Diplomarbeit
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Softwaretechnik

Konzeption und Realisierung eines an der Anwendungssprache orientierten Hilfesystems nach der Werkzeug-Material Metapher

Carola Lilienthal

Juni 1995

Betreuung

Prof. Dr. Christiane Floyd
Universität Hamburg

Prof. Dr. Heinz Züllighoven
Universität Hamburg

Diese Diplomarbeit wurde dem Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zur Erlangung des Titels Diplom-Informatikerin eingereicht.

Erklärung

Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, den

Carola Lilienthal

Heckscherstr. 6

20253 Hamburg

Betreuung

Prof. Dr. Christiane Floyd (Erstbetreuerin)

Prof. Dr. Heinz Züllighoven (Zweitbetreuer)

Prof. Dr. Christiane Floyd

Prof. Dr. Heinz Züllighoven

Fachbereich Informatik

Arbeitsbereich Softwaretechnik

Universität Hamburg

Vogt-Kölln-Str. 30

22527 Hamburg

Inhaltsverzeichnis

Einleitung 1

I Grundlagen 5

1 Hilfesysteme 5
1.1 Warum ist Hilfe notwendig? 5
1.2 Komponenten eines Hilfesystems 6
1.3 Handhabungskomponente eines Hilfesystems 8
1.4 Erklärungskomponente eines Hilfesystems 12
1.5 Zusammenfassung 15

2 Werkzeug-Material Metapher 17
2.1 Ebenen der Werkzeug-Material Metapher 17
2.2 Interpretations- und Gestaltungsmuster 18
2.3 Hilfesysteme aus der Sicht der Werkzeug-Material Metapher 20

3 Transparenz und Sprache 25
3.1 Sprache 25
3.2 Transparenz 27
3.3 Förderung von Orientierung 30
3.4 Hilfesysteme aus der Sicht von Transparenz und Sprache 33

II Prozeß 35

4 Projektbericht 35
4.1 Fragestellung des Projekts 35
4.2 Projektverlauf 36
4.3 Aufgabennetze zur aufgabenbezogenen Anforderungsermittlung 38
4.4 Empirische Untersuchung 42

5 Prototyp des Hilfesystems 47
5.1 Überblicksvision 47
5.2 Handhabungsvisionen 48
5.3 Bewertung des Prototyps 53

III Produkt **55**

6 Das Hilfesystem **57**

6.1 Zusammenspiel der Komponenten des Hilfesystems 57

6.2 Hilfetext 57

6.3 Hilfewerkzeug 65

6.4 Koordination wiederverwendbarer Subwerkzeuge 68

7 Kontextsensitivität **71**

7.1 Hilfesystem mit Kontextsensitivität 71

7.2 Schnittstelle zwischen Hilfewerkzeug und Applikation 72

7.3 Der Kontext 74

7.4 Aktivierung des Hilfesystems aus einer Applikation 78

8 Das Hilfe-Framework **81**

8.1 Werkzeuge zur Erstellung der Hilfematerialien 81

8.2 Handhabungsvisionen der zusätzlichen Werkzeuge 82

8.3 Verbindung zur Applikation 85

8.4 Koordination von Werkzeugen mit mehreren Materialien 87

8.5 Das Hilfe-Framework im Entwicklungsprozeß 88

Zusammenfassung und Ausblick **91**

Anhang **95**

Einleitung

Hilfesysteme sind bei der Gestaltung von Softwaresystemen häufig ein Randthema. Sie werden, wenn überhaupt, erst entwickelt, nachdem die *eigentliche* Software schon erstellt ist. Dabei wird das Potential, das Hilfesysteme für die Softwareentwicklung bieten könnten, nicht ausgenutzt. Vielmehr stehen die resultierenden Hilfesysteme ihrem eigentlichen Sinn - Benutzerinnen[1] dabei zu unterstützen, Software zu verstehen und zu verwenden - häufig entgegen.

Im Fachbereich Informatik an der Universität Hamburg befaßt sich der Arbeitsbereich Softwaretechnik unter anderem mit der Entwicklung von anwendungsnahen, benutzungsgerechten Softwaresystemen. In diesen Bereich fällt auch die Entwicklung von Hilfesystemen. Ein wichtiger Ausgangspunkt bei der Softwareentwicklung ist die von Heinz Züllighoven et al. entwickelte Werkzeug-Material Metapher (s. [BudZül90, BüGrZü92]). Sie stellt sowohl ein Leitbild für Benutzungsschnittstellen als auch Entwurfsmuster für die Konstruktion zur Verfügung.

In der vorliegenden Arbeit wird ein *Rahmenwerk für Hilfe* vorgestellt, das nach der Werkzeug-Material Metapher entwickelt wurde. Unter einem Rahmenwerk wird hier eine Menge von kooperierenden Klassen verstanden, die ein wiederverwendbares Design für eine spezifische Art von Softwareprodukten bilden (s. [GaHeJoVl95]). Während der Entwicklung bildete sich für das Rahmenwerk der Name *Hilfe-Framework* heraus, so daß er im folgenden durchgehend verwendet wird.

Das Hilfe-Framework besteht aus einem *Hilfesystem* und einer *Arbeitsumgebung für Hilfe*. Beide Teile machen es möglich, daß das Hilfesystem unabhängig von einem bestimmten Anwendungssystem konzipiert werden konnte. Damit das Hilfesystem mit einem Anwendungssystem kooperieren kann, muß die Anwendungsentwicklerin in der Arbeitsumgebung für Hilfe alle hilferelevanten Komponenten, wie z.B. Hilfetexte, für ihr System zusammenstellen. Das Anwendungssystem und das spezifische Hilfesystem werden der Benutzerin *gemeinsam* übergeben. Um die Software zu verstehen und zu verwenden, kann die Benutzerin auf das Hilfesystem und die hilferelevanten Komponenten zurückgreifen. Das Hilfe-Framework deckt somit zwei unterschiedliche *Einsatzkontexte* ab: die Verwendung des Hilfesystem mit verschiedenen Anwendungssystemen durch die *Benutzerin* und die Erstellung von hilferelevanten Komponenten durch die *Entwicklerin*. In seiner Ausgestaltung legt das Hilfe-Framework die Sichtweise nahe, daß die zukünftigen Benutzerinnen des Hilfesystems *mündige* Personen sind, die ein eigenes Verständnis von ihrer Arbeit haben.

Die Arbeit am Thema Hilfesysteme begann, als am Arbeitsbereich Softwaretechnik der Universität Hamburg im Fachbereich Informatik der Wunsch nach einem Hilfesystem für eine dort verwendete Software aufkam. Bei dieser Software handelt es sich um den Aufgabennetzeditor SANE (**S**ystem für einen **A**ufgaben**n**etz**e**ditor, s. [Riehle93]), der im Rahmen der Vorlesung "Einführung in die Softwaretechnik" zur aufgabenbezogenen Anforderungsermittlung eingesetzt wird (s. [Floyd92]). Für den Aufgabennetzeditor wurde wie für das Hilfe-Framework die Werkzeug-Material Metapher als softwaretechnisches Entwurfskonzept verwendet.

---

[1] Ich verwende in dieser Arbeit durchgehend die weibliche Form. *Die Benutzerin* oder *die Benutzerinnen* werden als Rollen verstanden. Zur besseren Lesbarkeit wird auf andere umständliche Formulierungen verzichtet. Spreche ich von wirklichen Personen, so berücksichtige ich das Geschlecht.

Die sprachliche Ausgestaltung von Hilfetexten ist für die späteren Benutzerinnen des Hilfesystems von großer Bedeutung. In bezug auf Hilfetexte hat für diese Arbeit besondere Relevanz, in welcher Weise Sprache verwendet wird, um in Hilfetexten die Beziehung zwischen Anwendungsgebiet und Software zu charakterisieren. Aus diesem Grund wurde bei der Konzeption des Hilfe-Frameworks auf sprachliche Richtlinien zurückgegriffen; darüber hinaus wurden neue Grundsätze erarbeitet. Der auf diese Art entstandenen Konzeption liegt ein Verständnis von Computersystemen als sprachliche Artefakte zugrunde (s. [KleLyy92, NakSch93]).

Die gesamte Arbeit gliedert sich in drei Abschnitte: Grundlagen, Prozeß und Produkt. Der erste Teil führt in die Themengebiete Hilfesysteme, Werkzeug-Material Metapher sowie Transparenz und Sprache ein, die die Basis für die Entwicklung des Hilfe-Frameworks gebildet haben.

Kapitel 1 setzt sich kritisch mit der Literatur zum Thema Hilfesysteme auseinander und liefert eine eigenständige Definition. In der Definition werden Komponenten von Hilfesystemen identifiziert und Eigenschaften festgelegt, die sich als sinnvoll und einlösbar erwiesen haben. Entscheidend hierbei ist, daß das Hilfesystem passive, dynamische und individuelle Hilfe in Form von Hypertexten anbietet, die das Verständnis der Entwicklerinnen von Anwendungsgebiet und die Handhabung an der Benutzungsschnittstelle erklären. Auf der Basis dieser Definition wird eine Anforderung an Hilfesysteme erstellt.

In Kapitel 2 wird die Werkzeug-Material Metapher erläutert und ihre für das Hilfe-Framework wesentlichen Aspekte beschreibt. Dabei wird deutlich wie die verschiedene Muster der Werkzeug-Material Metapher, auf die Konzeption und Realisierung des Hilfe-Frameworks Einfluß genommen haben. Anhand der Werkzeug-Material Metapher kann die in Kapitel 1 aufgestellte Anforderung an Hilfesysteme um grundlegende Aufgaben für Hilfesysteme erweitert werden und die einzelnen Komponenten des Hilfesystems in die vorhandene Terminologie eingeordnet werden.

Kapitel 3 stellt die sprachtheoretische Grundlage und das Konzept der Transparenz vor. Der Ausgangspunkt für die sprachtheoretischen Betrachtungen bildet die Gleichsetzung von Computersystemen und symbolischen Maschinen (s. [Krämer93]). Um den Benutzerinnen eines Computersystems die Interpretation der symbolischen Maschine zu ermöglichen, wird auf die softwaretechnische Forderung nach Transparenz in der Benutzungssituation und auf die sich daraus ergebenden Mitteln zur Förderung von Transparenz zurückgegriffen. Diese Mittel lassen sich im Rahmen von Hilfesystemen einsetzen und werden in die Anforderung an Hilfesysteme aus Kapitel 1 und 2 aufgenommen.

Aufbauend auf der in Teil I erarbeiteten Anforderung an Hilfesysteme konnte ein Projekt durchgeführt werden, in dessen Verlauf ein Prototyp entwickelt wurde. Im zweiten Teil dieser Arbeit wird daher das Projekt, das zu der Entwicklung des Hilfe-Frameworks geführt hat, beschreiben und aus der Prozeßperspektive [Floyd87] betrachtet.

Kapitel 4 umfaßt einen Projektbericht, der einerseits die Fragestellung des zugrundeliegenden Projektes klärt und den zeitlichen Verlauf des Projekts schildert. Auf der anderen Seite wird in das Anwendungsgebiet des Aufgabennetzeditors SANE eingeführt, mit dem das Hilfe-Framework zum Einsatz kommen soll. Abschließend wird eine empirische Untersuchung vorgestellt, die mit Studierenden während ihrer Arbeit mit dem Aufgabennetzeditor SANE durchgeführt wurde. Die Voraussetzungen der Untersuchung, die

Durchführung und die Ergebnisse werden erläutert und ihr Einfluß auf die Konzeption des Hilfe-Frameworks herausgearbeitet.

Kapitel 5 stellt den im Laufe des Projekts entstandenen Prototyp des Hilfesystems in seiner aktuellen Version vor. Anhand von Visionen wird die Arbeit von Benutzerinnen mit dem im Hilfe-Framework integrierten Hilfesystem beschrieben, und so eine Vorstellung von der Benutzungsschnittstelle dieses Teils des Hilfe-Frameworks vermittelt. Zum Abschluß des Kapitels wird der Prototyp anhand der im Grundlagenteil entwickelten Anforderung an Hilfesysteme bewertet und mögliche Erweiterungen aufgezeigt, die in nachfolgenden Versionen umgesetzt werden sollten.

Der dritte Teil der Arbeit ist dem gesamten Hilfe-Framework als Produkt gewidmet. Hier wird der fachliche Entwurf des Hilfe-Frameworks in seiner vorliegenden Version anhand von verschiedenen Ausbaustufen geschildert.

Kapitel 6 befaßt sich mit dem im Hilfe-Framework enthaltenen Hilfesystem. Die einzelnen Komponenten des Hilfesystems - Hilfewerkzeug und Hilfematerial - werden detailliert vorgestellt. Die Gestaltungsrichtlinien für das Hilfematerial, die im Grundlagenteil und in der empirischen Untersuchung erarbeiteten worden sind, werden zusammengetragen und umgesetzt. Mit Hilfe der Muster aus der Werkzeug-Material Metapher und von anderer Seite definierten Mustern (s. [Gamma92, GaHeJoVl95]) wird der technische Entwurf des Hilfematerials und des Hilfewerkzeugs sowie die sich daraus ergebenden kooperierenden Klassen beschreiben. Um den Benutzerinnen des Hilfewerkzeugs Orientierung bei der Arbeit mit dem Hilfesystem zu bieten, wurde das Prinzip der Koordination wiederverwendbarer Subwerkzeuge entwickelt, das am Ende dieses Kapitels vorgestellt wird.

Kapitel 7 untersucht, wie das Hilfesystem kontextsensitiv auf Benutzungssituationen reagiert, und auf welche Weise dieser Aspekt im Rahmen des Hilfe-Frameworks umgesetzt wurde. Die kontextsensitive oder dynamische Reaktion des Hilfesystems auf eine Benutzungssituation setzt voraus, daß das Hilfesystem Zugang zum Anwendungssystem und seiner aktuellen Situation hat. Zu diesem Zweck muß ein Zustandsbegriff eingeführt werden, der für Hilfesituationen aussagekräftig ist. Darüber hinaus benötigt das Hilfesystem eine abstrakte Schnittstelle zu Anwendungssystemen, um entkoppelt mit unterschiedlichen Anwendungssystemen zusammenarbeiten zu können. Die dazu erforderliche Schnittstelle zwischen dem Hilfesystem und verschiedenen Anwendungssystemen wird vorgestellt und die Arbeit des Hilfesystems mit einem Anwendungssystem beschrieben.

Kapitel 8 führt in die Arbeitsumgebung für Hilfe ein, die das Hilfe-Framework anbietet. Um das Hilfesystem für ihr Anwendungssystem anzupassen, müssen Entwicklerinnen Hilfetexte auf der Basis von Hyperstrukturen formulieren und ein Material - Kontextmaterial - erstellen, das eine dynamische Reaktion des Hilfesystems auf den Zustand des Anwendungssystems erlaubt. Um dieses Kontextmaterial anlegen zu können, ist demzufolge ebenfalls eine Schnittstelle zum Anwendungssystem notwendig, die im einzelnen vorgestellt wird. Da es sich bei vielen Dokumenten, die im Entwicklungsprozeß entstehen, um Dokumente handelt, die wie Hilfetexte aus hyperartigen Strukturen bestehen, wird am Ende dieses Kapitels die Stellung des Hilfe-Frameworks im Entwicklungsprozeß betrachtet. Es ergeben sich Möglichkeiten für seinen Einsatz und die übergreifende Verwendung von Dokumenten.

Den Abschluß der Arbeit bildet eine Zusammenfassung der wichtigsten Ergebnisse und ein Ausblick auf Themen und Probleme, die während dieser Arbeit zu Tag traten, aber in diesem Rahmen nicht mehr bearbeitet oder gelöst werden konnten.

Ich danke Christiane Floyd und Heinz Züllighoven für ihre engagierte und ermutigende Betreuung. Mein besonderer Dank gilt Yvonne Dittrich, Martina Wulf und allen anderen für die vielen Stunden anregender Diskussion.

I Grundlagen

Die Ergebnisse dieser Arbeit bauen auf drei verschiedenen Themenbereichen auf: Hilfesysteme, die Werkzeug-Material Metapher und Transparenz. Um die Ergebnisse verständlich zu machen, werden zuerst diese Gebiete vorgestellt. Die für Hilfesysteme relevanten Aussagen der drei Gebiete fließen in eine Anforderung an Hilfesysteme ein, die im Laufe der drei Kapitel erarbeitet wird.

1 Hilfesysteme

Um die Beschäftigung mit Hilfesystemen grundsätzlich zu motivieren, wird in diesem Kapitel einleitend die Frage untersucht, warum Hilfe bei der Arbeit mit Computern überhaupt benötigt wird. Nachdem geklärt wurde, warum eine Auseinandersetzung mit Hilfesystemen lohnend erscheint, wird eine erste Annäherung an den Begriff "Hilfesystem" unter Berücksichtigung der vorhandenen wissenschaftlichen Diskussion vorgenommen. Die daraus resultierende Untergliederung von Hilfesystemen in zwei Teile wird beschrieben und als Strukturierung für die weiteren Abschnitte verwendet. Daran anschließend werden in der Literatur vorhandene Aussagen zu Hilfesystemen diskutiert und bewertet. Einerseits werden verschiedene Ansätze vorgestellt, die die Frage behandeln, auf welche Weise ein Hilfesystem arbeiten sollte. Andererseits hält jedes Hilfesystem textuelle oder visuelle Erklärungen für die Benutzerinnen bereit, die auf unterschiedliche Weise gestaltet werden können. Abschließend werden die gewonnenen Aussagen in einer "Anforderung für Hilfesysteme" zusammenfaßt.

1.1 Warum ist Hilfe notwendig?

Im Bereich der Software Ergonomie und der Mensch-Maschine Interaktion wird die Meinung vertreten, daß Computersysteme, die über gute Benutzungsschnittstellen verfügen, weitestgehend selbsterklärend sind. Als Idealfall wird angestrebt, daß Benutzerinnen sich ohne fremde Hilfe neue Programme erschließen können. Benutzungsgerecht gestaltete Systeme und eine leicht verständliche, weitgehend selbsterklärende Schnittstelle sollten das *Ziel* jeder Softwareentwicklung sein. Doch trotz intensiver Bemühungen in dieser Richtung ist immer wieder zu beobachten, daß selbst Systeme mit software-ergonomisch entworfener Schnittstelle der Benutzerin Probleme bereiten. Zwei Gründe für die auftretenden Probleme werden im folgenden diskutiert.

Bei Computersystemen, die Benutzerinnen in verstärktem Maße Schwierigkeiten bereiten, handelt es sich i.d.R. um Systeme, die zur Unterstützung von komplexen Anwendungsgebieten entwickelt wurden[2]. Jede Entwicklerin eines solchen Systems wird mit dem Problem konfrontiert, daß die verschiedenen Benutzerinnen des zukünftigen Systems kein eindeutig bestimmbares, gleichbleibendes Verständnis von ihrem Anwendungsgebiet haben [KleLyy92]. Vielmehr hat jede Benutzerin ihre eigenen, ganz persönlichen Vorstellungen von den Zusammenhängen und Wertigkeiten ihrer Arbeit. Hinzu kommt, daß das Verständnis der Benutzerinnen einem ständigen Wandel unterworfen ist. Im Laufe einer Softwareentwicklung verstärkt sich dieser Effekt sogar noch, weil die Benutzerinnen durch die Gespräche mit den Entwicklerinnen vermehrt auf anderen Ebenen über ihre Arbeit nachdenken [KiGrZü94]. Aus dieser vorhandenen und sich wandelnden Bandbreite an Interpre-

[2] In der weiteren Erörterung wird davon ausgegangen, daß die Entwicklerin eines Systems direkten Kontakt zu den zukünftigen Benutzerinnen hat. Ist dies nicht der Fall, so treten die diskutierten Probleme in verstärktem Maße auf.

tationen des Anwendungsbereiches müssen sich die Entwicklerinnen ein Verständnis bilden, das, wenn möglich, einen für alle Benutzerinnen akzeptablen Konsens darstellen kann. In [Piepenburg94] wird dies als "konventionelles" Verständnis bezeichnet. Das schließlich entstehende Computersystem wird, gutes Design vorausgesetzt, dieser konventionellen Interpretation des Anwendungsgebiets entsprechen. Für alle Benutzerinnen vom Grundsatz her verständlich, wird das Computersystem trotzdem nicht in allen Teilbereichen gleichermaßen und ohne Probleme nutzbar sein. Daher kann erwartet werden, daß Benutzerinnen, selbst wenn sie mit einem speziell für sie entwickelten System arbeiten, nicht ohne Hilfe auskommen.

Abgesehen von den Differenzen im Verständnis des Anwendungsgebiets handelt es sich bei Computersystemen um relativ komplexe Gegenstände im Vergleich zu anderen Dingen des täglichen Gebrauchs. Jeder Mensch kann aus eigener Erfahrung bestätigen, daß der Gebrauch von Dingen *erlernt* werden muß. Weiterhin ist ständiges Üben erforderlich, und es können stets neue Fähigkeiten im Umgang mit Gegenständen hinzugewonnen werden, selbst wenn sie bereits länger in Gebrauch sind. Für Computersysteme, die Gegenstände wie Stifte oder Schreibmaschinen an Komplexität bei weitem überschreiten [DuPaMe92], ist daher die Forderung nach Selbsterklärung in diesem Zusammenhang nicht haltbar. Benutzerinnen werden bei Computersystemen immer wieder Unterstützung benötigen, um die jeweilige Handhabung zu erlernen.

Es kann also erwartet werden, daß Benutzerinnen sowohl mit dem Verständnis als auch mit der Handhabung eines Computersystems Schwierigkeiten haben. Diese Schwierigkeiten können auf verschiedene Art und Weise beseitigt werden. Kommt eine Benutzerin das erste Mal mit einem System in Kontakt, so sollte sie sich durch eine persönliche oder notfalls durch eine computergestützte Einführung (Lernprogramm) mit dem System vertraut machen können. Arbeiten Benutzerinnen längere Zeit mit einem Computersystem, so vertieft sich das Verständnis und weiterführende Fragen treten auf. Auch in diesen Fällen ist eine *Person* als Unterstützung allen anderen Möglichkeiten vorzuziehen, weil Menschen aufgrund ihrer Kommunikationsfähigkeit und durch die Übersicht, die sie über die gesamte Situation haben, gegenüber anderen Lernmitteln stets im Vorteil sind [Suchman87]. In vielen Fällen sind geeignete Personen mit dem entsprechenden Wissen nicht erreichbar, so daß Benutzerinnen auf andere Mittel zurückgreifen müssen. Zu diesem Zweck werden sowohl Handbücher als auch Hilfesysteme angeboten. Da die Gestaltung von Handbüchern ein zwar verwandtes Thema, aber in dieser Arbeit nicht von zentraler Bedeutung ist, wird sie im weiteren außer Acht gelassen. Eingehender untersucht wurde die Gestaltung von Handbüchern in [FloGeb91].

1.2 Komponenten eines Hilfesystems

In der Literatur[3] wird der Begriff "Hilfesystem" vielfach verwendet, es fehlt aber eine Definition. In diesem Abschnitt wird daher eine Definition für Hilfesysteme entwickelt, um einen Ausgangspunkt für die vertiefende Diskussion einzelner Aspekte zu gewinnen.

Die Hilfe, die ein Hilfesystem anbietet, bezieht sich immer auf ein Anwendungssystem (im folgenden Applikation genannt), von dem aus das Hilfesystem aufgerufen wurde. Bei der Entwicklung eines Hilfesystems muß entschieden werden, ob das Hilfesystem programm-

[3] siehe [AarCar8 (6a|6b|7|8), Bauer88, BauSch88, DuPaMe92, Hoffmann88, Horton90, Kearsley88, SzeNec94, Thies94, Willms92]

technisch *unabhängig* von einer Applikation sein soll oder ob es in eine bestimmte Applikation *integriert* werden sollte. Unabhängige Hilfesysteme können für verschiedene Applikationen verwendet werden und erlauben dadurch eine konsistente Handhabung für *ein* Hilfesystem, das eine Menge von Applikationen unterstützt. Im Rahmen dieser Arbeit werden nur applikationsunabhängige Hilfesysteme betrachtet, weil nur sie den softwaretechnischen und software-ergonomischen Forderungen nach Kapselung, Abstraktion, Wiederverwendbarkeit und einheitlicher Schnittstellengestaltung genügen.

Aus den Ausführungen in der Literatur kann geschlossen werden, daß bei Auseinandersetzungen mit Hilfesystemen stets über *zwei* Komponenten nachgedacht werden muß: die Erklärungskomponente, die die zugehörige Applikation textuell und visuell beschreibt, und die Handhabungskomponente, die den Zugriff auf die Erklärungskomponente möglich macht. Für die Erklärungskomponente werden unterschiedliche Begriffe verwendet, wie Hilfeinformation [Bauer88], Wissensbasis [Thies94], Inhalt [DuPaMe92, Kearsley88] und Texte [Willms92, Hoffmann88]. Da alle Autorinnen und Autoren außerdem den Begriff "Erklärungen" einsetzten, wird er in dieser Arbeit durchgehend benutzt.

Die Aufspaltung in Handhabungs- und Erklärungskomponente hat sich als sinnvoll erwiesen, weil sich ein Hilfesystem in bezug auf die Erklärungen von anderen Systemen unterscheidet. Normalerweise entwerfen Software-Entwicklerinnen Programme, die Benutzerinnen verwenden sollen, um mit Hilfe der Programme an Dokumenten oder Datenbeständen zu arbeiten. Ein Hilfesystem dagegen wird vornehmlich zu dem Zweck entwickelt, bereits vorhandene Erklärungen anzuzeigen. Die *Zusammenstellung* der applikationsspezifischen Erklärungen verlangt daher genauso viel Aufmerksamkeit wie der Entwurf der Handhabungskomponente. Sowohl die Handhabungs- als auch die Erklärungskomponente sind daher für die Verwendbarkeit eines Hilfesystems gleich wichtig. Ist das Hilfesystem in der Benutzung unhandlich, weil die Handhabungskomponente schlecht entwickelt wurde, so verfehlt es seinen Sinn, Hilfe zu geben. Sind die Erklärungen unklar und ungenau, so versagt das Hilfesystem ebenfalls.

Um eine Grundlage für die weitere Diskussion bereitzustellen, werden diese Aussagen in einer Definition zusammengefaßt.

Definition 1-1:

> Ein **Hilfesystem** ist programmtechnisch unabhängig von einer Applikation und besteht aus zwei Teilen: einer Handhabungs- und einer Erklärungskomponente. Die **Handhabungskomponente** eines Hilfesystems ermöglicht der Benutzerin den Zugriff auf die **Erklärungskomponente**, welche Hilfe zu einer Applikation enthält.

Im weiteren wird untersucht, ob die in der Literatur vorgeschlagenen Eigenschaften von Handhabungs- und Erklärungskomponente umgesetzt werden können und inwieweit sie dazu beitragen, daß ein Hilfesystem seiner Aufgabe gerecht werden kann. Da das Ziel dieses Abschnitts darin besteht, das in dieser Arbeit vertretene Verständnis von Hilfesystemen zu klären, wird zuerst nicht auf die konkrete konstruktive Umsetzung der verschiedenen Ideen eingegangen. Die Umsetzung der am Ende dieses Kapitels formulierten Forderungen wird vielmehr in den Teilen II (Projekt) und III (Produkt) beschrieben und anhand der aufgestellten Forderungen evaluiert.

1.3 Handhabungskomponente eines Hilfesystems

Die Handhabungskomponenten von Hilfesysteme können auf verschiedene Art und Weise gestaltet sein. Die Bandbreite reicht von Online-Manuals, die wie gedruckte Handbücher verwendet werden können, bis zu Hilfesystemen, die jede Benutzerin individuell unterstützen und von selbst ihre Hilfe anbieten. Abbildung 1-1 gibt eine Klassifikation wieder, die in Anlehnung an [BauSch88] erstellt wurde. Unter Berücksichtigung von [Bauer88, DuPaMe92, Kearsley88, Hoffmann88, Thies94, Willms92] wurden einige Änderungen vorgenommen, die im folgenden diskutiert werden.

passiv wird durch eine explizite Anfrage der Benutzerin ausgelöst.	**aktiv** wird ausgelöst, wenn das Hilfesystem feststellt, daß Hilfe benötigt wird.
statisch liefert Information, die den aktuellen Kontext nicht berücksichtigt.	**dynamisch** liefert Information, die den Kontext zum Zeitpunkt der Hilfeanforderung berücksichtigt.
uniform liefert für jede Benutzerin dieselbe Information.	**individuell** liefert Information, die an die speziellen Bedürfnisse der Benutzerin angepaßt ist.

Abb. 1-1: Klassifikation von Hilfesystemen

Die in ihrer Grobstruktur von Bauer und Schwab eingeführte Klassifikation hat in ihrer ursprünglichen Form eine etwas andere Definition für passive und aktive Hilfe. Dabei vermischen die Autoren die Ebenen der Kontextsensitivität und der Art, wie das Hilfesystem aktiviert wird.

Bauer und Schwab geben bei passiver Hilfe zusätzlich zu der Definition in Abbildung 1-1 an, daß die gelieferte Information den aktuellen Kontext nicht berücksichtigt. Diese Definition kann, muß aber nicht zutreffen, denn auch ein passives Hilfesystem, das nur auf Anfrage der Benutzerin aktiv wird, kann den Kontext der problematischen Applikation in Betracht ziehen.

Aktive Hilfe wird von Bauer und Schwab über die oben gegebene Definition hinaus als Hilfe verstanden, die den Kontext zum Zeitpunkt der Hilfeanforderung berücksichtigt. Diese Definition ist dahingehend irreführend, daß ein aktives Hilfesystem selbständig in Aktion tritt und in diesem Zusammenhang den Kontext einbeziehen muß, um *festzustellen*, daß Hilfe benötigt wird. Die bauer-schwab'sche Definition spricht aber vom Berücksichtigen des Kontextes zum Zeitpunkt der *Hilfeanforderung*.

In der abgewandelten Klassifikation lassen sich drei Ebenen für die Betrachtung von Hilfesystemen finden:

- **passiv und aktiv** geben Auskunft darüber, auf welche Art die Handhabungskomponente eines Hilfesystems mit der Benutzerin in Kontakt tritt,
- **statisch und dynamisch** machen eine Aussage über die Kontextsensitivität der Handhabungskomponente eines Hilfesystems und
- **uniform und individuell** liefern Angaben darüber, ob die Handhabungskomponente eines Hilfesystems über Profile für Benutzerinnen verfügt.

In der wissenschaftlichen Diskussion um Hilfesysteme werden die oben genannten Eigenschaften von Hilfesystemen und ihre Durchführbarkeit nicht hinterfragt. Entweder wird ein Hilfesystem relativ pragmatisch anhand der technischen Möglichkeiten entwickelt[4] oder es wird davon ausgegangen, daß alle Eigenschaften erreichbar sind, wenn nur entsprechend gute Systeme gebaut werden[5]. Auch die Sinnhaftigkeit der verschiedenen Eigenschaften wird häufig außer Acht gelassen. Um abwägen zu können, welche Eigenschaften für ein Hilfesystem geeignet sind bzw. welche sich überhaupt realisieren lassen, werden die herausgearbeiteten Gegensätze im folgenden einer genaueren Betrachtung unterzogen.

1.3.1 Aktive versus passive Hilfe

Bei der Verwendung von Systemen gibt es hin und wieder Situationen, in denen sich Benutzerinnen im Rückblick wünschen, jemand hätte sie rechtzeitig auf eine "fehlerhafte" Verwendung des Systems aufmerksam gemacht. Eine Lösung dieses Problems wäre ein Hilfesystem, das in der Lage ist, *aktiv* Ratschläge und Erklärungen zu geben [BauSch88]. Dieser Wunsch nach Hilfe kann von einem Computersystem nicht erfüllt werden. Bis heute sind derartige Systeme nicht entwickelt worden, und einige Wissenschaftskreise[6] zweifeln daran, daß ein solches Unterfangen jemals von Erfolg beschieden sein wird. Diese Behauptung läßt sich aus verschiedenen Blickwinkeln begründen.

L. Suchman [Suchman87] macht in ihrem Buch "plans and situated actions" deutlich, daß Computer nie imstande sein werden, jeden auftretenden Situationszusammenhang eines Menschen bei der Arbeit mit einem Computer zu erkennen. Selbst wenn es gelänge, alle möglichen Wörter und Aktionen aufzuzählen, und dieses Wissen in einem Computer abzulegen, wäre dieser nicht in der Lage, die Informationen zu *verstehen*, weil die zusätzlich benötigten Hintergrundinformationen nicht eingrenzbar sind. Jeder Versuch, die zugrundeliegenden Annahmen (Hintergrund) einer Äußerung als Teil ihres semantischen Kontextes zu integrieren, geht von der Grundlage aus, daß es eine *feststell-* oder *festlegbare* Menge von Annahmen gibt, die einer gegebenen Äußerung zugrunde liegt. Diese Vorstellung von Äußerungen und ihrem abgegrenzten Kontext ist nicht haltbar, weil jede gefundene Hintergrundannahme weitere Annahmen erzeugt, die ebenfalls erarbeitet werden müssen. Diese Eigenschaft ist typisch für jede situierte Handlung von Menschen. Suchman verwendet hierfür den Begriff "situated action" und unterstreicht damit die Sichtweise, daß

---

[4] siehe [Horton90, DuPaMe92, Kearsley88]

[5] siehe [Bauer88, BauSch88, Thies94]

[6] s. [Suchman87, AarCar8 (6a|6b|7|8), BudZül90, Robinson93]

jede Aktion im wesentlichen ad hoc ist und von den bearbeiteten Gegenständen und den sozialen Umständen abhängt.

Darüber hinaus weist Suchman darauf hin, daß ein menschlicher Partner im Gegensatz zum Computer in der Lage ist, das Umfeld der anderen Person miteinzubeziehen. Ein System dagegen kann nur die mit ihm getätigten Aktionen interpretieren. Für das System nicht sichtbar ist die wirkliche Arbeit der Benutzerin, wenn sie Symbole auf dem Bildschirm interpretiert oder weiterführende Referenzen identifiziert. Das System sieht nur die Ergebnisse dieser Arbeit. Es hat daher eine stark eingeschränkte "Sicht" auf die Arbeit der Benutzerin.

Diese Aussagen stimmen mit den Ergebnissen von mehreren Untersuchungen überein, die A. Aaronson und J. M. Carroll in ihren Arbeiten beschrieben haben (s. [AarCar8 (6a|6b|7|8)]). In diesen Untersuchungen wurden mehrere Experimente durchgeführt, bei denen Benutzerinnen und Benutzer mit verschiedenen Systemen arbeiteten und Hilfe von unterschiedlichen Instanzen bekamen. Im ersten Experiment gaben Menschen persönlich Auskunft, im zweiten konnte Hilfe per E-mail angefordert werden und im dritten wurde die Hilfe von einem aktiven Hilfesystem angeboten, hinter dem sich ohne Wissen der Benutzerinnen und Benutzer ein Mensch verbarg.

Für das letzte Experiment wurden Hilfetexte vorbereitet, die die Person, die das Hilfesystem simulierte, an die Benutzerinnen und Benutzer verschicken sollte. Darüber hinaus war es dieser Person gestattet, in Ausnahmefällen individuelle Hilfetexte zu erzeugen. Wie sich herausstellte, mußten trotz sorgfältiger Erstellung der vorgefertigten Hilfetexte in über 55% der Fälle individuelle Hilfetexte verwendet werden, weil eine Reihe von Situationen eintraten, mit denen im Voraus nicht gerechnet worden war. Außerdem hatte die das Hilfesystem simulierende Person große Probleme aus den Aktionen der Benutzerinnen und Benutzern zu schließen, ob und welche Hilfe erforderlich war. Diese Probleme übertrafen bei weitem die Schwierigkeiten der Experten, die den Benutzerinnen und Benutzern in den anderen Experimenten zur Seite standen. So hatten die Benutzerinnen und Benutzer selbst bei diesem "menschlichen" Hilfesystem ähnliche Probleme, wie sie auch von rechnerinternen Systemen verursacht werden: die Hilfe kam zu spät, wenn die Lösung bereits gefunden worden war; die Benutzerinnen und Benutzer warteten auf Hilfe und bekamen keine oder die Hilfe störte und war sinnlos, so daß sie von den Benutzerinnen und Benutzern ignoriert wurde.

Auch Budde et al. [BudZül90] machen deutlich, daß eine Entwicklerin nicht vorhersehen kann, zu welchem Zweck eine Benutzerin eine Applikation verwenden will (s. auch [Robinson93, RaeVel95]). Daher sollten weder in einer Applikation noch in einem Hilfesystem Annahmen darüber getroffen werden, was die Benutzerin mit ihrer Tätigkeit bezweckt und ob die Benutzerin damit noch ein *sinnvolles* Ziel verfolgt oder nicht.

