Eine objekt- und fensterorientierte Bedienoberfläche für graphikfähige UNIX-Workstations

Ulrike Weng-Beckmann, München

Zusammenfassung

Im Rahmen des Projekts SWAP wird eine Software-Entwicklungsumgebung auf der Basis von UNIX realisiert. Besondere Aufmerksamkeit galt beim Entwurf auch den ergonomischen Eigenschaften, die für die Akzeptanz von Software-Systemen zunehmend an Bedeutung gewinnen. Ein wesentliches Merkmal ergonomischer Systeme ist eine konsistente Mensch-Maschine-Schnittstelle, die ein effizientes und situationsgerechtes Arbeiten ermöglicht. Im Rahmen von SWAP wurden durchgängige, leicht erlernbare Interaktionstechniken definiert, für deren Anwendung in Applikationen entsprechende Werkzeuge bereitgestellt werden. Das hier vorgestellte Werkzeug (SWAP-Shell) zielt darauf ab, die Interaktionstechniken bei der Bedienung von UNIX-System-funktionen einzusetzen. Insbesondere werden elementare Operationen unterstützt wie Navigieren im Dateibaum, Dateimanipulation und Programmausführung; weitere Funktionen können in konsistenter Weise integriert werden.

1. Zielsetzung

Im Rahmen des Projekts SWAP (Software-Arbeitsplatz für K PN) wird eine Software-Entwicklungsumgebung realisiert. Als Grundsystem wurde UNIX auf einer graphikfähigen, netzwerkbasierenden Workstation gewählt. Für UNIX als Basis eines Software-Arbeitsplatzes spricht eine Reihe von Gründen, wie die ursprüngliche Ausrichtung von UNIX auf die SW-Entwicklung, seine weite Verbreitung, die Verfügbarkeit auf unterschiedlichen Rechnerklassen, insbesondere auch auf allen gängigen Workstations sowie die weltweite Standardisierung. Nicht zuletzt gehören dazu die Flexibilität und Offenheit von UNIX, da Entwicklungssysteme in weit häufigerem Maße an geänderte Aufgabenstellungen und Benutzerprofile angepaßt werden müssen als andere Software.

Besondere Aufmerksamkeit galt bei der Konzeption des Systems auch den ergonomischen Eigenschaften, die für die Akzeptanz von Software-Systemen zunehmend an Bedeutung gewinnen (vgl. z.B. Bullinger (1985), Coutaz (1985), Shneiderman (1987)). Ein wesentliches Merkmal ergonomischer Systeme ist eine konsistente Mensch-Maschine-Schnittstelle, die ein effizientes und situationsgerechtes Arbeiten ermöglicht. Bei der Realisierung solcher Schnittstellen stellt der Einsatz von Graphik ein wichtiges Hilfsmittel dar.

Das von uns gewählte Basissystem UNIX verfügt über eine Bedienschnittstelle (Shell), die ergonomischen Maßstäben kaum standhält. Die Eignung der UNIX-Shell hängt in starkem Maße von der Routine des Benutzers im Umgang mit dem System, von Erfahrungen mit anderen Systemen und vom jeweiligen Aufgabengebiet ab. Für routinierte Entwickler ist sie fast unersetzlich, für andere fast unzumutbar. Beim Entwurf unserer Bedienschnittstelle war es deshalb ein besonderes Anliegen, daß sowohl UNIX-Laien als auch UNIX-Experten eine adäquate Arbeitsweise ermöglicht wird.

Voraussetzung für eine konsistente und homogene Mensch-Maschine-Schnittstelle ist insbesondere in einem offenen System die Festlegung von Richtlinien für die Gestaltung von Bedienoberflächen, die für alle Komponenten im System verbindlich sind. Ein User Interface Toolkit, das geeignete Bausteine bereitstellt, unterstützt die Einhaltung und Anwendung dieser Richtlinien. Im Projekt SWAP entstanden auf der Basis projektverbindlicher Konventionen (vgl. Zellner, 1987) zunächst die ***Diatools*** (Stork, 1987), die Bausteine zur Gestaltung objekt- und fensterorientierter Bedienober - flächen zur Verfügung stellen. Darauf basierend wurde die ***SWAP-Shell*** mit der Zielsetzung realisiert, die vorgegebenen Interaktionstechniken bei der Bedienung von Systemfunktionen einzu - setzen.

