# MENUHANDLER - EIN ANWENDUNGSUNABHÄNGIGES MENÜ-INTERFACE

C. Hoffmann, W. Valder, St. Augustin

Zusammenfassung: Menüs erlauben dem Benutzer eines Software-Systems, aus einer vorgegebenen Menge von Alternativen eine Wahl zu treffen. In diesem Aufsatz wird eine flexible, anwendungsunabhängige Menü-Schnittstelle vorgestellt, die es erlaubt, viele Aspekte von Menüs während der Laufzeit der Anwendung anzupassen. Die Möglichkeit zur Anpassung umfaßt nicht nur die Repräsentation der Menüs, sondern auch ihre Struktur, die Mechanismen zur Selektion und die Information, die die Anwendung im Falle einer Selektion erwartet. Für komplexe Bereiche ist die Flexibilität des Systems Voraussetzung für den Einsatz an der Benutzerschnittstelle. Das Papier beschreibt die technische Konzeption für ein derartiges System.

## Einleitung

Bei der Benutzung von UNIX™ treten viele Probleme auf, die ihre Ursache in der Komplexität des Systems haben. Nicht nur die große Anzahl der zur Verfügung stehenden Werkzeuge und Objekte macht es für den Benutzer in der Regel schwer, für eine konkrete Aufgabe eine geeignete Auswahl zu treffen, sondern auch die Tatsache, daß es keine problemorientierte Struktur auf der Menge der Werkzeuge gibt (vgl. Draper, 1984). Eine Möglichkeit, die Komplexität des Systems beherrschbar zu machen, besteht darin, dem Benutzer die Werkzeuge des Systems problemorientiert in sogenannten Arbeitskontexten zu präsentieren (vgl. Hoffmann und Valder, 1986). In einem Arbeitskontext können die Erfahrungen hochgeübter Benutzer bei der Nutzung eines Systems anderen Benutzern verfügbar gemacht werden. Dazu werden die Werkzeuge und Objekte, die sich für ein Arbeitsverfahren als nützlich herausgestellt haben, den Benutzern präsentiert zusammen mit der für das Arbeitsverfahren relevanten Information. Eine geeignete Interaktionstechnik, die es dem Benutzer erleichtern kann, aus den zur Verfügung stehenden Alternativen eine geeignete Wahl zu treffen, ist die gezielte Verwendung von Menüs. Mit Hilfe von Menüs können in strukturierten Arbeitsbereichen die relevanten Werkzeuge und Objekte aufgabenorientiert präsentiert werden.

In diesem Aufsatz wird die technische Konzeption für ein System vorgestellt, das sowohl den Benutzern als auch den Entwicklern komplexer Anwendungssysteme den Umgang mit Menüs erleichtern soll. Dieses System, das wir MenuHandler nennen, gestattet die flexible Anpassung von Menüs an die sich ändernde Arbeitssituation des Benutzers. Diese Anpassung kann durch den Benutzer selbst durchgeführt werden. Er kann z.B.aus einer Menge unterschiedlicher Repräsentationen von Menüs die geeignete auswählen. Ebenso kann der Benutzer auf die Interaktion mit dem Menü-System Einfluß nehmen, z.B. ob er mit Maus oder Cursor-Tasten oder durch Angabe von Schlüsselwörtern aus den Menüs auswählen will. Insbesondere hat er jedoch die Möglichkeit, neue Menüs zu definieren, Menüs umzustrukturieren, und festzulegen, welche Information eine Anwendung als Ergebnis einer Selektion erhalten soll.

In komplexen Problembereichen ist der Gesichtspunkt der Flexibilität besonders wichtig, weil gerade hier vom Entwickler eines Anwendungssystems eine Zuordnung von definierten Aufgaben zu Werkzeugen nicht a-priori festgelegt werden kann. Häufig treten neue Probleme auf, für die dann kreativ neue Lösungen gefunden werden. Bei der Instrumentierung der Lösung müssen den spezifizierten Teilaufgaben geeignete Werkzeuge zugeordnet werden. Dieser Vorgang, bei dem das Werkzeuginventar während der Problemlösung dynamisch neu strukturiert wird, kann durch die Flexibilität des MenuHandlers unterstützt werden.