Aus den hier ausgeführten Argumenten läßt sich entnehmen, daß keine Möglichkeit besteht, alle Situationen zu ermitteln, in denen Hilfe erforderlich sein kann, und daß der Computer nur eine sehr eingeschränkte "Sicht" auf die Situation der Benutzerin hat. Aktive Hilfe ist daher ein Konzept, das nur eingeschränkt verwirklicht werden kann. Die heute vorhandenen Systeme, die als aktive Hilfesysteme bezeichnet werden, bereiten den Benutzerinnen verschiedenartige Probleme, die sich selbst mit noch besseren Systemen nur teilweise werden ausräumen lassen.

Auf der Basis des hier zugrunde gelegten Menschenbilds und den daraus angeführten Gründen sind nur *passive* Hilfesysteme ein sinnvolles und realisierbares Ziel bei der Softwareentwicklung.

1.3.2 Statische versus dynamische Hilfe

Statische Hilfe hat gegenüber dynamischer Hilfe den großen Nachteil, daß der Benutzerin in jeder Anwendungssituation immer die selbe Information angeboten wird. In diesem Fall verhält sich die Handhabungskomponente des Hilfesystems wie ein Handbuch mit erweiterten Suchfunktionen und verbesserten Darstellungsmöglichkeiten. Um dies Potential von Computersystemen auszunutzen, sollte die Handhabungskomponente eines Hilfesystems fähig sein, auf das Wissen zurückzugreifen, das die von der Benutzerin verwendete Applikation zur Verfügung stellen kann. Die dort erhältlichen Informationen beschränken sich in der Regel auf Angaben über den *Zustand* der Applikation und über die *Aktionen der Benutzerin* bis zum Zeitpunkt der Hilfeanforderung. Eine Auswertung der Aktionen der Benutzerin schließt sich aus den selben Gründe aus wie das Erkennen einer Hilfesituation bei aktiven Hilfesystemen. Auch hier hat die eingeschränkte "Sicht" des Hilfesystems auf die Arbeit der Benutzerin zur Folge, daß eine Vorhersage aller potentiell Situationen unmöglich ist.

Um der Benutzerin mehr als statische Hilfe anbieten zu können, bleibt daher nur die Möglichkeit, ein *dynamisches* Hilfesystem zu entwickeln, das Angaben über den *Zustand* der Applikation verwendet, mit der die Benutzerin arbeitet.

1.3.3 Uniforme versus individuelle Hilfe

Uniforme Hilfesysteme bieten jeder Benutzerin dieselbe Hilfe unabhängig von deren Kenntnisstand und Erfahrung. Individuelle Hilfe dagegen unterscheidet entweder verschiedene Gruppen von Benutzerinnen (z.B. Anfängerinnen und Expertinnen) oder modelliert die Eigenschaften einer jeden Benutzerin individuell.

Selbst wenn individuelle Hilfe ein wünschenswertes Konzept wäre, könnte die Handhabungskomponente eines Hilfesystems diese Forderung nicht ohne weiteres erfüllen. Die Einteilung von Benutzerinnen in Gruppen ist schwierig, wenn nicht unmöglich, weil die Wissensstände in verschiedenen Arbeitszusammenhängen zu große Unterschiede aufweisen [DuPaMe92]. Die Alternative, alle Eigenschaften von Benutzerinnen individuell einzuschätzen, ist ebenfalls problematisch, weil die Informationen für diese Einschätzung aus sehr begrenztem Material gewonnen werden müssen (s. Abschnitt 1.3.1: eingeschränkte "Sicht").

Eine weiterführende Art individueller Hilfe sind adaptive Systeme, die sich an die Erfahrung der Benutzerinnen anpassen. Diese Systeme ändern je nach Einschätzung der Benutzerin die Menge der zur Verfügung gestellten Informationen. Diese Art von Hilfesystemen ist für die Benutzerin äußerst verwirrend, weil es passieren kann, daß ihr in zwei gleichen Situationen unterschiedliche Erklärungen angeboten werden. Darüber hinaus ist auch hier anzuführen, daß Systeme nicht in der Lage sind, den Wissenstand von Benutzerinnen individuell einzuschätzen. Systeme, die in diesem Sinn individuell vorgehen, sind daher wie aktive Hilfesysteme weder sinnvoll noch realisierbar.

Wenn man die Betrachtungsweise umkehrt und es der Benutzerin überläßt zu entscheiden, welche Information sie benötigt, kann man Hilfe auch als *individuelle* Hilfe betrachten. In

diesem Fall entscheidet nicht das System, was einer Anfängerin oder einer Expertin zugänglich ist und was nicht, sondern die Benutzerin wählt explorativ aus, welche Information sie zur Zeit in Anspruch nehmen möchte. So kann sich die Benutzerin im Hilfesystem individuell mit Information versorgen. Weitreichendere Möglichkeiten, Hilfe individuell zu gestalten, bestehen darin, daß die Benutzerin auf Wunsch Notizen schreiben und Lesezeichen an für sie wichtige Informationen anbringen kann. Auf diese Weise kann die Benutzerin die angebotenen Informationen selbständig erweitern und sich eine *eigene* Sicht auf die Informationen erarbeiten.

Anhand der hier angeführten Argumente läßt sich zusammenfassen, daß ein Hilfesystem *individuelle* Hilfe geben sollte. Dabei sollte die Benutzerin nicht von der Systemseite aus einschätzt werden, sondern die Benutzerin sollte selbst beurteilen dürfen, welche Informationen sie benötigt.

1.4 Erklärungskomponente eines Hilfesystems

Damit die Handhabungskomponente eines Hilfesystems die Benutzerin bei ihrer Arbeit mit einer Applikation unterstützen kann, muß eine sinnvoll gestaltete Erklärungskomponente zur Verfügung stehen. Im folgenden wird untersucht, wie Erklärungen dargestellt werden sollten, welchen Inhalt sie haben sollten und woher sie gewonnen werden können.

1.4.1 Darstellung

In der Literatur[7] werden die unterschiedlichsten Formen empfohlen, um Erklärungen darzustellen. Die Bandbreite reicht von Online-Manuals bis zu Hypermedia[8]. Ein Online-Manual besteht aus einer am Bildschirm erhältlichen Version des Handbuchs. Hypermedia dagegen bestehen aus Texten, Bildern und Bildsequenzen, die sich aus vielen Teilen (Knoten) zusammensetzen. Die Knoten sind untereinander durch Verweise verbunden, so daß eine netzartige Struktur entsteht.

Beim Durchlesen eines Handbuchs kann man anhand der großen Zahl von Verweisen feststellen, daß es sich bei Erklärungen um eine hochgradig vernetzte Struktur handelt. Das Verfolgen von Verweisen in Büchern oder Online-Manuals erweist sich als aufwendiges Unterfangen, weil jeweils anhand der Seitenzahl oder der Kapitelnummer nach dem gewünschten Thema gesucht werden muß. Hypermedia können dieses Problem netzartiger Strukturen in einen Vorteil verwandeln, weil sie durch ihre technische Unterstützung beim Verfolgen von Verweisen einen besseren Zugang zu ihrer Struktur bieten.

Darüber hinaus sind Hypermedia Büchern oder Online-Manuals auch auf anderem Gebiet überlegen. Hypermedia hat sich als gutes Mittel erwiesen, um eine Benutzerin beim explorativen Lernen zu unterstützen. Unter explorativem Lernen wird verstanden, daß die Benutzerin sich ihre Lerninhalte selbst erarbeitet. Diese Art des Lernens wird als effizienteste Form des Lernens betrachtet [FloGeb91]. Hypermedia unterstützen exploratives Lernen, indem sie der Benutzerin kurze Erklärungen anbieten, die Verweise auf andere Erklärungen enthalten. Auf diese Weise versetzt Hypermedia die Benutzerin in die Lage, selbst zu entscheiden, was und wieviel sie erfahren will.

---

[7] siehe [BauSch88, DuPaMe92, FloGeb91, Gamma92, Horton90, Kearsley88, SzeNec94, Willms92, Yetim91]

[8] Der Begriff *Hypermedia* steht in dieser Arbeit für Hyperstrukturen, die sowohl Text als auch Bild enthalten. Der Begriff *Hypertext* wird verwendet, wenn Hyperstrukturen gemeint sind, die nur aus Text bestehen (s. [NakSch93]).

Carroll et al. (s. [CGLMR85]) haben in diesem Zusammenhang eine Untersuchung durchgeführt, bei der die Verwendung von Handbüchern und Erklärungen auf Karten verglichen wurde.

Für die Untersuchung wurde ein Satz von Karten entwickelt, auf denen je eine elementare Aufgabe behandelt wurde, die mit dem verwendeten Softwareprodukt gelöst werden konnte. Die Erklärungen wurde möglichst kurz gehalten. Alle Karten waren nach einem einheitlichen Schema aufgebaut. Es wurden Hinweise zur Problemlösung sowie die Möglichkeit zum Erkennen und Beheben von Fehlern gegeben. Diese "Guided Exploration"(GE)-Karten wurden entwickelt, um entdeckendes Lernen zu unterstützen.

Die Testpersonen wurden in zwei Gruppen eingeteilt: Eine Gruppe bekam als erklärende Unterstützung die GE-Karten, die in zufälliger Folge aufeinander lagen. Die Versuchspersonen sollten die Möglichkeit haben, die GE-Karten in einer Reihenfolge zu benutzen, die ihren eigenen Zielen entsprach. Die andere Gruppe benutzte das Handbuch des Softwareproduktes, mit dem die einzelnen Teilnehmerinnen und Teilnehmer alleine lernen sollten.

Es stellte sich heraus, daß die Versuchspersonen mit den GE-Karten effizienter lernten. Besonders auffällige Ergebnisse waren:

- Die Lernzeit der GE-Benutzerinnen und Benutzer war nicht einmal halb so lang wie bei den Testpersonen mit Handbüchern. Die GE-Teilnehmerinnen und Teilnehmer brauchte für die erste Aufgabe zwar 22% länger, waren bei der letzten Aufgabe dagegen 35% schneller als die andere Gruppe.
- Die GE-Versuchspersonen lasen in weniger als 12% ihrer Zeit und hatten zu 88% ihre Aufmerksamkeit auf den Bildschirm gerichtet. Bei der anderen Gruppe wurde dagegen zu 44% der Zeit gelesen.
- GE-Benutzerinnen und Benutzer waren beim Übertragen von Gelerntem erfolgreicher.
- Teilnehmerinnen und Teilnehmer mit Buch fragten meist nach dem Sinn ihrer Aufgabe, was keine GE-Versuchperson tat.
- GE-Probandinnen und Probanden waren bei der Fehlerbehebung 40-70% erfolgreicher.

Aus dieser Untersuchung von Carroll et al. läßt sich schließen, daß eine Benutzerin mit Hilfe von Erklärungen, die in kleine Einheiten unterteilt sind und zu denen sie in beliebiger Reihenfolge Zugang hat, besser lernen kann, als mit Hilfe eines durchgehenden Textes. Um diese Erkenntnis auf Hypermedia übertragen zu können, muß berücksichtigt werden, daß die Versuchspersonen von Carroll et al. lediglich mit einer kleinen Anzahl von Karten arbeiten mußten. Hypermedia bestehen dagegen i.d.R. aus einer großen Anzahl von erklärenden Einheiten, die in einem Netz miteinander verbunden sind. Damit eine Benutzerin in diesem Netz nicht die Übersicht verliert, ist es erforderlich, daß ihr entsprechende Orientierungsmöglichkeiten zur Verfügung gestellt werden (s. auch Kapitel 3).

Sind Orientierungsmöglichkeiten vorhanden, so bieten Erklärungen, die als Hypermedia dargestellt werden, der Benutzerin bessere Möglichkeiten, Informationen zu sammeln als Online-Manuals.

1.4.2 Inhalt

In der Literatur[9] wird gefordert, daß die Erklärungskomponente eines Hilfesystems Erläuterungen zu folgenden Punkten geben soll:

- wenn ein Fehler in der Applikation auftritt,
- wie in einer bestimmten Situation vorzugehen ist,
- welche Handhabungsmöglichkeiten vorhanden sind und
- welches Verständnis der Entwicklerinnen vom Anwendungsgebiet der Applikation zugrunde liegt.

Nur die beiden letzten Forderungen lassen sich von einem Hilfesystem erfüllen und sind dort am richtigen Platz.

Erläuterungen zu Fehlern in der Applikation sollten nicht im Hilfesystem sondern in der Applikation selbst angesiedelt werden, da Fehler von der Applikation diagnostiziert und daher auch an dieser Stelle behandelt werden können. Die Anwendung sollte die Benutzerin darüber informieren, warum der Fehler aufgetreten ist und wie er zu beheben ist.

Ratschläge, wie in einer bestimmten Situation vorgegangen werden kann, sind ebensowenig sinnvoll, wie die Idee, daß sich Hilfesituationen voraussehen lassen (s. Abschnitt 1.3.1). Deshalb ist ein Hilfesystem weder in der Lage, Erklärungen zu allen möglichen Verwendungsarten bereit zu halten, noch diese zu erkennen.

Sinnvolle Bestandteile der Erklärungskomponente eines Hilfesystems sind dagegen Erläuterungen, wie die Applikation zu *handhaben* ist und welches *Verständnis* die Entwicklerinnen vom Anwendungsgebiet haben. Diese beiden Fragestellungen sollten in Hilfesystemen beantwortet werden, da es sich hierbei um die Grundvoraussetzungen zum Verständnis einer Applikation handelt (s. Abschnitt 1.1).

1.4.3 Ursprung der Erklärungskomponente

Eine häufige Forderung an das Design von Hilfesystemen[10] besteht darin, daß die Erklärungskomponente automatisch aus der Applikation generiert werden sollte. Diese Forderung beruht auf der Annahme, daß die Applikation, zu der Hilfe gewünscht wird, bereits implizit alle für die Benutzerin interessanten Informationen enthalten würde.

Ist dies der Fall, so hat das Generieren von Erklärungen aus der Applikation den Vorteil, daß zu jedem Zeitpunkt Hilfe konsistent zur aktuellen Version des Systems angeboten werden kann und daß Änderungen am System nicht separat in den Erklärungen nachgezogen werden müssen. Als weiteren Vorteil führen die Vertreterinnen und Vertreter[10] dieser Sichtweise an, daß ein Hilfesystem, welches seine Erklärungen aus der Applikation aufbaut, alle Fragen beantworten kann, zu denen Antworten in der Applikation stecken. Bei manuell erstellten Erklärungen müssen dagegen alle möglichen Fragestellungen von der Erstellerin der Erklärungen antizipiert werden. Dabei werden wichtige Erklärungen oft vergessen, weil der zugehörige Sachverhalt für die Erstellerin der Erklärungen selbstverständlich ist (s. [Bauer88, BauSch88]).

---

[9] siehe [BauSch88, DuPaMe92, Horton90, Kersley88, Willms92]

[10] siehe [Bauer88, BauSch88, SzeNec94, Thies94]

Bis heute existieren nur einige wenige diese Anforderungen erfüllende Prototypen, die *gemeinsam* mit Kleinst-Applikationen (sogenannten S-Programmen [Lehman87]) entwickelt wurden. Dabei mußten auch diese Applikationen nach bestimmten Kriterien [SezNec94] oder mit Hilfe bestimmter Programmsysteme [Thies94] entwickelt worden sein. Abgesehen von den technischen Schwierigkeiten bei der Umsetzung solcher Hilfesysteme, läßt sich die Frage stellen, ob eine Applikation wirklich alle hilferelevanten Informationen implizit enthalten kann. Sollen in einem Hilfesystem nur Erklärungen zu Handlungssequenzen angeboten werden, so kann davon ausgegangen werden, daß diese Information in der Applikation zu finden ist. Es besteht theoretisch also die Möglichkeit, Informationen über die Handhabung aus der Applikation zu generieren. Wird von einem Hilfesystem aber gefordert, daß es auch Auskunft über das Zusammenspiel zwischen Anwendungswelt und Applikation geben soll, so ist diese Information in der Applikation selbst nicht zu finden. Das von der Entwicklerin untersuchte Zusammenspiel schlägt sich zwar zum Teil im Design der Applikation nieder, ist aber nicht wie die Handhabung an Sequenzen festzumachen. Daher kann ein Hilfesystem diese Information nicht aus der Applikation extrahieren.

Unter Berücksichtigung technischer Gegebenheiten und des geforderten Inhalts von Erklärungskomponenten muß davon ausgegangen werden, daß Erklärungskomponenten explizit erzeugt werden müssen und nicht aus der Applikation generierbar sind.

1.5 Zusammenfassung

Nachdem in diesem Kapitel eine Abgrenzung gegen die in der Literatur vorhandenen Ideen von Hilfesystemen vorgenommen und Aspekte für die Entwicklung von Hilfesystemen herausgearbeitet wurden, können die wichtigsten Aussagen abschließenden in einer Anforderung an Hilfesysteme zusammengeführt werden. Dabei wird Definition 1-1 aufgegriffen und erweitert.

Anforderung 1-2:

Ein Hilfesystem ist programmtechnisch unabhängig von einer Applikation und besteht aus zwei Teilen: einer Handhabungs- und einer Erklärungskomponente. Die Handhabungskomponente eines Hilfesystems ermöglicht der Benutzerin den Zugriff auf die Erklärungskomponente, welche Hilfe zu einer Applikation enthält.

Die **Handhabungskomponente** eines Hilfesystems:

- bietet **passive** Hilfe an, da Hilfesituationen nie eindeutig bestimmbar sind,
- stellt **dynamische** Hilfe zur Verfügung, die aufgrund des Zustands der Anwendung kontextsensitiv ist, und
- stellt insofern **individuelle** Hilfe bereit, als daß sie der Benutzerin Hilfe zur explorativen Selbsthilfe gibt.

Die **Erklärungskomponente** besteht aus kurzen textuellen und visuellen Einheiten, die als Hypermedia organisiert sind. Diese erklärenden Einheiten versetzen die Benutzerin in die Lage, die von der Entwicklerin intendierte Handhabung der Applikation und das Verständnis der Entwicklerin vom Anwendungsgebiet zu erfassen.

Die Anforderungen an Hilfesysteme, die in diesem Kapitel herausgearbeitet wurden, sind noch relativ abstrakt. In den folgenden Kapiteln werden auf dieser Grundlage weiterführende Konzepte und eine mögliche konstruktive Umsetzung vorgestellt, die sich im Softwareentwicklungsprozeß direkt einsetzen lassen.

2 Werkzeug-Material Metapher

Die weitere Auseinandersetzung mit Hilfesystemen erfolgt anhand der Werkzeug-Material Metapher (s. [BudZül90, GryZül92]), die von Züllighoven und anderen entwickelt wurde. Die Werkzeug-Material Metapher ist Grundlage des fachlichen und technischen Entwurfs für das in dieser Arbeit vorgestellte Hilfe-Framework.

Mit der Werkzeug-Material Metapher werden sowohl ein Gestaltungsleitbild für Benutzungsschnittstellen als auch Designmuster[11] für die technische Konstruktion zur Verfügung gestellt. Dadurch kann sie bruchlos im Laufe des gesamten Softwareentwicklungsprozesses eingesetzt werden und bietet eine gute Grundlage für die Diskussion zwischen Benutzerinnen und Entwicklerinnen. Beide Aspekte der Werkzeug-Material Metapher kommen in dieser Arbeit zum Tragen. Einerseits wird das Hilfesystem als ein Werkzeug verstanden, das auf Materialien arbeitet und sie darstellt. Das Hilfesystem sollte daher an der Benutzungsschnittstelle wie ein *Werkzeug* zu handhaben sein, das seinen Benutzerinnen die Freiheit läßt, nach ihren Vorstellungen mit ihm zu arbeiten. Auf der anderen Seite bietet die Werkzeug-Material Metapher eine Reihe von Mustern an [Riehle95], die in Teil III (Produkt) verwendet werden, um die Konstruktion des Hilfe-Frameworks zu beschreiben. Mit Hilfe der Muster kann eine Strukturierungsebene zwischen den fachlichen Begriffe aus der Anwendungswelt und computersprachlicher Implementation eingeführt werden. Dadurch sind Entwürfe leichter zu verstehen und können unter Verwendung der Muster diskutiert werden.

Dieses Kapitel gibt einen Überblick über die Werkzeug-Material Metapher und die in ihrem Rahmen definierten Muster. Die Einführung ist auf den Inhalt dieser Arbeit ausgerichtet und beschreibt so nur einen Teil der mit der Werkzeug-Material Metapher verbundenen Gedankenwelt. Für die verschiedenen anderen Bereiche sind bereits detaillierte Beschreibung[12] vorhanden. Die verwendete, graphische Notation ist im Anhang erklärt.

2.1 Ebenen der Werkzeug-Material Metapher

In [Riehle95] werden drei Ebenen für die Werkzeug-Material Metapher genannt:

- das Leitbild vom Arbeitsplatz für qualifizierte menschliche Arbeitstätigkeit,
- Interpretations- und Gestaltungsmuster sowie
- objektorientierte Entwurfsmuster.

Der Ausgangspunkt der Metapher, die oberste Ebene, ist *das Leitbild vom Arbeitsplatz für qualifizierte menschliche Arbeitstätigkeit*. Auf dieser Ebene wird Software unterteilt in Arbeitsmittel (Werkzeuge) und Arbeitsgegenstände (Materialien). Um *qualifizierte* Arbeit zuzulassen, müssen Werkzeuge und Materialien so entwickelt werden, daß sie fachliche Konzepte und Gegenstände des Anwendungsgebiets repräsentieren und dabei keine Arbeitshandlungen festschreiben. Softwaresysteme, die solchermaßen entwickelt wurden, ermöglichen einer Benutzerin, ihre Qualifikation zu entfalten und selbständig zu arbeiten.

Die zweite Ebene der Werkzeug-Material Metapher umfaßt *Interpretations- und Gestaltungsmuster*. In [Riehle95] wird definiert:

---

[11] siehe [Gamma92, RieZül94a, GaHeJoVl95]

[12] siehe [BudZül90, BüGrZü92, GryZül92, KiGrZü94, RieZül94a, Riehle95, BäGrZü95] und diverse Studien- und Diplomarbeiten am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg.

> " Ein Interpretations- und Gestaltungsmuster ist ein Muster, welches zur Interpretation und Gestaltung von tatsächlichen oder antizipierten Anwendungssituationen und Softwaresystemen verwendet werden kann".

Unter Interpretations- und Gestaltungsmustern werden die Begriffe Umgebung, Verwendungszusammenhang, Material und Werkzeug eingeordnet.

Die Umgebung stellt den Rand eines Arbeitsplatzes im Computersystem dar. Sie hat die Aufgabe, "eine der menschliche Wahrnehmung angemessene Umsetzung räumlicher Organisationsformen in logische Dimensionen" [Riehle95] zu ermöglichen, werkzeugübergreifende Konsistenzbedingungen zu formulieren und zu wahren sowie zu überprüfen, ob bestimmte Werkzeuge und Materialien zueinander passen. Auf diese Weise stellt die Umgebung den Rahmen für die verwendeten Werkzeuge und Materialien zur Verfügung. Weitere Ausführungen zum Begriff der Umgebung finden sich in [Riehle95].

Die Gestaltungs- und Interpretationsmuster Material, Werkzeug und Verwendungszusammenhang werden im Abschnitt 2.2.1 und 2.2.2 genauer betrachtet, weil sie eine Grundlage für die Entwicklung des Hilfesystems darstellen. Neben den in [Riehle95] aufgeführten Gestaltungs- und Interpretationsmustern enthält die Werkzeug-Material Metapher noch ein weiteres Ausdrucksmittel: den Automaten. Der Automat wird im Rahmen dieser Arbeit ebenfalls als Gestaltungs- und Interpretationsmuster verstanden und in Abschnitt 2.2.3 detaillierter beschrieben.

Die dritte Ebene der Werkzeug-Material Metapher enthält *objektorientierte Entwurfsmuster*, die bei der Softwareentwicklung in konkrete Systemkomponenten und Systemstrukturen umgesetzt werden. In [Riehle95] wird definiert:

> "Ein Entwurfsmuster ist ein Muster, das als Vorlage für die Konstruktion eines softwaretechnischen Entwurfs dient. Ein Entwurfsmuster besteht aus dem Zusammenspiel technischer Elemente wie Klassen, Objekte und Operationen. Die Struktur und Dynamik eines Entwurfsmusters klärt die beteiligten Komponenten, ihre Zusammenarbeit und die Verteilung der Verantwortlichkeiten."

Entwurfsmuster lassen sich unterteilen in Entwurfsmuster für die Konstruktion von Softwarewerkzeugen und in Entwurfsmuster für die Integration von Werkzeugen und logisch abhängigen Materialien. Die zweite Sorte Entwurfsmuster spielt im Zusammenhang dieser Arbeit keine wesentliche Rolle, daher werden diese Muster nicht weiter vorgestellt. Die Entwurfsmuster zur Konstruktion von Softwarewerkzeugen: Werkzeug und Material Kopplung, Trennung von Interaktion und Funktion, Werkzeugkomposition, Werkzeugkopplung und der Ereignismechanismus werden in dieser Arbeit zur Beschreibung des Entwurfs benötigt und daher bei ihrer Verwendung in Teil III sukzessive eingeführt.

2.2 Interpretations- und Gestaltungsmuster

Bei der folgenden Beschreibung der Interpretations- und Gestaltungsmuster werden bewußt Bilder oder Begriffe aus dem Ursprungsbereich der Metapher - Arbeitsplatz - und ihrem Bildbereich - Softwarekomponenten - nebeneinander verwendet. Auf diese Weise sollen die Implikationen, die diese Muster für den Softwareentwurf haben, vermehrt sichtbar gemacht werden.

2.2.1 Materialien und Werkzeuge

Die Werkzeug-Material Metapher beruht auf der Annahme, daß in Arbeitssituationen ein intuitiver Unterschied festgestellt werden kann zwischen Dingen, an denen man arbeitet (**Materialien**), und Dingen, die Arbeitsmittel sind (**Werkzeuge**).

Ein *Material* ist in diesem Sinne eine passive Einheit, die in Arbeitszusammenhängen, für die sie geeignet ist, zur Hand genommen und bearbeitet wird. Jedes Material hat spezifische *Umgangsformen*, d.h. man kann es auf bestimmte Weise handhaben. Während der Arbeit liegt das eigentliche Interesse der arbeitenden Person darin, das Material zu verändern bzw. Informationen über und von dem Material zu bekommen.

Werkzeuge hingegen stehen *nicht* im vorrangigen Interesse einer Benutzerin. Sie dienen lediglich als Mittel zur Arbeitserledigung. Dabei geben Werkzeuge eine Sicht auf das Material wieder, können Veränderungen am Material durchführen und geben Rückmeldung über ihre Verrichtung.

Auf der Ebene des softwaretechnischen Entwurfs werden Materialien und Werkzeuge in voneinander getrennten Klassen abgebildet. Die im fachlichen Entwurf spezifizierten Umgangsformen mit Material und Werkzeug finden sich in den Operationen der Klassen wieder, so daß die jeweiligen Klassen entsprechend zusammenarbeiten bzw. benutzt werden können.

Ob es sich bei einem Gegenstand um ein Material oder ein Werkzeug handelt, kann nur in der jeweiligen Arbeitssituation festgestellt werden. Normalerweise sind Werkzeuge und Materialien zwar klar von einander zu unterscheiden, aber in bestimmten Arbeitszusammenhängen können Gegenstände, die Werkzeugcharakter haben, zum Material eines anderen Werkzeugs werden. Ein typisches Beispiel aus der Anwendungswelt ist ein Bleistift, der beim Schreiben als Werkzeug verwendet werden kann; in dem Moment, wo er angespitzt werden muß, aber zum *Material* eines Bleistiftanspitzers wird.

Entsprechend verhält es sich mit Applikationen, die nach der Werkzeug-Material Metapher konstruiert sind und mit dem Hilfesystem zusammenarbeiten. Hat eine Benutzerin Probleme mit einer Applikation, so ruft sie das Hilfesystem auf. Das Hilfesystem arbeitet nun als Werkzeug auf dem Werkzeug der Applikation. In einer Hilfesituation kann daher das Werkzeug einer Applikation zum *Material* des Hilfesystems werden.

2.2.2 Verwendungszusammenhang und Aspekt

Wenn mit Werkzeugen an Materialien gearbeitet wird, kann man feststellen, daß nicht jedes Werkzeug für jedes Material geeignet ist. Werkzeuge passen vielmehr in einer spezifischen Weise zu Materialien, weil sie entwickelt wurden, um eine bestimmte Arbeit an genau diesen Materialien zu verrichten. In einer Welt physischer Gegenstände wird der Zusammenhang zwischen Werkzeugen und Materialien implizit durch Handlungen hergestellt. In einer softwaretechnischen Welt, in der keine Handlungsorientierung durch physische Randbedingungen vorgegeben wird, muß der Zusammenhang zwischen Werkzeugen und Materialien explizit gemacht werden. Dieses Zusammenpassen von Werkzeugen und Materialien läßt sich durch den **Verwendungszusammenhang** explizit machen. Ein Verwendungszusammenhang ist gegeben, wenn ein Werkzeug mit einem Material kombiniert wird, um eine bestimmte Arbeitstätigkeit zu erledigen.

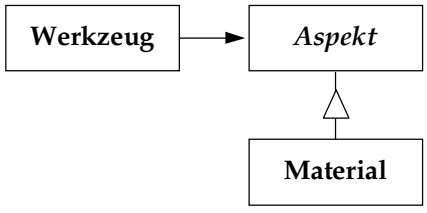


Abb. 2-1: Verwendungszusammenhang

Um auf der Ebene des fachlichen und technischen Entwurfs die explizite Beziehung zwischen Werkzeug und Material sicherzustellen, wird für den Verwendungszusammenhang eine Abstraktion namens **Aspekt** verwendet. Dieser Begriff signalisiert, daß ein Material aus Sicht eines Werkzeugs unter einem spezifischen Aspekt betrachtet wird. Der Aspekt stellt den Zugang eines Werkzeugs zu einem Material dar. Zu diesem Zweck wird durch den Aspekt ein Protokoll definiert, das alle notwendigen Umgangsformen anbietet, die es einem Werkzeug ermöglichen, an einem Material zu arbeiten. Die Konsequenz dieser Abstraktion ist, daß weder ein Werkzeug sein Material noch ein Material sein Werkzeug kennt.

Die Entkoppelung von Werkzeug und Material durch einen Aspekt ist eine wichtige Unterstützung, um Systeme flexibel und unabhängig zu machen. Bei der Entwicklung des Hilfesystems hat sich dieser Mechanismus als vorteilhaft erwiesen.

2.2.3 Automaten

Neben Werkzeug, Aspekt und Material enthält die Werkzeug-Material Metapher ein weiteres Ausdrucksmittel, den **Automaten**. Automaten werden dazu eingesetzt, Arbeitsprozesse zu kapseln, die eine gewisse Regelmäßigkeit und Wiederholung aufweisen. In der Arbeitswelt lassen sich solche Aufgaben in Bargeldautomaten, bei der Ausführung von Daueraufträgen, aber auch bei Wasch- oder Geschirrspülmaschinen wiederfinden.

Aus softwaretechnischer Sicht ist ein Automat dadurch gekennzeichnet, daß er, im Unterschied zum Werkzeug, mit einer bestimmten Einstellung im Hintergrund aktiv ist. Hat ein Automat eine veränderbare Einstellung, so haben Benutzerinnen Zugang zu bestimmten Parametern des Automaten. Der innere Algorithmus des Automaten bleibt den Benutzerinnen aber verborgen. Einmal angestoßen, leistet ein Automat seine Arbeit auf der Grundlage des vorgegebenen Algorithmus und liefert ein vorher festgelegtes Ergebnis, falls die Arbeit erfolgreich war.

2.3 Hilfesysteme aus der Sicht der Werkzeug-Material Metapher

Die im ersten Kapitel erarbeiteten Anforderungen lassen sich aus dem Blickwinkel der Werkzeug-Material Metapher umformulieren und erweitern.

Nachdem im ersten Kapitel eine Aufteilung des Hilfesystems in Handhabungs- und Erklärungskomponente vorgenommen wurde, lassen sich diese Begriffe im Kontext der Werkzeug-Material Metapher redefinieren. Die Handhabungskomponente eines Hilfesystems ist in diesem Sinne ein Werkzeug (**Hilfewerkzeug**), das der Benutzerin verschiedene Sichten auf das Material, die Erklärungen, anbietet (**Hilfematerial**).

Im ersten Kapitel wurde argumentiert, daß das Hilfewerkzeug passiv, dynamisch hinsichtlich des Zustands der Applikation und individuell für explorative Selbsthilfe sein soll (s. Anforderung 1-2). Die Werkzeug-Material Metapher, die auf dem Leitbild des qualifizierten Arbeitsplatzes beruht, unterstützt diese Forderungen.

Betrachtet man die Handhabungskomponente eines Hilfesystems als Werkzeug, so ist im Rahmen der Werkzeug-Material Metapher zu erwarten, daß ein Werkzeug *passiv* (s. Abschnitt 1.3.1) ist. Werkzeuge sind i.d.R. an einem Arbeitsplatz angeordnet und stehen zur Verfügung, wenn die Benutzerin sie benötigt. Stellt eine Benutzerin fest, daß sie Hilfe braucht, so *nimmt* sie ein Handbuch bzw. ein Hilfesystem zur Hand. Werkzeuge oder Hilfesysteme drängen sich nicht zum Gebrauch auf, wie es ein aktives Hilfesystem tut.

Selbst wenn eine Person eine andere Person bei der Verwendung eines Werkzeugs unterstützt, ist eine gewisse Passivität von Vorteil. Es läßt sich immer wieder feststellen, daß häufige Hinweise darauf, wie etwas besser zu machen sei, oder gar direktes Eingreifen, die lernende Person stören. Werden aber Fragen der Lernenden kompetent beantwortet und im Zusammenhang mit Fragen Informationen vermittelt, so ergibt sich eine gute Lernatmosphäre.

Dynamische Hilfe (s. Abschnitt 1.3.2) soll nach Anforderung 1-2 aufgrund des Zustands einer Applikation gegeben werden. Arbeitet eine Benutzerin mit einem Werkzeug auf einem Material, so kann sie i.d.R. den Zustand der beiden Gegenstände feststellen. Aufgrund des Zustands kann die Benutzerin Hilfe zu ihrer jeweiligen Arbeitssituation aus Handbüchern oder von Menschen bekommen. Auch als Softwaresysteme angebotene Werkzeuge und Materialien haben einen Zustand, auf den ein Hilfesystem zurückgreifen kann, um dynamische Hilfe anbieten zu können. Daher kann aus dem Blickwinkel der Werkzeug-Material Metapher dynamische Hilfe in Abhängigkeit vom Zustand realisiert werden.

Die Forderung nach *individueller* Hilfe (s. Abschnitt 1.3.3) in Form von Selbsthilfe läßt sich aus Sicht der Werkzeug-Material Metapher unterstützen. Menschen richten sich bei der Arbeit an einem qualifizierten Arbeitsplatz ihre Arbeitsgegenstände und Arbeitsmittel selbständig ein, probieren sie aus und verändern sie, wenn es ihren Zwecken förderlich ist. Die "Individualisierung" führt die Benutzerin selbst an ihren Werkzeugen und Materialien durch. Ein Hilfesystem sollte seiner Benutzerin entsprechende Möglichkeiten bieten.

Der kontrovers zu dieser Sicht in Abschnitt 1.3.3 diskutierte Ansatz, verschiedene Benutzerinnengruppen, wie Anfängerin und Expertin, für individuelle Hilfe einzuführen, findet keine Motivation in der Welt der realen Werkzeuge und Materialien. Ein Werkzeug stellt allen Benutzerinnen die gleichen Möglichkeiten zur Verfügung und die Verwirrung wäre groß, wenn das Werkzeug sich abhängig von der Erfahrung der Benutzerinnen verändern würde. In der realen Welt sind lediglich *verschiedene* Werkzeuge vom gleichen Typ mit unterschiedlichen Leistungen und differierender Präzision verfügbar. Bei diesen Werkzeugen ist jeder Benutzerin von vorne herein klar, mit welchem Werkzeug sie arbeitet. Außerdem verändert sich bei *keinem* Werkzeug während der Arbeit die Leistungsfähigkeit, solange es nicht einen Defekt hat und repariert oder ausgewechselt werden muß.

Ein Hilfesystem, das passiv, dynamisch und individuell Hilfe anbietet, hat also aus Sicht der Werkzeug-Material Metapher Bestand und ist mit der Vorstellung eines qualifizierten Arbeitsplatzes zu vereinbaren.