Der Schwerpunkt der SWAP-Shell liegt auf dem permanenten Sichtbarmachen der aktuellen Arbeitsumgebung sowie der effizienten, leicht erlernbaren Bedienung elementarer Operationen. Darüber hinaus wird keine zusätzliche Funktionalität angeboten. Die bereitgestellten Mechanismen werden einerseits als eigenständige Applikation (SWAP-Shell) angeboten, die anstelle einer UNIX Standard-Shell eingesetzt werden kann. Andererseits können sie aber auch von anderen Appli - kationen genutzt werden (SWAP-Shell-Lib).

Die Software ist ablauffähig auf Siemens Workstations WS30 unter UNIX System V (DOMAIN/IX) und setzt als Basis die Diatools voraus.

2. Leistungsumfang

Die SWAP-Shell realisiert eine einheitliche Bedienschnittstelle sowohl für UNIX-Funktionen als auch für beliebige, zu integrierende Applikationen. Für die Gestaltung der Schnittstelle stellen die Diatools eine Reihe von graphischen Dialogelementen zur Verfügung, auf deren Basis benutzer - gesteuerte Dialoge effizient und situationsgerecht realisiert werden können. Die wichtigsten in der SWAP-Shell verwendeten Gestaltungselemente sind:

- *Schreibtisch*
  Der Schreibtisch bildet den logischen Rahmen einer Diatools-Anwendung. Es können mehrere Schreibtisch-Anwendungen parallel existieren.
- *Fenster*
  Die anwendungsspezifischen Daten werden in Fenstern angezeigt und manipuliert. Innerhalb einer Anwendung können Fenster mit unterschiedlichen Eigenschaften verwendet werden. Es können mehrere Fenster gleichzeitig angezeigt werden, wobei einzelne Fenster nur teilweise oder garnicht sichtbar sein können. Eingaben werden nur dem jeweils aktiven Fenster zugeordnet. Fenster können i.a. geschlossen oder ikonisiert werden
- *Menüs*
  Über Menüs werden Funktionen der Anwendung aufgerufen. Je nach Verwendung haben Menüs fest vorgegebenen oder variablen Inhalt, werden sie an fester oder beliebiger Position ausgegeben. Pop-up-Menüs sind Objekten zugeordnet und enthalten jeweils die für die selektierten Objekte anwendbaren Operationen; sie werden nur auf Anforderung ausgegeben.
- *Formulare*
  Mittels Formularen wird der Dialog mit dem Benutzer realisiert. Man unterscheidet zwischen zwei Arten von Formularen: modale oder Pflichtformulare zwingen den Benutzer zur Bear - beitung bevor er in einen anderen Kontext wechseln kann. Nichtmodale Formulare kann der Benutzer zu einem beliebigen Zeitpunkt bearbeiten.

Über die von der SWAP-Shell angebotenen Interaktionsmechanismen können nur sehr elementare Funktionen, wie Navigieren im Dateibaum und Dateimanipulation, sowie allgemeine Dienste angesprochen werden; über eine Standard-Kommandoschnittstelle hat man darüber hinaus uneingeschränkten Zugriff auf die UNIX-Shell. Aufgrund des offenen Konzepts kann der vorgegebene Funktionsumfang in konsistenter Weise im Rahmen von Applikationen verändert werden. Über Voreinstellungen läßt sich die Arbeitsweise benutzerspezifisch anpassen.

Orientieren und Navigieren im Dateibaum sind elementare Funktionen, die die Effizienz des Arbeitens wesentlich beeinflussen. Die Orientierung im Dateisystem wird dadurch erleichtert, daß mehrere Kataloge gleichzeitig in jeweils einem Fenster (Kataloganzeigefenster) angezeigt werden können. Damit bleibt für den Benutzer relevante Information permanent präsent und inspizierbar. Zudem können auch gleichzeitig unterschiedliche Sichten auf einen Katalog dargestellt werden. Für das Navigieren im Dateibaum und die Bearbeitung von Dateien gibt es verschiedene Möglichkeiten, auf die in einem späteren Abschnitt noch eingegangen wird.

Die Bedienung der SWAP-Shell kann entweder objekt- oder funktionsorientiert erfolgen. Der Übergang zwischen diesen beiden Modi ist problemlos möglich, in gewissem Umfang wird auch ein gemischter Modus unterstützt. Objekte im Sinne der SWAP-Shell sind die Komponenten des Dateisystems, also Dateien und Kataloge. Die SWAP-Shell ordnet jedem Objekt eine Klasse (z.B. Katalog, normale Datei, ASCII-Datei) und eine Menge von zulässigen Operationen (z.B. Kopieren, Löschen, Übertragen) zu. Die Klassen bilden hinsichtlich ihrer Eigenschaften eine Hierarchie, in der Unterklassen eine Verfeinerung der übergeordneten Klassen darstellen.