Damit der große Aufwand für die Entwicklung einer qualitativ hochwertigen Benutzerschnittstelle zu rechtfertigen ist, sollten die Komponenten der Benutzerschnittstelle möglichst anwendungsunabhängig entwickelt werden. Beispielsweise hat ein anwendungsunabhängiger MenuHandler den Vorteil, daß er in die Benutzerschnittstelle vieler Anwendungssysteme integriert werden kann.

Die Forderung, Komponenten der Benutzerschnittstelle möglichst anwendungsunabhängig zu entwickeln, wird erhoben, seit interaktive Systeme immer mehr an Bedeutung gewinnen und die Programmierung des Dialogs zwischen dem Benutzer und dem Computer sich immer mehr als eines der kritischen Probleme herausstellt. So haben Untersuchungen interaktiver Systeme gezeigt, daß bis zu 60% des Codes der Kommunikation zwischen dem Benutzer und dem System dienten (vgl. Benbesat & Wund, 1984).

Menüs

Ein Menü definiert eine Abbildung von einer Menge von Begriffen aus dem Anwendungsgebiet eines Systems (anwendungsspezifische Begriffe) in eine Menge von Begriffen der Systementwickler (systemspezifische Begriffe): die Elemente eines Menüs (die Menü-Items) sollen dem Benutzer die Auswahl von geeigneten Elementen des Systems erleichtern. Die Items beschreiben oft Tätigkeiten aus einem Anwendungsgebiet, denen eine Funktion (ein Kommando) des zugrundeliegenden Systems zugeordnet ist.

Beispiel: Bei einem System zur Unterstützung der Arbeit einer Sekretärin kann die Tätigkeit 'Briefe zeigen' im Menü abgebildet werden auf die systemspezifische Funktion 'ls /usr/sekr/letters'.

Die Anwendbarkeit von Menü-Systemen kann erweitert werden, wenn man die Benutzerselektion nicht nur auf Funktionen eines Systems beschränkt, sondern beliebige Objekte aus einem Anwendungsgebiet zuläßt.

Beispiel: Will die Sekretärin im o. a. Beispiel einen vorhandenen Brief weiterbearbeiten, so kann ihr, nachdem sie aus einem Menü die Tätigkeit 'Brief bearbeiten' ausgewählt hat, in einem Menü eine Liste aller relevanten Briefe angeboten werden, aus der sie eine Auswahl treffen kann.

Im folgenden werden verschiedene Arten von Menüs besprochen, für die eine Unterstützung durch das Menü-System vorgesehen ist. Weiterhin wird erläutert, welche Mechanismen zur Selektion dem Benutzer zur Verfügung gestellt werden sollen.

Verschiedene Arten von Menüs

Menüs lassen sich in drei Klassen einordnen (vgl. auch Shneiderman, 1987):

- Einzel-Menüs,
- hierarchische Menüs,
- eingebettete Menüs.

Einzel-Menüs

Einzel-Menüs erlauben dem Benutzer aus zwei oder mehr Items eine Wahl zu treffen. Menüs mit zwei Alternativen, binäre

Menüs, werden häufig benutzt, um eine Bestätigung vom Benutzer zu erhalten, z.B.:

'Wollen Sie die Änderungen abspeichern (J/N)?'

Die Zahl der Items in einem Menü ist jedoch nicht beschränkt; sogar Menüs, die über mehrere Bildschirmseiten gehen, sind vorstellbar. Auch die Zahl der Alternativen, die ein Benutzer aus einem Menü auswählen darf, ist nicht auf eine Selektion beschränkt. Es existieren Anwendungen, in denen es durchaus sinnvoll ist, mehrere Alternativen auszuwählen, die dann z.B. als Belegungen für die Parameter eines Kommandos interpretiert werden.