Über die Idee eines passiven, dynamischen und individuellen Hilfesystems hinaus spielt beim Entwurf nach der Werkzeug-Material Metapher die Frage nach dem Umgang mit einem bestimmten Material oder einem Werkzeug eine wichtige Rolle. Um diese Sichtweise für das Hilfesystem produktiv zu machen, müssen die charakteristischen Umgangsformen von Benutzerinnen mit Handbüchern oder anderen Hilfsmitteln untersucht werden. Typischerweise läßt sich beobachten:

- Benutzerinnen *lesen* Erklärungen.
- Benutzerinnen *suchen* adäquate Erklärungen zu ihrem Problem, indem sie einen *Index verwenden*, nach Themengebieten vorgehen oder im Handbuch *stöbern*.
- Benutzerinnen *markieren* bestimmte Stellen in den Erklärungen, um sie später wiederzufinden.
- Benutzerinnen *schreiben* eigene Bemerkungen zu den Erklärungen.

Die Art und Weise wie Benutzerinnen einen bestimmten Gegenstand handhaben, entsteht i.d.R. über einen längeren Zeitraum hinweg und manifestiert sich in bewährten Formen des Umgangs. Diese bewährten Formen können bei Softwareentwürfen aufgegriffen werden, um den Benutzerinnen ein Softwaresystem zu bieten, das fachlich der ihnen bekannten Anwendungswelt entspricht. Umgangsformen, die sich bei Benutzerinnen in der Arbeit mit Hilfsmitteln etabliert haben, sollten daher auch in einem Hilfesystem umgesetzt werden.

Die oben aufgelisteten Umgangsformen mit Hilfsmitteln lassen sich für die Anforderungen an Hilfesystems in zwei Hauptaufgaben für Hilfesysteme umformulieren.

Einerseits muß ein Hilfesystem die *Informationen* zu einer Applikation in angemessener Form *anzeigen*, damit die Benutzerin die Erklärungen lesen und darin stöbern kann. Dies kann in Gestalt von Texten, Bildern, Bildsequenzen, Diagrammen etc. geschehen.

Andererseits muß ein Hilfesystem seine Benutzerin dabei unterstützen, *Informationen aufzufinden*. Dazu muß es der Benutzerin Möglichkeiten zum Suchen nach verschiedenen Kriterien und Indizes, zum Markieren von bestimmten Stellen, zum Stöbern und zum Schreiben von eigenen Anmerkungen bieten. Um dies umzusetzen, können verschiedene Ideen wie Schlagwortsuche, Lesezeichen, Kontextsensitivität und Verweise zwischen Themen aufgegriffen werden, die in vielen Hilfesystemen bereits verwirklicht worden sind.

Die im ersten Kapitel zusammengestellten Anforderungen an Hilfesysteme lassen sich daraufhin erweitern zu:

Anforderung 2-1:

Die **Aufgaben** eines Hilfesystems bestehen darin, Informationen **anzuzeigen** und das **Auffinden** von Informationen zu ermöglichen.

Ein Hilfesystem ist programmtechnisch unabhängig von einer Applikation und besteht aus zwei Komponenten: einem **Hilfewerkzeug** und einem **Hilfematerial**. Das Hilfewerkzeug ermöglicht der Benutzerin den Zugriff auf das Hilfematerial, welches Hilfe zu einer Applikation enthält.

Das Hilfewerkzeug:

- bietet passive Hilfe an, da Hilfesituationen nie eindeutig bestimmbar sind,
- stellt dynamische Hilfe zur Verfügung, die aufgrund des Zustands der Anwendung kontextsensitiv ist und
- stellt insofern individuelle Hilfe bereit, als daß es der Benutzerin Hilfe zur explorativen Selbsthilfe gibt.

Das Hilfematerial besteht aus kurzen textuellen und visuellen Einheiten, die als Hypermedia organisiert sind. Diese erklärenden Einheiten versetzen die Benutzerin in die Lage, die von der Entwicklerin intendierte

Handhabung der Applikation und das Verständnis der Entwicklerin vom Anwendungsgebiet zu erfassen.

Wie der Anforderung 2-1 zu entnehmen ist, hat die Werkzeug-Material Metapher Einfluß auf die fachliche Entwicklung des Hilfesystems genommen. Im folgenden wird Anforderung 2-1 um software-ergonomische und sprachliche Vorgaben erweitert, um schließlich in Teil III als Basis für die Produktbeschreibung zu dienen.

3 Transparenz und Sprache

Wie bereits im ersten Kapitel festgestellt wurde, braucht eine Benutzerin Hilfe, um mit Computersystemen arbeiten zu können. In welchem Maß sie Hilfe benötigt, hängt hauptsächlich davon ab, ob die Benutzungsschnittstelle der Applikation angemessen widerspiegelt, welches Verständnis die Entwicklerin vom Anwendungsgebiet hat und welche Handhabung sie vorgesehen hat. Um zu klären, was eine Benutzerin an der Benutzungsoberfläche vorfindet, kann auf sprachtheoretische Überlegungen zurückgegriffen werden. Die sprachlichen Aussagen legen nahe, daß die Benutzerin bei der Interpretation der Benutzungsschnittstelle unterstützt werden sollte.

Zu diesem Zweck gibt es eine Reihe software-ergonomischer Richtlinien. Eine wichtige software-ergonomische Forderung ist in diesem Zusammenhang das Konzept der Transparenz. Es wurde geprägt, um zu beschreiben, welche Voraussetzungen erfüllt sein müssen, damit eine Benutzerin einen Computer zu ihren Zwecken einsetzen kann. Neben der Berücksichtigung von software-ergonomischen Richtlinien können auch Hilfesysteme eingesetzt werden, um eine transparente Benutzungssituation zu fördern. Entscheidend ist dabei, daß die Erklärungskomponente des Hilfesystems sprachlich und inhaltlich gut gestaltet ist.

Im diesem Kapitel werden die auf der Sprachtheorie basierenden Annahmen und das Konzept der Transparenz vorgestellt. Aufbauend auf diesen beiden Konzepten können Aussagen für das Design von Software (z.B. das Hilfewerkzeug) und insbesondere für die Gestaltung der Hilfetexte gemacht werden.

Die Begriffe "System", "Computersystem" und manchmal auch "Computer" werden im folgenden synonym verwendet. Darunter wird stets ein Anwendungssystem oder ein integriertes System verstanden, daß auf einem Großrechner oder auf vernetzten Rechnern realisiert ist.

3.1 Sprache

Der hier vertretende sprachliche Ansatz beruht auf dem von Sybille Krämer geprägten Begriff **symbolische Maschinen** (s. [Krämer92, Krämer93]). Krämer motiviert diesen Ausdruck ausgehend von der Schule des symbolorientierten Begriffs vom Geist (G. W. Leibniz, C. S. Peirce, E. Cassirer, L. Wittgenstein), die einen Akzent auf den unauflöslichen Zusammenhang zwischen Denken und externem Zeichengebrauch legt. Ausgangspunkt ist die Überlegung, daß Sprechen und Schreiben Vorgänge sind, bei denen Menschen Symbole gebrauchen. Dabei werden Symbole nicht als "intrinsische mentale Repräsentationen, sondern als externe, sinnlich wahrnehmbare Zeichen" verstanden. Die Symbole bilden ein zwischen Menschen gültiges System, dessen Regeln die Menschen beherrschen müssen, um die Zeichen verstehen und gebrauchen zu können.

Vor diesem Hintergrund, daß geistige Leistung immer im Zusammenhang mit dem handgreiflichen Umgehen mit externen Symbolen steht, läßt sich Informatik, als die technische Unterstützung geistiger Tätigkeiten verstehen. Die "Maschinisierung der Kopfarbeit" [Nake92] wird "in Gestalt der Operationalisierung des externen Symbolgebrauchs" [Krämer93] durchgeführt. Die Informatik verlagert also nicht die ursprünglich im Kopf stattfindenden Tätigkeit auf Maschinen, sondern sie setzt lediglich einen immer schon außerhalb des Kopfes sich vollziehenden handgreiflichen *Umgang mit Symbolen* um.

Dies geschieht mit Hilfe von symbolischen Maschinen und deren physischen Realisierungen in Computern. Krämer definiert in [Krämer93] sinngemäß:

Definition 3-1:

> Eine symbolische Maschine ist ein endliches Zeichensystem oder ein Kalkül, das zwei Bedingungen erfüllen muß, um verwendet werden zu können:
>
> (1) Es ist eine endliche Menge von *Zeichen* und *Regeln* vorhanden, nach denen unendlich viele Zeichenmuster gebildet und umgeformt werden können.
> (2) Es muß eine *Interpretation* für die symbolische Maschine zu finden sein, bei der ein Gegenstandsbereich so eindeutig auf den Bereich der Zeichen abzubilden ist, daß jeder zulässige Ausdruck zugleich einen wahren Satz über den entsprechenden Gegenstandsbereich ergibt.

Der Computer ist dadurch nur eine neue Erfindung, im Verlauf vielen Erfindungen, bei denen symbolische Maschinen zur Verrichtung geistiger Tätigkeit geschaffen wurden. Wird ein Computersystem als symbolische Maschine betrachtet, so macht Krämers Definition deutlich, daß an der Benutzungsschnittstelle nur Zeichen und Regeln in Form von Symbolen vorhanden sind (s. Definition 3-1, (1)).

Die Symbole an der Benutzungsschnittstelle bedürfen einer Interpretation, damit Aussagen über den Gegenstandsbereich getroffen werden können (s. Definition 3-1, (2)). Genau diese Interpretation muß die Benutzerin des Computersystems leisten, wenn sie das System im Rahmen ihrer Arbeitsaufgabe zielgerichtet einsetzen will. Krämers theoretische Definition für symbolische Maschinen nimmt allerdings keinen Bezug darauf, was bei der Gestaltung einer konkreten symbolischen Maschine berücksichtigt werden muß, um eine Interpretation durch die Benutzerin zu erleichtern.

Es gibt verschiedene Möglichkeiten und Voraussetzungen, symbolische Maschinen zu interpretieren. Im folgenden werden zwei verschiedene Arten von Computersystemen betrachtet, die diese Unterschiede deutlich machen. Zuerst wird auf Systeme eingegangen, die zur Steuerung von technischen Geräten als Leitstände in Fabriken eingesetzt werden. Bei diesen Systemen bilden Symbole Gegebenheiten ab, die tatsächlich an anderer Stelle existieren. Anschließend werden symbolische Maschinen betrachtet, die im Sinne von [Nake92] Kopfarbeit maschinisieren, d.h. Maschinen, die Kopfarbeit unterstützen, indem sie eine andere Art der Symbolverarbeitung als z.B. Papier und Stift zur Verfügung stellen. Nake setzt den Begriff "Maschinisierung" explizit von "Automatisierung" ab. Er versteht Maschinisierung in dem Sinne, daß neue Arbeitsmittel zur Verfügung gestellt werden, die Kopfarbeit tatsächlich *unterstützen* und nicht automatisieren.

Handelt es sich bei dem Computersystem um einen Leitstand in einer Fabrik (s. [Zuboff88]) so müssen die Benutzerinnen, die vorher in den Fabrikhallen direkten Kontakt zu den Maschinen hatten, allein aus den vom System gelieferten Daten die gleichen Informationen extrahieren. Im *direkten* Kontakt mit den Maschinen wissen die Menschen, welche Tätigkeiten richtig sind, weil sie anhand von Geräuschen (der Maschinen), des Geruchs, der Empfindung auf der Haut, Reaktionen des eigenen Körpers oder des Verhaltens des Materials (Zähigkeit beim Umrühren) den Zustand des Bearbeitungsprozesses feststellen können. Sie haben gelernt, Wissen über den Gegenstand in außerhalb von ihnen befindlichen Zusammenhängen abzuspeichern, d.h. ihr Wissen, was zu tun ist, wird erst durch das Auf-

treten der entsprechenden Signale aktiviert (s. [Norman89]). Sind die Menschen aber mit einem Computersystem konfrontiert, so müssen alle diese schwer formalisierbaren Eigenschaften, die vorher nur implizit verfügbar waren, auf andere Weise zugänglich gemacht werden. Die Menschen können ihre Kontrolltätigkeit nicht mehr direkt, sondern nur noch mittelbar über den Computer ausüben. Um das entsprechende Wissen verwendbar zu machen, muß der Computer daher einerseits Analysemöglichkeiten anbieten, die einen quasi direkten Kontakt mit den realen Dingen herstellen. Andererseits müssen die Benutzerinnen eine völlig neue Form des Arbeitens erlernen, weil sie ausschließlich anhand von explizitem Wissen an der Benutzungsschnittstelle tätig können. Hier läßt sich der Begriff *symbolische Maschine* unmittelbar nachempfinden. Die Benutzerinnen manipulieren an der Benutzungsschnittstelle eine symbolische Maschine oder Fabrik; die wirkliche Fabrik wird vom Computersystem gesteuert.

Die zweite Art von Computersystemen bildet keine Steuerung von technischen Geräten ab, sondern stellt ”lediglich" eine neue Art von Symbolbearbeitung zur Verfügung. Die neue Art der Symbolbearbeitung steht hier im Vordergrund, weil durch das Computersystem nur die Art, mit den Symbolen umzugehen, verändert wurde. Die Symbole selbst können i.d.R. aus dem Anwendungsgebiet übernommen werden, so wie beispielsweise für Computersysteme keine neuen Schriftzeichen erfunden wurden. Textverarbeitungssysteme bieten aber eine ganz andere Art des Umgangs mit den Schriftzeichen an, als es Papier und Bleistift oder eine Schreibmaschine jemals ermöglichen könnte. Dieser Zusammenhang kann in bezug auf symbolische Maschinen dadurch begründet werden, daß die Symbole für die zugehörige Kopfarbeit über längere Zeit durch Menschen entwickelt wurden und somit bereits eine adäquate Darstellungsform sind. Die Art der Bearbeitung der Symbole hingegen kann durch Computersysteme verbessert werden. Im Kontext der Werkzeug-Material Metapher wird in ähnlicher Weise vorgegangen (s. [BüGrZü92]), indem zuerst die in der Anwendungswelt vorgefundenen "Materialien" untersucht werden und häufig in einer direkten Abbildung in ein Computersystem übernommen werden. "Werkzeuge" hingegen können selten mimetisch aus der Anwendungswelt abgebildet werden, sondern stellen eine Weiterentwicklung der in der Anwendungswelt gefundenen Werkzeuge dar.

Aufgrund dieser Aussagen lassen sich Symbole im Computer in zwei Weisen wiederfinden: als Repräsentation für die *Gegenstände* der zu maschinisierenden Tätigkeiten sowie als Repräsentation für die *Handlungen*, die im Zusammenhang mit dieser Tätigkeiten durchgeführt werden können. Ob die Symbole für Gegenstände und Handlungen mit realen anderen Maschinen gekoppelt sind oder nicht, ist für die weitere Betrachtung unerheblich.

Sowohl Symbole für Gegenstände als auch Symbole für Handlungen werden der Benutzerin an der Benutzungsschnittstelle angeboten. Sind diese Symbole eine adäquate Darstellung der aus der Anwendungswelt abgebildeten Materialien und Werkzeuge (oder der alten Art der Symbolbearbeitung), so ist die Benutzerin in der Lage, eine Interpretation für die symbolische Maschine zu finden. Diese Überlegungen führen direkt zu der Frage, wie eine adäquate Darstellung der Anwendungswelt auszusehen hat. An dieser Stelle bietet das Konzept der *Transparenz* einen Ansatzpunkt.

3.2 Transparenz

Der Begriff "Transparenz" hat im Sprachgebrauch hauptsächlich die Bedeutung von "Durchsichtigkeit" oder "Durchscheinen". In der software-ergonomischen Literatur und ana-

log auch in dieser Arbeit wird Transparenz dagegen verwendet, um auszudrücken, daß ein Computersystem "leicht zu durchschauen" bzw. "leicht zu verstehen" ist.

Für das Konzept der *Transparenz* existiert in der Software-Ergonomie bisher keine abschließende Definition. Vielfach wird Transparenz als Qualitätsmerkmal eines Computersystems verstanden. Besitzt ein Computersystem diese Qualität, so sollte die Benutzerin das System zu ihren Zwecken einsetzen können.

In dieser Arbeit wird dagegen die Meinung vertreten, daß Transparenz nicht die Eigenschaft eines Systems oder eines Programms ist, sondern ein Idealzustand bei der Interaktion zwischen Mensch und Computer. Dies bedeutet, daß Transparenz von den *Umständen* der Interaktion abhängig ist. Untersucht man eine Situation, in der eine Person mit einem Computersystem interagiert, so läßt sich feststellen, daß jede Interaktion mit einer speziellen Aufgabe zusammenhängt. Interaktion hat i.d.R. für die Benutzerin den Zweck, eine bestimmte Aufgabe zu erledigen. Ob eine Benutzerin ein System als transparent empfindet, hängt daher von ihrer situativen Kontrolle und ihrem situativen Verständnis des Systems bei der Erledigung einer bestimmten Aufgabe ab. Transparenz ist also keine Eigenschaft eines Systems, sondern Transparenz stellt sich unter bestimmten *Bedingungen* ein.

Susanne Maaß hat in ihrem Bericht "Transparenz - Eine zentrale Software-ergonomische Forderung" (s. [Maaß94]) diese Sicht von Transparenz geprägt und eine umfassende Diskussion durchgeführt. Im folgenden wird auf Susanne Maaß' Verständnis von Transparenz aufbauend das Konzept der Transparenz vorgestellt.

Definition 3-2:

> "Ein System ist für eine Benutzerin transparent, wenn die Benutzerin sich ein zweckmäßiges mentales Modell vom System gemacht hat (Orientierung) und seine Anwendungs- und Handhabungsfunktionen im Rahmen ihrer Arbeitsaufgabe zielgerichtet einsetzen und einrichten kann (Beeinflussung)." [Maaß94].

Graphisch läßt sich Definition 3-2 folgendermaßen darstellen:

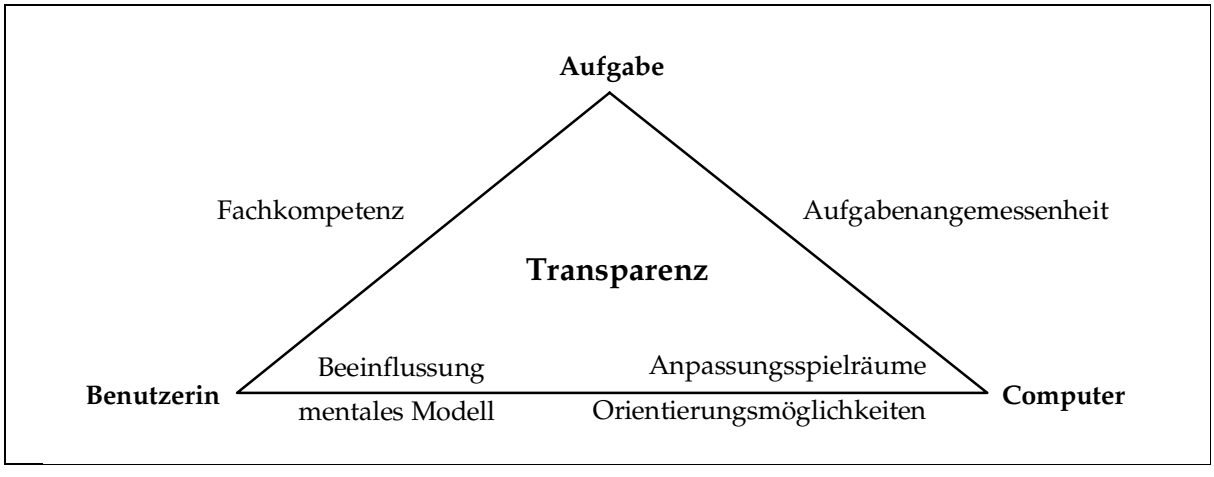


Abb. 3-1: Transparenz (in Anlehnung an [Maaß94])

Abbildung 3-1 zeigt den Zusammenhang zwischen Benutzerin, Computer und der Aufgabe, die die Benutzerin gerade in Bearbeitung hat. Andere Faktoren, wie der Kontext der Aufgabe in einer Organisation sind bewußt weggelassen, weil hier Einzelarbeitsplätze betrachtet werden, für die Hilfe erforderlich ist.

Die drei Faktoren sind durch verschiedene Beziehungen miteinander verbunden: Um eine Aufgabe mit einem Computer transparent bearbeiten zu können, benötigt die Benutzerin *Fachkompetenz* in bezug auf die Aufgabe. Das dieser Betrachtungsweise zugrundeliegende Menschenbild sieht die Benutzerinnen in einer aktiven und kompetenten Rolle, d.h. von einer Benutzerin wird Kompetenz im Anwendungsgebiet und aktives Bemühen um die Beherrschung des Systems erwartet. Das Computersystem seinerseits muß *angemessen* für die Aufgabe sein, so daß es der Benutzerin nicht zusätzliche oder unnötige Arbeit aufbürdet. Um dies zu erreichen, muß die Anwendungsfunktionalität gut auf das zu bearbeitende Aufgabenfeld abgestimmt sein.

In Abbildung 3-1 sind zwei verschiedene Stufen dargestellt, wie Transparenz in einer Anwendungssituation gefördert werden kann. Diese beiden Stufen unterscheiden sich in der Bezeichnung von den in [Maaß94] ursprünglich gewählten Begriffen. Maaß benutzt für den Begriff "Beeinflussung" das Wort "Beherrschung", für "Anpassungsspielräume" verwendet sie "Spielräume" und anstelle von "Orientierungsmöglichkeiten" taucht bei ihr "individuelle Durchschaubarkeit" auf. Im folgenden werden die beiden Stufen und die hier gewählte Benennung erläutert.

Will eine Benutzerin ein Programm kennenlernen, so besteht ihr erster Schritt i.d.R. darin, daß sie versucht, sich ein *mentales Modell* von einem System zu bilden und sich somit im System zu orientieren. Ein Computer kann diesen Prozeß der Bildung eines zweckmäßigen mentalen Modell unterstützen, indem er der Benutzerin *Orientierungsmöglichkeiten* gibt. Der in dieser Arbeit verwendete Begriff "Orientierungsmöglichkeiten" scheint hierfür besser geeignet als Maaß' Benennung "individuelle Durchschaubarkeit", weil der Vorgang des Orientierens oder des Durchschauens immer von der Benutzerin und nicht vom Computersystem ausgehen kann. Vielmehr sollte die Benutzerin das Computersystem individuell beim Gebrauch durchschauen und alle für ihren Arbeitskontext relevante Funktionalität und Handhabung kennen. Zu diesem Zweck muß das System Orientierungsmöglichkeiten anbieten.

Ist eine Benutzerin nicht nur in der Lage, sich in einem System zu orientieren, sondern kann sie das Computersystem auch *beeinflussen*, so ist die zweite Stufe der Transparenz erreicht. Die Beeinflußbarkeit eines Systems erlaubt es der Benutzerin, die Anwendungs- und Handhabungsfunktionen für ihre eigenen Zwecke zu gebrauchen und anzupassen. Ein Computersystem muß der Benutzerin hierfür *Anpassungsspielräume* bieten, so daß sie z.B.:

- Funktionen umbenennen,
- Wege durch Zusammenfassen von Funktionssequenzen unter selbstgewählten Namen abkürzen,
- nicht gewünschte Optionen oder Ausgaben ausblenden sowie
- Darstellungsformen verändern kann (s. [Oberquelle94]).

Auch für diese Stufe des Transparenzkonzepts wurde eine andere Benennung gewählt als in [Maaß94]. Der Begriff "Anpassungsspielräume" taucht in Maaß' *Diskussion* des Transparenzkonzepts meist an Stelle des umfassenderen Ausdrucks "Spielräume" auf. In der *graphischen Darstellung* für Transparenz verwendet Maaß aber das Wort "Spielräume". Da der Begriff "Anpassungsspielräume" umfassender ist, wurde er in dieser Arbeit in die graphische Darstellung und Beschreibung übernommen. Für die zweite Stufe der Transparenz wurde hier die Benennung "Beeinflussung" anstelle des Begriffs "Beherrschung" gewählt. Aus der hier vertretenen Sicht erlauben Anpassungspielräume der Benutzerin, das System

zu beeinflussen. Befindet sie sich in einer transparenten Anwendungssituation, weil sie sich in dem Systemen mit Hilfe eines mentalen Modells orientieren kann und das System beeinflussen kann, so *beherrscht* sie das System.

Für diese Arbeit ist besonders die erste Stufe der Transparenz *Orientierung* mit Hilfe eines mentalen Modells von Bedeutung. Die Idee der symbolischen Maschinen hat deutlich gemacht, daß die Benutzerin an der Schnittstelle eines Computersystems lediglich Symbole vorfindet, für die sie eine Interpretation finden muß, um ihre Arbeit erledigen zu können. Das Konzept der Transparenz sieht vor, daß von der Seite des Computers aus Hinweise auf eine sinnvolle Interpretation gegeben werden können.

Um die Interpretation durchführen zu können, muß die Benutzerinnen in der symbolischen Maschine oder der Benutzungsschnittstelle des Computersystems *Orientierungsmöglichkeiten* vorfinden. Zu diesem Zweck muß in der symbolischen Maschine eine *aufgabenangemessene* oder adäquate Menge von Zeichen und Regeln vorhanden sein. Nur so kann gewährleistet werden, daß die Benutzerin mit der entsprechenden *Fachkompetenz* die notwendige Interpretation überhaupt leisten kann.

Daß an einer Benutzungsschnittstelle Orientierungsmöglichkeiten vorhanden sind, auf die die Benutzerin zurückgreifen kann, um die Symbole zu interpretieren, läßt sich auf verschiedene Art und Weise fördern.

3.3 Förderung von Orientierung

Beispiele für Maßnahmen, um Orientierung zu fördern, sind in der Software-Ergonomie relativ leicht zu finden. Zwei sehr populäre Konzepte aus diesem Bereich sollen im folgenden exemplarisch auf ihre Möglichkeiten hin untersucht werden, ob sie der Benutzerin die Arbeit mit einem Computersystem erleichtern. Zuerst wird *Konsistenz* diskutiert und anschließend werden *Metaphern* einer genaueren Betrachtung unterzogen. Beide Konzepte lassen sich bei der Gestaltung von Hilfesystemen einsetzen.

3.3.1 Orientierung durch Konsistenz

Konsistenz ist eine zentrale Forderung für die Gestaltung von benutzungsgerechten Systemen[13] . In [Maaß94] werden konsistente Systeme folgendermaßen beschrieben:

> "Konsistente Systeme verhalten sich regelmäßig, gleichmäßig, der Benutzer kann Erwartungen ausbilden, das System reagiert erwartungskonform. Konsistenz ist das 'Prinzip der geringsten Überraschung'. Es wird angenommen, daß konsistente Systeme aufgrund ihrer Struktur leichter zu erlernen sind. Allgemein bedeutet Konsistenz soviel wie strukturelle Übereinstimmung oder Widerspruchsfreiheit."

Aus diesen Aussagen geht hervor, daß eine Benutzerin sich in konsistenten Systemen leichter orientieren kann, weil ihre während der Benutzung gebildeten Erwartungen erfüllt werden. Die Erwartungen der Benutzerin beziehen sich nach [Maaß94] offenbar insbesondere auf die Strukturierung des Systems und auf die Art wie es auf Eingaben reagiert. Ist das System in diesen Bereichen regelmäßig, so ist eine gute Voraussetzung für eine transparente Benutzungssituation geschaffen. Bei einem konsistenten System lenken keine Unregelmäßigkeiten in der Struktur und der Reaktion von einer naheliegenden Interpretation ab.

[13] s. [Horton90, Maaß94, Norman89]

Daher ist zu erwarten, daß es einer Benutzerin gelingen kann, eine geeignete Interpretation für das verwendete System (oder die symbolische Maschine) zu finden.

Es existieren in der Literatur unterschiedlichen Vorstellungen davon, was im Einzelnen im Rahmen von Konsistenz von einem Computersystem gefordert werden sollte. Ein Konzept, das sich als tragfähig erwiesen hat, wurde von Wendy Kellogg vorgeschlagen [Kellogg87]. Kellogg prägte die Begriffe *prozedurale* und *konzeptuelle* Konsistenz.

Unter prozeduraler Konsistenz oder Verfahrenskonsistenz versteht Kellogg, daß verschiedene Systemteile, z.B. das Layout der Bildschirmmasken, die Interaktionsart, der Aufbau von Menüs oder die Anwendbarkeit von Funktionen auf Objekte, gleichartig gestaltet sind. Prozedurale Konsistenz kann sich sowohl auf die Ähnlichkeit in einem System als auch auf die Regelmäßigkeiten zwischen verschiedenen Systemen beziehen. Um diese verschiedenen Bezugsgrößen ebenfalls zu fassen, wird Konsistenz in einem einzigen System als interne Konsistenz bezeichnet und für mehrere Systeme der Ausdruck externe Konsistenz verwendet. Auf prozedurale Konsistenz ausgerichtet sind die von Herstellerinnen und Anwenderinnen eingesetzten Richtlinien für graphische Oberflächen ("styleguides"). Indem die äußere Form von Computersystemen konsistent gestaltet wird, kann der Wiedererkennungsgrad wirkungsvoll gesteigert werden. Ob eine Benutzerin für die wiedererkannten Systemobjekte oder -funktionen allerdings die geeignete Interpretation finden kann, wird durch prozedurale Konsistenz nicht beeinflußt.

Konzeptuelle Konsistenz oder begrifflich-gedankliche Konsistenz, die Kellogg gegen prozedurale Konsistenz absetzt, bezieht sich auf semantische Konzepte. Darunter wird die Abbildung von Arbeitszielen der Benutzerin auf Systemobjekte und -funktionen und die Kohärenz der Systemstruktur gefaßt. Die Abbildung von Arbeitszielen der Benutzerin auf Systemobjekte und -funktionen kann im Rahmen der symbolischen Maschinen als Interpretation der Repräsentation für die *Gegenstände* der maschinisierten Tätigkeiten sowie als Interpretation der Repräsentation für die *Handlungen* innerhalb dieser Tätigkeiten verstanden werden. Unter der Kohärenz der Systemstruktur versteht Kellogg, daß analoge Tätigkeiten in gleiche Handlungen zerlegt werden und daß Aktionen auf allen Objekten ausführbar sind, bei denen sie sinnvoll erscheinen.

3.3.2 Orientierung durch Metaphern

Eine Metapher ist eine sprachliche Figur, die einen Ausdruck aus seinem umgangssprachlichen Zusammenhang nimmt und in einem anderen Zusammenhang verwendet, d.h. es wird eine Bedeutungsübertragung vorgenommen (s. [HügLüb91]). Diese Aussage macht darauf aufmerksam, was im Zusammenhang mit Orientierung von Metaphern erwartet werden kann: Mit Hilfe einer Metapher kann eine Bedeutungsübertragung vorgenommen werden. Welche Bedingungen zutreffen müssen, damit diese Bedeutungsübertragung möglich ist, bringt E. MacCormac auf den Punkt:

> "Explanations without metaphor would be difficult if not impossible, for in order to describe the unknown, we must resort to concepts that we know and understand, and that is the essence of a metaphor - an unusual juxtaposition of the familiar and the unfamiliar." [MacCormac85]

Eine Metapher kann also helfen, die Bedeutung eines Zusammenhangs aus einem vertrauten auf ein ungewohntes Gebiet zu übertragen, weil sie auf ungewöhnliche Weise zwei Bereiche nebeneinanderstellt.

Werden Metaphern bei der Gestaltung von Computersystemen verwendet, so wird ein Zusammenhang aus der realen Welt auf den Computer übertragen (s. [CarMac82, CaMaKe87, LakJoh80]). Ist die Benutzerin in der Lage, die Metapher zu erkennen, so kann sie die Bedeutung des vertrauten Bereichs für "das Neue, Unbekannte" im Computersystem einsetzen. Durch die Metapher wird der Versuch, das Computersystem zu interpretieren, direkt auf einen bestimmten Aspekt des Anwendungsgebiets gelenkt. Deutlich wird bei diesen Überlegungen, daß eine Metapher nur dann ihren Zweck erfüllen kann, wenn die intendierte Bedeutungsübertragung (s. [HügLüb91]) oder Nebeneinanderstellung (s. [MacCormac85]) für die Benutzerin überhaupt wahrnehmbar ist.

In Computersystemen werden Metaphern deshalb häufig durch bildliche Darstellungen des entsprechenden Gegenstands unterstützt, auf den sich die Metapher bezieht. Für jede Benutzerin, die sich in einer normalen Büroumgebung zurechtfindet, sind Abbilder von Papierkörben oder Papierblättern einleuchtende Hinweise auf den Sinn und die Handhabung der dahinter verborgenen Strukturen oder Funktionen. Häufig werden in Computersystemen aber auch Metaphern verwendet, die nicht auf den ersten Blick, wohl aber mit Hilfe des Namens nachvollziehbar sind, wie z.B. *Ordner* - zum Ablegen von Dokumenten, *Schreibtisch* - der Bildschirm als Arbeitsfläche, *Menü* - eine Speisekarte für Einträge, aus denen gewählt werden kann oder *direkte Manipulation* - Objekte können räumlich manipuliert werden.

Im Kapitel 2 wurde die Werkzeug-Material Metapher[14] eingeführt. Bei dieser Metapher werden bekannte Werkzeuge und Materialien nicht aus der Arbeitswelt *visuell* auf dem Bildschirm *abgebildet*, sondern die Softwarekomponenten *verhalten* sich wie Werkzeuge und Materialien. Mit Hilfe des Leitbilds vom Arbeitsplatz für qualifizierte menschliche Arbeitstätigkeit wird die Benutzerin mit einer *umfassenden* Metapher ausgestattet. Ein Großteil der im vorigen erwähnten Metaphern läßt sich durch die Werkzeug-Material Metapher in ein Gesamtbild integrieren. Hat man die Vorstellung, daß mit Werkzeugen an Materialien gearbeitet wird, so ist es naheliegend, daß ein *Schreibtisch*, als Ort der qualifizierten Arbeit, vorhanden ist. Werkzeugen und Materialien ist immanent, daß sie zur Verwendung angefaßt, d.h. *direkt manipuliert* werden. *Ordner* und *Papierkörbe* sind übliche Materialien an einem Arbeitsplatz und haben daher auch in der Werkzeug-Material Metapher unter dem allgemeinen Begriff *Behälter* einen nicht unwichtigen Platz.

Trotz der hervorragenden Möglichkeiten, mit Hilfe der Werkzeug-Material Metapher Analogien zwischen Computersystemen und Arbeitsplätzen herzustellen, muß die Benutzerin zu Beginn mit dieser Metapher vertraut gemacht werden. Bei Schulung und beim Einsatz von nach der Werkzeug-Material Metapher entwickelten Systemen hat sich gezeigt, daß die Benutzerinnen und Benutzer auf die verwendete Metapher hingewiesen werden müssen. Ist den Benutzerinnen und Benutzern bewußt, daß das System ihnen einen Arbeitsplatz mit Werkzeugen zur Bearbeitung von Materialien anbietet, die ihnen aus ihrer Arbeitswelt bekannt sind, so ist eine schnelle Einarbeitung möglich.

Es bleibt festzuhalten, daß Metaphern Brücken schlagen zwischen der Gedankenwelt der Benutzerin und den Systemobjekten und -funktionen. Sie können aber nur dann eine trans-

[14] Die Werkzeug-Material Metapher unterscheidet sich von den anderen hier beschriebenen Metaphern dadurch, daß sie sowohl an der Benutzungsschnittstelle aus auch im Softwareentwurf auf der Designebene eingesetzt wird. Die Metapher wird also nicht allein zu dem Zweck verwendet, die Benutzerin zu unterstützen, sondern sie kann von den Entwicklerinnen für die Architektur des Systems herangezogen werden.

parente Benutzungssituation und die Interpretation des Computersystems fördern, wenn sie für die Benutzerin erkennbar und verständlich sind.

3.4 Hilfesysteme aus der Sicht von Transparenz und Sprache

Auch am Ende dieses Kapitels lassen sich die Anforderungen an Hilfesysteme teilweise umformulieren, neu interpretieren und ergänzen.

Im ersten Kapitel dieser Arbeit wurde argumentiert, daß die Erklärungskomponente eines Hilfesystems Benutzerinnen in die Lage versetzen sollten, die von der Entwicklerin intendierte *Handhabung* der Applikation und das *Verständnis* der Entwicklerin vom Anwendungsgebiet zu erfassen. Das Konzept der Transparenz und die Idee der symbolischen Maschine unterstützt diese Forderungen.