Die Objekte werden in Kataloganzeigefenstern dargestellt und sind einzeln oder in Gruppen selektierbar. Die den selektierten Objekten zugeordneten Operationen werden in einem Pop-up-Menü angeboten. Einige Operationen können zusätzlich durch direkte Manipulation der Objekte ausgeführt werden.

Die funktionsorientierte Bedienung erfolgt über konventionelle Shell-Kommandos und Menü - techniken.

3. Komponenten der SWAP-Shell

In Bild 1 ist ein Beispiel für den Schreibtisch der SWAP-Shell dargestellt. Seine wichtigsten Kom - ponenten sind:

Menüleiste

Sie enthält Menüs (SWAP, File, Edit ...), über die allgemeine Funktionen zugänglich gemacht werden, wie z.B. Online-Help, Voreinstellungen, Verwaltung der Anzeigefenster oder Aufruf allgemeiner Dienste.

Steuerpult (Control Panel)

Das Steuerpult dient zum Anzeigen und Erfassen von Einstellungen in der SWAP-Shell und zum Ausführen bestimmter Funktionen der SWAP-Shell (z.B. Einrichten einer neuen Datei).

Das Steuerpult besitzt den Titel *Control Panel*. Es kann verschoben, aber nicht überdeckt werden. Über einen Eintrag im *Control*-Menü kann es aus- bzw. wieder eingeblendet werden, wenn die Arbeitsfläche auf dem Schreibtisch temporär vergrößert werden soll.

Kataloganzeigefenster

Der Inhalt von Katalogen wird jeweils in einem Anzeigefenster dargestellt; der Fenstertitel ist der vollqualifizierte Pfadname des Katalogs. Die Anzeige eines Kataloginhalts besteht aus der Auflistung aller Einträge zusammen mit ausgewählten Attributen. Die Anzeige kann modifiziert werden, indem andere oder weitere Attribute ausgegeben werden oder das Sortierkriterium verändert wird. Für die Anzeige kann zwischen drei verschiedenen Fontgrößen gewählt werden. Das Anzeigefenster enthält neben den Katalogeinträgen eine (fest vorgegebene) Aktionsleiste, in der Funktionen zum Verändern der Anzeige und zum Navigieren im Dateibaum zusammengefaßt sind. Die Überschriftszeile gibt Aufschluß über die angezeigten Dateiattribute.

Es können beliebig viele Anzeigefenster gleichzeitig existieren. Anzeigefenster, die man vorübergehend nicht benötigt, können ikonisiert werden. Die Titel der aktuell auf dem Schreibtisch dargestellten Anzeigefenster werden im *Windows*-Menü in der Menüleiste aufgelistet; bei Selektion eines Titels wird das zugehörige Fenster aktiviert und vollständig angezeigt. Über *Stack Windows* im selben Menü werden die Anzeigefenster auf dem Schreibtisch neu angeordnet.

Papierkorb (Trash)

Dateien werden gelöscht, indem sie in den Papierkorb übertragen werden, wo sie bis zum Ende der Sitzung verfügbar bleiben. Der Papierkorb ist ein spezieller Katalog, der für die Dauer einer SWAP-Shell-Sitzung eingerichtet wird. Die Papierkorb-Anzeige verhält sich m.E. wie andere Katalog-anzeigen. Das zugehörige Anzeigefenster mit dem Titel *Trash* ist standardmäßig ikonisiert.

