der Einsatz von IDEA im Rahmen einer intensiven Benutzer-Entwickler-Kooperation geprüft wurde, deuten darauf hin, dass der Einsatz von solchen interaktiven Werkzeugen die Qualität der erzielten Lösungen verbessern kann.

4. Individualisierungsmöglichkeiten für Benutzer

Bei der Entwicklung der Werkzeuge HIDE und IDEA wurde davon ausgegangen, dass es unmöglich ist, Benutzerschnittstellen für Anwendungssysteme zu entwerfen, die für alle Benutzer die optimale Lösung ("one best way") darstellen. Gemäss dem "Prinzip der differentiellen und dynamischen Arbeitsgestaltung" (Ulich 1978, 1983) müssen Arbeitssysteme (in diesem Fall Anwendungssysteme als Teil des Arbeitssystems) sowohl auf die unterschiedlichen Bedürfnisse und Qualifikationen der verschiedenen Benutzer als auch auf die sich im Laufe der Zeit ändernden Bedürfnisse und Qualifikationen anpassbar sein (Kriterium der "Flexibilität/Individualisierbarkeit", Ulich 1986). Der Dialoghandler HIDE wurde daher so konzipiert, dass die Benutzerschnittstellen eines Anwendungssystems nachträglich durch die Benutzer modifiziert und an ihre eigenen Bedürfnisse angepasst werden können, ohne dass dazu Programmierung notwendig ist. Das Anwendungssystem ist mit mehreren, verschieden strukturierten Benutzerschnittstellen lauffähig, welches durch die Beschreibung der Benutzerschnittstelle in einer Datei ermöglicht wird: jeder Benutzer kann mit seiner eigenen Benutzerschnittstellen-Datei arbeiten und diese auch selber interaktiv verändern.
Selbstverständlich muss ein geeignetes Werkzeug zur Verfügung stehen, welches es den Benutzern eines Anwendungssystems erlaubt, die Modifikationen ihrer Benutzerschnittstelle durchzuführen. Ein solches "Individualisierungswerkzeug" konnte auf einfache Weise interaktiv durch Anpassung des Entwurfswerkzeug IDEA (mit Hilfe von IDEA selber) erzeugt werden: gewisse Befehle und Maskenfelder wurden ausgeblendet, da sie nur für Entwickler relevant sind. Dadurch entstand ein neues Werkzeug mit reduzierter Funktionalität, welches nur die für Benutzer relevanten Informationen darstellt und verändert. Das Individualisierungswerkzeug ist nichts anderes als eine

"Individualisierung" (d.h. Anpassung an spezifische Benutzerbedürfnisse) des Entwurfswerkzeugs.

5. Zusammenfassung und Diskussion

Das hier vorgestellte Konzept für UIMS/UIDS, welcher bei der Realisierung von HIDE und IDEA verwendet wurde, beruht auf einem interdisziplinären Ansatz, der versucht, technische und arbeitspsychologische Anforderungen gleichzeitig zu erfüllen. Dies führt zu deutlich anderen Lösungen als bei einem rein informatikorientierten Ansatz für UIMS/UIDS. HIDE und IDEA orientieren sich primär an den Bedürfnissen der Benutzer von Anwendungssystemen anstelle der Bedürfnisse der Entwickler, ganz im Gegensatz zu anderen UIMS/UIDS. Die Unterschiede manifestieren sich vor allem in den folgenden Merkmalen:

1. **Ausrichtung auf Entwurf statt auf Implementierung**. Oft wird der Implementierung von Benutzerschnittstellen bei der Konstruktion von UIMS/UIDS das Hauptgewicht beigemessen. Beim Werkzeug IDEA hingegen liegt die Betonung auf dem Entwurf der Benutzerschnittstelle, welche letzten Endes auch über die eigentliche Anwendungsfunktionalität entscheidet. Durch die Simulationsmöglichkeiten wird eine enge Benutzer-Entwickler-Kooperation unterstützt.
2. **Individualisierungsmöglichkeiten**. Der Dialoghandler HIDE ist so konzipiert, dass dasselbe Anwendungssystem mit verschiedenen, auf individuelle Bedürfnisse jederzeit anpassbare Benutzerschnittstellen benutzt werden kann.
3. **Unterstützung von existierenden Standards für Benutzerschnittstellen**. Myers (1989) bemängelt zu recht, dass UIMS/UIDS für existierende und verbreitete Oberflächentypen und Systemumgebungen kaum verfügbar sind. Bei der Entwicklung des Dialoghandlers HIDE wurde besonderer Wert auf die Portabilität in bezug auf den Oberflächentyp gelegt. Dadurch ist es möglich, HIDE für bestehende Oberflächentypen anzupassen.
4. **Mehrdimensionale Architektur**. Eine Dialogelement- (vertikal) und Schichtengliederung (horizontal) erlaubt zwar die Konstruktion von offenen und portablen Dialoghandlern, trägt aber der Unterscheidung zwischen Laufzeitsystem und Entwicklungswerkzeug nur ungenügend Rechnung. Eine "mehrdimensionale" Architektur mit verschiedenen Schnittstellen nach verschiedenen Seiten (Anwendung, Editor, Interaktion, Kernmechanismen) ist dafür geeigneter.
5. **Einbettung in Entwurfsvorgehen**. Die Benutzerorientierung des hier vorgestellten Ansatzes bedingt, dass das Augenmerk nicht nur auf den Werkzeugen - welche allein noch nicht die Entwicklung von gut gestalteten Benutzerschnittstellen garantieren -, sondern auch auf der Einbettung der Werkzeuge in ein Entwurfs- und Entwicklungsvorgehen liegt. Das Entwurfswerkzeug IDEA wurde explizit so entworfen, dass eine enge Benutzer-Entwickler-Kooperation und ein adäquater Einbezug von Benutzern in den Entwicklungsprozess möglich sind. Dazu ist ein iteratives und/oder inkrementelles Vorgehen bei der Software-Entwicklung notwendig.

6. **Ueberprüfung des Ansatzes**. Auch wenn viele UIMS/UIDS-Entwickler explizit die Eignung ihrer Werkzeuge für "Rapid Prototyping" und die Bedeutung dieser Methode betonen, scheinen die meisten Werkzeuge selber im stillen Kämmerchen entwickelt worden sein - Prototyping ist für alle gut, ausgenommen für sich selber. Gerade UIDS, welche oft auch komplexe interaktive Systeme darstellen, sollten aber ebenfalls in einem solchen Verfahren entwickelt werden. Bei der Entwicklung von IDEA stand der Einbezug von Benutzern und Tests von Vorversionen im Mittelpunkt. Dieses Vorgehen führte nicht nur in einem frühen Stadium zu einer sehr flexiblen Grundkonzeption des Werkzeugs (Zeltner 1989), sondern auch zu einer Reihe von Kritikpunkten (Beeli & Brunner 1990), welche zum Teil deutlich zur Verbesserung des Systems beitrugen. Die Ueberprüfung des Ansatzes darf sich nicht auf das Werkzeug beschränken, sondern muss sich auch auf das Entwicklungsvorgehen beziehen, in das dieses Werkzeug eingebettet ist. Dies heisst, dass der Entwicklungsansatz (Benutzereinbezug, enge Benutzer-Entwickler-Kooperation) in empirischen Untersuchungen überprüft werden muss, um allfällige kritische Faktoren zu identifizieren.

6. Literatur

Beeli, B. & Brunner, F. (1990). **Evaluation des Prototypingsystems HIDE/IDEA**. Semesterarbeit, Institut für Arbeitspsychologie, ETH Zürich, Betreuer T. Greutmann.

Brown, J.R. & Cunningham, S. (1989). **Programming the User Interface - Principles and Examples**. New York, Wiley.

Burgstaller, J., Grollmann, J. & Kapsner (1989). On the Software Structure of User Interface Management Systems. In: W. Hansmann, F.R.A. Hopgood & W. Strasser (Hrsg.). **EUROGRAPHICS '89**. Amsterdam, North-Holland, S.75-86.