Aus der Sichtweise der symbolischen Maschine werden der Benutzerin an der Benutzungsschnittstelle lediglich Symbole zur Verfügung gestellt, mit deren Hilfe sie Arbeit verrichten kann (s. Abschnitt 3.1). Um die symbolische Maschine für ihre Zwecke einsetzen zu können, muß die Benutzerin allerdings in der Lage sein, die Symbole zu interpretieren. Die im Rahmen von Hilfesystemen in Anforderung 1-2 aufgestellte Anspruch, die Handhabung einer Applikation und das in der Applikation abgebildete Verständnis der Entwicklerin vom Anwendungsgebiet zu erklären, läßt sich hier wiederfinden und schärfer formulieren.

Die Handhabung und das Verständnis der Entwicklerin schlagen sich in den an der Benutzungsschnittstelle zur Verfügung gestellten Symbolen nieder. Um aber eine Orientierung an und in einer symbolischen Maschine zu ermöglichen, greifen Entwicklerinnen zu Metaphern, um das Zusammenspiel der Symbole explizit zu machen. Außerdem versuchen sie, ihr System konsistent in bezug auf das System selbst und auf das Anwendungsgebiet zu gestalten. Die von Entwicklerinnen eingesetzten Mittel, nämlich Symbole, Metaphern und Konsistenz, müssen den Benutzerinnen erläutert werden, damit sowohl die Handhabung als auch das Verständnis der Entwicklerinnen für die eigenen Zwecke interpretiert werden können. In den nachfolgend erweiterten Anforderungen an Hilfesysteme schlägt sich diese Forderung für den Inhalt der Erklärungskomponente wieder.

Neben einer präziseren Formulierung der Forderungen an die Erklärungskomponente gibt das Konzept der Transparenz einen Hinweis auf eine weitere Aufgabe des Hilfesystems. Bisher wurde von einem Hilfesystem gefordert, Information anzuzeigen und das Auffinden von Information zu ermöglichen. Das Konzept der Transparenz verlangt, daß jedes System der Benutzerin *Orientierungsmöglichkeiten* bietet. Auch und gerade ein Hilfesystem sollte dieser Forderung genügen. Mögliche Techniken, um neben Metaphern und Konsistenz Orientierung zu bieten, sind Historien, in denen Benutzerinnen ihren Weg zurückverfolgen können, klare Namengebung für alle Arten von Symbolen an der Benutzungsschnittstelle sowie graphische Darstellungen, die verschiedene Zusammenhänge des Hilfesystems erläutern.

Anforderung 3-3:

> Die Aufgaben eines Hilfesystems bestehen darin, Informationen anzuzeigen, das Auffinden von Informationen zu ermöglichen und der Benutzerin **Orientierungsmöglichkeiten** in der Fülle der Hilfeinformationen zu geben.
>
> Ein Hilfesystem ist programmtechnisch unabhängig von einer Applikation und besteht aus zwei Komponenten: einem Hilfewerkzeug und einem Hil-

fematerial. Das Hilfewerkzeug ermöglicht der Benutzerin den Zugriff auf das Hilfematerial, welches Hilfe zu einer Applikation enthält.

Das Hilfewerkzeug:

- bietet passive Hilfe an, da Hilfesituationen nie eindeutig bestimmbar sind,
- stellt dynamische Hilfe zur Verfügung, die aufgrund des Zustands der Anwendung kontextsensitiv ist und
- stellt insofern individuelle Hilfe bereit, als daß es der Benutzerin Hilfe zur explorativen Selbsthilfe gibt.

Das Hilfematerial besteht aus kurzen textuellen und visuellen Einheiten, die als Hypermedia organisiert sind. Diese Einheiten erläutern die an der Benutzungsschnittstelle verwendeten Symbole für **Gegenstände** und **Handlungen** und stellen sie in eine **konsistente** Beziehung zum Anwendungsgebiet. Außerdem erklärt das Hilfematerial die in der Applikation verwendeten **Metaphern**.

Mit Anforderung 3-3 sind alle für Hilfesysteme wichtigen Erkenntnisse zusammengefaßt, die in den letzten drei Kapiteln herausgearbeitet wurden. Anforderung 3-3 bildet die Basis, auf der der Prozeß zur Entwicklung des Hilfesystems und das Produkt Hilfesystem bzw. Hilfe-Framework selbst aufbauen konnten.

II Prozeß

Nachdem im ersten Teil die theoretischen Grundlagen aufgearbeitet worden sind, befaßt sich dieser Teil mit dem Verlauf des zugrundeliegenden Projekts. Einerseits zeigt dieser Teil, wie sich das Projekt aus den Grundlagen entwickelt hat und in welcher Weise sich die Grundlagen konkretisiert haben. Auf der anderen Seite wird in diesem Teil deutlich, daß die in Teil I vorgestellten Themen: Hilfesysteme, Werkzeug-Material Metapher sowie Transparenz und Sprache, zum Teil erst im Verlauf des Projekts zu dem Stellenwert gelangt sind, den sie in dieser Arbeit einnehmen.

Um den Umfang und den Inhalt des Projekts zu klären, beginnt dieser Teil mit einem Projektbericht (s. Kapitel 4). Der Projektbericht klärt die Fragestellung und macht den Verlauf des Projekts deutlich. Außerdem wird das Anwendungsgebiet und die Applikation vorgestellt für die das Hilfesystem während des Projekts entwickelt wurde. Den Abschluß des Projektberichts bildet die Darstellung einer im Projektverlauf durchgeführten empirischen Untersuchung.

In der zweiten Hälfte dieses Teils (Prozeß) wird der Prototyp des entwickelten Hilfesystems beschreiben (s. Kapitel 5). Um einen Eindruck vom Prototyp zu vermitteln, wird anhand verschiedener Dokumente die Arbeit der Benutzerin mit dem Hilfesystem vorgestellt. Der Prototyp des Hilfesystems ist anhand der Grundlagen aus Teil I entwickelt worden. Trotzdem erfüllt er nicht alle Anforderungen (s. Anforderung 3-1), die sich aus den Grundlagen ergeben haben. Der letzte Abschnitt befaßt sich daher mit der Frage, inwieweit und aus welchen Gründen der Prototyp bestimmte Anforderungen nicht oder nur teilweise erfüllt.

4 Projektbericht

Das Projekt, das dieser Arbeit zugrunde liegt, begann im Juli 1994 mit dem Ziel, ein Hilfesystem für eine konkrete Applikation den Aufgabennetzeditor SANE zu entwickeln. Ende Februar 1995 war ein einsatzfähiger Prototyp fertiggestellt, der die Fragestellung des Projekts zufriedenstellend klären konnte.

Die Fragestellung des Projekts ist in dieser Arbeit bereits mehrfach angeklungen und wird zu Beginn dieses Kapitels explizit geklärt. Den zweiten Teil bildet ein Bericht über den Projektverlauf. Um einen Einblick in das Anwendungsgebiet zu geben, für das das Hilfesystem entwickelt wurde, wird die aufgabenbezogenen Anforderungsermittlung, Aufgabennetze und der Aufgabennetzeditor SANE vorgestellt. Während des Projekts wurde eine empirische Untersuchung durchgeführt, die den Entwicklungsprozeß des Hilfesystems beeinflußt hat. Den Abschluß des Projektberichts bildet daher die Beschreibung der empirische Untersuchung und ihrer Implikationen für das entstandene Produkt.

4.1 Fragestellung des Projekts

Ausgangspunkt der Fragestellung war der Aufgabennetzeditor SANE (s. Abschnitt 4.3.2), der im Rahmen der Vorlesung "Einführung in die Softwaretechnik" von Studierenden eingesetzt wird. Das zu entwickelnde Hilfesystem sollte und soll die Benutzerinnen und Benutzern des Aufgabennetzeditors unterstützen.

Da der Aufgabennetzeditor nach der Werkzeug-Material Metapher entwickelt wurde, lag die Überlegung nahe, auch das Hilfesystem nach dieser Metapher zu entwerfen. Die Werkzeug-Material Metapher bietet eine Reihe von Ausdrucksmitteln an: Werkzeuge, Materia-

lien, Aspekte, Automaten und ein Repertoire an Mustern (s. Kapitel 2). Als das Projekt ins Leben gerufen wurde, existierten bereits Überlegungen zu der Stellung eines Hilfesystem innerhalb der Werkzeug-Material Metapher. Ein Hilfesystem wurde in diesem Zusammenhang als Werkzeug betrachtet, das auf einer Applikation, als sein *Material*, arbeitet, indem es Hilfe zu der Applikation anbietet. Eine Umsetzung dieser Überlegung war zu diesem Zeitpunkt noch nicht erfolgt.

Gleichzeitig entstand auf Anregung von Yvonne Dittrich[15] die Idee, dem sprachlichen Aspekt bei der Erstellung eines Hilfesystems besonderes Gewicht beizumessen. Das Hilfesystem sollte sich besonders dadurch von anderen Hilfesystemen unterscheiden, daß bei der Entwicklung vermehrt auf die Gestaltung der erklärenden Texte eingegangen wird.

Als Fragestellung der Arbeit, die sich auch im Titel widerspiegelt, ergab sich daher: Wie läßt sich ein Hilfesystem im Rahmen der Werkzeug-Material Metapher realisieren, und inwieweit lassen sich sprachphilosophische Überlegungen bei der Entwicklung eines Hilfesystems verwenden?

4.2 Projektverlauf

Tabelle 4-1 enthält eine Auflistung der wichtigen Aktivitäten und in Dokumenten festgehaltenen Ergebnisse des Projekts. Im folgenden werden zuerst die verschiedenen Dokumenttypen[16] erläutert und anschließend die einzelnen Aktivitäten näher beschrieben.

Zeitraum	**Aktivität**
Juli - August 1994	Analyse der Handhabung von SANE anhand von Beispielen Einarbeitung in Literatur zu Hilfesystemen
September 1994	Erstellen von Szenarien und Systemvisionen Einarbeitung in eine vorhanden Hypertextanwendung
Oktober 1994	Beginn der empirischen Untersuchung
November 1994	Abschluß und Auswertung der empirische Untersuchung Fertigstellung eines ersten Prototyps
Dezember 1994	Redesign des ersten Prototyps
Februar 1995	Ausbau der Funktionalität zu einer einsatzfähigen Anwendung

Tabelle 4-1: Projektverlauf

Einige der in Tabelle 4-1 aufgeführten Aktivitäten hatten Dokumente zum Ergebnis, die am Arbeitsbereich Softwaretechnik der Universität Hamburg im Fachbereich Informatik bei der Softwareentwicklung im Zusammenhang mit der Werkzeug-Material Metapher (s. Kapitel 2) eingesetzt werden: Szenarien, Systemvisionen und Prototypen.

In Szenarien halten die Entwicklerinnen in der Fachsprache der Anwendung typische Arbeitsabläufe, Handlungsstudien und den Arbeitskontext des Anwendungsfelds fest. Szenarien werden beispielhaft und episodisch verfaßt, um im Entwicklungsprozeß Fragen und Begriffe zu klären.

Systemvisionen beschreiben die Erledigung von Arbeitsaufgaben, wie sie sich die Entwicklerin nach Einführung des Anwendungssystems vorstellt. Dabei wird das Erscheinungsbild

---

[15] wissenschaftliche Mitarbeiterin am Fachbereich Informatik der Universität Hamburg

[16] [Floyd92, GryZül92, KiGrZü94]

des Systems nachgezeichnet, und die antizipierten Umgangsformen mit dem System werden festgehalten. Systemvisionen verbinden die mögliche Handhabung eines Computersystems mit bekannten und neuen Arbeitshandlungen und helfen im Entwicklungsprozeß, die Szenarien mit Prototypen zu verbinden. Im nächsten Kapitel wird ein Teil der für das Hilfesystem erstellten Systemvisionen zur Beschreibung des Prototyps verwendet.

Prototypen gehören im Rahmen der Softwareentwicklung zur konstruktiven Umsetzung der sich entwickelnden Systemlösung. Ein Prototyp ist eine spezielle Ausprägung eines ablauffähigen Softwaresystems. Er realisiert einen vorab ausgewählten Aspekt des Zielsystems und bildet so eine Basis für die Kommunikation zwischen Benutzerinnen und Entwicklerinnen. Szenarien, Systemvisionen und Prototypen wurden bei der Entwicklung des Hilfesystems eingesetzt.

Im Juli 1994 begann das Projekt mit einer ersten Untersuchung des Anwendungsgebiets. Da der Aufgabennetzeditor SANE mit einem Hilfesystem versehen werden sollte, standen seine Verwendung und dabei auftauchende Wünsche nach Hilfe im Vordergrund. Zu diesem Zeitpunkt war es nicht möglich, tatsächliche Benutzerinnen des Aufgabennetzeditors bei der Arbeit zu beobachten, weil SANE erst im Ende Oktober 1994 zum Einsatz kommen sollte. Daher wurde SANE im Selbstversuch anhand von Beispielen erschlossen.

Aufgrund dieser Erfahrungen wurden im September 1994 Szenarien verfaßt, die die Arbeit mit SANE und einem entsprechenden Handbuch beschrieben. Das Handbuch wurde hier als möglicher "Vorgänger" eines Hilfesystems betrachtet, um die typischen Umgangsformen mit Gegenständen zu identifizieren, die als Hilfsmittel verwendet werden. Aus den so entstandenen Szenarien wurden Systemvisionen eines Hilfesystems entwickelt, die auf die Handhabung und die enthaltenen Erklärungstexte eingingen. Sowohl die Szenarien als auch die Systemvisionen konnten nur mit wenigen Studentinnen diskutiert werden, weil das Hauptinteresse der Studentinnen, die mit SANE arbeiteten, weniger in einer kontinuierlichen Nutzung von SANE lag, für die ein Hilfesystem sicherlich erforderlich ist, sondern mehr in der schnellen Erledigung ihrer Aufgaben.

Im gleichen Zeitraum, in dem die Szenarien und Systemvisionen entstanden, fiel die Einarbeitung in eine bereits existierende Hypertextanwendung, den "Glossarbrowser" (s. [Heß93]). Der Glossarbrowser befaßt sich mit der Darstellung und Verwaltung von Hypertexten. Die anfängliche Idee, einen großen Teil dieser Anwendung wiederverwenden zu können, mußte fallengelassen werden, weil der Aspekt der Wiederverwendbarkeit bei der Entwicklung des Glossarbrowsers nicht im Vordergrund gestanden hatte. Bei der Umsetzung der Hypertextstrukturen und ihrer Darstellung im Hilfesystem wurden die entsprechenden Klassen des Glossarbrowsers als Vorlagen verwendet. Um das Hilfesystem in der vorgesehenen Zeit umsetzen zu können, wurde beschlossen, für die Erklärungskomponente nur Texte vorzusehen. Die Integration von Bildern wurde im Hilfesystem bereits angelegt und ist für eine weitere Ausbaustufe wünschenswert.

Ende Oktober konnte die empirische Untersuchung der Studierenden beginnen, die SANE verwendeten. Im Rahmen dieser Untersuchung wurden Erkenntnisse darüber gesammelt, welche Teile von SANE den Studierenden besondere Probleme bereiteten. Da die Untersuchung Einfluß auf die Entwicklung des Hilfesystem hatte, wird sie am Ende dieses Kapitel in Abschnitt 4.4 detaillierter beschrieben.

Im November 1994 wurde die empirische Untersuchung ausgewertet und ein erster Prototyp fertiggestellt. Dieser Prototyp war in der Lage, Hilfetexte in Form von Hyperknoten

anzuzeigen und den Benutzerinnen das Navigieren zwischen den Hyperknoten zu ermöglichen. Außerdem stellte er weitere Funktionalität zum Suchen von Informationen zur Verfügung. Der Prototyp wurde von einigen Benutzerinnen und Benutzern auf seine Verwendbarkeit hin getestet. Die Kritik der Benutzerinnen und Benutzer führte dazu, daß an der Handhabung des Hilfewerkzeugs Änderungen vorgenommen wurden. Der erste Prototyp war in der Lage, lediglich *einen* Knoten des Hypertextes zur Zeit anzuzeigen. Für die Benutzerinnen und Benutzer stellte dies ein Problem dar, weil sie verschiedene Texte gleichzeitig auf dem Bildschirm angezeigt haben wollten.

Von Dezember 1994 bis Ende Februar 1995 wurde der Prototyp des Hilfesystems daher weiter ausgebaut. Dabei wurde das Hilfesystem um Kontextsensitivität erweitert. Das Hilfesystem konnte nun auf den jeweiligen Zustand von SANE Bezug nehmen. Zu diesem Zweck mußte die Schnittstelle von SANE so angepaßt werden, daß das Hilfesystem die benötigten Informationen erfragen konnte.

Gleichzeitig entstand die Idee, das Hilfesystem nicht nur als zusätzlichen Teil von SANE zu entwickeln, sondern ein allgemeines Hilfesystem für Applikationen entstehen zu lassen, die nach der Werkzeug-Material Metapher konstruiert wurden. Um diese Maßgabe zu erfüllen, wurde mit Hilfe eines Aspektes eine Abstraktion zwischen dem Hilfesystem und SANE eingefügt. Dieser Aspekt definiert eine abstrakte Schnittstelle, der eine Applikation genügen muß, wenn sie mit dem Hilfesystem zusammenarbeiten soll. Auf diese Weise konnte das Hilfesystem als unabhängige Anwendung betrachtet werden, die über eine Schnittstelle mit beliebigen Applikationen interagieren kann. Die Zusammenarbeit mit verschiedenen Applikationen konnte nur sichergestellt werden, indem für die Interaktion zwischen Hilfesystem und Applikation ebenfalls eine Abstraktion gebildet wurde. Zu diesem Zweck wurde eine abstrakte Form der Kontextsensitivität entwickelt (s. Kapitel 7).

Neben dieser Verallgemeinerung des Hilfesystems wurde deutlich, daß noch weitere Werkzeuge erforderlich sein würden, um die Verwendung des Hilfesystems zu ermöglichen. Bevor das Hilfesystem mit einer Applikation zum Einsatz kommen kann, müssen erklärende Texte für die Applikationen erarbeitet und gestaltet werden. Um Kontextsensitivität zu erreichen, benötigt das Hilfesystem außerdem Informationen, wie von bestimmten Zuständen der Applikation auf entsprechende Hilfetexten zu schließen ist. Dem Hilfesystem wurden daher weitere Werkzeuge und Materialien zur Seite gestellt, die für diese Aufgaben geeignet erschienen (s. Kapitel 8).

Ende Februar 1995 wurde die Arbeit an dem Prototyp des Hilfesystems (s. Kapitel 5) beendet. Der Prototyp hatte eine Ausbaustufe erreicht, die einen Großteil der gestellten Anforderungen aus dem Grundlagenteil erfüllte.

4.3 Aufgabennetze zur aufgabenbezogenen Anforderungsermittlung

Der Aufgabennetzeditor SANE ("System für einen Aufgabennetzeditor", s. [Riehle93]) wird zur Darstellung von Aufgabennetzen verwendet, die bei der aufgabenbezogenen Anforderungsermittlung im Rahmen von STEPS ("Softwaretechnik für evolutionäre, partizipative Softwareentwicklung", s. [Floyd92]) eingesetzt werden. Aufgabenbezogene Anforderungsermittlung hat im Gegensatz zu vielen datenzentrierten Vorgehensweisen zum Ziel, Menschen und ihre Aufgaben in den Vordergrund zu stellen [Floyd92].

Um während der Anforderungsermittlung, die Aufgaben von Menschen darstellen zu können, wurden von Christiane Floyd und Reinhard Keil-Slawik Aufgabennetze entwickelt (s.

[Floyd92, Keil-Slawik89]). Bei Aufgabennetzen handelt es sich um ein graphisches, halbformales Darstellungsmittel. Aufgabennetze werden bei der aufgabenbezogenen Anforderungsermittlung eingesetzt, um wiederzugeben, wie Menschen zusammenarbeiten, wenn sie ein gemeinsames Ziel erreichen wollen.

Um diese Kooperation zu modellieren, werden *funktionelle Rollen* verwendet. Funktionelle Rollen sind für eine Anzahl zusammengehöriger Aufgabengebiete verantwortlich, es besteht aber kein zwingender Zusammenhang zwischen Personen und funktionellen Rollen. Vielmehr kann eine funktionelle Rolle von verschiedenen Personen ausgefüllt werden, und eine Person kann gleichwohl mehr als eine funktionelle Rolle einnehmen.

Um ihre Aufgaben in einem kooperativen Arbeitsprozeß zu erfüllen, tauschen funktionelle Rollen Informationen und physisch vorhandene Objekte aus. Aufgabennetze, die ausschließlich funktionelle Rollen und ihren Austausch betrachten, werden als Ist-Netze bezeichnet. Soll ein Computer in ein solches System integriert werden, so ist besonders der Austausch von Informationen und Objekten zwischen einem Rechner und den funktionellen Rollen von Interesse. Die daraus entstehenden Soll-Netze haben zum Ziel, eine Vision von einem Computersystem zu ermitteln, bei der die einzelnen Arbeitsvorgänge nicht festgeschrieben sind.

4.3.1 Aufgabennetze als Darstellungsmittel

Wird die Anforderungsermittlung in der oben beschriebenen Art und Weise durchgeführt, so erhält man als Ergebnis ein Aufgabennetz. Um dieses Aufgabennetz übersichtlich zu gestalten, wird es in mehrere Teilnetze aufgegliedert, die untereinander durch eine hierarchische Ordnung verbunden sind. Das in der Hierarchie oberste Teilnetz wird *Übersichtsnetz* genannt. Es enthält alle beteiligten funktionellen Rollen, ihren Austausch an Information und, wenn es sich um ein Soll-Netz handelt, den Rechner. Das Übersichtsnetz wird in weitere Teilnetze zerlegt, die jeweils einen speziellen Ausschnitt des Übersichtsnetzes in größerem Detail wiedergeben. Dieser Prozeß, in dessen Verlauf das Übersichtsnetz in mehrere Netze zerlegt wird, hat den Namen *Verfeinerung* und endet, wenn die Entwicklerin auf einer für sie ausreichend granularen Ebene angelangt ist. Ein Aufgabennetz, das mehrere Verfeinerungen enthält, besteht somit aus einer *Verfeinerungshierarchie*, in der jedes Teilnetz einen Ausschnitt des übergeordneten Teilnetzes ausführlich beschreibt.

In Abbildung 4-1 sind alle graphischen Elemente von Aufgabennetzen abgebildet. Die verschiedenen Elemente sind nicht mit Namen versehen, die im Laufe einer bestimmten Anforderungsermittlung entstanden sind, sondern mit ihren Namen als graphische Symbole von Aufgabennetzen.

Das wichtigste Symbol in einem Aufgabennetz ist das der *funktionellen Rolle*. Jede funktionelle Rolle ist für eine Anzahl von *Aufgabengebieten* zuständig. Die Aufgabengebiete ihrerseits können wieder in Aufgabengebiete und schließlich in *Tätigkeiten* verfeinert werden. Parallel zur Verfeinerung der funktionellen Rollen werden *Informationsstrukturen* wiederum in Informationsstrukturen oder in *Informationen* sowie *Objekte* verfeinert. Existiert für ein Aufgabengebiet oder eine Informationsstruktur eine Verfeinerung, so wird dies durch einen senkrechten Strich im jeweiligen graphischen Symbol deutlich gemacht (s. in Abbildung 4-1: Aufgabengebiet 1 und Informationsstruktur 2). Für Tätigkeiten, Informationen und Objekte können keine Verfeinerungen angelegt werden, weil es sich bei ihnen in der Terminologie der Aufgabennetze um elementare Einheiten handelt. Informationsstrukturen, Informationen und Objekte werden zwischen funktionellen Rollen und dem Rechner ausgetauscht.

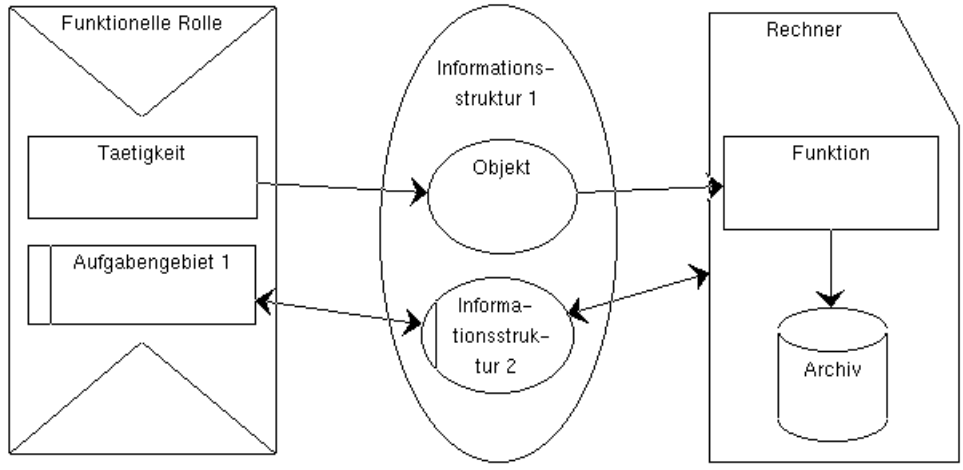


Abb. 4-1: Aufgabennetz

Der *Rechner* wird auf den oberen Ebenen der Netzhierarchie als *black box* dargestellt, mit dem die funktionellen Rollen Informationsstrukturen austauschen. Lediglich die *Archive*, welche verwendet werden, um Objekte persistent zu machen, sind auf allen Ebenen der Verfeinerung sichtbar. Am Ende der Verfeinerung, wenn man auf Informations- und Objektebene angelangt ist, werden *Funktionen* eingeführt, die dafür zuständig sind, Objekte in den Archiven abzuspeichern und sie herauszusuchen. Die *Kanten* in den Aufgabennetzen verdeutlichen den Fluß der Informationen und Objekte zwischen funktionellen Rollen und Rechner. Die Kanten können gerichtet sein und symbolisieren so die Übergabe einer Information(sstruktur) oder eines Objekts. Es sind aber auch Kanten mit zwei Pfeilspitzen vorgesehen, die den Austausch zwischen zwei Partnern über eine Information(sstruktur) oder ein Objekt darstellen.

Während der aufgabenbezogenen Anforderungsermittlung werden die einzelnen Verfeinerungen eines Aufgabennetzes auf DIN A4 Blätter gezeichnet. Jedes Blatt enthält den Rechner mit seinen Archiven, eine Verfeinerung einer funktionellen Rolle oder eines Aufgabengebiets und die entsprechenden Informationsstrukturen, Informationen oder Objekte. Die Rechnerfunktionen, Objekte und Archive werden in einem Glossar ausführlich definiert, so daß sie als Grundlage für einen modulorientierten Entwurf des zukünftigen Systems verwendet werden können.

4.3.2 SANE - eine Applikation zum Erstellen von Aufgabennetzen

Der Aufgabennetzeditor SANE ("System für einen Aufgabennetzeditor", s. [Riehle93]) hat im Laufe seiner Existenz mehrere durchgreifende Änderungen erlebt. In diesem Abschnitt wird sein Zustand zum Zeitpunkt der Untersuchung beschreiben, weil die zugehörigen Ergebnisse auf diese Version zurückzuführen sind.

SANE unterstützt die Benutzerin dabei, während der aufgabenbezogenen Anforderungsermittlung Aufgabennetze zu entwerfen und in Richtung des modulorientierten Entwurfs weiterzuentwickeln. In SANE hat die Benutzerin die Möglichkeit, die graphischen Symbole von Aufgabennetzen auf virtuellen DIN A4 Blättern anzuordnen. Alle im letzten Abschnitt erwähnten Symbole können erzeugt werden, indem der entsprechende Knopf aus der Objektpalette gedrückt und das Symbol auf einem DIN A4 Blatt abgelegt wird (s. Abbildung 4-2).

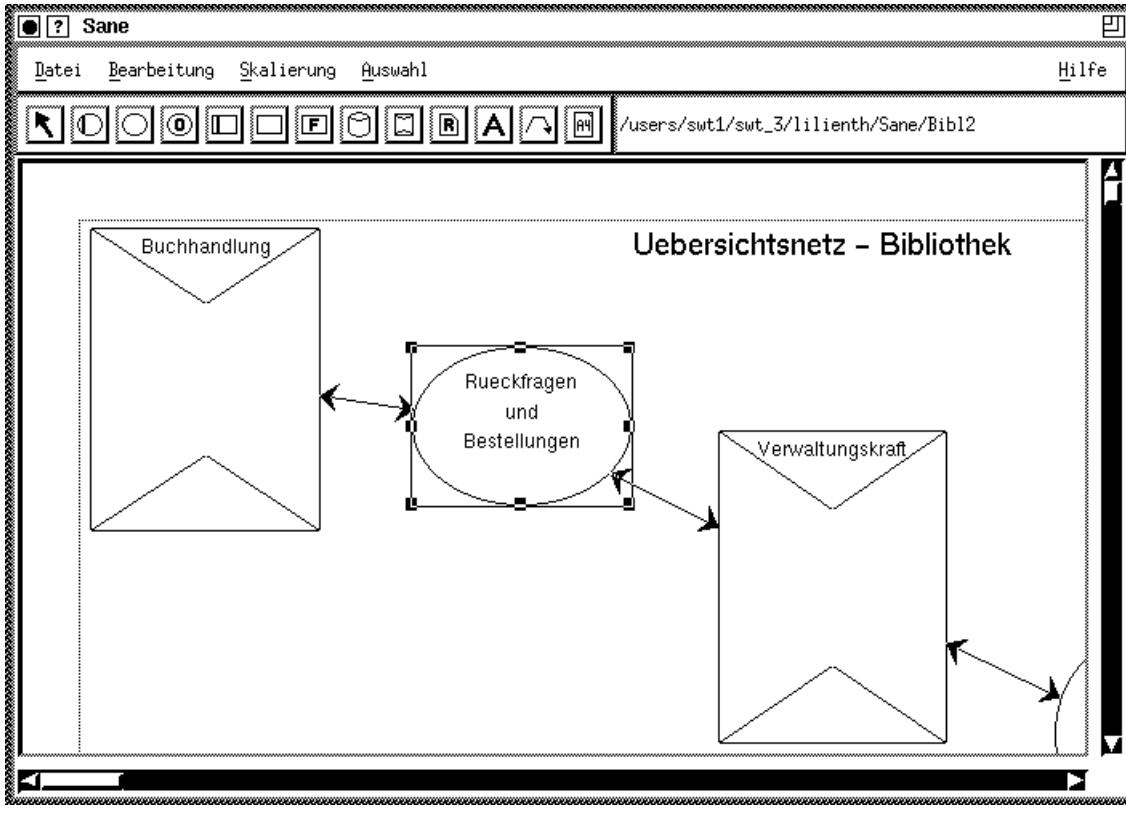


Abb. 4-2: SANE - der Aufgabennetzeditor

Die einzelnen Aufgabennetzsymbole können in SANE mit Hilfe der Maus direkt manipuliert werden. Neben diesen graphischen Möglichkeiten unterstützt der Aufgabennetzeditor die Benutzerin dabei, konsistente Netze zu erstellen. Ändert die Benutzerin den Namen eines Symbols, das mehrere Repräsentationen auf verschiedenen DIN A4 Blättern hat, so wird diese Änderung automatisch auf alle betroffenen Blätter übertragen. Während des Verfeinerns benötigt die Benutzerin häufig eine weitere Repräsentation eines Aufgabennetzsymbols, z.B. eine funktionelle Rolle, weil diese auf einem verfeinerten Netz mit den enthaltenen Aufgabengebieten und Tätigkeiten wieder erscheint.

Um die Konsistenz der Netze zu gewährleisten, müssen alle Repräsentationen eines Exemplars auf die selbe logische Einheit verweisen. Es gibt zwei verschiedene Methoden, dies zu bewerkstelligen. Eine Methode, um eine weitere Repräsentation eines graphischen Objekts zu erzeugen, besteht darin, ein Symbol zu kopieren und es dann auf ein neues Blatt Papier zu übertragen. Das selbe Ergebnis bekommt man, indem man ein neues Aufgabennetzsymbol vom entsprechenden Typ auf einem neuen Blatt Papier erzeugt. Um die gewünschte Konsistenz in den Netzen herzustellen, muß das eine graphische Objekt anschließend mit seiner anderen Repräsentation verbunden werden.

SANE markiert entsprechend der Aufgabennetzlogik das in der Verfeinerungshierarchie jeweils übergeordnete graphische Objekt mit einem Strich, um anzudeuten, daß eine Verfeinerung existiert. Um das Verfeinern zu vereinfachen, bietet SANE eine *Verfeinerungsfunktion* an, die den ersten Schritt des eben geschilderten Vorgangs durchführt. Wird die Verfeinerungsfunktion eines Symbols aufgerufen, so wird eine weitere Repräsentation des gewählten graphischen Objekts auf einem neuen Blatt Papier abgelegt.

4.4 Empirische Untersuchung

Im Wintersemester 1994 wurde am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik an der Universität Hamburg von Yvonne Dittrich und der Autorin eine empirische Untersuchung durchgeführt. Im Rahmen dieser Untersuchung wurden Personen beobachtet, die eine bestimmte Aufgabe zuerst ohne Computersystem und später unter Einsatz eines Softwareproduktes durchführen sollten. Bei den Versuchspersonen handelte es sich um Studierende, die an der Vorlesung und den Übungen zu "Einführung in die Softwaretechnik" bei Prof. C. Floyd teilnahmen. Um die theoretisch vermittelten Methoden auszuprobieren, hatten die Studierenden die Aufgabe, ein System zu entwickelt, das für eine Bibliothek bestimmt war. Als Unterstützung bei der Anforderungsermittlung verwendeten die Studierenden Aufgabennetze (s. Abschnitt 4.3.1) und den Aufgabennetzeditor SANE (s. Abschnitt 4.3.2).

In der Untersuchung sollte die Fragen geklärt werden:

(1) Welche Hilfe benötigen Benutzerinnen und Benutzer, wenn sie mit SANE arbeiten?

(2a) Wie ändert sich die Sprache der Benutzerinnen und Benutzer, wenn sie SANE verwenden?

Da diese Arbeit auf Hilfesysteme ausgerichtet ist, wird hier nur auf die für Hilfesysteme relevanten Fragen und Ergebnisse eingegangen[17]. Aus Sicht der Entwicklung eines Hilfesystems läßt sich Frage (2a) umformulieren zu:

(2b) Wie müssen die Hilfetexte sprachlich gestaltet sein?

Dieses Kapitel macht die Randbedingungen der Untersuchung deutlich, beschreibt die Beobachtung und präsentiert die aus den Beobachtungen gewonnenen Ergebnisse für Frage (1) und (2b).

4.4.1 Planung

Zwölf Gruppen mit jeweils zwei bis sechs Teilnehmerinnen und Teilnehmern wurden gebildet, die auf drei verschiedene Arten für einen Zeitraum von vier Wochen arbeiteten:

- Einige Gruppen benutzten ausschließlich Papier und Stift,
- anderen Gruppen arbeiteten zwei Wochen mit Papier und Stift und die restliche Zeit mit SANE,
- die restlichen Gruppen verwendeten die ganze Zeit nur SANE.

Als Erklärungsmaterial stand den Studierenden die Vorlesung, das Vorlesungsskript und eine kurze Einführung in den Aufgabennetzeditor SANE zur Verfügung. Im Rahmen der Einführung in SANE wurde das Laden und Speichern von Dateien, das Erzeugen von Objekten mit Hilfe der entsprechenden Knöpfe und das Öffnen von Objektmenüs erläutert. Die genaue Handhabung konnten sich die Studierenden erst bei der Arbeit mit SANE durch Exploration aneignen.

Drei Arbeitstreffen der Studierenden wurden beobachtet, die jeweils etwa neunzig Minuten dauerten. Nach der ersten und nach der letzten Sitzung wurden Interviews durchgeführt, in

[17] Die Auswertung der konkreten Veränderungen in der Sprache der Studierenden erfolgt in der Dissertation von Yvonne Dittrich.

denen versucht wurde, das erlangte Verständnis von aufgabenbezogener Anforderungsermittlung, Aufgabennetzen und SANE zu überprüfen. Die Beobachtungen wurden auf Tonband aufgenommen und in Notizen festgehalten. Da es sich bei dieser Untersuchung um eine qualitative Untersuchung handelt, wurden keine statistischen Auswertungen vorgenommen, sondern die Ergebnisse wurden einzeln analysiert.

4.4.2 Beobachtungen

Von den insgesamt beobachteten zwölf Gruppen werden in diesem Abschnitt drei Gruppen beschrieben. Diese drei Gruppen wurden ausgewählt, weil sie einen repräsentativen Querschnitt in bezug auf die bei der Arbeit mit SANE aufgetretenen Probleme darstellen. Zwei dieser Gruppen arbeiteten die Hälfte der Zeit mit Papier und Bleistift und danach mit SANE (Halbe-Halbe-Gruppen). Die andere Gruppe entwarf von Anfang an Aufgabennetze mit SANE (Mit-Gruppe). Die folgende Beschreibung der einzelnen Gruppen und ihrer Probleme soll lediglich exemplarisch einen Eindruck von der Arbeit der Studierenden mit SANE vermitteln. Eine Schilderung aller Details, die zu den in Abschnitt 4.4.3 dargestellten Ergebnissen geführt haben, würde den Rahmen dieser Arbeit sprengen (Das Protokoll *einer* Sitzung umfaßt bereits zwölf Seiten.).