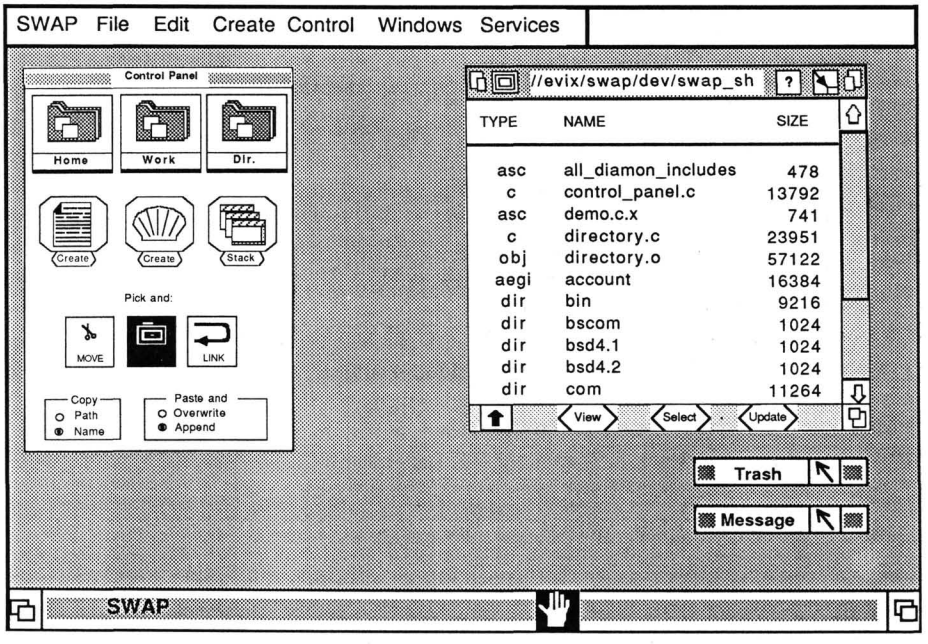

Bild 1: SWAP-Shell-Schreibtisch

Meldungsfenster (Message)

Für Meldungen von asynchron ablaufenden Vorgängen (d.h. Programmen, die in einem eigenen Prozeß ablaufen) steht das Meldungsfenster zur Verfügung. Die Meldungen werden protokolliert und im Fenster angezeigt. Sie können gelesen und kopiert werden; Änderungen sind nicht möglich.

Das Meldungsfenster hat den Titel *Message* und ist wie das Trash-Fenster standardmäßig ikonisiert. Bei Eintreffen einer Meldung wird das Icon invertiert dargestellt. Wenn das Fenster geöffnet ist, wird der Inhalt so verschoben, daß die zuletzt eingetroffene Meldung sichtbar ist.

Shell-Fenster

Die gewohnte UNIX-Bedienschnittstelle wird in Form einer Standard-Shell-Schnittstelle in einem eigenen Fenster angeboten. Damit erhält man auch Zugriff auf Funktionen, die über die Interaktionsmechanismen der SWAP-Shell nicht unterstützt werden. Es werden mehrere Shell-Fenster zugelassen.

Ein Shell-Fenster stellt neben einer Textzeile zur Kommandoeingabe ein Transcript-Pad zur Verfügung, in dem Ein- und Ausgaben der Shell protokolliert werden. Die protokollierten Daten können wie beim Meldungsfenster gelesen und kopiert werden.

4. Arbeitsweise

4.1 Prinzipien der Arbeitsweise

Der Dialog mit der SWAP-Shell wird vom Benutzer gesteuert, der zwischen verschiedenen Fenstern, Menüs und Arbeitsmodi beliebig wechseln kann. Eine Ausnahme stellen nur die modalen Formulare dar; ihr Einsatz beschränkt sich aber fast ausschließlich auf die Anzeige von Fehlermeldungen, auf die der Benutzer sofort reagieren muß (Warnformulare).

Die von der SWAP-Shell direkt unterstützten Funktionen wie Auflisten von Kataloginhalten, Dateimanipulationen oder Ändern von Dateiattributen laufen synchron innerhalb der SWAP-Shell ab. Dabei gelten die von UNIX vorgegebenen Konventionen und Festlegungen. Alle anderen Anwendungen, Werkzeuge und benutzerspezifizierten Funktionen werden durch separate Prozesse ausgeführt und laufen asynchron ab. Die SWAP-Shell übernimmt nur das Erzeugen der Prozesse, den Aufruf der Programme und die Verarbeitung evtl. Meldungen bzw. Ergebnisse. Shell-Fenstern ist jeweils ein interaktiver Shell-Prozeß zugeordnet, der die Kommandos ausführt.

Ein wesentliches Merkmal der SWAP-Shell ist ihre Flexibilität. Sie kann sowohl funktionell an unterschiedliche Anwendungen) als auch über Voreinstellungen an individuelle Benutzeranforderun - gen angepaßt werden. Voreinstellungen sind pro Benutzer möglich. Für eine spätere Ausbaustufe ist das Wiederaufsetzen auf einem abgespeicherten Zustand geplant.