Eine wichtige Frage ist, ob ein Menü permanent auf dem Bildschirm zu sehen ist (permanente Menüs) oder ob ein Menü erst nach Anfrage des Benutzers auf dem Bildschirm dargestellt und nach einer Selektion wieder gelöscht wird (pop-up Menüs). Pop-up Menüs werden häufig verwendet, weil permanente Menüs viel Platz verbrauchen. Eine Kombination von permanenten Menüs und pop-up Menüs ist beim Apple Macintosh™ realisiert. Dort existiert eine Menü-Leiste am oberen Ende des Bildschirms, die die Namen aller zur Verfügung stehenden Menüs anzeigt. Durch das Aktivieren mit der Maus werden unterhalb des selektierten Menü-Namens die entsprechenden Items dargestellt. Nach einer Selektion wird das Menü bis auf den Namen in der Kopfzeile gelöscht.

Hierarchische Menüs

Wenn die Anzahl der Auswahlmöglichkeiten einer Anwendung sehr groß wird, dann ist schnell die Grenze der Anwendbarkeit von Einzel-Menüs erreicht. Sie werden für den Benutzer unübersichtlich und unhandhabbar. Die zur Auswahl anstehenden Alternativen können dem Benutzer dann mit Hilfe hierarchischer Menüs präsentiert werden. Die Selektion eines Item bedeutet in diesem Fall nicht immer die Auswahl eines Anwendungselements; als Folge einer Selektion kann auch ein weiteres Menü zur Selektion angeboten werden.

Hierarchische Menüs können verschieden strukturiert sein. Für bestimmte Problembereiche kann eine baumartige Struktur angemessen sein. Das bedeutet, jedes Menü hat genau ein Vorgänger-Menü. Vielfach ist jedoch eine flexiblere Struktur nötig: Menüs können dann auch allgemein netzartig strukturiert sein. In diesem Fall kann ein Menü auf verschiedenen Wegen erreicht werden.

Eingebettete Menüs

Manchmal ist es zweckmäßig, die Alternativen eines Menüs nicht in einer festen räumlichen Ordnung zueinander darzustellen, sondern die Items beliebig positioniert auf dem Bildschirm zu präsentieren. Diese Möglichkeit kann insbesondere dazu verwendet werden, beliebige Worte eines auf dem Bildschirm dargestellten Textes selektierbar zu machen. In diesem Fall spricht man von eingebetteten Menüs.

Als Beispiel kann ein Programm zur automatischen Überprüfung eines Textes auf Rechtschreibfehler dienen. Alle Worte eines Textes, die in einem Lexikon nicht vorkommen, werden als selektierbar gekennzeichnet. Wählt der Benutzer ein Wort aus, wird ihm in einem Menü eine Liste von Vorschlägen präsentiert, aus der er eine Alternative auswählen kann.

Mechanismen zur Selektion

Zur Selektion von Alternativen aus einem Menü stehen verschiedene Möglichkeiten zur Verfügung. Auf zeichenorientierten Terminals werden die Items in einem Menü häufig durch einen Buchstaben oder eine Ziffer gekennzeichnet. Zur Selektion gibt der Benutzer den entsprechenden Buchstaben (die entsprechende Ziffer) ein. Genauso ist es möglich, den Items eines Menüs Funktionstasten zuzuordnen: das Drücken einer Funktionstaste führt dann zur Selektion des betreffenden Item. Meistens wird die Selektion jedoch durch Positionierung des Cursors oder mittels Zeigeinstrument, wie z.B. der Maus, vorgenommen.

Bei hierarchischen Menüs kann es für den geübten Benutzer nützlich sein, wenn er durch gleichzeitiges Eingeben mehrerer Selektionen sofort zu einem bestimmten Menü kommt, ohne daß ihm alle Zwischenschritte gezeigt werden. Die kombinierte Selektion kann für den Benutzer den Charakter eines Kommandos bekommen. Ein weiterer Schritt in diese Richtung ist die Vergabe von Namen für Menüs: ein Menü wird dann durch Angabe des betreffenden Namens aktiviert. Genauso lassen sich Namen für Items definieren, so daß auch die Selektion von Items mittels Namen möglich ist. So kann ein großer Nachteil hierarchischer Menüs behoben werden; denn für einen erfahrenen Benutzer ist es mühsam und zeitaufwendig,

jedesmal eine Folge von Menüs durchzuarbeiten, wenn er aufgrund seiner Erfahrung mit dem System weiß, zu welchem Menü er gelangen will. Kann er jedoch Namen für Menüs und Items vergeben, hat er ein Mittel in der Hand, sich eine einfache Kommandosprache zu definieren.