Die erste Halbe-Halbe-Gruppe (HG1) erreichte bei ihrem ersten Arbeitstreffen ein gutes Verständnis von aufgabenbezogener Anforderungsermittlung und Aufgabennetzen. Als sie mit SANE zu arbeiten begannen, versuchten sie, ihr Verständnis von Aufgabennetzen auf den Aufgabennetzeditor zu übertragen.

Die Studierenden der HG1 konnten ihr Verständnis von den Aufgabennetzsymbolen direkt auf die in SANE dargestellten Knöpfe zum Erzeugen der Symbole abbilden. Mehr Schwierigkeiten hatte diese Gruppe mit der Handhabung des Aufgabennetzeditors. Ein Problem entstand, als die Studierenden der HG1 versuchten, Symbole zu kopieren, um so mehrere gleich große Exemplare des selben Symbols zu bekommen. Auf diese Weise wollten sie ihrem Aufgabennetz ein gleichmäßiges Aussehen verleihen. Bei SANE hat das Kopieren von Symbolen allerdings eine andere Semantik, als in vielen anderen Graphikprogrammen. Eine Kopie eines Symbols ist in SANE immer nur eine Kopie der graphischen Repräsentation eines aufgabennetzlogischen Objekts (s. Abschnitt 4.3.2). Als die Studierenden der HG1 begannen, alle kopierten Symbole mit unterschiedlichen Namen zu versehen, mußten sie feststellen, daß die kopierten Symbole untereinander verknüpft waren und die Namensänderungen immer für alle kopierten Symbole galten.

In ihrer ersten Sitzung hatten die Studierenden der HG1 sich den Begriff *Ebenen* gewählt, um verschiedene Hierarchiestufen in Aufgabennetzen bezeichnen zu können. Anhand ihrer Erfahrungen beim Kopieren von Symbolen versuchten sie, die Ebenen-Metapher auf SANE zu übertragen. Aufgrund der Ebenen-Metapher und dem Kopieren von Symbolen hatten sie die Vorstellung entwickelt, daß das Löschen eines Symbols auf einer höheren Ebene automatisch alle anderen Repräsentationen des selben Symbols auf den tieferen Ebenen entfernen würde. In diesem Zusammenhang fehlte den Studierenden der HG1 das Wissen um die Entwurfsmetapher von SANE: Werkzeug und Material. Wäre ihnen bewußt gewesen, daß sie mit SANE wie mit Stift und Radiergummi auf Papierblättern arbeiten, so hätten sie den eben geschilderten Effekt nicht erwartet.

Die zweite Halbe-Halbe-Gruppe (HG2) konnte in ihrem ersten Gruppentreffen kein gutes Verständnis von aufgabenbezogener Anforderungsermittlung und Aufgabennetzen erreichen. Die Gründe dafür lagen in mangelnder Erinnerung der Vorlesung und ineffektiver

Nutzung der schriftlichen Unterlagen. In der ersten Sitzung mit SANE hatten die Mitglieder dieser Gruppe sowohl mit den Systemobjekten und den Systemfunktionen Schwierigkeiten. Während ihrer gesamten Arbeit mit SANE erreichten sie niemals den Zustand, daß sie Aufgabennetze und den Aufgabennetzeditor SANE wirklich verstanden. Dies wurde besonders in den Interviews deutlich, in deren Verlauf sie darum gebeten wurden, Begriffe wie Verfeinerung oder Kante zu definieren.

Die dritte Gruppe (Mit-Gruppe, MG) diskutierte in ihrer ersten Sitzung, bei der sie sofort mit SANE zu arbeiten begannen, ausführlich über Kriterien, nach denen Aufgabennetze zu entwerfen seien. Sie erarbeiteten sich ein Verständnis von den Symbolen an der Benutzungsoberfläche mit Hilfe des schriftlichen Materials, der Informationen aus der Vorlesung und des Aufgabennetzeditors. Nachdem diese Gruppe einige Zeit mit SANE gearbeitet hatte, begannen sie, einzelnen Funktionen von SANE zu kritisieren. Sie stellten fest, daß der Vorgang des Verfeinerns verwirrend und kompliziert gestaltet ist, weil jedes Symbol für eine Verfeinerung auf ein neues Blatt kopiert bzw. neu erzeugt und verbunden werden muß. Darüber hinaus wünschten sich die Mitglieder der MG eine bessere Möglichkeit, verschiedene Hierarchiestufen eines Aufgabennetzes übersichtlich darzustellen. In SANE können nur DIN A4 Blätter in Ikonenform oder in Normalformat dargestellt und auf einer Arbeitsfläche angeordnet werden.

4.4.3 Ergebnisse

Bei der Untersuchung konnte zu beiden Fragen (1) und (2b) einige Erkenntnisse gewonnen werden. Zu der Frage (1): Welche Hilfe benötigen Benutzerinnen und Benutzer, wenn sie mit SANE arbeiten? wurden folgende Ergebnisse zusammengetragen: Es zeigte sich, daß die Studierenden im wesentlichen drei Fragen bei der Arbeit mit SANE stellten:

- Welche Möglichkeiten habe ich, hier weiterzuarbeiten?
- In welchem Arbeitszusammenhang mit SANE befinde ich mich gerade?
- Welche Bedeutung hat das Symbol auf dem Bildschirm?

Diese Fragen wurden von den Studierenden während der verschiedenen Sitzungen mehrfach diskutiert. Für die Entwicklung des Hilfesystems wurde daher geschlossen, daß Antworten auf diese Fragen ein notwendiger Bestandteil eines Hilfesystems sein müssen. Besonders die letzte Frage nach der Bedeutung eines Symbols, sowie auch die Beobachtungen der Studierenden aus der HG2, machen deutlich, daß die Probleme nicht nur bei der Handhabung von SANE auftraten. Vielmehr hatten die Studierenden mit der Entwicklung von Aufgabennetzen an sich, unabhängig von SANE, Schwierigkeiten. Die Schwierigkeiten waren also auf zwei verschiedenen Ebenen angesiedelt:

- Logik und Bedeutung der Aufgabennetze und
- Handhabung von SANE

An dieser Stelle wurde während der Untersuchung klar, daß die Studierenden eine Voraussetzung für eine transparente Benutzungssituation nicht erfüllten (s. Abschnitt 3.2). Die Studierenden besaßen keine Fachkompetenz in bezug auf ihre Aufgabe, Aufgabennetze zur aufgabenbezogenen Anforderungsermittlung zu erstellen. Dieser Effekt tritt nicht allein im Bereich der studentischen Arbeit mit Computersystemen auf, sondern Benutzerinnen besitzen häufig keine ausreichende Fachkompetenz, wenn sie mit einem System in Kontakt kommen. Sei es das sich eine Benutzerin selbständig mit einem neuen Anwendungsgebiet

vertraut machen will und anfängt mit einem Computersystem zu arbeiten oder das Berufsanfängerinnen mit Hilfe eines Computersystems in ein neues Fachgebiet eingeführt werden. Um Benutzerinnen das Erlernen eines neuen Computersystems zu erleichtern, sollte daher nicht davon ausgegangen werden, daß die Benutzerinnen für das gesamte Anwendungsgebiet fachkompetent sind. Das zu dem Computersystem entwickelte Hilfesystem sollte den Benutzerinnen daher nicht nur Erklärungen dazu geben können, wie das System zu Handhaben ist, sondern auch, welchen Sinn bestimmte Symbole oder Teile des Systems im Anwendungsgebiet haben.

Diese Forderung läßt sich in den im Teil I (Grundlagen) aufgestellten Anforderungen an Hilfesysteme (s. Anforderung 3-3) wiederfinden. Anforderung 3-3 legt fest, daß das Hilfematerial die an der Benutzungsschnittstelle verwendeten Symbole für Gegenstände und Handlungen erläutern und sie in eine *konsistente Beziehung zum Anwendungsgebiet* stellen soll. Werden den Benutzerinnen und Benutzern von SANE die Symbole für Gegenstände und ihre Beziehung zum Anwendungsgebiet erklärt, so sollten die Schwierigkeiten mit der Logik und Bedeutung von Aufgabennetzen im Anwendungsgebiet zumindest gemildert sein.

In der Schilderung von Gruppe HG1 wird deutlich, daß die Studierenden aus mangelnder Kenntnis der Metapher, nach der SANE entwickelt wurde, falsche Schlüsse aus bestimmte Zusammenhänge zogen. Wäre ihnen bekannt gewesen, daß SANE größtenteils eine direkte Abbildung ihrer Arbeit mit Papier und Bleistift anbietet, so wären einige Probleme nicht aufgetreten. Wie bereits in Abschnitt 3.3.2 diskutiert, müssen Metaphern für die Benutzerinnen explizit gemacht werden, damit die Benutzerinnen aus einer Metapher Nutzen ziehen können.

Neben Frage (1) wurde auch Frage (2b): *Wie müssen die Hilfetexte sprachlich gestaltet sein?* in Laufe der Untersuchung teilweise beantwortet. Bei den Diskussionen der Studierenden um SANE und Aufgabennetze konnte beobachtet werden, daß sich die Studierenden bemühten, die Begriffe aus den Aufgabennetzen bei ihrer Arbeit an Netzen zu verwenden. Auf diese Weise versuchten die Studierenden die Kluft zwischen Anwendungsgebiet und SANE zu überbrücken. Viele Begriffe aus dem Anwendungsgebiet, wie "Verfeinern", "funktionelle Rolle", "Informationsstruktur" etc., konnten sie in SANE wiederfinden. Gleichzeitig konnte festgestellt werden, daß die Studierenden im Laufe der Zeit ebenfalls neue Ausdrücke und Konzepte verwendeten, die durch SANE geprägt wurden. Beispielsweise wurden durch SANE neue Begriffe, wie "Abtrennen", "Verbinden", eingeführt, die die Studierenden nach kurzer Zeit in ihr Repertoire von Ausdrücken übernahmen. Darüber hinaus aber entwickelten die Studierenden aus ihrer Arbeit an den Aufgabennetzen heraus eine neue Sprache für bestimmte Zusammenhänge. Es handelte sich bei dieser Sprache nicht um gänzlich neue Wörter, aber um Begriffe, die in der Vorlesung oder den Unterlagen zu Aufgabennetzen bzw. zu SANE nicht verwendet wurden (vgl. die Studierenden der HG1 mit ihrem Ausdruck von "Ebenen der Verfeinerung").

Um Benutzerinnen und Benutzern bei der Verwendung einer Applikation beizustehen, muß das Hilfematerial daher die durch die Applikation geprägten Begriffe in der Sprache der Anwendungswelt erklären. Mögliche neue Wörter, die bei Beobachtungen von Benutzerinnen und Benutzern einer neuen Applikation gefunden werden, sollten ebenfalls in das zugehörige Hilfematerial aufgenommen werden.

Alle im ersten Teil erarbeiteten Anforderungen in bezug auf das Hilfematerial werden durch die hier geschilderte Untersuchung bestätigt. Benutzerinnen brauchen konsistente Erklärungen der Symbole für Gegenstände und Handhabung auf dem Bildschirm und ihrer Bedeutung im Anwendungsgebiet. Metaphern müssen den Benutzerin erklärt werden, sonst sind sie wirkungslos. Darüber hinaus konnten drei hauptsächliche Fragen der Benutzerin identifiziert und einige Aussagen zur Gestaltung der Hilfetexte gemacht werden, die in Teil III (Produkt) erneut aufgegriffen werden. Der im nächsten Kapitel vorgestellt Prototyp des Hilfesystems gibt einen Eindruck, wie sich die gesammelten Erkenntnisse aus den Grundlagen und der Untersuchung auf die Benutzungsschnittstelle des Hilfesystems ausgewirkt haben.

5 Prototyp des Hilfesystems

Aus dem Projektbericht im letzten Kapitel geht hervor, daß im Laufe des Projekts ein Prototyp erstellt und revidiert wurde. Der in diesem Kapitel vorgestellte Prototyp ist somit aus dem Redesign des ersten Prototyps entstanden und sollte *nicht* als "fertiges" Produkt betrachtet werden. Vielmehr ist der Prototyp im Verlauf eines evolutionären Softwareentwicklungsprozesses entstanden, dessen Grundidee[18] eine inkrementelle Entwicklung und Fortschreibung von Systemversionen ist. In einem evolutionären Entwicklungsprozeß wird das Produkt stets weiterentwickelt und an das sich wandelnde Verständnis der am Softwareentwicklungsprozeß Beteiligten angepaßt. Auch neu entstehende Anforderungen können in einem evolutionären Entwicklungsprozeß berücksichtigt werden. Im Verlauf des Entwicklungsprozesses werden Zeitpunkte festgelegt, an denen eine neue Version des Produkts an die Benutzerinnen übergeben wird. Auch das Hilfe-Framework befindet sich zur Zeit in einem Zustand, in dem es möglich ist, das Hilfesystem mit einer Applikation einzusetzen. Um den momentanen Stand des Entwicklungsprozesses festzuhalten, wird an dieser Stelle der Prototyp des Hilfesystems in seiner augenblicklichen Version beschrieben.

Zur Darstellung des Prototyps werden in diesem Kapitel zwei verschiedene Arten von Systemvisionen verwendet: *Überblicksvision* und *Handhabungsvision*. Eine Überblicksvision hat den Zweck, festzustellen, in welchem Verhältnis die bisherigen Tätigkeiten der Benutzerinnen zu neuen Tätigkeiten mit dem Computersystem stehen. Die hier vorgestellte Überblicksvision stellt das Hilfesystem im Vergleich zu Handbüchern vor. Eine Handhabungsvision beschreibt das Aussehen und die Verwendung des künftigen Anwendungssystems. Um das Hilfewerkzeug über die Überblicksvision hinaus detaillierter zu erläutern, werden verschiedene Tätigkeiten von Benutzerinnen bei der Arbeit mit dem Hilfesystem in Handhabungsvisionen beschrieben.

5.1 Überblicksvision

Das Ziel bei der Bereitstellung des Hilfesystems besteht darin, die Benutzerin bei der Arbeit mit einer Applikation zu unterstützen. Dabei werden die typischen Aufgaben von Handbüchern: Informationen *darzustellen*, das *Auffinden* von Information zu unterstützen und die *Orientierung* in der Menge der Information zu erleichtern, im Hilfewerkzeug abgebildet und durch technische Mittel effizienter gestaltet.

Das Hilfewerkzeug *stellt* über den Bildschirm erklärende Texte *dar*, wie sie sich auch in einem Handbuch finden lassen. Gleichzeitig bietet das Hilfewerkzeug im Gegensatz zu einem Handbuch die Möglichkeit, eine selbstbestimmte Anzahl und Auswahl von erklärenden Texten gleichzeitig angezeigt zu bekommen.

Wie ein Handbuch muß auch das Hilfewerkzeug das *Auffinden* von Information unterstützen. Die von Handbüchern bekannten Möglichkeiten, nach Schlagwörtern zu suchen sowie Lesezeichen und Verweise an bestimmten Stellen anzubringen, sind im Hilfewerkzeug abgebildet. Gerade das Prinzip der Verweise läßt sich im Hilfewerkzeug durch Hypertextstrukturen besser verwirklichen, als es in einem Handbuch möglich ist. Übliche Listendarstellungen nach Themengebieten geordnet sind im Hilfewerkzeug verwirklicht und können über gezielte Selektionen eingeschränkt werden. Außerdem bietet das Hilfewerkzeug als weiteren Mechanismus Kontextsensitivität. Hier wird der Zustand der ver-

[18] s. [Floyd92, Floyd94a, GryZül92, KiGrZü94]

wendeten Applikation ausgewertet, der identifizierte Zustand wird angezeigt und die entsprechenden Hilfeknotentexte zur Verfügung gestellt.

Die *Orientierung* in der Menge der Information wird im Hilfewerkzeug wie in einem Handbuch durch klare Namengebung für die einzelnen Themen erleichtert. Gleichzeitig können einzelne erklärende Texte mit einer Klassifikation versehen werden, so daß der Charakter der jeweiligen Texte vor dem Lesen erkannt werden kann. Das Hilfewerkzeug bietet darüber hinaus einerseits Historien an, die zurückverfolgt werden können, und erlaubt andererseits das Hinzufügen von eigenen Kommentaren zu den angezeigten Texten.

5.2 Handhabungsvisionen

Als typische Tätigkeiten von Benutzerinnen, die mit einem Hilfewerkzeug arbeiten, wurden ausgewählt:

- Hilfe zu aktueller Benutzungssituation anfordern,
- Im Hilfetext navigieren,
- Ein Lesezeichen anbringen,
- Ein Lesezeichen suchen,
- Mit einem Schlagwort suchen,
- Einen eigenen Kommentar eintragen,
- Einen eigenen Kommentar suchen und
- Kontextsensitivität ausschalten.

Diese Handhabungsvisionen sind während des Entwicklungsprozesses entstanden und kontinuierlich fortgeschrieben worden. Sie wurden mit verschiedenen Benutzerinnen und Benutzern diskutiert. Um die Anschaulichkeit der Visionen zu erhöhen, wurden Bilder vom Hilfewerkzeug während der Benutzung in die Visionen integriert.

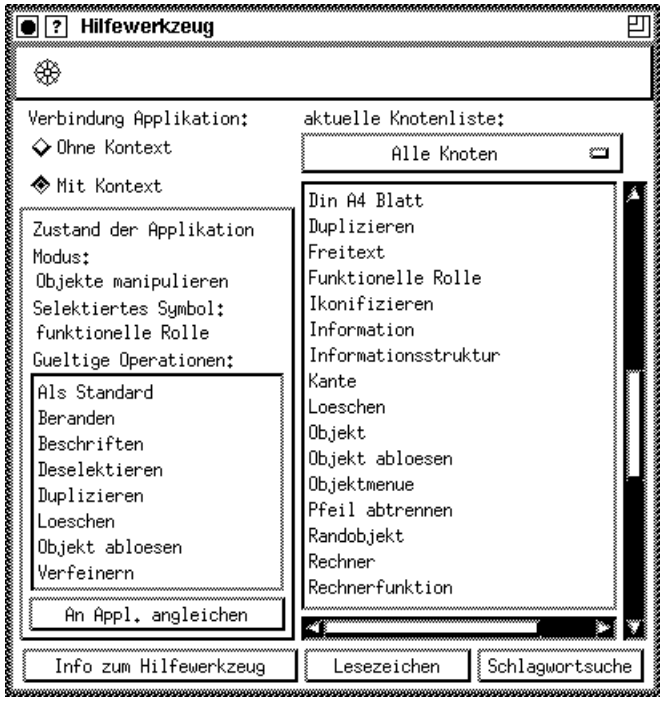


Abb. 5-1: Hilfewerkzeug mit aktiven Operationen

5.2.1 Hilfe zu aktueller Benutzungssituation anfordern

Die Benutzerin arbeitet mit SANE und möchte eine funktionelle Rolle auf dem Bildschirm löschen. Sie hat im zugehörigen Objektmenü den Befehl "Löschen" gefunden, weiß aber nicht, ob dieser Befehl nur die selektierte Funktionelle Rolle oder auch alle damit verknüpften anderen funktionellen Rollen auf anderen DIN A4 Blättern entfernt. Die Benutzerin ruft das Hilfewerkzeug auf, um sich über den Effekt des Befehls "Löschen" zu informieren. Das Hilfewerkzeug erscheint auf dem Bildschirm und bietet der Benutzerin neben einer Liste von

Abb. 5-2: Hilfeknotenbrowser mit Text

Hilfeknoten eine Liste der zur Zeit in SANE aktiven Befehle an.

Die Benutzerin wählt mit einem Doppelklick aus der Liste der aktiven Befehle "Löschen" aus, und der entsprechende Hilfeknoten wird in einem Hilfeknotenbrowser angezeigt.

Der Hilfeknotentext informiert die Benutzerin, daß der Befehl "Löschen" *eine* graphische Repräsentation eines Objekts löscht, nicht aber alle anderen Repräsentationen. Nachdem die Benutzerin den Hilfeknoten gelesen hat, schließt sie den Hilfeknotenbrowser durch einen Mausklick auf dem Knopf "Schließen". Anschließend beendet sie ihre Arbeit mit dem Hilfewerkzeug, indem sie aus der Menüleiste des Hilfewerkzeugs den Befehl "Beenden" auswählt.

5.2.2 Im Hilfetext navigieren

Die Benutzerin navigiert durch den Hilfetext, um verschiedene Informationen ausfindig zu machen. Sie findet in einem Hilfeknoten einen Verweis auf ein Thema, das sie interessiert. Der Verweis ist daran erkennbar, das der Name des Themas von zwei Sonderzeichen eingefaßt wird (s. "<<graphisches Objekt>>" in Abbildung 5-2). Um den Hilfeknoten zu diesem Thema angezeigt zu bekommen, wählt die Benutzerin den Verweis mit einem Doppelklick aus. Der Hilfeknoten zu dem ausgewählten Verweis erscheint in dem selben Fenster, so daß der vorherige Knoten nicht mehr sichtbar ist.

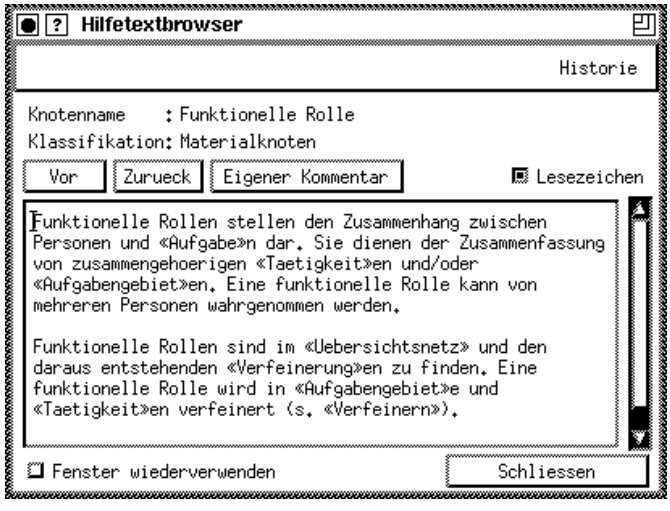


Abb. 5-3: Hilfeknotenbrowser mit aktivem Vor- und Zurück-Knopf

Da die Benutzerin beide Hilfeknoten gleichzeitig angezeigt haben möchte, drückt sie auf den Knopf "Zurück" (s. Abbildung 5-3) in besagtem Fenster. Nun wird wieder der erste Knoten angezeigt. Die Benutzerin wählt den Radioknopf "Fenster wiederverwenden" links unten im Fenster an, so daß er anschließend ausgeschaltet ist (s. Abbildung 5-3). Daraufhin führt sie erneut einen Doppelklick auf den Verweis aus und der Hilfeknoten zu dem gewählten Verweis wird in einem neuen Fenster angezeigt.

Nachdem die Benutzerin den

Hilfeknoten in dem neuen Fenster gelesen hat, verfolgt sie weitere Verweise per Doppelklick. Da sie die Option "Fenster wiederverwenden" (s. Abbildung 5-2 bzw. 5-3) in diesem Fenster nicht ausgeschaltet hat, werden alle im folgenden angewählten Hilfeknoten in diesem Fenster angezeigt. Nach einigen Navigationsschritten möchte die Benutzerin zu einem früheren Knoten zurückkehren. Sie wählt in der Menüleiste "Historie" (s. Abbildung 5-3) an und bekommt in dem dazugehörigen Menü eine Namensliste aller Hilfeknoten angezeigt, die sie mit diesem Fenster besucht hat. Sie selektiert einen Knoten aus der Liste, und der Hilfeknoten erscheint wieder in dem zu der Historie gehörigen Fenster. Da die Benutzerin in diesem Knoten die gewünschte Information nicht findet, sich aber erinnert, daß einer der nachfolgenden Knoten die Information enthalten muß, beschließt sie, die Liste der aufgesuchten Knoten noch einmal zu durchlaufen. Zu diesem Zweck springt sie mit Hilfe des "Vor"-Knopfes (s. Abbildung 5-3) von Hilfeknoten zu Hilfeknoten.

5.2.3 Ein Lesezeichen anbringen

Die Benutzerin hat bei Navigieren im Hypertext mit dem Hilfewerkzeug einen Text gefunden, den sie später erneut lesen möchte. Der Text ist in einem Fenster auf dem Bildschirm sichtbar. In der oberen rechten Ecke befindet sich ein Radioknopf, der die Bezeichnung "Lesezeichen" trägt (s. Abbildung 5-2 und 5-3). Die Benutzerin drückt diesen Knopf (s. Abbildung 5-3) und fährt in ihrer Suche durch den Hypertext fort.

Abb. 5-4: Lesezeichensuche

5.2.4 Ein Lesezeichen suchen

Um ein Lesezeichen wiederzufinden, wählt die Benutzerin im Hauptfenster des Hilfewerkzeugs den Knopf "Lesezeichen" (s. Abbildung 5-1). Das sich öffnende Fenster zeigt eine Liste mit dem Namen aller Knoten an, die die Benutzerin mit Lesezeichen versehen hat (s. Abbildung 5-4). Aus der angezeigten Liste wählt die Benutzerin mit einem Doppelklick den gewünschten Knoten aus. In einem Fenster erscheint der entsprechende Text des Hilfeknotens. Die Benutzerin schließt das Fenster zum Lesezeichen Suchen mit dem Knopf "Schließen".

5.2.5 Mit einem Schlagwort suchen

Die Benutzerin möchte zu einem bestimmten Schlagwort einen Hilfetext suchen. Sie wählt im Hauptfenster des Hilfewerkzeugs den Knopf "Schlagwortsuche" (s. Abbildung 5-1). Es erscheint ein Fenster mit zwei Listen (s. Abbildung 5-5). Die linke Liste enthält alle im Hilfetext verfügbaren Schlagwörter in alphabetischer Reihenfolge. Die rechte Liste ist leer.

Die Benutzerin gibt in dem Eingabefeld über dieser Liste einen Buchstaben ein und der sichtbare Bereich, der Schlagwörterliste wird bis zu dem ersten Schlagwort, das mit diesem Buchstaben anfängt, verschoben.

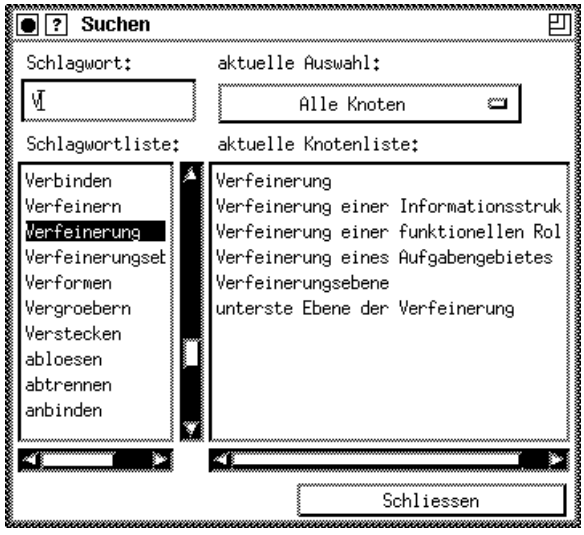


Abb. 5-5: Schlagwortsuche

Nachdem die Benutzerin ein passendes Schlagwort gefunden hat, wählt sie es mit einem Doppelklick aus.

In der zweiten Liste erscheint daraufhin die Liste aller Hilfeknoten, mit denen das ausgewählte Schlagwort verknüpft ist. Mit einem Doppelklick wählt die Benutzerin einen Hilfeknoten aus, und der dazugehörige Text erscheint in einem Fenster. Da die Benutzerin nicht weiter nach Schlagwörtern suchen will, schließt sie das Fenster für die Schlagwortsuche, indem sie den Knopf "Schließen" betätigt.

5.2.6 Einen eigenen Kommentar eintragen

Die Benutzerin hat beim Navigieren im Hypertext mit dem Hilfewerkzeug einen Text gefunden, den sie gerne um einen eigenen Kommentar erweitern möchte. Da der vorgefertigte Hilfetext nicht verändert werden kann, markiert sie mit einem Mausklick die Stelle im Text an der ihr Kommentar eingefügt werden soll und wählt den Knopf "Eigener Kommentar" (s. Abbildung 5-2 und 5-3). Ein Fenster mit leerem Kommentartext wird auf dem Bildschirm angezeigt. Die Benutzerin gibt ihrem Kommentar den Namen "von Carola" und trägt ihre Anmerkung in das Textfeld ein.

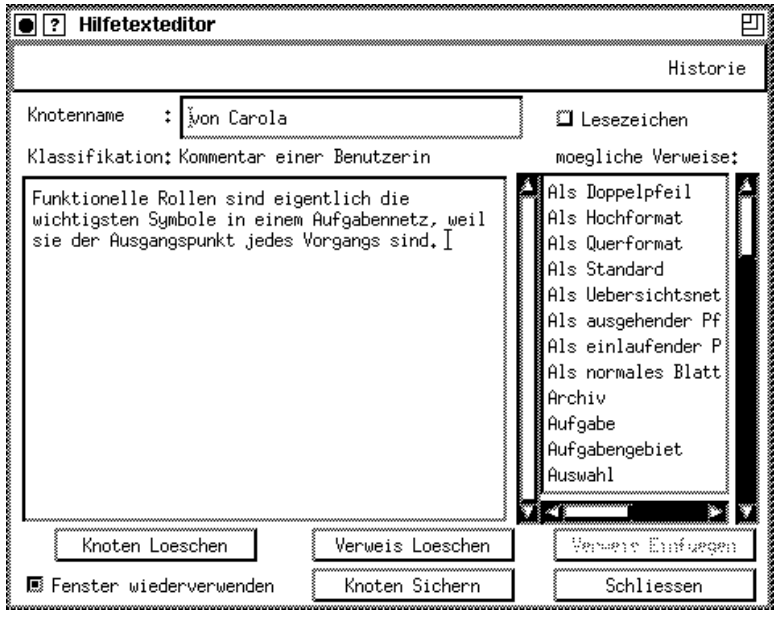


Abb. 5-6: Editor für Kommentare

Nachdem die Benutzerin ihren Kommentar fertiggestellt hat, betätigt sie den Knopf "Knoten Sichern" und schließt das Fenster für den Kommentar. In dem Hilfeknotentext, zu dem die Benutzerin ihren Kommentar verfaßt hat, erscheint an der vorher markierten Stelle im Text der Name des Kommentars "von Carola".

5.2.7 Einen eigenen Kommentar suchen

Die Benutzerin hat bei einer früheren Sitzung mit SANE und dem Hilfewerkzeug einen Kommentar zu einem Hilfeknoten eingegeben. Um diesen Kommentar wiederzufinden ruft sie das Hilfewerkzeug auf. In dem nun auf dem Bildschirm erscheinenden Fenster befindet

sich die Liste mit allen Knotennamen (s. Abbildung 5-1). Da die Liste der Knoten relativ lang ist und die Benutzerin sich nicht mehr an den genauen Namen des Kommentars erinnert, ändert sie mit Hilfe eines sich über der Liste befindenden Auswahlknopfes die Selektion in der Liste. Anstelle der vorhandenen Selektion "Alle Knoten" wählt sie die Selektion "Kommentare von Benutzerinnen" (s. Abbildung 5-7).

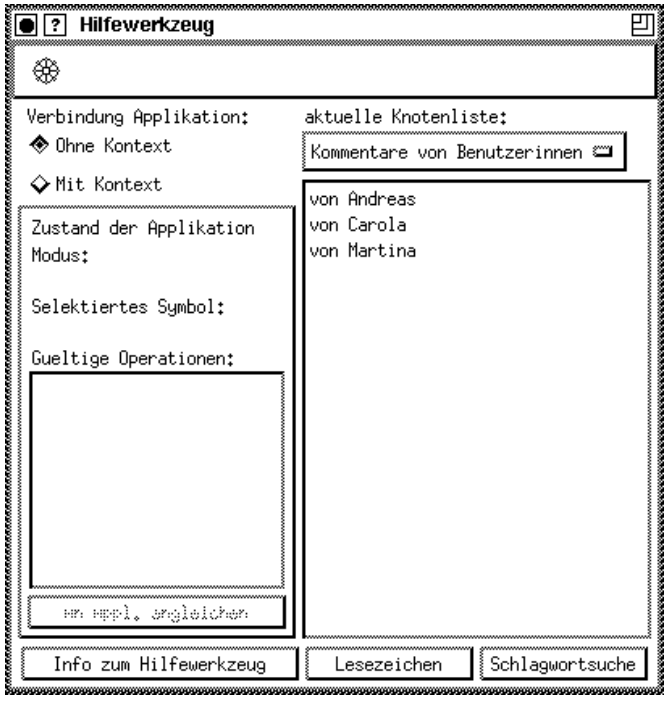


Abb. 5-7: Hilfewerkzeug mit Knotenselektion

Daraufhin enthält die Liste nur noch alle die Hilfeknoten, die Kommentare von Benutzerinnen enthalten. In dieser sehr viel kürzeren Liste findet die Benutzerin ihren Kommentar, wählt ihn mit einem Doppelklick aus und bekommt ihn in einem Fenster angezeigt. Sie liest ihren Kommentar und trägt zusätzliche wichtige Bemerkungen ein. Nachdem sie ihren Kommentar bearbeitet hat, fällt ihr in der Liste ein Kommentar einer anderen Benutzerin auf, den sie noch nicht gelesen hat. Die Benutzerin ruft diesen Knoten mit einem Doppelklick auf. Der Knoten erscheint in einem weiteren Fenster. Die Benutzerin möchte zu diesem Kommentar einen eigenen Kommentar eingeben. Da sie den Kommentar einer anderen Benutzerin nicht verändern kann, muß sie einen neuen Kommentarknoten erzeugen (s. Vision 5.2.6 "Einen eigenen Kommentar eintragen").

5.2.8 Kontextsensitivität ausschalten

Die Benutzerin arbeitet mit SANE und braucht Hilfe. Sie ruft das Hilfewerkzeug auf, das ihr abhängig vom Zustand des Aufgabennetzeditors kontextsensitive Hilfeknoten anbietet. Die Benutzerin möchte nicht nur einen Teilausschnitt der vorhandenen Hilfeknoten zur Verfügung gestellt bekommen und schaltet deshalb die Kontextsensitivität des Hilfewerkzeugs und damit den im Hilfewerkzeug integrierten Zustandsinterpreter aus. Zu diesem Zweck aktiviert die Benutzerin den Auswahlknopf "Ohne Kontext" (vgl. Abbildung 5-1 und 5-7). Daraufhin bietet das Hilfewerkzeug keine Informationen zum Kontext (Selektiertes Symbol, Modus, gültige Operationen und zum Zustand passende Hilfeknoten) mehr an, sondern zeigt alle im Hilfetext verfügbaren Hilfeknoten an. Die Benutzerin beginnt in dem Hilfetext zu navigieren und sich verschiedene Informationen anzeigen zu lassen.

Ausgehend von diesen Visionen des Hilfewerkzeugs wird in Teil III (Produkt) der Entwurf des Hilfewerkzeugs, der zugehörigen Materialien, eines Automaten und einiger weiterer Werkzeuge beschrieben.

5.3 Bewertung des Prototyps

Der Prototyp des Hilfesystems wurde auf der Grundlage von Anforderung 3-3 erstellt. Aufgrund technischer Randbedingungen konnten einige Vorgaben noch nicht erfüllt werden. Im folgenden werden die Vorgaben, die durch das Hilfesystem eingelöst werden, lediglich kurz erwähnt, da ihre Umsetzung in Teil III (Produkt) detailliert vorgestellt wird. Die nicht erfüllten Punkte werden insbesondere im Hinblick darauf diskutiert, welche Erweiterungen des Hilfe-Frameworks in einer nächsten Version verwirklicht werden sollten.

Die Anforderungen an das Hilfewerkzeug, passive, dynamische und individuelle Hilfe zu geben, werden vom Prototyp des Hilfesystems in seiner jetzigen Form eingelöst. Das Hilfesystem muß von der Benutzerin aufgerufen werden, d.h. es setzt sich nicht aktiv sondern *passiv* mit ihr in Verbindung. Auf den Zustand der Applikation reagiert das Hilfesystem *dynamisch* und bietet der Benutzerin eine auf den Zustand bezogene Auswahl von Hilfeknoten an. Die Benutzerin hat die Möglichkeit, sich *individuell* mit Hilfeinformation zu versorgen, weil das Hilfesystem ihr Verweise, Schlagwörter, Lesezeichen, das Eintragen von eigenen Kommentaren und eine Klassifikation für Hilfeknoten (s. Abb. 5-2, 5-3, 5-6 bzw. Abschnitt 6.2.3.) zur Verfügung stellt.