4.2 Kataloganzeige

Die Darstellung von Kataloganzeigen kann variiert werden; sie wird bestimmt wird durch Angabe der anzuzeigenden Attribute, des Sortierkriteriums und der Fontgröße. Diese Festlegungen werden zunächst aus der Voreinstellung übernommen und können anschließend beliebig verändert werden. Sie gelten jeweils für neu erzeugte Anzeigefenster.

Die Sicht auf den aktiven Katalog kann in analoger Weise modifiziert werden. Die Fenstergröße wird dabei entsprechend angepaßt; sie kann auch vom Benutzer im Rahmen vorgegebener Minimal- und Maximalgrößen verändert werden. Über den Inhalt des Fensters kann vertikal gescrollt werden, wobei Überschriftszeile und Aktionsleiste fest bleiben.

Die in einem Kataloganzeigefenster dargestellte Sicht gibt den Zustand des jeweiligen Katalogs zu einem bestimmten Zeitpunkt wider. Diese Sicht wird nur dann aktualisiert, wenn Änderungen unter der Kontrolle der SWAP-Shell ablaufen. Änderungen, die von anderen Prozessen oder anderen Benutzern im Netz verursacht werden, werden von der SWAP-Shell nicht automatisch erkannt und deshalb in der Anzeige auch nicht nachgezogen. Der Benutzer kann allerdings die Aktualisierung der Anzeige jederzeit selbst veranlassen (*update*-Knopf in der Aktionsleiste).

Die mögliche Inkonsistenz in der Kataloganzeige stellt einen Kompromiß dar, der zu Gunsten kurzer Reaktions- und Anzeigezeiten in Kauf genommen wird. Die einzige Möglichkeit einer konsistenten Anzeige besteht darin, die Informationen zu den angezeigten Kataloginhalten in sehr kurzen, periodischen Abständen vom System abzufragen und bei Änderungen die Anzeige zu aktualisieren. Das Beschaffen dieser Dateiinformationen (i-Nodes) ist aber ein relativ zeitaufwendiger Prozeß und würde zu unakzeptablen Reaktionszeiten führen.

4.3 Ändern von Dateiattributen

Die angezeigten Dateiattribute können, soweit sie änderbar sind, direkt manipuliert werden. Dazu wird ein Attribut zunächst selektiert, indem erst das Gesamtobjekt und dann die entsprechende Spalte in diesem Objekt selektiert wird. Folgende Attribute können geändert werden:

Dateiname Benutzername Gruppenname Zugriffsrechte Mod. Datum

Das Ändern des gewählten Attributs ist anschließend durch einfaches Überschreiben möglich (mit Ausnahne des Mod. Datums).

4.4 Navigieren im Dateibaum

Das Arbeiten im Dateibaum wird zunächst dadurch erleichtert, daß mehrere Kataloge gleichzeitig angezeigt werden können.Basis für das Navigieren ist der Pfadnamen von Dateien, der festlegt, wie die Datei von der Wurzel des Dateibaums aus erreicht werden kann. Die SWAP-Shell erleichtert das Navigieren, indem sie das Durchlaufen bestimmter Pfade direkt unterstützt; darüber hinaus können Home und Working Directory direkt über Menüs im *Control Panel*, beliebige andere Kataloge durch Angabe ihres Pfadnamens geöffnet werden. Jeder Katalog wird in einem neu erzeugten Fenster angezeigt.

Von einem angezeigten Katalog aus kann direkt in die erste Hierarchiestufe des darunter liegenden Unterbaums verzweigt werden und damit, durch Wiederholung des Vorgangs, der gesamte Unterbaum durchlaufen werden. Zudem kann jeder im Pfad darüber liegende Katalog direkt geöffnet werden: die SWAP-Shell bietet dazu in der Aktionsleiste der Anzeigefenster ein Menü (*dirup*), dessen Einträge die entsprechenden Pfad-Komponenten repräsentieren (vgl. Bild 2). Die Auswahl eines Eintrags bewirkt das Öffnen und Anzeigen des zugehörigen Katalogs.