Nach einer allgemeinen Diskussion der Anforderungen, die an einen MenuHandler als Komponente einer Benutzerschnittstelle gestellt werden, soll im folgenden eine technische Konzeption für einen MenuHandler beschrieben werden. Der MenuHandler wird unter UNIX™ auf einer SUN™ 3 entwickelt. Er ist Bestandteil einer Konzeption für eine anwendungsunabhängige Benutzerschnittstelle, die unter UNIX™ zur Verfügung gestellt werden soll.

Der MenuHandler

Der MenuHandler ist eine technische Komponente der Benutzerschnittstelle, die den gesamten menü-bezogenen Dialog mit dem Benutzer führen kann. Eine Anwendung, die Leistungen des MenüHandlers benötigt, muß zunächst eine Kommunikationsverbindung zu ihm herstellen. Daraufhin muß sie dem MenuHandler eine Beschreibung der von ihr verwendeten Menüs zur Verfügung stellen. Die weitere Kommunikation des Benutzers mit der Anwendung läuft dann über den MenuHandler: er interpretiert die Eingabe des Benutzers, führt an ihn gerichtete Kommandos aus und liefert das Ergebnis an die Anwendung (vgl. Abb.1). Für die Anwendung ist die Arbeitsweise des MenuHandlers transparent; die Anwendung kann nicht unterscheiden, ob die Daten, die sie erhält, die Eingabe des Benutzers ist, oder ob sie vom MenuHandler als Ergebnis einer Interaktion mit dem Benutzer erzeugt wurden.

Der MenuHandler besteht aus einem eigenen Kommando-Interpretierer, der die Eingabe des Benutzers analysiert, und der Menü-Daten-Struktur (MDS). In der MDS sind die durch die Anwendung übermittelten Menü-Beschreibungen festgehalten (vgl. Abb.2). Der Kommando-Interpretierer zerlegt die Eingabe des Benutzers in syntaktische Einheiten (Token). Für jedes Token wird überprüft, ob es sich um ein Kommando für den MenuHandler handelt. Dazu durchsucht der Kommando-Interpretierer die MDS nach dem entsprechenden Schlüsselwort. Wenn es gefunden wird, führt der MenuHandler die entsprechende Funktion aus. Falls es sich nicht um ein Kommando an

den MenuHandler handelt, wird das Eingabe-Token unverändert an die Anwendung weitergeleitet. Bei Kommandos, die zu einer Selektion führen, wird das entsprechende Token in der Eingabe durch das Ergebnis der Selektion ersetzt.

Beispiel: Die Eingabe 'Dateien listen' wird folgendermaßen vom MenuHandler bearbeitet (vgl. Abb.3). Das Token 'Dateien' wird als Schlüsselwort für ein Menü in der MDS entdeckt. Das Token 'listen' ist ein Schlüsselwort, das sich auf ein Item des selektierten Menüs bezieht. Die Benutzereingabe führt also zur Selektion eines Menü-Item. Als Ergebnis wird die Zeichenkette 'ls -l' an die Anwendung gesendet. Die Selektion erfolgt in diesem Fall also vollständig durch Kommandoeingabe ohne eine weitere Interaktion mit dem Benutzer.