Das Hilfematerial, wie es im Prototyp verwirklicht wurde, entspricht größtenteils den in der Anforderung festgelegten Erwartungen: Die Symbole für Gegenstände und Handhabung an der Benutzungsschnittstelle sowie die eingesetzten Metaphern werden im Hilfematerial konsistent zum Anwendungsgebiet erklärt. Abstriche mußten bei der Umsetzung des Hypermedia gemacht werden. Bei der verwendeten hyperartigen Struktur handelt es sich zwar um einem Hypertext, es sind aber keine Möglichkeiten vorhanden, *visuelle* Erklärungen einzubinden. Diese Einschränkung kam zustande, weil das Hilfesystem wie SANE unter der Windows-Oberfläche Motif entwickelt werden mußte, um mit SANE zusammenarbeiten zu können. Am Arbeitsbereich Softwaretechnik stand lediglich eine Anwendung (Glossarbrowser) zur Verfügung, die einen Hypertexteditor enthielt (s. Abschnitt 4.2). Auch in den verschiedenen anderen Diskussionsforen zu Motif war kein entsprechender Hypermediaeditor verfügbar. Aus diesem Grund wurde das Hilfematerial als Hypertext verwirklicht. Für die Ideen und Konzepte, die im Verlauf der Entwicklung entstanden, hat sich diese Entscheidung nicht immer als glücklich erwiesen.

Die Einschränkung auf textuelle Darstellungen erwiesen sich insofern als Nachteil, als daß der Wunsch auftrat, an Verweisen in den Hilfeknotentexten Ikonen einzufügen. Ikonen sollten an den Verweisen angebracht werden, um zu illustrieren, welche Art von Information sich in den Hilfeknotentexten hinter den Verweisen verbirgt. Eine Ikone könnte beispielsweise deutlich machen, daß der zugehörige Verweis zu einem Hilfeknoten führt, der ein Symbol für eine mögliche Handlung oder aber für einen Gegenstand erklärt. Ohne einen Verweis zu verfolgen, könnten die Benutzerinnen beurteilen, ob der dahinterstehende Hilfeknoten für sie relevante Information enthält.

Bei der Erstellung des Hilfetextes für SANE hat es sich gezeigt, daß die vorhandene Hypertextstruktur teilweise an ihre Grenzen stößt. In der gewählten Hypertextstruktur wird ein Verweis immer durch den Namen des Knotens repräsentiert, auf den der Verweis zeigt. Handelt es sich bei dem Namen des Knoten beispielsweise um ein Verb: *Löschen*, so kommt dieses Wort in verschiedenen Sätzen des Hilfetextes in unterschiedlichen Formen und Zeiten vor: löscht, gelöscht etc. Derartige Verweise müssen um der Lesbarkeit und der Verständlichkeit Willen realisiert werden, indem der Verweis in Klammern hinter dem entscheidenden Wort eingefügt wird: "Objekte können gelöscht (s. *Löschen*) werden, indem

....". Um Texte verständlich schreiben zu können und umständliche Konstruktionen zu vermeiden, sollte die Hypertextstruktur dahingehend erweitert werden, daß Verweise von einem Wort aus zu einem Knoten führen können, bei denen Wort und Knotenname nicht übereinstimmen.

Die genannten Schwierigkeiten weisen daraufhin, daß speziell die Hypertextstruktur in der Form, wie sie im Hilfe-Framework realisiert ist, noch einige Veränderungen durchmachen wird. Da das Hilfesystem in einem begrenzten Zeitraum entwickelt werden mußte und es sich bei Softwareentwicklung um einen evolutionären Prozeß handelt, können die aufgetretenen Probleme als Hinweise für weiterführende Entwicklungsschritte gewertet werden.

III Produkt

Nachdem die Grundlagen eingeführt wurden und das zugrundeliegende Projekt beschrieben wurde, folgt in diesem Teil die Beschreibung des Produkts. Dabei werden die Konzepte und Entwurfsentscheidungen erläutert, die im Hilfe-Framework in seiner derzeitigen Form realisiert sind.

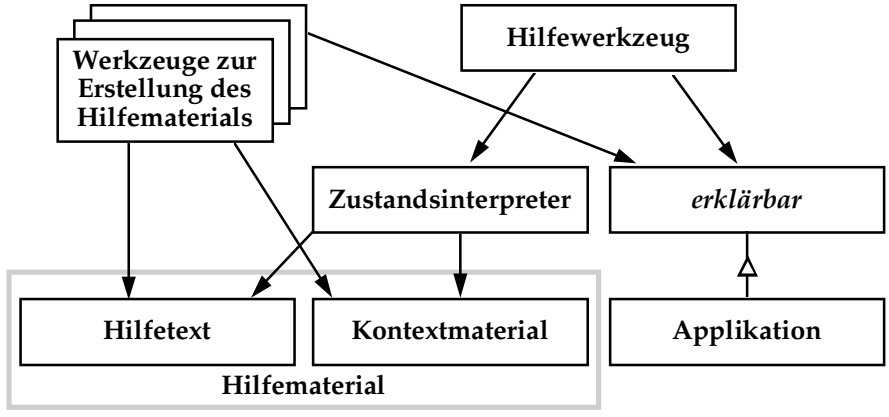


Abbildung III-1: Hilfe-Framework im Überblick

Insgesamt besteht das Hilfe-Framework aus mehreren Komponenten (s. Abb. III-1), die in unterschiedlichen Kombinationen zu verschiedenen Zielen eingesetzt werden. Dabei lassen sich zwei Einsatzkontexte identifizieren: die Verwendung des Hilfesystems mit einer Applikation und die Erstellung des Hilfematerials für eine Applikation. Verwendet eine Benutzerin das Hilfesystem bei ihrer Arbeit mit einer Applikation, so arbeiten die Komponenten Hilfewerkzeug, Zustandsinterpreter, Hilfematerial und die Applikation unter ihrer Schnittstelle *erklärbar* zusammen. Gestaltet die Entwicklerin ein Hilfematerial für ihre Applikation, so setzt sie die Werkzeuge zur Erstellung des Hilfematerials sowie das Hilfematerial und die Applikation ein.

Die einzelnen Komponenten, aus denen sich das Hilfe-Framework zusammensetzt, werden in den folgenden Kapiteln in drei Ausbaustufen beschrieben:

- Hilfesystem (Kapitel 6)
- Kontextsensitivität (Kapitel 7) und
- Werkzeuge zur Erstellung des Hilfematerials (Kapitel 8).

Die erste Stufe (Hilfesystem) beschreibt den Entwurf des Hilfetextes und des Hilfewerkzeugs. Dabei wird die Raummetapher vorgestellt und ein Mechanismus zur Wiederverwendung von Subwerkzeugen erläutert. Auf der zweiten Ebene (Kontextsensitivität) wird das Hilfesystem um Kontextsensitivität erweitert, so daß es dynamisch auf den Zustand einer Applikation reagieren kann. Hier wird der Zustandsbegriff diskutiert, die Verbindung zwischen Hilfesystem und Applikation erläutert sowie die Arbeit des Zustandsinterpreters beschrieben. Nachdem mit Hilfesystem und Kontextsensitivität der Einsatzkontext des Hilfesystems für die Benutzerin beschrieben wurde, befaßt sich die dritte Ausbaustufe mit den Komponenten des Hilfe-Frameworks, die die Entwicklerin zur Erstellung des Hilfematerials einsetzt. Die Vorstellung des Hilfe-Frameworks endet damit, daß ein dritter möglichen Einsatzkontext des Hilfe-Frameworks diskutiert wird: die Verwendung des Hilfe-Frameworks im Entwicklungsprozeß.

Um den Entwurf des Hilfe-Frameworks zu illustrieren, wird in den folgenden drei Kapiteln ein Beispiel verwendet. Damit die Teile des Textes, die zum Beispiel gehören, leichter zu identifizieren sind, werden sie in einer anderen Schrift dargestellt als die Beschreibung des Entwurfs. Die Grundlage dieses Beispiels bildet eine Anwendungssituation bei der Benutzung von SANE[19].

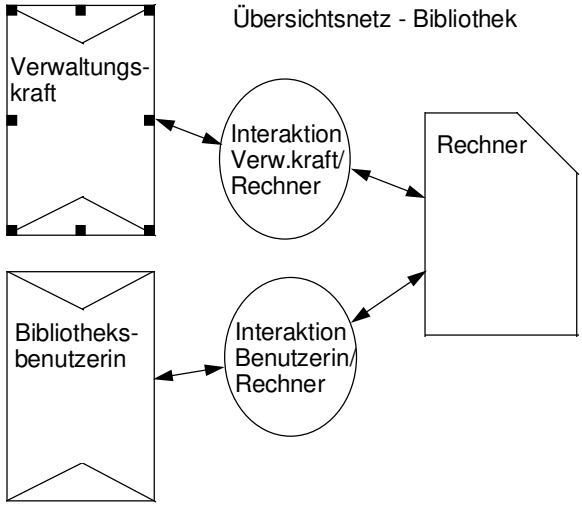


Abb. III-2: Aufgabennetz für das begleitende Beispiel

Eine Benutzerin hat bei ihrer Arbeit mit SANE ein Übersichtsnetz für eine Bibliothek erstellt (s. Abbildung III-2). In dem Übersichtsnetz hat sie bereits den Rechner, zwei funktionelle Rollen (Verwaltungskraft, Bibliotheksbenutzerin) und die Interaktion zwischen Verwaltungskraft, Bibliotheksbenutzerin und Rechner über zwei Informationsstrukturen eingetragen. Der nächste Schritt der Benutzerin besteht nun darin, die funktionelle Rolle "Verwaltungskraft" zu verfeinern. Zu diesem Zweck hat die Benutzerin das Symbol für die Verwaltungskraft selektiert. Ihre Selektion ist an den acht schwarzen Kästen am Rand des Symbols zu erkennen. Für das begleitende Beispiel sei die Annahme gemacht, daß die Benutzerin *nicht* weiß, auf welche Weise sie bei einer Verfeinerung vorgehen muß. Sie ruft aus diesem Grund das Hilfesystem auf, um Erklärungen zum Verfeinern einer funktionellen Rolle zu finden.

Im folgenden Kapitel (Hilfesystem) wird deutlich, wie die Benutzerin das Hilfesystem verwenden kann und welche textuelle Hilfe es ihr bietet. Kapitel 7 zeigt, wie die Kontextsensitivität des Hilfesystems der Benutzerin einen direkteren Zugang zu der benötigten Information ermöglicht. In Kapitel 8 wird schließlich beschrieben, auf welche Weise eine Entwicklerin das Hilfesystem mit dem nötigen Wissen ausstatten kann, damit das Hilfesystem kontextsensitiv Hilfetexte zu der Hilfesituation der Benutzerin bereitstellen kann.

Es wird im folgenden davon ausgegangen, daß objektorientierte Konzepte wie *Klassenbildung*, *Vererbung*, *Benutzung*, *Ober-* und *Unterklasse* bekannt sind. Bei der Darstellung des Entwurfs wird auf die in [Riehle95] eingeführten Interaktions-, Gestaltungs- und Entwurfsmuster zurückgegriffen, die sich im Rahmen der Werkzeug-Material Metapher bewährt haben. Die verwendete, graphische Notation ist im Anhang erläutert.

[19] Die in diesem Teil verwendeten Hilfematerialien beziehen sich auf die neuste Version von SANE. In Teil II wurde eine ältere Version von SANE vorgestellt, auf deren Basis die empirische Untersuchung durchgeführt wurde. Wesentlicher Unterschied zwischen diesen Versionen ist, daß der Vorgang, mit dessen Hilfe Aufgabennetz verfeinert werden, in der neueren Version von SANE vereinfacht worden ist. Diese Erweiterung von SANE ist maßgeblich auf die Beobachtungen während der empirischen Untersuchung zurückzuführen.

6 Das Hilfesystem

Dieses Kapitel stellt die erste Ausbaustufe des Hilfe-Frameworks *das Hilfesystem* vor. Das Hilfesystem besteht, wie in Anforderung 3-3 gefordert, aus Hilfewerkzeug und Hilfematerial. Vom Hilfematerial wird in diesem Kapitel nur der Entwurf des Hilfetextes beschrieben. Auf das Kontextmaterial wird im nächsten Kapitel unter dem Thema Kontextsensitivität eingegangen.

Zuerst wird ein Überblick über das Hilfesystem gegeben. Die Entwürfe für den Hilfetext und das Hilfewerkzeug werden gesondert erläutert, wobei im Abschnitt über das Hilfewerkzeug auf den Entwurf des Hilfetextes zurückgegriffen wird. Das begleitende Beispiel wird in Abschnitt 6.2.3. wieder aufgegriffen, wo verschiedene Hilfeknotentexte vorgestellt werden.

6.1 Zusammenspiel der Komponenten des Hilfesystems

Damit Hilfewerkzeug und Hilfematerial entsprechend der Werkzeug-Material Metapher entkoppelt werden können, muß ein Verwendungszusammenhang identifiziert werden.

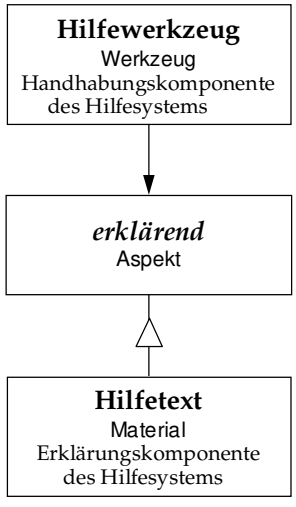


Abb. 6-1: Hilfesystem

Das Muster der *Werkzeug und Material Kopplung* (s. [Riehle95]) erlaubt es, den Verwendungszusammenhang zwischen Hilfewerkzeug und Hilfetext explizit zu machen. Dieses Muster findet seinen Ausdruck in einer Aspektklasse. Jeder Aspekt, der einen Verwendungszusammenhang spezifiziert, wird in einer Aspektklasse umgesetzt. Bei einer Aspektklasse handelt es sich um eine abstrakte Klasse, die von einer oder mehreren Materialklassen geerbt wird. Die abstrakten Operationen der Aspektklassen müssen von den erbenden Materialklassen jeweils spezifisch implementiert werden. In Materialklassen können durch Mehrfachvererbung verschiedene Aspekte zusammengeführt werden.

Der Verwendungszusammenhang zwischen Hilfewerkzeug und Hilfetext wird zu dem Aspekt `erklärend` abstrahiert. Der Aspekt `erklärend` formuliert die Operationen, die das Hilfewerkzeug sondierend und manipulierend auf dem Hilfetext ausführen kann. In Kapitel 8 werden weitere Werkzeuge eingeführt, die über andere Aspekte auf dem Hilfetext arbeiten.

6.2 Hilfetext

Gegenstand und das Ergebnis der Arbeitstätigkeit einer Benutzerin ist das Material. Es steht im Fokus ihres Interesses. Auch bei der Arbeit mit dem Hilfesystem richtet sich das Augenmerk der Benutzerin auf das *Material* Hilfetext und die darin enthaltenen Informationen. Aus diesem Grund gilt das erste Interesse der inhaltlichen Gestaltung der Texte und dem Entwurf des Hilfetextes. Im folgenden wird die inhaltliche Gestaltung und der Entwurf beschrieben sowie eine Klassifikation für Hilfeknoten eingeführt.

6.2.1 Gestaltung der Hilfetexte

Um den Inhalt von Hilfetexten verständlich darzustellen, müssen verschiedene Grundsätze beachtet werden. Teilweise lassen sie sich in der Literatur finden, zum Teil sind sie aber durch die Diskussion der Grundlagen (s. Teil I) und die empirische Untersuchung (s. Abschnitt 4.4) zu ihrem Stellenwert gelangt. Aus den Grundlagen und der empirischen Untersuchung lassen sich folgende Grundsätze für Hilfetexte extrahieren:

- Die Symbole an der Benutzungsschnittstelle müssen erklärt werden.
- Die Sprache der Anwendungswelt muß für die Erklärungen eingesetzt werden, so daß das Hilfesystem von den Begriffen der Anwendungswelt in die Begriffe des Computersystems übersetzt.
- Die Erklärungen müssen Orientierung zu den Möglichkeiten bieten, wie in einem bestimmten Zustand mit der Applikation weiter gearbeitet werden kann.
- Auf die Metaphern, die in der Applikation eingesetzt werden, muß hingewiesen werden.

Aus der Literatur (s. [FloGeb91]) lassen sich noch weitere Forderungen zusammengetragen, die an jede Art von erklärendem Text gestellt werden:

- Es sollten einfache, kurze Sätze verwendet werden.
- Einmal eingeführte Begriffe sollten konsequent eingesetzt und keine Synonyme benutzt werden.
- Gleichwertige Sachverhalte sollten auch als solche gekennzeichnet werden.
- Es sollte so wenig Nominal- und Substantivstil wie möglich verwendet werden.
- Die erklärenden Einheiten sollten für sich selbst verständlich und verwendbar sein (Modularität).
- Es sollten klare Strukturen angeboten werden, an denen sich die Benutzerin orientieren kann (Strukturiertheit).

Diese Grundsätze wurden bei der Erstellung der Hilfetexte für SANE berücksichtigt, sind aber von der eigenen Disziplinierung jeder Autorin abhängig. Das Hilfe-Framework kann Autorinnen von Hilfetexten lediglich insofern unterstützen, als daß es Konzepte für die Strukturierung und den Aufbau von Texten anbietet. Diese Konzepte sollen in den folgenden Abschnitten eingeführt werden.

6.2.2 Entwurf des Hilfetextes

Aus der in den ersten drei Kapiteln entwickelten Definition für Hilfesysteme (s. Anforderung 3-3) geht hervor, daß ein Hilfetext als *hyperartige* Struktur realisiert werden sollte, also die Funktionalität eines Hypertextes mit Hyperknoten besitzen muß. Gleichzeitig soll das Hilfesystem die Benutzerin beim Auffinden von Information unterstützen. Zu diesem Zweck sind Schlagwörter, Lesezeichen und Kommentare von Benutzerinnen in den Hilfetext integriert worden. Im folgenden wird zuerst eine Lösung für Hypertexte erarbeitet und anschließend werden die Ergebnisse auf Hilfetexte übertragen.

Hypertexte sind häufig verwendete Strukturen, um vernetzte Gebilde darzustellen. Die besondere Schwierigkeit von Hypertexten ist das Verwalten der Verweise zwischen einzelnen Hyperknoten. I.d.R. gehen derartige Verweise von einem oder mehreren Worten im Text eines Knoten aus und verweisen auf einen anderen Knoten. Es gibt zwei gegensätzliche Ansätze, Verweise in Hypertexten zu realisieren, und eine Reihe von Abstufungen zwischen diesen beiden Extremen.

Verfährt man nach dem einen Extrem, so stellt jedes Wort im Text eines Hyperknotens einen potentiellen Verweis dar. Wird ein Wort von der Benutzerin selektiert, um den zugehörigen Eintrag angezeigt zu bekommen, so wird der Hypertext nach Einträgen durchsucht, die einen entsprechenden Titel oder Inhalt haben. Meistens sind Regeln definiert, die festlegen, wann ein Eintrag zu einem bestimmten Wort gehört. Die Nachteile dieses Verfahrens sind der erhebliche Suchaufwand und, daß eine Benutzerin bei dieser Vorgehensweise darüber im unklaren gelassen wird, ob sie alle Querverweise eines Knotentextes gesehen hat oder nicht.

Das andere Extrem überläßt der Benutzerin die volle Kontrolle und Verantwortung darüber, wo Querverweise eingesetzt werden und wohin sie verweisen sollen. In diesem Fall sind die Verweise konkret in der Struktur des Hypertextes vorhanden und das Durchsuchen der Texte ist nicht erforderlich. Die Benutzerin muß bei dieser Variante Sorgfalt beim Erzeugen und Löschen von Verweisen walten lassen, damit alle Verweise zu existierenden Einträgen führen.

Um zwischen den beiden Extremen eine Lösung zu finden, wurde das Konzept des Hypertextes und seiner Knoten in dieser Arbeit mit Hilfe der *Raummetapher*[20] in die Klassen `Hypertext` und `Hyperknoten` umgesetzt. Diese Metapher wurde verwendet, weil es für hyperartigen Strukturen im Gegensatz zu vielen anderen Materialien keine direkte Entsprechung in der realen Welt gibt. Die Raummetapher bietet eine gute Vorstellung für die Realisierung von Hypertexten mit Knoten und Verweisen an. Sie legt nahe, daß der Hypertext als Haus und die Hyperknoten als Räume in diesem Haus verstanden werden. Die Verweise zwischen den einzelnen Hyperknoten werden in dieser Metapher als Türen zwischen den Räumen verstanden.

Betrachtet man die einzelnen Hyperknoten als Räume, die durch *Türen* verbunden sind, so wird klar, daß Türen einen wohldefinierten Ort an der Wand eines Raumes haben. Türen führen außerdem immer zum *selben* Ort, es sei denn das Haus wird umgebaut. Aus diesem Grund sollten bei der Realisierung des Hypertextes nur explizite Verweise zwischen Hyperknoten zugelassen werden. Jeder Verweis muß im Text als solcher erkennbar sein. Zu diesem Zweck werden in Texten von Hyperknoten üblicherweise spezielle Zeichen oder Ikonen verwendet. Um Verweise explizit, also an einen Ort gebunden, anlegen zu können, muß die Benutzerin Quelle und Ziel des Verweises angeben.

Türen zwischen Räumen sind normalerweise von beiden Seiten erkennbar; in Hypertexten ist dies nicht der Fall. Die Vorstellung, die hier weiterhilft, ist die Idee der Tapetentür. Eine Tapetentür hat nur auf einer Seite eine Klinke, so daß sie nur von einer Seite aus zu benutzen ist. Häufig sind Tapetentüren auch nur von einer Seite sichtbar und entsprechen dadurch der Idee von Verweisen, die zu einem anderen Knoten führen, dort aber nicht als eingehende Verweise erkennbar sind. Trotzdem muß innerhalb eines Hauses dafür gesorgt werden, daß Türen, die ins Nichts führen, verschlossen werden. Um diese Konsistenz zwi-

[20] s. [HenCar86, HenRod95, MaaObe92]

schen Verweisen in einem Hypertext sicherzustellen, muß in jedem Hyperknoten eine Liste aller Knoten aufbewahrt werden, die einen Verweis auf diesen Knoten besitzen. Wird dann ein Hyperknoten gelöscht, so benachrichtigt er alle anderen Knoten, die Verweise auf ihn enthalten, daß sie diese Verweise löschen müssen.

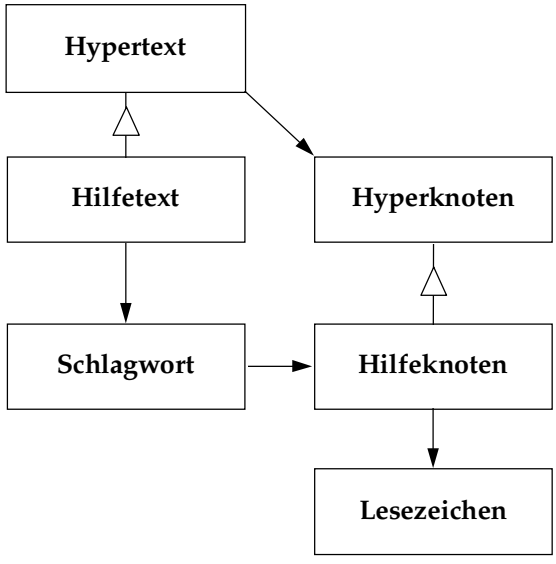


Abb. 6-2: Der Hilfetext ist eine Spezialisierung von Hypertext

Das Konzept der Klassen `Hypertext` und `Hyperknoten` kann von `Hilfetext` und `Hilfeknoten` geerbt werden, da es sich bei ihnen um spezialisierte Hyperstrukturen handelt (s. Abbildung 6-2). Für den Entwurf des Hilfetextes und seiner Hilfeknoten kann die Raummetapher weiterhin eingesetzt werden. In der Raummetapher lassen sich Lesezeichen als Markierungen in Räumen betrachten. Sie sind deshalb den Hilfeknoten in einer Benutzt-Beziehung zuzuordnen. Schlagwörter dagegen können mit mehreren Hilfeknoten verknüpft sein. Will man die Raummetapher sehr weit treiben, so stellen die Schlagwörter eine Art Tunnelsystem unterhalb der Räume dar, wobei jeder Tunnel einen oder mehrere Ausgänge in verschiedenen Räume haben kann. Die Schlagwörter werden daher im Hilfetext verwaltet und verweisen auf ihnen zugeordnete Hilfeknoten. Kommentare von Benutzerinnen schließlich lassen sich als eigene Hilfeknoten im Hilfetext betrachten, so daß keine weitere Klasse erforderlich ist. Die sich ergebenden Vererbungs- und Benutzt-Beziehungen sind in Abbildung 6-2 als Klassendiagramm dargestellt.

Hypertexte und ihre Strukturierung werden in verschiedenen Zusammenhängen diskutiert. Einige Ideen haben zu der hier gewählten Lösung beigetragen. In [Heß93] wird eine ähnliche Struktur für die Vernetzung einzelne Hyperknoten vorgeschlagen. Allerdings wird in [Heß93] keine Unterscheidung zwischen Hypertext und Hyperknoten getroffen, sondern Hypertexte werden hierarchisch wiederum selbst als Hyperknoten interpretiert. Diese Gleichsetzung von Hypertext und Hyperknoten wurde hier nicht übernommen, weil die Unterschiede in den Umgangsformen und den Strukturen als zu gravierend eingeschätzt wurden.

In der ET++ [Gamma92] existiert ein sogenanntes Cookbook, das als Hypertext implementiert ist. Das Cookbook wurde als Dokumentationssystem für die ET++ entwickelt. Es unterstützt verschiedene Knotentypen, wie Textdokumente, graphische Dokumente, Quellcode-Dateien und ausführbare Beispiele. Zwischen den Knoten können Verweise erzeugt werden, die ebenfalls verschiedene Typen haben können: "See also", "Sub-Topic", "Action" und "Example". Das im Cookbook verwirklichte Konzept von Knoten- und Verweistypen hat für diese Arbeit den Anstoß gegeben, die im nächsten Abschnitt vorgestellten Klassifikation für Hilfeknoten zu entwickeln.

6.2.3 Klassifikation von Hilfeknoten

In der Anforderung 1-2 für Hilfesysteme wird gefordert, daß ein Hilfetext sowohl die Handhabung erklären als auch das Verständnis der Entwicklerinnen vom Anwendungsgebiet berücksichtigen sollte. Bei der Diskussion um Transparenz und Sprache wurde diese Forderung noch dahingehend erweitert, daß die Erklärungen anhand der in Symbolen repräsentierten Gegenstände und Handlungen an der Benutzungsschnittstelle erfolgen sollen. Aus diesem Anspruch heraus wurde im Hilfe-Framework eine Klassifikation für Hilfeknoten eingeführt. Hilfeknoten werden nach ihrem jeweiligen Inhalt klassifiziert in:

- Übersichtsknoten,
- Materialknoten und
- Werkzeugknoten.

Wie sich im folgenden zeigen wird, kann sich die Benutzerin des begleitenden Beispiels mit Hilfe dieser Klassifikation besser orientieren. Sie hat ein Übersichtsnetz zusammengestellt und möchte eine Verfeinerung der funktionellen Rolle "Verwaltungskraft" vornehmen. Im Hilfesystem findet sie einen *Übersichtsknoten* zum Thema "Verfeinerung einer funktionellen Rolle" (Verweise werden durch kursive Schrift angedeutet):

> "Um eine *funktionelle Rolle* zu *verfeinern*, muß aus dem *Objektmenü* der *funktionellen Rolle* die Operation '*Verfeinern*' ausgewählt werden. Ist die *funktionelle Rolle* daraufhin auf ein neues *Din A4 Blatt kopiert* worden, müssen *Aufgabengebiete* oder *Tätigkeiten* in das Symbol der *funktionellen Rolle* eingefügt werden. Zu diesem Zweck werden *Aufgabengebiete* und *Tätigkeiten* mit Hilfe der Knöpfe in der *Symbolleiste erzeugt*. Der Austausch von *Informationsstrukturen*, *Informationen* und *Objekten* zwischen der *funktionellen Rolle* und dem *Rechner* muß entsprechend *verfeinert* werden. Dazu wird die *Informationsstruktur*, die auf dem *Übersichtsnetz* für die Interaktion zwischen der zu verfeinernden *funktionellen Rolle* und dem *Rechner* steht, mit entsprechenden *Informationsstrukturen*, *Informationen* oder *Objekten verfeinert*. Abschließend wird die verfeinerte Interaktion zwischen *funktioneller Rolle* und *Rechner* durch *Kanten* dargestellt."

In der Logik der Aufgabennetze ist eine Verfeinerung einer funktionellen Rolle ein größeres Handlungsziel, das mehrere einzelne Schritte zu seiner Durchführung beinhaltet. Das Thema eines Übersichtsknotens ist entweder ein größeres Handlungsziel, das mehrere einzelne Operationen erfordert (s. Beispiel), oder ein bestimmter Teil des Materials, für den alle Handhabungsmöglichkeiten zusammengefaßt werden. Im Hilfesystem findet sich daher ein Übersichtsknoten, der der Benutzerin erklärt, welche Handlungen für eine Verfeinerung ausgeführt werden müssen. Der Übersichtsknoten gibt einen kurzen Überblick über das Thema und besitzen eine Anzahl von Verweisen auf andere Knoten, die genauere Informationen enthalten. Die genaueren Informationen zu den verschiedenen Begriffen sind in den Werkzeug- und Materialknoten zu finden, die im Übersichtsknoten als Verweise eingetragen sind.

Für die Benutzerin des begleitenden Beispiels ist es möglicherweise wichtig, erneut zu klären, ob sie verstanden hat, was in Aufgabennetzen unter einer funktionellen Rolle verstanden wird. Sie verfolgt daher einen Verweis *funktionelle Rolle* im Übersichtsknoten und gelangt zu einem *Materialknoten* für das Symbol "funktionelle Rolle":

> "Funktionelle Rollen stellen den Zusammenhang zwischen Personen und *Aufgaben* dar. Sie dienen der Zusammenfassung von zusammengehörigen *Aufgabengebieten* und *Tätigkeiten*. Eine funktionelle Rolle kann von mehreren Personen wahrgenommen werden.
>
> Funktionelle Rollen sind im *Übersichtsnetz* und den daraus entstehenden Verfeinerungen zu finden. Über *Informationsstrukturen*, *Informationen* und *Objekte* interagiert eine funktionelle Rolle mit dem *Rechner* oder kommuniziert mit anderen funktionellen Rollen. Bei einer *Verfeinerung* wird eine funktionelle Rolle in *Aufgabengebiete* und *Tätigkeiten* aufgegliedert."

Materialknoten enthalten Erläuterungen, die die einzelnen Symbole für Gegenstände auf dem Bildschirm und ihre Beziehungen untereinander erklären. Man könnte Materialknoten auch als konzeptuelle Knoten im Sinne der konzeptuellen Konsistenz von W. Kellogg (s. Abschnitt 3.3.1) bezeichnen. In einem Hilfetext für SANE erläutert ein Materialknoten daher zum Beispiel, was eine funktionelle Rolle, ein Übersichtsnetz etc. ist.

Um den Aufbau der einzelnen Hilfeknoten für die Benutzerin leichter durchschaubar zu machen und auch beim Schreiben der Hilfeknotentexte einer Form folgen zu können, wurde für Werkzeug- und Materialknoten ein Grundgerüst entwickelt. An dem Knoten für das Symbol "funktionelle Rolle" läßt sich erkennen, daß ein Materialknoten zuerst eine Definition für das zugehörige Symbol liefert. Die Definition macht eine Aussage darüber, welchen Zweck das Symbol in der Anwendungswelt hat. Nach dieser Definition folgt in einem zweiten Absatz eine Beschreibung des Kontextes. Hier werden die Beziehungen des Symbols zu anderen Symbolen erklärt und sein Stellenwert im Gesamtzusammenhang verdeutlicht.

Nachdem die Benutzerin geklärt hat, was eine funktionelle Rolle ist, möchte sie mit der Verfeinerung beginnen. In dem Übersichtsknoten zum Thema "Verfeinerung einer funktionellen Rolle" hat sie gelesen, daß sie das Objektmenü öffnen muß und die Operation "Verfeinern" anwählen soll. Bevor sie diesen Schritt durchführt, möchte die Benutzerin sich vergewissern, welchen Effekt die Operation "Verfeinern" hat. Da ihr die Erklärung im Übersichtsknoten nicht ausreicht, wählt sie den Verweis *Verfeinern* aus dem Übersichtsknoten. Die Operation "Verfeinern" wird in einem *Werkzeugknoten* erklärt:

> "*Funktionelle Rollen*, *Aufgabengebiete*, *Informationsstrukturen* oder der *Rechnern* werden durch die Operation 'Verfeinern' auf ein neues *Din A4 Blatt* übertragen, um dort einen Ausschnitt des *übergeordneten Netzes* detaillierter zu beschreiben.
>
> Die Operation 'Verfeinern' kann nur auf *Funktionelle Rollen*, *Aufgabengebiete*, *Informationsstrukturen* und den *Rechner* angewendet werden. Der Aufruf von 'Verfeinern' bewirkt, daß eine *Kopie* des Symbols und alle seine *Randobjekte* auf ein neues *Din A4 Blatt* angelegt wird. Es empfiehlt sich daher, 'Verfeinern' bei einer *Informationsstruktur* aufzurufen, weil dadurch alle für eine *Verfeinerung* relevanten Symbole (*Randobjekte*): *funktionelle Rolle* oder *Aufgabengebiet*, *Informationsstruktur* und *Rechner* auf das neue *Din A4 Blatt* gelangen. Die *Kopien* der Symbole sind mit ihrem Original dahingehend verbunden, daß sie dasselbe *aufgabennetzlogische Objekt* darstellen. *Namensänderungen* werden automatisch in die *Kopien* übertragen."

Werkzeugknoten befassen sich mit den Handlungen, die an der Bildschirmoberfläche symbolisch dargestellt sind. Hier wird erklärt, wann eine Operation wie aufgeführt werden

kann und welchen Effekt sie hat. Diese Knoten beschreiben Zusammenhänge, die für prozedurale Konsistenz im Sinne von Kellogg (s. Abschnitt 3.3.1) notwendig sind.

Auch Werkzeugknoten sind, wie im Beispiel zu erkennen, nach einem Grundschema aufgebaut. Werkzeugknoten beginnen wie Materialknoten mit einer Definition, die den Zweck einer Handlung beschreibt. Anschließend wird die Handlung selbst genauer erläutert, indem die notwendigen Voraussetzungen aufgezeigt werden, die Durchführung geschildert wird und das oder die Ergebnisse der Handlung spezifiziert werden.

Die Klassifikation der Hilfeknoten und der Aufbau der einzelnen Knoten nach einem Grundgerüst ermöglicht es der Benutzerin, gezielt nach Informationen zu suchen. Durch die hier vorgeschlagene Knotenklassifikation ist die Benutzerin in der Lage, selbst zu entscheiden, welche Art von Information sie lesen will oder nicht. In der Literatur[21] werden eine Reihe von Kategorien für Anfragen der Benutzerin und entsprechende Antworten vorgeschlagen, die in eine Klassifikation umgesetzt werden könnten: prozedural (Wie gehe ich vor, um...?), deskriptiv oder elaborativ (Was ist ...?), funktional (Was macht ...?), erklärend oder interpretativ (Warum ist... passiert?, Warum passiert gerade ...?), deklarativ (Wie ist die Syntax von Kommando ...?), explorativ (Was passiert, wenn ...?), orientierend (Wo bin ich?), rechtfertigend (Warum soll Kommando ... verwendet werden?), klärend (Welchen Status hat das System?).

An dieser großen Zahl von Kategorien wird deutlich, wie schwer es fällt, Fragen von Benutzerinnen und entsprechende Antworten in ein Raster zu pressen. Am sinnvollsten erscheint es, die Anzahl der Kategorien auf eine überschaubare Anzahl zu beschränken. Die hier gewählten Klassifikation mit drei Kategorien deckt die oben aufgeführten Fragentypen weitgehend ab: Werkzeugknoten sind prozedural, funktional, interpretativ, orientierend, rechtfertigend, explorativ und klärend; Materialknoten sind deskriptiv, orientierend und rechtfertigend; Übersichtsknoten sind elaborativ, funktional und explorativ. Die benötigte Information zu der jeweiligen Situation muß die Benutzerin weiterhin selbst aus den angebotenen Texten entnehmen. Wie in Abschnitt 1.3.1 diskutiert, ist ein Hilfesystem nicht in der Lage, die Situation zu erkennen oder die exakte Frage der Benutzerin mit vollem Hintergrund zu verstehen. Daher kann das Hilfesystem nie direkte Antwort auf Fragen anbieten, sondern die Benutzerin muß das Hilfesystem verwenden, um *selbst* Hilfe *zu finden*.