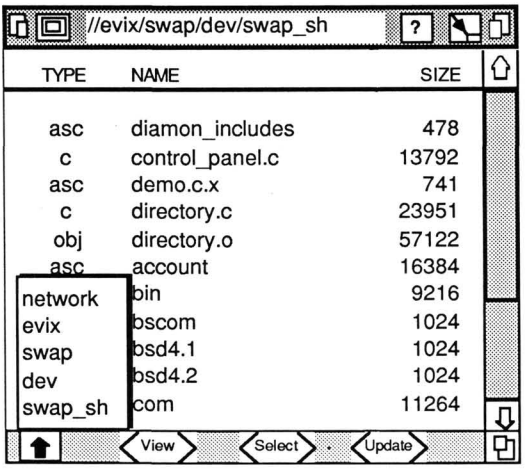

Bild 2: *dirup*-Menü im Katalog *//evix/swap/dev/swap_sh*

4.5 Dateioperationen

Die SWAP-Shell ordnet jedem Objekt eine Klasse (Typ) und eine Menge von Operationen zu. Die Festlegung der Klassen und der zugehörigen Operationen ist konfigurierbar; die Basis-Konfiguration ist vorgegeben. Die in den Fenstern angezeigten Objekte können selektiert und entsprechend ihres Typs verarbeitet werden. Die Verarbeitung erfolgt durch Auswahl einer Operation im zugehörigen Pop-up-Menü. Zusätzlich können gewisse elementare Operationen durch direkte Manipulation ausgeführt werden:

- Doppelklick auf ein Objekt der Klasse Katalog bzw. ASCII-Datei = *open / read*
  Anzeigen eines Katalogs in einem neuen Anzeigefenster bzw. einer ASCII-Datei in einem Read-Pad. Für andere Objektklassen: keine Aktion.
- Ziehen eines Objekts in ein Kataloganzeigefenster = *move / copy / link*
  Übertragen bzw. Kopieren in den Zielkatalog bzw. Einrichten einer Referenz im Zielkatalog (je nach Voreinstellung). Es gelten die Konventionen der entsprechenden UNIX-Kommandos.
- Ziehen eines Objekts in ein editierbares Textfeld = *copy name*
  Kopieren des Pfadnamens des Objekts in das Textfeld
- Klick auf ein selektiertes Objekt
  Modifizieren des entsprechenden Attributs, abhängig von der Spalte, auf die der Cursor zeigt.

Neben Einfachselektion ist auch Mehrfachselektion von Objekten möglich. Während bei der Einfach - selektion das zugehörige Pop-up-Menü durch die Objektklasse festgelegt ist, wird bei Mehrfachselek - tion ein festes Menü verwendet. Die angestossene Operation wird allerdings nur für die Objekte aus der Selektionsmenge ausgeführt, für die sie zulässig ist.

4.6 Arbeiten mit der Shell

Mit dem Shell-Fenster wird eine kommando- und zeilenorientierte UNIX-Bedienschnittstelle geboten, die in konventioneller Weise das Arbeiten mit der System V Shell erlaubt. Darüber hinaus wird ein Protokoll-Mechanismus (*Historie*) realisiert, der den Zugriff auf zurückliegende Kommandos und auf Meldungen der Shell ermöglicht.

Jedem Shell-Fenster ist ein interaktiver Shell-Prozeß zugeordnet, an den die Eingaben in die Kommandozeile des Shell-Fensters geschickt werden. Die Eingabe wird von der Shell in der üblichen Weise verarbeitet, Ergebnisse und Meldungen werden zurück an die SWAP-Shell geschickt und im Historienteil des zugehörigen Shell-Fensters zusammen mit der Eingabe aus der Kommandozeile ausgegeben.

Um vollständige Konsistenz und uneingeschränkte Funktionalität einer Standard Shell-Umgebung zu gewährleisten, werden alle Eingaben direkt vom zugehörigen Shell-Prozeß ausgeführt. Damit hat die SWAP-Shell keine Kontrolle über das, was der Benutzer im "Shell-Modus" arbeitet und die Anzeige bleibt davon unberührt. Um Statusänderungen in einem Shell-Prozeß in der SWAP-Shell-Anzeige berücksichtigen zu können, wäre die Reimplementierung der Shell oder zumindest die vollständige Analyse und teilweise Auswertung der Shell-Sprache notwendig gewesen. Bei der Mächtigkeit und Komplexität der Shell-Sprache ist dies aber kaum ohne Konsistenz- oder Funktionaliätsverlust denkbar.