Cardelli, L. (1988). Building User Interfaces by Direct Manipulation. In: **Proceedings of the ACM SIGGRAPH Symposium on User Interface Software**, Banff, Alberta, Canada, 17.-19. Oktober 1988, S.152-166.

Green, M. (1985). The University of Alberta User Interface Management System. **Computer Graphics**, Vol. 19, No. 3, Juli 1985, S.205-213.

Hayes, P.J., Szekely, P.A. & Lerner, R.A. (1985). Design Alternatives for User Interface Management Systems Based on Experience With COUSIN. In: L. Borman & B. Curtis (Hrsg.). **Human Factors in Computing Systems. Proceedings of the CHI'85 Conference**, 14.-18. April 1985, San Francisco, S.169-175.

Henderson, D.A. (1986). The Trillium User Interface Design Environment. In: M. Mantei & P. Orbeton (Hrsg.). **Human Factors in Computing Systems. CHI'86 Conference Proceedings**, 13.-17. April 1986, Boston, S.221-227.

Hill, R.D. (1986). Supporting Concurrency, Communication and Synchronization in Human-Computer Interaction - The Sassafras UIMS. **ACM Transactions on Graphics**, Vol. 5, No. 3, Juli 1986, S.179-210.

Hill, R.D. & Herrmann, M. (1989). The Structure of Tube - A Tool for Implementing Advanced User Interfaces. In: W. Hansmann, F.R.A. Hopgood & W. Strasser (Hrsg.). **EUROGRAPHICS '89**. Amsterdam, North-Holland, S.15-25.

Hix, D. (1989). Developing and Evaluating an Interactive System for Producing Human-Computer Interfaces. **Behaviour and Information Technology**, Vol. 8, No. 4, 1989, S.285-299.

Kasik, D.J. (1982). A User Interface Management System. **Computer Graphics**, Vol. 16, No. 3, Juli 1982, S.99-106.

Keller, R. (1989). **Prototypingorientierte Systemspezifikation. Konzepte, Methoden,**

Werkzeuge und Konsequenzen. Hamburg, Kovac.
Krähenmann, M., Kupferschmid, B. & Mayer, O. (1990). **Rapid Prototyping: Evaluation eines Entwurfswerkzeugs**. Semesterarbeit, Institut für Arbeitspsychologie, ETH Zürich, Betreuer T. Greutmann.
Myers, B.A. (1989). User-Interface Tools: Introduction and Survey. **IEEE Software**, Januar 1989, S.15-23.
Olsen, D.R. (1986). MIKE: The Menu Interaction Kontrol Environment. **ACM Transactions on Graphics**, Vol. 5, No. 4, Oktober 1986, S.318-344.
Olsen, D.R. (1987). Larger Issues in User Interface Management. **Computer Graphics**, Vol. 21, No. 2, April 1987, S.134-137.
Schulert, A.J., Rogers, G.T. & Hamilton, J.A. (1985). ADM - A Dialog Manager. In: L. Borman & B. Curtis (Hrsg.). **Human Factors in Computing Systems. Proceedings of the CHI'85 Conference**, 14.-18. April 1985, San Francisco, S.177-183.
Ulich, E. (1978). Ueber das Prinzip der differentiellen Arbeitsgestaltung. **Industrielle Organisation**, 1978, 47, S.566-568.
Ulich, E. (1983). Differentielle Arbeitsgestaltung - ein Diskussionsbeitrag. **Zeitschrift für Arbeitswissenschaft**, 1983, 37, S.12-15.
Ulich, E. (1986). Aspekte der Benutzerfreundlichkeit. In: W. Remmele & M. Sommer (Hrsg.). **Arbeitsplätze morgen**. Berichte des German Chapter of the ACM, Band 27. Stuttgart, Teubner, S.102-122.
Wirth, N. (1983). **Programming in Modula-2**. Berlin, Springer.
Zeltner, A. (1989). **Dialoggestaltung als erster Schritt beim SW-Entwurf**. Semesterarbeit, Institut für Arbeitspsychologie, ETH Zürich, Betreuer T. Greutmann.

Thomas Greutmann
Institut für Arbeitspsychologie
ETH Zürich
Nelkenstr. 11
CH-8092 **Zürich**