Neben den drei bisher eingeführten Knotenarten Übersichts-, Werkzeug- und Materialknoten gibt es im Hilfe-Framework noch eine vierte Sorte Knoten. Bei diesem Knoten handelt es sich um die Kommentare der Benutzerinnen. Jede Benutzerin kann an Hilfeknotentexte Kommentare anhängen, um den Hilfetext individuell zu gestalten. In der vorliegenden Version des Hilfe-Frameworks sind die Kommentare der Benutzerinnen auch als Hilfeknoten im Hilfetext eingefügt, so daß sie ebenfalls Verweise auf andere Knoten enthalten können. Der Einsatz in der Praxis wird zeigen, ob diese Lösung für die Benutzerinnen sinnvoll ist oder ob Kommentare besser als angehängt Texte betrachtet werden sollten, die keine vollwertigen Knoten mit Hyperfunktionalität darstellen.

6.2.4 Entkoppelung von Hilfetexten für Werkzeuge und Materialien

In Arbeitsumgebungen, die nach der Werkzeug-Material Metapher gestaltet sind, stehen der Benutzerin i.d.R. eine größere Anzahl von Werkzeugen und Materialien zur Verfügung. Die Benutzerin hat die Möglichkeit, mit jedem Werkzeug, das zu einem Material paßt, auf

[21] [DuPaMe92, Gamma92, HarSmi88, Kearsley88, Nievergelt83]

dem jeweiligen Material zu arbeiten. Sie kann also verschiedene Kombinationen von Werkzeugen und Materialien zur Erledigung ihrer Arbeit verwenden.

Bisher wurde davon ausgegangen, daß ein Hilfetext alle Hilfeknoten enthält, die eine feststehende Applikation, also ein Werkzeug und ein Material, erklären. Um der Tatsache Rechnung zu tragen, daß vielfach mehrere Werkzeuge auf einem Material arbeiten, d.h. verschiedene Kombinationen von Werkzeugen und Materialien wahrscheinlich sind, ist der Wunsch aufgetreten, für Werkzeuge und Materialien je einen *eigenen* Hilfetext zu entwikkeln. Sind die Hilfetexte für Werkzeuge und Materialien getrennt verfügbar, so besteht die Möglichkeit, sie entsprechend der von einer Benutzerin gewählten Konstellation aus Werkzeug und Material zu kombinieren.

Werden Hilfetexte aufgespalten in Hilfetexte für Werkzeuge und in solche für Materialien, so stellen sich die Fragen: Welche Informationen sind in einem Hilfetext für ein Werkzeug und welche in einem Hilfetext für ein Material enthalten? Hier liegt die Überlegung nahe, daß die Klassifikation der Hilfeknoten in Werkzeug- und Materialknoten eine mögliche Trennung bereits eingeleitet hat.

Während der Entstehung der Hilfeknotentexte für den Aufgabennetzeditor SANE konnte die Erfahrung gesammelt werden, daß Materialknoten *keine* Verweise auf Werkzeugknoten enthalten. Werkzeugknoten und Übersichtsknoten hingegen beziehen in ihre Texte Verweise auf die Materialknoten ein, um zu erläutern, was Operationen bewirken und an welchen Symbolen sie ausführbar sind. Diese Beziehungen zwischen den verschiedenen Knotenarten lassen sich auch in den Beispielen für Übersichtsknoten, Werkzeugknoten und Materialknoten in Abschnitt 6.2.3 wiederfinden. Es bestände daher die Möglichkeit, Werkzeug- und Übersichtsknoten in einen Hilfetext für das Werkzeug zu übernehmen und den Hilfetext für das Material allein aus Materialknoten aufzubauen. Da aber Werkzeug- und Übersichtsknoten Verweise auf Materialknoten enthalten, benötigen Hilfetexte für Werkzeuge und Materialien untereinander eine wohldefinierte Schnittstelle. Werkzeuge und Materialien, die über einen Aspekt kombiniert werden, müssen analog zum Aspekt auch über eine explizite Schnittstelle aus Verweisen zwischen ihren Hilfetexten verfügen. Diese Schnittstelle besteht aus Verweisen, die, ausgehend von Werkzeug- und Übersichtsknoten, zu den Materialknoten führen.

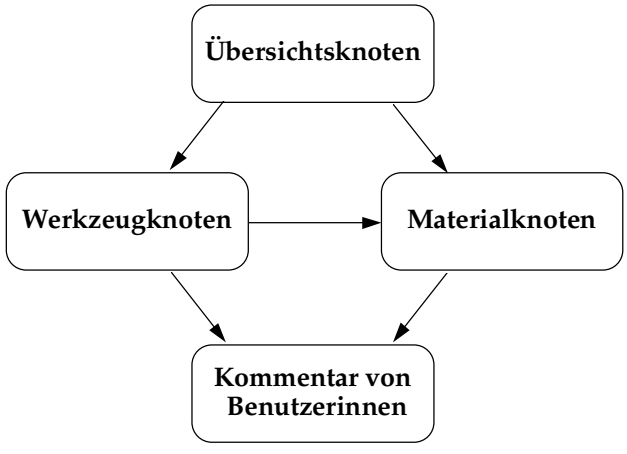


Abb. 6-3: Beziehung zwischen Hilfeknoten

Diese hier geschilderte Entkoppelung von Hilfetexten für Werkzeuge und Materialien ist bisher nur theoretisch diskutiert worden. Die softwaretechnische Umsetzung ist im Rahmen des hier beschriebenen Projektes nicht erfolgt, könnte aber ein weiterführendes Thema sein.

Nachdem Hilfetexte und Hilfeknoten als Material des Hilfewerkzeugs eingeführt worden sind, kann im folgenden der Entwurf des Hilfewerkzeugs vorgestellt werden.

6.3 Hilfewerkzeug

Der Entwurf des Hilfewerkzeugs wird entlang seiner Aufgaben erläutert. Dabei wird der Entwurf zuerst anhand der Entwurfsmuster aus der Werkzeug-Material Metapher beschreiben. Anschließend wird auf die Gestaltung der Oberfläche eingegangen und zum Abschluß eine Möglichkeit vorgestellt, Sub- und Kontextwerkzeuge zu koordinieren.

6.3.1 Entwurf des Hilfewerkzeugs

In Kapitel 2 und 3 wurden unter anderem zwei Hauptaufgabe eines Hilfesystems herausgearbeitet: Informationen *anzeigen* und das *Auffinden* von Informationen ermöglichen. Diese Aufgaben lassen sich für das Hilfewerkzeug anhand des Musters *Werkzeugkomposition* auf ein Kontextwerkzeug und mehrere Subwerkzeuge verteilen. Mit Hilfe des Musters der *Werkzeugkomposition* werden Werkzeuge aus Subwerkzeugen zusammengebaut, so daß sich ein Werkzeugbaum ergibt. Dabei werden bestimmte Teilaufgaben vom Kontextwerkzeug an seine Subwerkzeuge delegiert. Die Ergebnisse der Subwerkzeuge werden vom Kontextwerkzeug im Hinblick auf seine komplexere Aufgabe interpretiert. Die Werkzeugkomposition erlaubt es, bereits vorhandene einfache Subwerkzeuge wiederzuverwenden, indem sie in ein umgebendes Kontextwerkzeug integriert werden.

Für das *Hilfewerkzeug*, als Kontextwerkzeug, werden Teilaufgaben folgendermaßen an Subwerkzeuge weitergegeben: Der *Hilfeknotenbrowser* ist für das Anzeigen einzelner Hilfeknoten zuständig. Die Suche mit Hilfe von Schlagwörtern und aufgrund von Lesezeichen wird jeweils in ein eigenes Subwerkzeug - *Schlagwortsuche*, *Lesezeichensuche* - abgebildet. Eigene Kommentare können die Benutzerinnen mit Hilfe des *Kommentareditors* schreiben. Alle vier Subwerkzeuge werden von einem gemeinsamen Kontextwerkzeug - dem *Hilfewerkzeug* - koordiniert, was der Benutzerin den Zugang zu diesem Subwerkzeugen bietet.

Abbildung 6-4 stellt die Klassenstruktur des Hilfesystems dar, wobei die Subwerkzeuge Lesezeichensuche und Kommentareditor weggelassen wurde und die entsprechenden Schnittstellen der einzelnen (Sub-)Werkzeuge zum Material mit Hilfe von Aspekten deutlich gemacht werden.

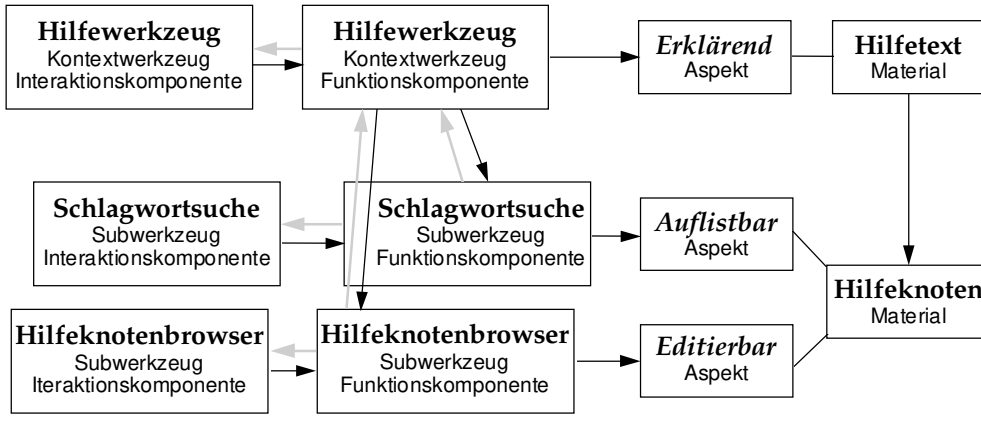


Abb. 6-4: Hilfewerkzeug mit Subwerkzeugen, Aspekten und Material
(Die grauen Pfeile stellen die Benachrichtigung durch den Ereignismechanismus dar.)

Alle (Sub-)Werkzeuge wurden gemäß des Musters der *Trennung von Interaktion und Funktion* in eine *Interaktions-* und eine *Funktionskomponente* zerlegt, um die funktionalen Teile jedes

Werkzeugs von den darstellenden Teilen zu trennen. Die Interaktionskomponente eines Werkzeugs präsentiert das Werkzeug an der Benutzungsoberfläche und nimmt die von der Benutzerin ausgelösten Ereignisse entgegen. Die Funktionskomponente ist der manipulierende und sondierende Bestandteil des Werkzeugs. In der Funktionskomponente findet die Bearbeitung des Materials in Abhängigkeit der von der Benutzerin ausgelösten Ereignisse statt. Durch die Trennung von Interaktion und Funktion kann die Flexibilität eines Werkzeugs in bezug auf Änderungen und auch auf einen Wechsel des Fenstersystems erhöht werden. Für das Kontextwerkzeug und alle seine Subwerkzeuge existiert daher jeweils eine Interaktions- und eine Funktionskomponente.

Damit das Hilfewerkzeug trotz seiner Aufteilung in Kontext- und Subwerkzeuge als zusammenhängende Einheit arbeiten kann, wird das Muster der *Werkzeugkopplung* eingesetzt. Die Verantwortung für das Erzeugen und Koordinieren von Subwerkzeugen wird der Funktionskomponente eines jeden Werkzeugs übertragen. Der baumartigen Struktur aus den Funktionskomponenten des Hilfewerkzeugs und seinen Subwerkzeugen wird ein gleichartiger Baum von Interaktionskomponenten zugeordnet. Jede Interaktionskomponente *beobachtet* (s.u.) ihre Funktionskomponente, und jede Funktionskomponente eines Kontextwerkzeugs *beobachtet* die Funktionskomponenten ihrer Subwerkzeuge. Da i.d.R. die umgebende Interaktionskomponente nicht direkt mit den in ihr eingebetteten Interaktionskomponenten der Subwerkzeuge zusammenarbeitet, wurde auf eine Benutzt-Beziehung zwischen der Interaktionskomponente des Hilfewerkzeugs und der Interaktionskomponenten der Subwerkzeuge Hilfeknotenbrowser und Schlagwortsuche verzichtet.

Das *Beobachten* zwischen Komponenten eines Werkzeugs läßt sich mit dem *Ereignismechanismus* verwirklichen. Eingesetzt wird dieser Mechanismus, um sicherzustellen, daß zwischen den verschiedenen Komponenten nur eindirektionale Benutzt-Beziehungen bestehen. Durch den Ereignismechanismus kennt die beobachtete Komponente ihre übergeordnete Instanz nur unter der anonymen Schnittstelle eines Beobachters. Die Realisierung des Ereignismechanismus' kann auf unterschiedliche Weise erfolgen. Zwei mögliche Formen der Umsetzung, um Ereignisse zu übermitteln, sind *Konstanten* oder *Objekten*. Im folgenden wird zuerst der Ereignismechanismus in der Variante mit Ereignisobjekten beschrieben.

Beim Ereignismechanismus mit Ereignisobjekten werden relevante Zustandsänderungen in einer Komponente auf je ein Ereignisobjekt abgebildet. Tritt eine Zustandsänderung ein, so wird ein Mechanismus beim Ereignisobjekt in Gang gesetzt, der dafür sorgt, daß alle an diesem Ereignis interessierten, beobachtenden Komponenten von dem Ereignis in Kenntnis gesetzt werden. Die beobachtenden Komponenten müssen sich bei dem Ereignis anmelden und die Operation bekanntgeben haben, die im Fall des Ereignisses bei ihnen aufgerufen werden soll. Mit Hilfe des Ereignismechanismus wird sowohl die Interaktion zwischen der Funktionskomponente des Kontextwerkzeugs und den Funktionskomponenten seiner Subwerkzeuge sichergestellt, als auch der Kontakt zwischen der Funktions- und der Interaktionskomponente eines jeden (Sub-)Werkzeugs realisiert (s. Abbildung 6-4, graue Pfeile).

Tritt beispielsweise in der Funktionskomponente des Hilfeknotenbrowsers dadurch eine Zustandsänderung auf, daß die Benutzerin einen Verweis auswählt, so aktiviert die Funktionskomponente des Hilfeknotenbrowsers das entsprechende Ereignis. Die Funktionskomponente des Hilfewerkzeugs ist für dieses Ereignis angemeldet; daher wird bei ihr die entsprechende Methode aufgerufen. Diese Methode bewirkt, daß das Hilfewerkzeug den Hilfeknoten, der an dem ausgewählten Verweis hängt, in einem weiteren Hilfeknotenbrowser anzeigt.

Der Ereignismechanismus in der Variante mit Konstanten läuft auf vergleichbare Weise ab. Allerdings kann die beobachtete Komponente sich in diesem Fall nicht der Dienste eines Ereignisobjekts bedienen, sondern muß ihre Beobachter selbst benachrichtigen. Zu diesem Zweck existiert eine abstrakte Oberklasse `Notifier`, die eine Methode `Notify` zur Verfügung stellt. Die beobachtende Komponente kennt nur diese Methode ihres Beobachters und ruft sie unter Angabe der Ereigniskonstanten auf. Um der beobachteten Komponente bekannt zu geben, an wen sie Nachrichten verschicken soll, meldet sich der Beobachter in dieser Variante bei der beobachteten Komponente selbst als Beobachter an. In der Bibliothek, die zur Entwicklung des Hilfe-Frameworks verwendet wurde, ist der Ereignismechanismus auf der Basis von Konstanten umgesetzt worden. Eine Weiterentwicklung zu Ereignisobjekten steht noch aus.

6.3.2 Orientierung im Hilfewerkzeug

Als dritte Hauptaufgabe des Hilfesystems neben *Informationen anzeigen* und das *Auffinden von Informationen ermöglichen* wurde in Anforderung 3-3 festgehalten, daß ein Hilfesystem der Benutzerin *Orientierungsmöglichkeiten* in der Fülle der Informationen geben sollte. Häufig wird in der Literatur auf das Problem hingewiesen, daß Benutzerinnen in Hyperstrukturen die Orientierung verlieren. Netzartige Strukturen bieten den Benutzerinnen kaum Anhaltspunkte, damit sie die Übersicht bewahren können. Um dieses Problem für das Hilfewerkzeug zu untersuchen und abzumildern, läßt sich anhand der Nievergeltschen Fragen [Nievergelt83]: Wo bin ich? Was kann ich hier tun? Wo kam ich her? in Anlehnung an [Nake92, NakSch93] vorgehen.

Die Frage nach dem Ort (*Wo bin ich?*) läßt sich auf jedem (Sub-)Werkzeug des Hilfesystems, aber auch in allen Hilfeknoten beantworten, da alle (Sub-)Werkzeuge und Hilfeknoten einen Namen haben und dieser jeweils angezeigt wird. In Abbildung 6-5 befindet sich die Benutzerin im Werkzeug "Hilfeknotenbrowser" in einem Werkzeugknoten, der die Operation "Löschen" erklärt.

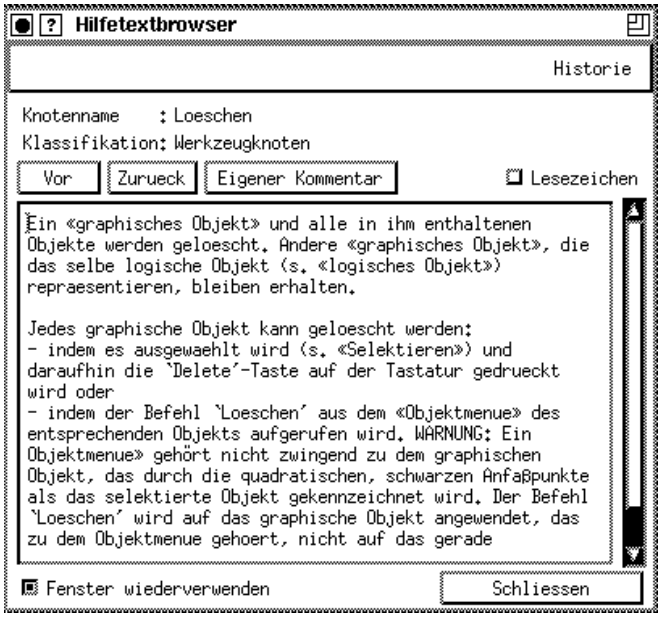


Abb. 6-5: Orientierung im Hilfeknotenbrowser

Die Handlungsmöglichkeiten (*Was kann ich hier tun?*) werden durch Knöpfe und Menüpunkte in den Fenstern aller (Sub-)Werkzeuge gleichermaßen dargestellt und sind in allen Fenstern auf die gleiche Weise umgesetzt (Konsistenz). Die Benutzerin ist nach kurzer Zeit mit dieser Darstellung vertraut und kennt dadurch die angebotenen Handlungsmöglichkeiten. In dem Hilfeknotenbrowser aus Abbildung 6-5 kann die Benutzerin beispielsweise folgende Handlungen ausführen: Das Fenster schließen, einen eigenen Kommentar anfügen, ein Lesezeichen anbringen, die Historie des Fensters zurückverfolgen und den Zustand des Fenster mit

"Fenster wiederverwenden" verändern (s. Abschnitt 6.3.3).

Auch die Frage *Wo kam ich her?* kann sich die Benutzerin in jedem (Sub-)Werkzeug beantworten. In jedem Werkzeug hat die Benutzerin die Möglichkeit, sich mit Hilfe der Historie einen Überblick über den zurückgelegten Weg zu verschaffen bzw. ihren Weg zurückzuverfolgen. Der Hilfeknotenbrowser in Abbildung 6-5 führt eine Liste der besuchten Hilfeknoten und erlaubt es der Benutzerin, entweder Schritt für Schritt den Weg zurückzuverfolgen (Vor- und Zurück-Knopf) oder mit Hilfe der Historie in der Menüleiste direkt zu bestimmten Knoten zurückzuspringen.

Wie dieser Abschnitt deutlich macht, wurde durch die Oberflächengestaltung der einzelnen Sub- und Kontextwerkzeuge versucht, der Benutzerin Orientierung bei ihrer Arbeit mit dem Hilfesystem zu geben. Dabei ist offensichtlich, daß auch für ein Hilfesystem hin und wieder Hilfe notwendig ist, um alle angebotenen Möglichkeiten ausnutzen zu können. Das Hilfewerkzeug stellt zu diesem Zweck auf Anfrage Hilfetexte zur Verfügung, die die Handhabung und das Verständnis der Autorin von Hilfesystemen erklären (s. Abbildung 5-1 und 5-7, Knopf "Info zum Hilfewerkzeug"). Eine weitere Strategie, um den Benutzerinnen die Orientierung zu erleichtern, ist das im nächsten Abschnitt vorgestellte Prinzip der Koordination wiederverwendbarer Subwerkzeuge.

6.4 Koordination wiederverwendbarer Subwerkzeuge

Der erste Prototyp des Hilfesystems bestand aus einem Werkzeug, das das Hilfewerkzeug und den Hilfeknotenbrowser in einem Fenster darstellte. Bei der Evaluation dieses Prototyps hat es sich gezeigt, daß Benutzerinnen und Benutzer mehrere Hilfeknoten gleichzeitig auf dem Bildschirm angezeigt bekommen möchten, damit sie Zusammenhänge zwischen Texten besser untersuchen können. Zu diesem Zweck wurden das Subwerkzeug Hilfeknotenbrowser vom Hilfewerkzeug abgetrennt und in einem eigenen Fenster implementiert. Auf diese Weise konnte erreicht werden, daß mehrere Hilfeknotenbrowser gleichzeitig auf dem Bildschirm sichtbar sind. Trotzdem sollte ein bei der Benutzung von Computersystemen ärgerliches Phänomen vermieden werden. Häufig füllt sich der Bildschirm im Laufe der Bearbeitung mehr und mehr mit Fenstern, so daß der Benutzerin schließlich die Übersicht verlorengeht.

Für die Entwicklung des Hilfewerkzeugs stellte sich daher die Frage, wie der Benutzerin gestattet werden könnte, selbst zu entscheiden, wieviele Subwerkzeuge gleichzeitig auf dem Bildschirm sichtbar sind. Zur Lösung dieser Frage wurde für das Hilfesystem die *Koordination wiederverwendbarer Subwerkzeuge* eingeführt. Als Vorbild diente die Entwicklungsumgebung Sniff+ (s. [Sniff+94]), deren verschiedene Werkzeuge nach dem gleichen Prinzip gehandhabt werden können.

Das Kontextwerkzeug des Hilfesystems wurde zu diesem Zweck mit einem Mechanismus ausgestattet, der es erlaubt, Subwerkzeuge koordiniert wiederzuverwenden. Die Interaktionskomponente jedes Subwerkzeugs wurde um einen Knopf "Fenster wiederverwendbar" erweitert (s. Abbildung 5-2, 5-3 und 6-5), dessen Einstellung festlegt, ob das Subwerkzeug wiederverwendet werden kann oder ob es seine Anzeige nicht verändern darf. Mit Hilfe dieses Knopfes kann die Benutzerin bei jedem Exemplar eines Subwerkzeugs die jeweilige *Einstellung* des Subwerkzeugs verändern. Die Benutzerin kann so erreichen, daß ein Subwerkzeug für eine Aktion, die sie ausgelöst hat, verwendet wird oder nicht.

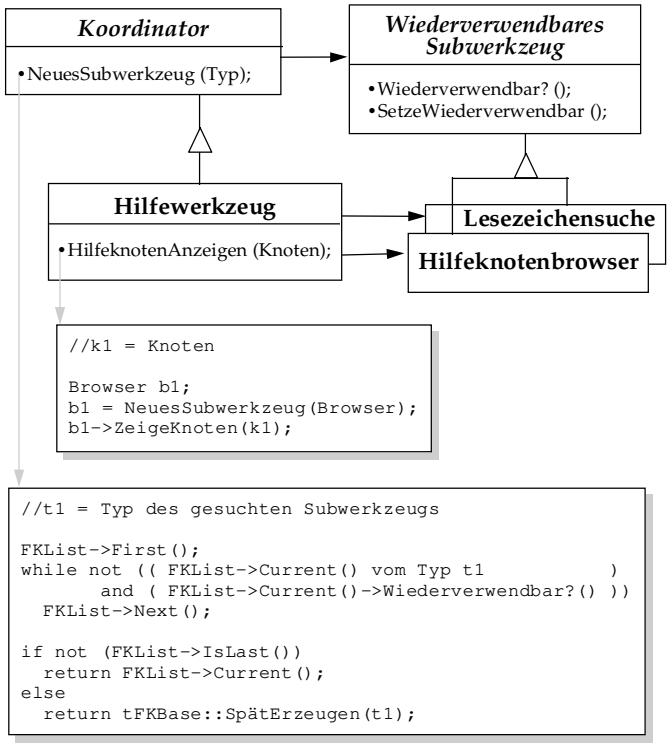


Abb. 6-6: Technische Umsetzung der Koordination

Um dieses Konzept zu verwirklichen, wurden zwei Oberklassen eingeführt: *Koordinator für wiederverwendbare Werkzeuge* und *wiederverwendbares Subwerkzeug* (s. Abbildung 6-6). Der Oberklasse *Koordinator* stellt die Funktionalität zur Verfügung, mit der Werkzeuge wiederverwendet oder neue Werkzeuge vom selben Typ[22] erzeugt werden. Die Klasse *wiederverwendbares Subwerkzeug* verbirgt den Status der Wiederverwenbarkeit. Sie stellt sondierende Operationen zur Verfügung, um den Status abzufragen, sowie manipulierende Operationen, um den Status zu verändern. In Abbildung 6-7 wird die Dynamik der Koordination wiederverwendbarer Werkzeuge dargestellt, wobei für die Werkzeuge nur Objekte der Funktionskomponenten eingetragen sind. Das `Hilfewerkzeug` verfügt in der durch Abbildung 6-7 dargestellten Konstellation über drei Objekte von Subwerkzeuge: `Lesezeichensuche`, `Browser 1` und `Browser 2`. Bei `Browser 1` und `Browser 2` handelt es sich um Hilfeknotenbrowser, wobei der eine, `Browser 2`, wiederverwendbar ist und der andere nicht.

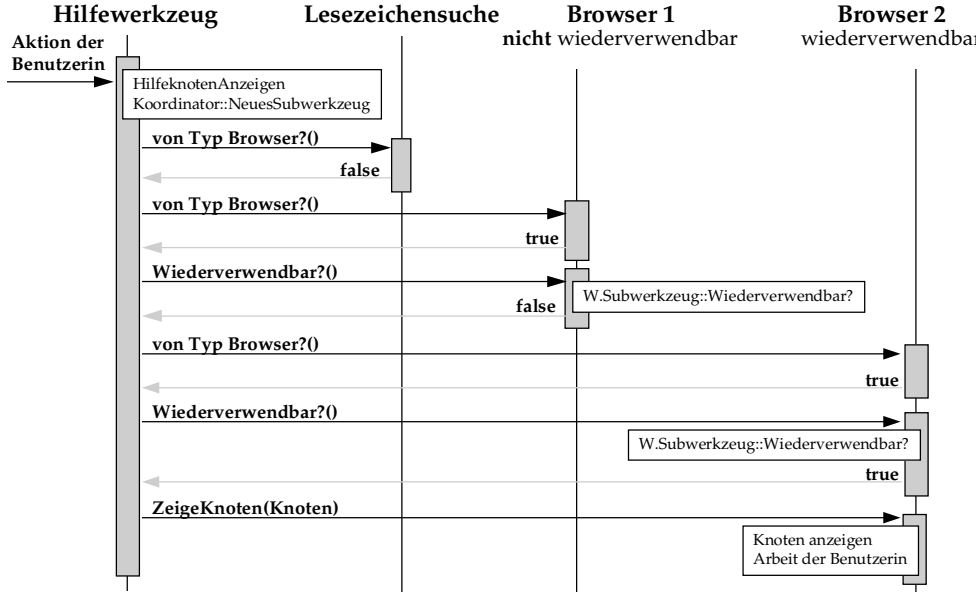


Abb. 6-7: Dynamik der Koordination

[22] Der Typ des gewünschten Subwerkzeugs wird dem Koordinator mit Hilfe einer Klausel bekanntgegeben. Das Konzept der Klauseln wurde für die späten Erzeugung entwickelt (s. [RieZül94b]) und konnte im Zusammenhang der Subwerkzeugkoordination wiederverwendet werden.

Abbildung 6-7 vermittelt einen Eindruck, wie eine Aktion einer Benutzerin dazu führt, daß ein Hilfeknoten in einem Hilfeknotenbrowser angezeigt wird. Das `Hilfewerkzeug` reagiert auf die Aktion der Benutzerin, indem es mit der geerbte Methode `NeuesSubwerkzeug` seiner Oberklasse `Koordinator` ein neues Subwerkzeug des entsprechenden Typs anfordert. In der Methode des Koordinators `NeuesSubwerkzeug` wird die Liste (`FKList`) aller Subwerkzeuge nach einem passenden Subwerkzeug durchsucht, das sich in einem wiederverwendbaren Status befindet. Das erste Subwerkzeug `Lesezeichensuche` ist nicht vom passenden Typ, da nach einem Hilfeknotenbrowser gesucht wird. Das zweite Subwerkzeug `Browser 1` hat zwar den geeigneten Typ, ist aber von der Benutzerin so eingestellt worden, daß es nicht wiederverwendet werden kann. Das dritte Subwerkzeug `Browser 2` ist von richtigen Typ und kann wiederverwendet werden. Aus der Methode `NeuesSubwerkzeug` wird `Browser 2` an das `Hilfewerkzeug` zurückgegeben und das `Hilfewerkzeug` reicht den Hilfeknoten an `Browser 2` zur Anzeigen weiter. Wird in der Methode `NeuesSubwerkzeug` *kein* entsprechendes Subwerkzeug gefunden, so erzeugt der `Koordinator` mit Hilfe von *später Erzeugung* [RieZül94b] ein neues Subwerkzeug des angegebenen Typs (s. Abbildung 6-6).

Mit Hilfe dieses Prinzips kann die Benutzerin im begleitenden Beispiel selbst koordinieren, wieviele Hilfeknotenbrowser sie gleichzeitig auf dem Bildschirm angezeigt haben will. Die Benutzerin ist bei ihrer Suche nach Informationen zur Verfeinerung einer funktionellen Rolle in Abschnitt 6.2.3 auf mehrere Hilfeknoten gestoßen, die sie interessieren: *Verfeinerung einer funktionellen Rolle*, *Funktionelle Rolle* und *Verfeinern*. Den Text des Hilfeknoten *Verfeinerung einer funktionellen Rolle* möchte die Benutzerin möglicherweise permanent in einem Fenster angezeigt bekommen. Zu diesem Zweck drückt sie den Knopf "Fenster wiederverwenden" (s. Abbildung 5-2, 5-3 und 6-5) in dem Fenster, daß den Übersichtsknoten *Verfeinerung einer funktionellen Rolle* enthält. Um sich über die Bedeutung des Symbols Funktionelle Rolle zu informieren, verfolgt die Benutzerin einen Verweis im Übersichtsknoten zu einem entsprechenden Materialknoten. Der Materialknoten *Funktionelle Rolle* wird in einem neuen Hilfeknotenbrowser angezeigt. Nachdem die Benutzerin den Text des Hilfeknotens *Funktionelle Rolle* gelesen hat, verfolgt sie im Übersichtsknoten *Verfeinerung einer funktionelle Rolle* einen Verweis zu dem Werkzeugknoten *Verfeinern*. Der Werkzeugknoten *Verfeinern* wird daraufhin in dem Hilfeknotenbrowser angezeigt, der vorher den Materialknoten *Funktionelle Rolle* enthalten hatte.

Wird Werkzeugkomposition (Kontextwerkzeug-Subwerkzeug-Beziehung) auf diese Weise verwirklicht, so führt das Kontextwerkzeug keine eigene Liste seiner Subwerkzeuge. Das Kontextwerkzeug kennt nur den Typ des zu erzeugenden Subwerkzeugs und erhält von der Methode seiner Oberklasse `Koordinator` entweder ein neues oder ein wiederverwendetes Subwerkzeug zurück. Bekommt das Kontextwerkzeug durch den *Ereignismechanismus* Nachrichten von seinen Subwerkzeugen, so behandelt es alle Nachrichten einheitlich, die vom selben Typ eines Subwerkzeugs kommen. Um dies zu bewerkstelligen, bedient sich das Kontextwerkzeug der im Metaobjektprotokoll abgelegten Typinformation und fragt den jeweiligen Benachrichtiger nach seinem Typ. Dementsprechend reagiert das Kontextwerkzeug auf die eingegangene Nachricht. Diese Art der Koordination von wiederverwendbaren Werkzeugen wurde im Rahmen des Hilfe-Frameworks noch ein zweites Mal (s. Abschnitt 8.4) verwendet und auch in einer anderen Diplomarbeit (s. [Wulf95]) eingesetzt.

Nachdem in diesem Kapitel der Entwurf der Hilfetexte und des Hilfewerkzeugs vorgestellt wurde, befaßt sich das nächste Kapitel mit Kontextsensitivität und dynamischer Hilfe.

7 Kontextsensitivität

Ruft die Benutzerin aus dem begleitenden Beispiel das Hilfesystem auf, so möchte sie eigentlich direkte Hilfe zu ihrem Problem bekommen. Bei der Vorstellung des Hilfewerkzeugs und der Hilfetexte wurde bisher keine Aussage darüber gemacht, wie die Benutzerin die benötigten Hilfeknoten für das Verfeinern einer funktionellen Rolle im Hilfesystem finden kann. Um diesen Vorgang zu vereinfachen, muß das Hilfesystem zu einer Applikation kontextsensitive Hilfe, d.h. dynamisch Hilfe, anbieten können. Wie bereits im ersten Kapitel argumentiert, kann die Berücksichtigung der Benutzungssituation nur dadurch erfolgen, daß das Hilfesystem den Zustand der Applikation in die Auswahl der angebotenen Hilfeknoten einbezieht.

In diesem Kapitel wird beschrieben, wie das Hilfesystem mit einer Applikation verbunden ist und auf welche Weise es über diese Schnittstelle dynamisch auf den Zustand reagieren kann. Dabei wird zuerst der Gesamtzusammenhang dargestellt, und anschließend werden die einzelnen Aspekte des Entwurfs diskutiert. Das begleitende Beispiel wird in Abschnitt 7.3 wieder aufgegriffen, um die Möglichkeiten der Kontextsensitivität mit Hilfe des Zustands und des Zustandsinterpreters zu beschreiben.

7.1 Hilfesystem mit Kontextsensitivität

Stehen einem Menschen bei der Arbeit mit realen Werkzeugen und Materialien nur textuelle oder visuelle Erklärungen in Form von Büchern etc. zur Verfügung, so muß die Verbindung zwischen der Wissenslücke und der entsprechenden Erklärung vom Menschen selbst hergestellt werden. Ist die Hilfe im Computer abgelegt und sind die Werkzeuge und Materialien im Computer abgebildet, so kann ein Hilfesystem die Benutzerin bei der Verbindung zwischen Wissenslücke und Erklärungen *unterstützen*. Diese Unterstützung kann ein Hilfewerkzeug liefern, weil es imstande ist, den Zustand einer Applikation zu berücksichtigen und dementsprechend Erklärungen anzubieten.

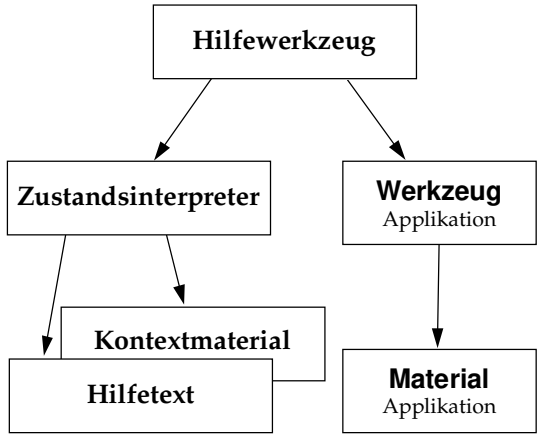


Abb. 7-1: Hilfesystem mit Kontextsensitivität

Im folgenden wird davon ausgegangen, daß die Applikation selbst nach der Werkzeug-Material Metapher entwickelt worden ist. Um den Zustand der Applikation zu erfragen, arbeitet das Hilfewerkzeug auf dem Werkzeug einer Applikation. Befragt das Hilfewerkzeug das Werkzeug einer Applikation nach seinem Zustand bezogen auf die Bearbeitung des Materials, so verwendet es ein Werkzeug als sein Material, das sondiert wird. Hier wird erneut deutlich, daß es keine vorbestimmte Eigenschaft eines Gegenstands ist, ein Werkzeug oder ein Material zu sein. Vielmehr hängt es vom jeweiligen Verwendungszusammenhang ab, ob ein Gegenstand als ein Werkzeug oder als ein Material verstanden wird. Man kann das Hilfewerkzeug auf dieser Betrachtungsebene auch als Werkzeug 2. Ordnung bezeichnen.