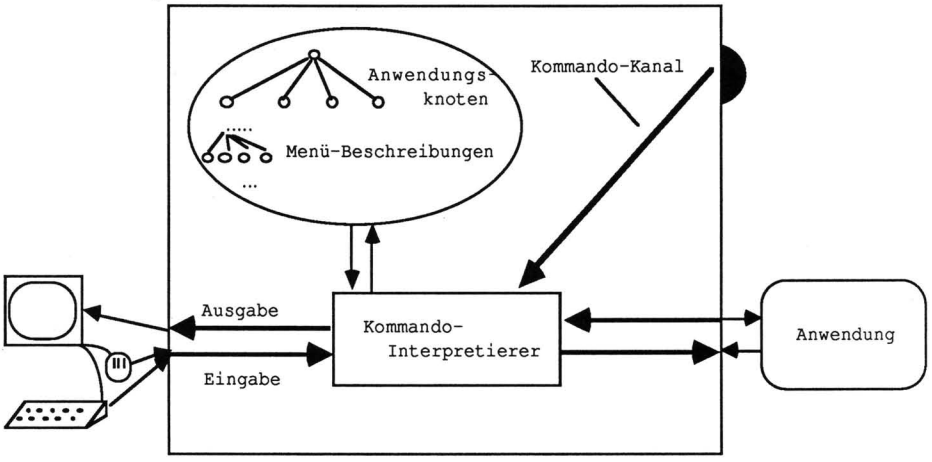
Lautet die Eingabe 'Dateien a.c b.c c.h' wird als Folge des Schlüsselworts 'Dateien' das entsprechende Menü dem Benutzer auf dem Bildschirm präsentiert. Selektiert der Benutzer dann z.B. das Item 'löschen', wird in der Eingabe des Benutzers das Token 'Dateien' durch das Ergebnis der Selktion ('rm ') ersetzt. Als Ergebnis wird an die Anwendung die Zeichenkette 'rm a.c b.c c.h' weitergeleitet.

Die durch die Ergebnisse von Selektionen erweiterte Eingabe wird erneut vom Kommando-Interpretierer analysiert. Auf diese Weise lassen sich hierarchische Menüs sehr leicht dadurch implementieren, daß das Ergebnis einer Selektion wieder ein neues Kommando zur Selektion in einem Menü ist.

Grundlage für die Menü-Behandlung bildet die MDS. In der MDS ist festgelegt, wie die von der Anwendung benötigten Menüs repräsentiert werden, wie die Selektion der Menü-Items im Dialog erfolgen soll und welche Ergebnisse von der Anwendung erwartet werden.

Eintragungen in die Menü-Daten-Struktur (MDS) können sowohl vom Benutzer als auch von der Anwendung verändert werden. Der Zugriff wird für den Benutzer durch Menüs ermöglicht, deren Beschreibung auch in der MDS abgelegt ist; d.h.zu jedem Zeitpunkt existiert in der MDS ein spezifischer Knoten, der die Menüs für den MenuHandler definiert. Zum Schutz vor unerlaubten Änderungen ist die Information selektiv mit Zugriffsrechten versehen.

Abb. 1: Komponenten des MenuHandlers




Die MDS ist baumartig strukturiert. Jede dem MenuHandler bekannte Anwendung definiert einen Anwendungsknoten. Jeder Anwendungsknoten ist Wurzel eines Unterbaums, der alle zur Verfügung stehenden Menüs der betreffenden Anwendung beschreibt. Zu jedem Menü wird gespeichert: (1) der Menütyp, (2) wie das Menü aktiviert wird, (3) Zugriffsrechte zur Manipulation der Menübeschreibung, (4) Zahl der gültigen Selektionen für dieses Menü. Jedes Menü besitzt wiederum einen Unterbaum, der die zugehörigen Items beschreibt. Zu jedem Item ist abgelegt: (1) die Repräsentation, (2) die gültigen Selektionsmechanismen (z.B. Namen, Funktionstasten), (3) was als Ergebnis der Selektion geliefert werden soll, und (4) wie evtl. zusätzliche Information beschafft werden kann, die zum Erzeugen des Ergebnisses der Selektion benötigt wird.

Der letzte Punkt bedarf einer Erläuterung. Bei der Definition eines Item kann nicht immer die vollständige Zeichenkette angegeben werden, die die Anwendung als Ergebnis der Selektion benötigt. Legt man eine Beschreibung der syntaktischen Struktur der als Ergebnis der Selektion zu liefernden Zeichenkette ab, kann der MenuHandler die syntaktische Korrektheit des Ergebnisses entsprechend dieser Beschreibung herstellen. Er stellt dazu ein Repertoire an Funktionen zur Verfügung.