5. Definition von Klassen

Basis für die Offenheit und Flexibilität des Systems ist das auf dem Dateisystem basierende Klassenkonzept: Dateien und Kataloge (Objekte) lassen sich bzgl. der auf den Objekten zulässigen Operationen in Klassen einteilen. Die Klasseneinteilung wird auf unterster Ebene vom UNIX-Datei - system unterstützt; weitere Klassen werden im Rahmen von Anwendungen definiert.

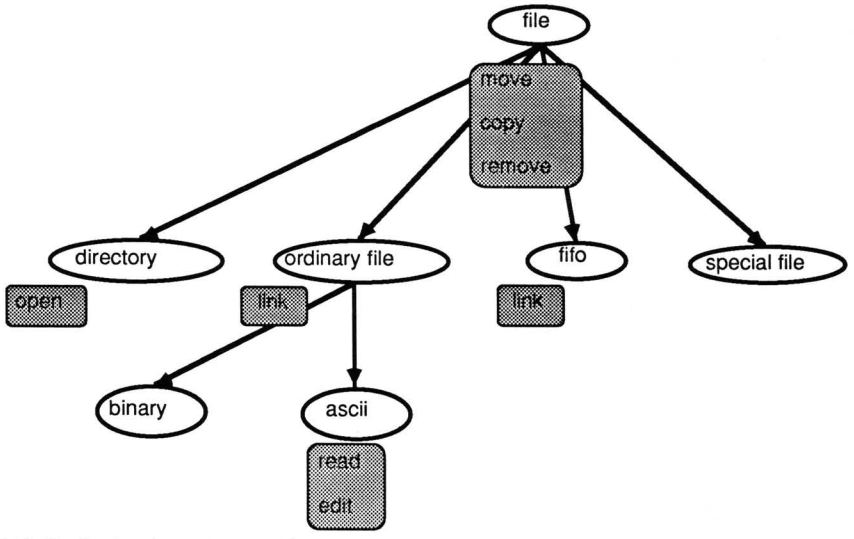

Bild 3: Basis-Konfiguration der Objektklassen

Die Definition der Klassen erfolgt außerhalb der SWAP-Shell. Klassendefinitionen werden in ASCII-Dateien gespeichert und durch ein eigenständiges Werkzeug bearbeitet, das auch die von der SWAP-Shell benötigten Informationen bereitstellt. Durch die Klassendefinition wird u.a. festgelegt, welche Klassen die SWAP-Shell kennt, wie sie Objekte einer Klasse zuordnet und welche Operationen auf die Objekte dieser Klassen anwendbar sind.

Die Klassen bilden eine Hierarchie, für die eine Basis-Struktur fest vorgegeben ist (vgl. Bild 3). Die Operationen, die den Knoten im Baum zugeordnet sind, sind auf alle Objekte der zugehörigen Klasse und, falls nicht explizit ausgeschlossen, aller Unterklassen anwendbar. Die für eine Klasse gültigen Operationen werden bestimmt, indem der Baum von der Wurzel bis zum Knoten, der die Klasse repräsentiert, durchlaufen wird und die zugehörigen Operationen kummuliert werden.

Für die Festlegung der Klassen wird ein Klasseneditor zur Verfügung gestellt, der die Darstellung und Manipulation der Klassenbäume in einfacher und effizienter Weise erlaubt. Mit seiner Hilfe kann eine bestehende Klassen-Hierarchie in graphischer Baumdarstellung angezeigt und über direkte Manipulation und Menütechniken bearbeitet werden.

Literaturhinweise

Bullinger, H.J. (1985):
Software Ergonomie'85. Mensch-Computer-Interaktion.
Teubner Verlag, Stuttgart, 1985

Coutaz, J. (1985):
Abstraction for User Interface Design.
Computer, September 1985, pp.21-34

Fößmeier, R. (1988):
Objektorientiertheit auf der UNIX-Kommandoebene.
GUUG Nachrichten Nr. 13, April 1988, S. 28-35

Shneiderman, B. (1987):
Designing the User Interface.
Addison-Wesley, Reading, 1987

Stork, Burkhard; Zellner, Rudolf (1987):
Architektur von Bedienoberflächen auf grafikfähigen Workstations unter UNIX
GUUG-Jahrestagung 1987, 22.-24. September, Karlsruhe

Zellner, R. (1987):
Oberflächengestaltung des Softwarearbeitsplatzsystems. Konzept und Richtlinien.
Interner Bericht (Siemens AG, München) 1987

Ulrike Weng-Beckmann
Siemens AG, ZFE F2 SOF 1
Otto-Hahn-Ring 6
D-8000 München 83