Beispiel: Der Benutzer definiert ein Menü 'Dateien' (vgl. Abb.2); ein Item des Menüs lautet 'kopieren'. Als Ergebnis der

Selektion dieses Item soll das entsprechende Kommando für das Anwendungsprogramm generiert werden. Die Struktur der Ergebniszeichenkette ergibt sich aus der Syntax des Kommandos. In UNIX™ lautet die Beschreibung für das Kommando zum Kopieren einer Datei 'cp file1 [file2 ..] target' (vgl. UNIX86). Das bedeutet, daß ein korrektes Kommando mindestens zwei Dateinamen enthalten muß. Bei der Selektion des betreffenden Item müssen die fehlenden Informationen (Dateinamen) beschafft und in das Ergebnis integriert werden. Diese Information kann z.B. vom Benutzer in einem speziellen Fenster angefordert werden. Hat der Benutzer seine Eingabe abgeschlossen, wird als Ergebnis der Selektion die entsprechende Zeichenkette aufgebaut und der Anwendung zur Verfügung gestellt.

Abb.2: Zu einem Menü abgelegte Informationen

| Ergebnis-Muster | Menü | keywords |
|---|---|---|
| | Dateien | Dateien, files, datei |
| ls -l | listen | show, listen, list, zeigen |
| cp | kopieren | kopieren, copy |
| mv | umbenennen | rename, umbenennen |
| rm | löschen | löschen, delete, remove |
| pr *% \| lpr | drucken | drucken, print |

Durch die Verknüpfung von Funktionen an die Selektion eines Item wird erreicht, daß bestimmte syntaktische Überprüfungen bereits im MenuHandler ohne Eingreifen der Anwendung durchgeführt werden können. Dazu stellt die Anwendung eine Beschreibung der syntaktischen Struktur des Kommandos in der MDS zur Verfügung. Aufgrund dieser Beschreibung wird ein Ergebnis generiert, daß den syntaktischen Anforderungen genügt. Die Schritte, die nötig sind, um die syntaktische Korrektheit zu erreichen (z.B. durch Einholen von Information vom Benutzer), werden vom MenuHandler selbständig durchgeführt.

Dieses Konzept erlaubt es, bestimmte Überprüfungen außerhalb der Anwendung vorzunehmen. Über die beschriebene syntaktische Überprüfung hinaus ist es auch möglich, semantische Aspekte zu überprüfen, z.B. ob eine Eingabe eine gültige Datei

benennt oder ob ein Kommando 'Sinn macht' (z.B. 'cp a a'). Die Anwendung muß die zur Überprüfung notwendige Information bereitstellen, die Überprüfung selbst wird dann außerhalb der Anwendung vorgenommen.

## Zusammenfassung

Der MenuHandler ist eine Komponente einer Benutzerschnittstelle, die Benutzern unter UNIX™ angeboten werden soll. Alle Komponenten dieser Benutzerschnittstelle werden soweit wie möglich anwendungsunabhängig entwickelt. Außerdem soll dem Benutzer größtmögliche Flexibilität bei der Anpassung der Benutzerschnittstelle an seine konkrete Arbeitssituation ermöglicht werden. Am Beispiel des MenuHandler wurde gezeigt, wie man diese Anforderungen zu einem technischen Konzept entwickeln kann.

## Literaturverzeichnis

Benbesat, I. & Wund, Y.(1984): A Structured Approach to Designing Human-Computer Dialogues. International Journal for Man-Machine Studies (1984), Vol. 21.

Draper,S.W. (1984): The nature of expertise in UNIX. INTERACT'84, First IFIP Conference on 'Human-Computer Interaction', 4-7 September 1984, London, Volume 2, p.182-186.

Hoffmann, C. & Valder, W. (1986): Command Language Ergonomics. Foundation for Human-Computer-Interaction, K. Hopper and I.A. Newman (Editors), Elsvier Science Publishers B.V. (North-Holland), IFIP, 1986, p.218-234.

Shneiderman, B. (1987): Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley Publishing Company.

UNIX86: SUN Microsystems, Inc., Command Language Reference Manual, 19 September 1986.

Claus Hoffmann, Wilhelm Valder
GMD-F2.G2
Postfach 1240
D-5205 St. Augustin 1