Da Werkzeuge im Rahmen der Werkzeug-Material Metapher in ihrer softwaretechnischen Umsetzung aus einem Kontextwerkzeug und seinen Subwerkzeugen bestehen, die jeweils in Interaktions- und Funktionskomponente aufgespalten werden, reicht es für die Entwicklung des Hilfewerkzeugs nicht aus festzustellen, daß das Hilfewerkzeug auf dem Werkzeug einer Applikation arbeitet. Vielmehr muß *die Komponente* eines Werkzeugs identifiziert werden, die in der Lage sind, die benötigten Informationen zu liefern. Das *Kontextwerkzeug* einer Applikation ist die einzige Komponente eines Werkzeugs, die Auskunft über den Zustand des *gesamten* Werkzeugs geben kann. Das resultiert aus der Tatsache, daß das Kontextwerkzeug Aufgaben an seine Subwerkzeuge delegiert, von Zustandsänderungen in seinen Subwerkzeugen aber informiert wird, damit es angemessen reagieren kann. Als Zugang zur Applikation wird für das Hilfesystem darüber hinaus die *Funktionskomponente* des Kontextwerkzeugs gewählt, weil nur die Funktionskomponente eines Werkzeugs Auskunft über den aktuellen Zustand, bezogen auf die Bearbeitung des Materials, geben kann. Die Interaktionskomponente ist nicht in der Lage, diesen Dienst zu erbringen, weil sie lediglich für die Darstellung zuständig ist und kein Wissen über den Zustand der Bearbeitung hat.

Der Algorithmus, der den Zustand der Applikation mit entsprechenden Hilfeknoten verknüpft, kann komplex sein und sich von Applikation zu Applikation ändern. Um ihn ersetzbar und flexibel zu halten, bietet es sich an, ihn nicht im Hilfewerkzeug anzusiedeln, sondern ihn in einem Automaten (s. Kapitel 2) zu kapseln: dem **Zustandsinterpreter**. Der Zustandsinterpreter liefert dem Hilfewerkzeug auf Anforderung die Knoten, die zu einem bestimmten Zustand der Applikation passen. Bei dem hier vorgestellten kontextsensitiven Hilfesystem wird die Verknüpfung zwischen Hilfeknoten und Zuständen der Applikation in einem weiteren Material dem **Kontextmaterial** abgelegt.

7.2 Schnittstelle zwischen Hilfewerkzeug und Applikation

Um dynamische Hilfe zu einer Applikation geben zu können, muß das Hilfewerkzeug die jeweilige Applikation kennen und nach entsprechenden Informationen zu ihrem Zustand befragen können. Dabei soll das Hilfesystem einerseits den Zustand der Applikation erfragen und auf die Applikation bezogen interpretieren, andererseits soll sich das Hilfesystem nicht an eine bestimmte Applikation binden. Zu diesem Zweck muß eine Schnittstelle geschaffen werden, die von verschiedenen Applikationen erfüllt werden kann. Da das Hilfewerkzeug auf dem Werkzeug der Applikation wie auf einem Material arbeitet, ist es naheliegend, zwischen Hilfewerkzeug und Applikation einen Aspekt als Schnittstelle einzufügen. Ein Aspekt ist im Rahmen der Werkzeug-Material Metapher das gängige Mittel, um den Verwendungszusammenhang zwischen Werkzeugen und ihren Materialien zu konkretisieren. Zu jeder Applikation, die diesen Aspekt erbt und implementiert, kann die Benutzerin das Hilfewerkzeug aufrufen, um die passende Hilfestellung zu finden. Die Methoden in der Klassenschnittstelle des Aspekts muß dem Hilfewerkzeug alle Informationen liefern, um kontextsensitiv auf eine bestimmte Applikation reagieren zu können. Dazu muß die Applikation einerseits bekannt geben, in welchem Zustand sie sich befindet, und andererseits muß geklärt werden, woher das Hilfewerkzeug die Hilfematerialien bekommt, die zur jeweiligen Applikation passen.

Die Frage nach dem Ort der Hilfematerialien läßt sich auf verschiedene Weise beantworten. Eine Möglichkeit wäre, daß das Hilfesystem über einen Behälter aller Hilfematerialien verfügt und jeweils das Hilfematerial heraussucht, der zu der "hilfsbedürftigen" Applikation paßt. In diesem Fall muß sichergestellt werden, daß zu jeder Applikation in einer bestimm-

ten Umgebung ein Hilfematerial im Behälter des Hilfewerkzeugs vorhanden ist. Ein anderer Weg, das Hilfewerkzeug mit Hilfematerialien zu versorgen, könnte darin bestehen, daß die Applikation in der Lage ist, das entsprechenden Hilfematerial zu liefern. Von diesen beiden Alternativen wurde im Hilfesystem die zweite Alternative realisiert, weil die Applikation vom Hilfewerkzeug unter einem bestimmten Aspekt betrachtet wird. Und zwar besteht die Aufgabe des Hilfesystems darin, Erklärungen *zu einer bestimmten Applikation* kontextsensitiv anzuzeigen. Die zu einer Applikation passenden Erklärungen sind somit nicht beim Hilfewerkzeug anzusiedeln, sondern bei der Applikation selbst.

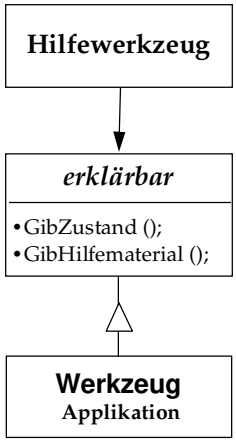


Abb. 7-2: Schnittstelle zur Applikation

Für das Hilfe-Framework wurde die Schnittstelle zwischen Hilfewerkzeug und Applikation in dem Aspekt `erklärbar` zusammengefaßt. Über den Aspekt `erklärbar` stellt jede Applikation dem Hilfewerkzeug ihr Hilfematerial und den aktuellen Zustand zur Verfügung.

Bei der hier vorgestellten Lösung mit Hilfe einer Aspektklasse muß die Schnittstelle jeder Applikation erweitert werden, die vom Hilfewerkzeug benutzt werden soll. Problematisch an diesem Ansatz ist, daß eine Applikation auf diese Weise um einen Verwendungszusammenhang in Form des Aspekts `erklärbar` erweitert wird, der üblicherweise nicht zum Anwendungsgebiet dieser Applikation gehört. Eine häufig verwendete Lösung, um dieser Problematik zu entgehen, ist, die Schnittstelle einer Klasse mit Hilfe eines Umwicklers (s. [Gamma92, GaHeJoVl95]) an einen neuen Zusammenhang anzupassen. Beim Umwickler handelt es sich um ein Entwurfsmuster, bei dem eine neue Klasse entwickelt wird, die die geforderte Schnittstelle erfüllt und die nicht zur Schnittstelle passende Klasse zur Erledigung der geforderten Dienste benutzt. Stellt eine Applikation Methoden zur Verfügung, die es erlauben, ihren Zustand festzustellen, so kann eine durch den Aspekt bedingte Erweiterung der Applikationsschnittstelle mit der Umwicklerlösung möglicherweise vermieden werden (s.u.).

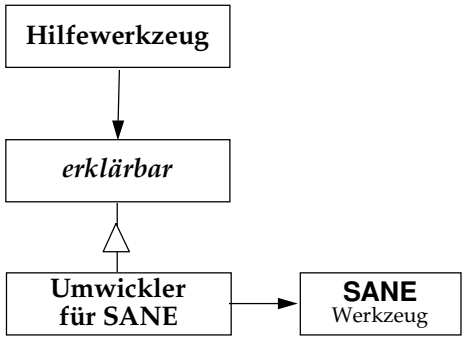


Abb. 7-3: Hilfewerkzeug mit Umwickler

Für SANE wurde die Variante mit dem Umwickler gewählt, weil der Wunsch bestand, SANE nicht zu verändern. Der für SANE entwickelte Umwickler macht das benötigte Hilfematerial für das Hilfewerkzeug verfügbar, ohne Methoden von SANE aufzurufen. Um den Zustand von SANE durch den Umwickler zur Verfügung zu stellen, war es jedoch trotzdem notwendig, SANE um einige sondierende Operationen zu erweitern.

Hier wird deutlich, welche Konsequenzen für Applikationen, die wie SANE nach ihrer Entwicklung an das Hilfewerkzeug angeschlossen werden, zu erwarten sind: Applikationen, die nicht im Hinblick auf das Hilfe-Framework entwickelt wurden, sind i.d.R. nicht in der Lage, Auskunft über ihren Zustand zu geben.

7.3 Der Kontext

Hat die Benutzerin aus dem begleitenden Beispiel bei der Arbeit mit SANE ein Übersichtsnetz erstellt und benötigt Hilfe, um es zu verfeinern, so befindet sich SANE in einem bestimmten Zustand, auf den das Hilfesystem Bezug nehmen soll. Zu diesem Zweck muß ein Zustandsbegriff vorhanden sein, der vom Hilfewerkzeug in bezug auf seine Aufgabe - Hilfeinformationen kontextsensitiv anzuzeigen - interpretiert werden kann. Damit das Hilfesystem auch mit anderen Applikationen als mit SANE zusammenarbeiten kann, muß der Zustandsbegriff soweit abstrakt sein, daß er keine SANE-spezifischen Anteile enthält.

Im folgenden wird zuerst das Konzept für den Zustand einer Applikation vorgestellt, das bei der Entwicklung des Hilfesystems für SANE abstrahiert werden konnte. Weiterhin wird das Kontextmaterial und der Zustandsinterpreter beschrieben.

7.3.1 Zustand einer Applikation

Bei der Entwicklung von objektorientierten Systemen nach der Werkzeug-Material Metapher wird i.d.R. wenig über den Zustand der einzelnen Komponenten nachgedacht. Vielmehr sind hauptsächlich Zustandsänderungen von Interesse, die dazu führen, daß eine Komponente eine andere durch den Ereignismechanismus von der an ihr aufgetretenen Zustandsänderung informiert. Für das Hilfewerkzeug sind aber weniger die Zustandsänderungen als der Zustand einer Applikation zu einem bestimmten Zeitpunkt interessant.

Der Zustand einer Applikation hängt, wie die Aufteilung von nach der Werkzeug-Material Metapher entwickelten Systemen nahelegt, vom Zustand des Werkzeugs und vom Zustand des Materials ab. Beide haben einen eigenen Zustand, der separat betrachtet werden könnte. Der jeweilige Zustand von Werkzeug und Material wird von den Tätigkeiten der Benutzerin bestimmt, wobei das Werkzeug den Zustand des Materials teilweise mit anzeigt. Das Werkzeug verfügt zwar über einen eigenständigen Zustand, der den aktuellen Stand der Bearbeitung wiedergibt. Da die Aufgabe eines Werkzeugs aber darin besteht, die Bearbeitung von Material zu ermöglichen, ist sein Zustand vom Zustand des Materials **nicht** unabhängig.

Bei mehreren Diskussionen am Arbeitsbereich Softwaretechnik wurde deutlich, daß für den Materialzustand bisher keine abstrakte Darstellung gefunden werden konnte. Auch die Untersuchung der Studierenden bei ihrer Arbeit mit SANE (s. Abschnitt 4.4) konnte keine konkreten Anhaltspunkte für die Spezifikation eines Materialzustands liefern. Der Materialzustand wird daher im folgenden nicht weiter betrachtet. In bezug auf den Werkzeugzustand konnte die empirische Untersuchung einige Hinweise auf ein abstraktes Konzept bieten. Die bereits in Abschnitt 4.4.3 zusammengetragenen Fragen der Studierenden lassen sich hier verwenden:

- Welche Möglichkeiten habe ich in diesem Zustand der Applikation weiterzuarbeiten?
- In welchem Arbeitszusammenhang mit der Applikation befinde ich mich gerade?
- Welche Bedeutung hat das Symbol, das ich gerade selektiert habe?

Diese Fragen der Studierenden können so interpretiert werden, daß die Studierenden nach dem aktuellen Zustand des Werkzeugs fragten. Der Zustand eines Werkzeugs wird in diesem Sinne festgelegt als: eine bestimmbare Anzahl von Arbeitsschritten, die in einem bestimmten Arbeitszusammenhang an einem selektierten Symbol ausführbar sind.

Auch die Benutzerin aus dem begleitenden Beispiel hat eine Frage an SANE bzw. das Hilfesystem: Wie verfeinert man eine funktionelle Rolle? Diese Frage ist aber lediglich für die Benutzerin in ihrer Anwendungssituation offensichtlich. SANE oder das Hilfesystem haben keine Möglichkeit, diese Frage zu erkennen. Aus dem Übersichtsnetz, das die Benutzerin in SANE gezeichnet hat, könnten sich eine Vielzahl von Fragen und weiteren Handlungszielen ergeben. Wird der Zustand mit Hilfe der oben aufgeführten Fragen und ihren Antworten festgehalten, so hat die Benutzerin, wie im folgenden gezeigt wird, zumindest eine größere Chance, die benötigte Information im Hilfesystem zu finden. Speziell eine Antwort auf die Frage: welche Möglichkeiten habe ich in diesem Zustand der Applikation weiter zu arbeiten? wird der Benutzerin bei der Verfeinerung einer funktionellen Rolle behilflich sein.

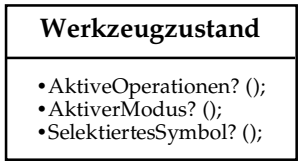


Abb. 7-4: Werkzeugzustand

Abbildung 7-4 stellt die Klasse `Werkzeugzustand` mit ihren Methoden dar. Die ausführbaren Arbeitsschritte sind die in einer Applikation zu einem bestimmten Zeitpunkt *aktiven Operationen*, also die Handlungsmöglichkeiten der Benutzerin. In dem Hilfetext einer Applikation sind zu diesen einzelnen Operationen entsprechende Hilfeknoten abgelegt. Die Benutzerin findet unter den verschiedenen aktiven Operationen auch die Operation "Verfeinern". Der zugehörige Werkzeugknoten, der die Operation "Verfeinern" beschreibt, wurde in Abschnitt 6.2.3 vorgestellt.

Der *Arbeitszusammenhang* einer Applikation wird häufig auch als *Modus* bezeichnet. In jedem Modus einer Applikation können bestimmte Arbeiten erledigt werden, wie z. B. das Manipulieren von graphischen Objekten oder das Erzeugen von Objekten. Zu diesen Arbeitszusammenhängen müssen ebenfalls in entsprechenden Hilfeknoten Erklärungen zu finden sein. Bisher wurde in dem hier verfolgten Beispiel kein Bezug auf einen Arbeitszusammenhang oder Modus genommen. Es läßt sich aber sagen, daß sich die Benutzerin im Modus "Manipulieren von graphischen Objekten" befindet. Zu diesem Modus gibt es im Hilfetext einen Übersichtsknoten, der der Benutzerin einen Einblick verschafft, was unter diesem Arbeitszusammenhang verstanden wird.

Das *selektierte Symbol* auf dem Bildschirm ist das Symbol, mit dem die Benutzerin gerade arbeiten will - auf das also ihre Aufmerksamkeit gerichtet ist. Zu jedem Symbol sollte sie in den Hilfetexten Erklärungen über seine Bedeutung (Materialknoten) und die an ihm ausführbaren Arbeitsschritte (Werkzeug- und Übersichtsknoten) erhalten. Die Benutzerin des Beispiels hat das Symbol für die Verwaltungskraft selektiert, das sie verfeinern möchte (s. Abbildung III-2). Durch ihre Selektion und dem dadurch im Werkzeugzustand vermerkten selektierten Symbol kann das Hilfesystem schließen, daß zu diesem Symbol Erklärungen gewünscht werden.

Aus Sicht der Werkzeug-Material Metapher läßt sich diese Definition des Werkzeugzustands unterstützen. Der "Sinn" eines Werkzeugs besteht darin, zur Bearbeitung von Material verwendet zu werden. Die zu einem Zeitpunkt aktiven Operationen und der Arbeitszusammenhang fassen zusammen, wie ein Material bearbeitet werden kann. Aus dem Blickwinkel des Kapitels 3 über Transparenz und Sprache führt diese Definition zu Hilfeknoten, die die Handhabung und damit die in Symbolen manifestierten Handlungen erklären. Durch die aktiven Operationen berücksichtigt der hier eingeführte Zustandsbegriff indirekt auch den Zustand des Material. Ein Werkzeug kann nur die Arbeitsschritte anbieten, die dem jeweiligen Zustand des Materials entsprechen. Indem die aktiven Operationen

aufgezählt werden, wird implizit deutlich, auf welchem Stand der momentanen Bearbeitung sich das Material gerade befindet. Das Merkmal selektiertes Symbol nimmt in diesem Zusammenhang eine eher zwiespältige Rolle ein, da es nicht direkt mit Fragen nach der Handhabung eines Werkzeugs verbunden ist. Das selektierte Symbol bietet vielmehr eine Verbindung zu den Materialknoten, die eine Erklärung zu jedem Symbol enthalten sollten. Auf diese Weise kann die Benutzerin durch das Selektieren eines Symbols deutlich machen, welcher bestimmte Teil des Materials gerade im Fokus ihres Interesses liegt.

In der Literatur werden immer wieder Zustandsautomaten für Objekte oder entworfen, um die verschiedenen möglichen Zustandsübergänge in einem Objekt festzuhalten. Als graphisches Darstellungsmittel werden i.d.R. endliche Automaten, Statecharts (s. [Harel87]) oder auch Objectcharts (s. [CoHaBe92]) verwendet. Ein vollständiger Zustandsautomat drückt alle möglichen Zustände und Zustandsübergänge eines Objekts aus. Diese umfassende softwaretechnische Zustandsbeschreibung eines Objekts soll durch den hier vorgestellten Werkzeugzustand nicht geleistet werden. Der Werkzeugzustand hat vielmehr lediglich den Sinn, über den aktuellen Zustand eines Werkzeugs unter dem Aspekt der Hilfestellung Auskunft zu geben.

7.3.2 Kontextmaterial

Kann die Applikation Auskunft über ihren Werkzeugzustand geben, so ist damit noch keine Kontextsensitivität erreicht. Die Information darüber, welche Hilfeknoten zu welchem Zustand oder zu welchen Teilaspekten eines Zustands passen, muß für das Hilfesystem zugänglich sein. Diese Verknüpfung zwischen Zustand und Hilfeknoten wird durch das Kontextmaterial geleistet.

Bei der Arbeit mit dem Hilfe-Framework hat es sich gezeigt, daß verschiedene Zustände oft Überschneidungen bei den aktiven Operationen haben. Die meisten Operationen, wie beispielsweise *Löschen*, sind in verschiedenen Zuständen der Applikation aktiv. Ebenso verhält es sich mit den Modi und selektierten Symbolen, die in einem Werkzeug verfügbar sind. Um im Kontextmaterial Verknüpfungen nur einmal eintragen zu müssen, wurde das Kontextmaterial *nicht* auf die Verbindung zwischen Zuständen und Hilfeknoten beschränkt. Es ist vielmehr möglich, sowohl Zustände als auch ihre einzelnen Merkmale (Aktive Operation, selektiertes Symbol, Modus) mit Werkzeugzuständen zu verknüpfen. Abbildung 7-5 stellt die so entstehenden Benutzt-Beziehungen zwischen Kontextmaterial, Werkzeugzuständen, deren Merkmalen und den Hilfeknoten dar.

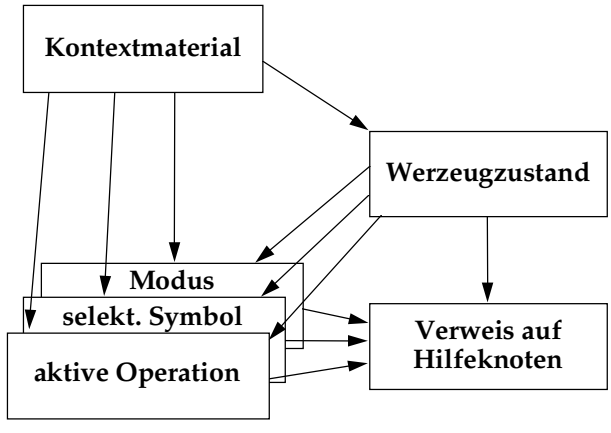


Abb. 7-5: Benutzt-Beziehungen im Kontextmaterial

Das Kontextmaterial verwaltet sowohl Werkzeugzustände als auch deren aktive Operationen, selektierte Symbole und Modi, die jeweils mit Verweisen auf Hilfeknoten versehen werden können. Wird zu einem bestimmten Werkzeugzustand die Menge der geeigneten Hilfeknoten gesucht, so werden die mit dem Werkzeugzustand und

mit seinen Einzelteilen verknüpften Hilfeknoten verwendet. Auf diese Weise kann das Kontextmaterial flexibel auf die Anforderung reagieren, und Erklärungen zu einzelnen aktiven Operationen, aber auch zu komplexeren Zusammenhängen wie Zuständen liefern.

Im letzten Abschnitt wurde der Zustand für die Benutzerin des begleitenden Beispiels herausgearbeitet: Sie hat ein Symbol für eine funktionelle Rolle selektiert, befindet sich im Modus "Manipulieren von Objekten", und ihr stehen eine Reihe von aktiven Operationen zur Verfügung, zu denen auch "Verfeinern" gehört. Um der Benutzerin Hilfe zu diesem Zustand in SANE geben zu können, enthält das Kontextmaterial verschiedene Verknüpfungen:

- zwischen dem Zustand und einem Übersichtsknoten, der das Verfeinern von funktionellen Rollen beschreibt (Beispiel s. Abschnitt 6.2.3)
- zwischen dem selektierten Symbol und einem Materialknoten, der das Symbol "funktionelle Rolle" erklärt (Beispiel s. Abschnitt 6.2.3)
- zwischen der aktiven Operation "Verfeinern" und dem Werkzeugknoten "Verfeinern" (Beispiel s. Abschnitt 6.2.3)

Die Aufgabe, diese Verknüpfungen ausfindig zu machen und die entsprechenden Hilfeknoten an das Hilfewerkzeug weiterzugeben, übernimmt im Hilfe-Framework der Zustandsinterpreter.

7.3.3 Zustandsinterpreter

Der Zustandsinterpreter ist schließlich für die Verwaltung und Verarbeitung des Hilfematerials (Hilfetext und Kontextmaterial) zu einer Applikation zuständig. Durch die Auslagerung dieser Aufgaben in den Automaten wird sichergestellt, daß das Hilfewerkzeug einerseits lediglich Hilfeknoten, Schlagwörter und Lesezeichen kennt, darstellt und die Arbeit mit ihnen unterstützt. Andererseits bleibt das Hilfewerkzeug flexibel, weil die Möglichkeit besteht, den Zustandsinterpreter gegen einen anderen Zustandsinterpreter auszutauschen, der den Zustand auf eine andere Weise auswertet.

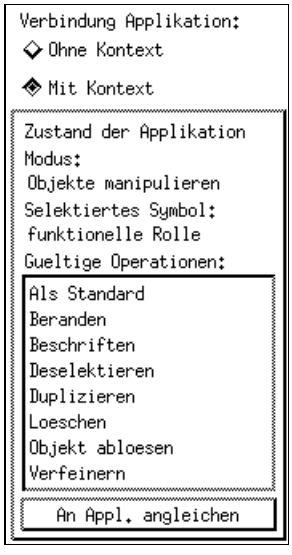


Abb. 7-6: Einstellung des Zustandsinterpreters und diagnostizierter Zustand

Da es sich bei dem Zustandsinterpreter nicht um ein Werkzeug handelt, besitzt er keine Benutzungsschnittstelle. Wie für Automaten typisch, bietet der Zustandsinterpreter keine Bearbeitung von Materialien an, sondern führt lediglich einen vorgefertigten Algorithmus aus. Trotzdem sollte die Benutzerin die Möglichkeit haben, über die Arbeit des Zustandsinterpreters informiert zu werden und die Wirkung des Zustandsinterpreters abzuschalten, um ohne kontextsensitiven Filter auf alle Hilfeknoten zugreifen zu können.

Um dieser Anforderung gerecht zu werden, macht das Hilfewerkzeug an seiner Benutzungsschnittstelle die Einstellung des Zustandsinterpreters und den diagnostizierten Werkzeugzustand sichtbar. Abbildung 7-6 zeigt einen Ausschnitt aus der Benutzungsoberfläche des Hilfewerkzeugs (Gesamtbild s. Abb. 5-1). Die Benutzerin hat hier im oberen Bereich des Hilfewerkzeugs die Möglichkeit, die Wirkung des Zustandsinterpreters abzuschalten (s. Radioknopf: "Ohne Kontext", "Mit Kontext"). Im unteren Bereich kann die Benutzerin ablesen, welcher Werk-

zeugzustand von der Applikation gemeldet wurde. Hat die Benutzerin, nachdem sie das Hilfesystem aufgerufen hat, mit der Applikation weitergearbeitet, so kann sie mit dem Knopf "An Appl. angleichen" den jeweils aktuellen Zustand der Applikation in das Hilfesystem übernehmen. Das Hilfesystem wertet daraufhin den aktuellen Zustand erneut aus und zeigt die entsprechenden Hilfeknoten an.

7.4 Aktivierung des Hilfesystems aus einer Applikation

Da es sich bei dem Hilfesystem nicht um eine (Sub-)Komponente einer Applikation handelt, sondern um ein eigenständiges Werkzeug, das unabhängig von einer bestimmten Applikation ist, kann das Hilfewerkzeug nicht von einer Applikation direkt aufgerufen werden. Es sind verschiedene Möglichkeiten denkbar, das Hilfewerkzeug zu aktivieren und es mit der Applikation in Verbindung zu bringen. Zwei im Rahmen der Werkzeug-Material Metapher umsetzbare Varianten haben entweder das Hilfewerkzeug oder die Applikation als Ausgangspunkt.

Ist das Hilfewerkzeug der Ausgangspunkt, so steht das *Werkzeug* im Fokus des Interesses. In einer graphischen Benutzungsoberfläche läßt sich diese Handhabungsform dadurch verwirklichen, daß die Benutzerin das Hilfewerkzeug als Ikone in ihrer Arbeitsumgebung vorfindet und es bei Bedarf mit Hilfe der Maus auf die entsprechende Applikation zieht. Durch diese Darstellung wird deutlich, daß ein *Werkzeug* zur Bearbeitung eines Materials zur Hand genommen wird.

Andererseits besteht die Möglichkeit, den bei vielen Applikationen üblichen Aufruf des Hilfesystems über die Menüleiste zu realisieren. Hier wird der Aufruf des Hilfesystems vom *Material* des Hilfewerkzeugs, der Applikation, aus gestartet. Diese Art, ein Hilfesystem zu aktivieren, entspricht einer Handhabungsform, die ebenfalls auf graphischen Benutzungsoberflächen verwendet wird. Benutzerinnen können dort eine Applikation starten, indem sie ein bestimmtes Material auswählen. Das zugehörige Werkzeug wird aufgrund der Auswahl automatisch geöffnet. Diese Sichtweise macht deutlich, daß das Hauptinteresse der Benutzerin bei ihrem *Material,* d.h. auf der für sie problematischen Applikation, liegt, für die sie Hilfe braucht.

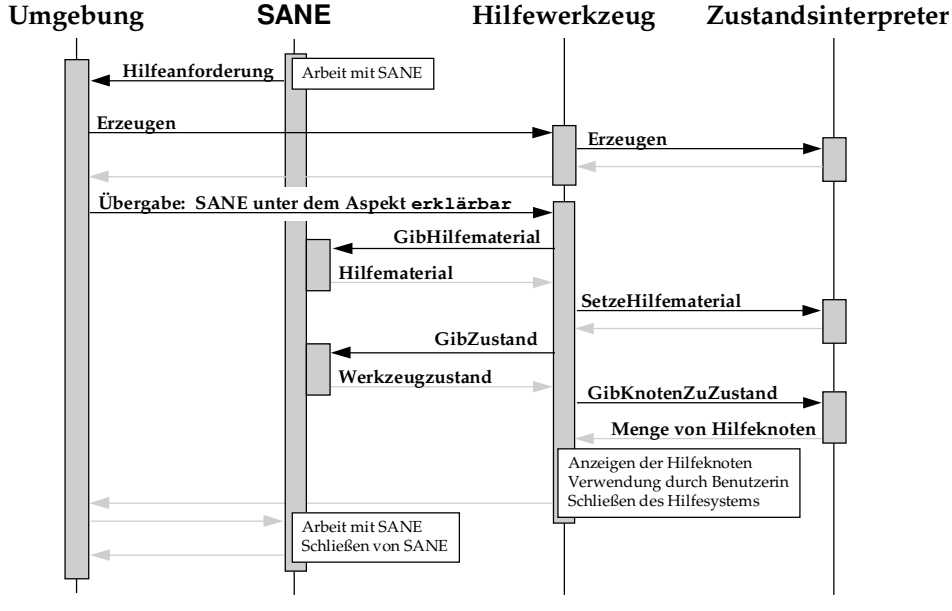


Abb. 7-7: Dynamik des Hilfesystems

Beide Vorschläge haben ihre Berechtigung und sind im Rahmen der Werkzeug-Material Metapher einsetzbar. Für das hier vorgestellte Hilfesystem wurde die zweite Variante gewählt, weil sie sich im Rahmen der verwendeten Bibliotheken und des Fenstersystem besser verwirklichen ließ. In welcher Weise die Objekte beim Aufruf des Hilfesystems erzeugt werden und wie sie zusammenarbeiten, ist in Abbildung 7-7 anhand des dynamischen Verhaltens festgehalten worden. Abbildung 7-8 stellt die Benutzt-Beziehungen zwischen den Objekten dar, nachdem alle Teile erzeugt sind.

Fordert die Benutzerin bei der Arbeit mit SANE Hilfe an, so kann sie dies direkt in der Menüleiste des Aufgabennetzeditors tun. Der Aufgabennetzeditor verschickt daraufhin mit Hilfe des Ereignismechanismus eine Nachricht an alle Beobachter, daß die Benutzerin das Hilfesystem angefordert hat. Einer der Beobachter einer jeden Applikation ist das Umgebungsobjekt, das die Werkzeuge und Materialien eines Arbeitsplatzes verwaltet und als umgebender Rand oder Hülle des Arbeitsplatzes dient. Dieses Umgebungsobjekt interpretiert die Hilfenachricht, indem es das Hilfesystem erzeugt. Nachdem das Hilfewerkzeug seinen Zustandsinterpreter erzeugt hat, übergibt die Umgebung dem Hilfewerkzeug die Applikation (SANE) unter dem Aspekt `erklärbar`. Das Hilfewerkzeug fordert nun bei SANE das passende Hilfematerialien an. Die Hilfematerialien, der Hilfetext und das Kontextmaterial, werden an den Zustandsinterpreter zur Verwaltung und Bearbeitung weitergegeben. Um kontextsensitiv auf die Hilfeanforderung reagieren zu können, läßt sich das Hilfewerkzeug von der Applikation den aktuellen Werkzeugzustand geben und fordert beim Zustandsinterpreter die zu dem Zustand passenden Hilfeknoten an.

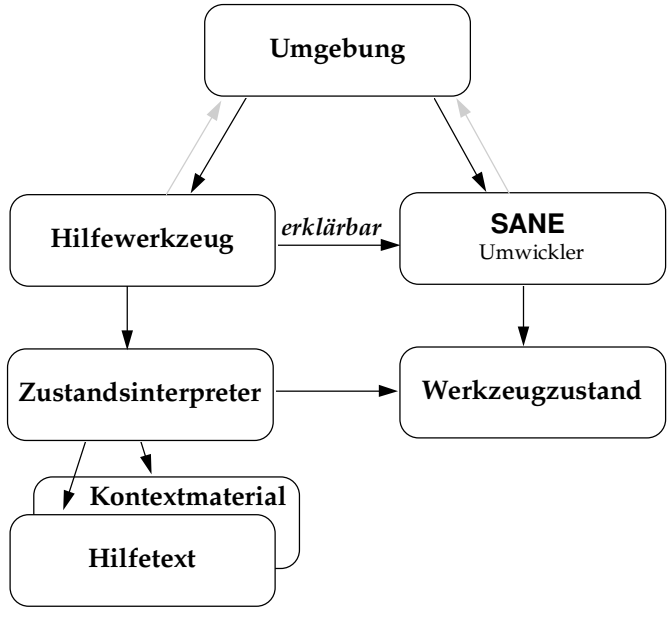


Abb. 7-8: Benutzt-Beziehung zwischen Hilfesystem, SANE und Umgebung nach der Erzeugung (Die grauen Pfeile stellen die Benachrichtigung durch den Ereignismechanismus dar.)

Der Zustandsinterpreter ermittelt aufgrund des gelieferten Zustands bzw. seiner einzelnen Merkmale, wie aktive Operationen, selektiertes Symbol, Modus, mit Hilfe des Kontextmaterials die geeigneten Knoten und stellt sie zur Verfügung. Das Hilfewerkzeug ist nun in der Lage, die Hilfeknoten anzuzeigen und der Benutzerin hoffentlich Information zur Weiterarbeit mit der Applikation zu bieten. Die Benutzerin kann mit dem Hilfewerkzeug und mit

SANE weiterarbeiten bzw. das Hilfewerkzeug beenden und zu ihrer Arbeit mit SANE zurückkehren.

Nachdem in diesem Kapitel vorgestellt wurde, auf welche Weise im Hilfe-Framework dynamische Hilfe realisiert wird, ist die Beschreibung des Hilfesystems und seines Einsatzkontextes abgeschlossen.

8 Das Hilfe-Framework

Neben dem in den letzten beiden Kapiteln vorgestellten Einsatzkontext, bei dem eine Benutzerin das Hilfesystem zur Erklärung einer Applikation einsetzt, bietet das Hilfe-Framework der Entwicklerin einer Applikation einen anderen Einsatzkontext an: die Arbeitsumgebung für Hilfe. Für die Entwicklerin einer Applikation wurden Werkzeuge geschaffen, die es der Entwicklerin ermöglichen, die für ihre Applikation notwendigen Hilfematerialien zusammenzustellen.

In diesem Kapitel wird zuerst ein Überblick über die verschiedenen Werkzeuge für den Einsatzkontext der Entwicklerin gegeben. Die für Softwareentwicklerinnen vorgesehenen Werkzeuge des Hilfe-Frameworks werden mit Hilfe von Handhabungsvisionen beschrieben. Um das begleitende Beispiel abzurunden, wird in einem Teil der Handhabungsvisionen geschildert, wie eine Entwicklerin für SANE Hilfematerialien erstellt, damit SANE kontextsensitiv Hilfe für die spezielle Situation der Benutzerin aus dem Beispiel zur Verfügung stellen kann. Außerdem werden der Entwurf der Schnittstelle zwischen der Arbeitsumgebung für Hilfe und der Applikation sowie die Zusammenarbeit der Werkzeuge in der Arbeitsumgebung vorgestellt. Abschließend wird die Überlegung diskutiert, das Hilfe-Framework bereits im Verlauf des Entwicklungsprozesses einzusetzen.

8.1 Werkzeuge zur Erstellung der Hilfematerialien

Das Hilfe-Framework enthält z. Z. fünf Werkzeuge, die für die Entwicklerinnen von Applikationen bestimmt sind:

- *Hilfeknotenauflister*: Dieses Werkzeug erlaubt es, Hilfeknoten und Schlagwörter zu erzeugen, zu löschen und umzubenennen. Außerdem ermöglicht es, Schlagwörtern mit Hilfeknoten zu verknüpfen und abzutrennen.
- *Hilfeknoteneditor*: Hilfeknoten können mit dem Hilfeknoteneditor als Hypertextstrukturen erstellt werden. Der Name und die Klassifikation des jeweiligen Hilfeknotens kann festgelegt bzw. geändert werden.
- *Zustandsauflister*: Mit diesem Werkzeug werden neue Zustände unter Verwendung der Applikation erzeugt. Außerdem können Zustände gelöscht und umbenannt werden.
- *Zustandseditor*: Aus den einzelnen Merkmalen eines Zustands (aktive Operationen, Modus und selektiertes Symbol) können Zustände zusammengesetzt werden. Zu den einzelnen Merkmalen und zu den Zuständen können Kommentare abgelegt werden.
- *Kontexteditor*: In diesem Werkzeug werden Hilfeknoten mit Zuständen oder Merkmalen von Zuständen verbunden; dadurch wird das Kontextmaterial erstellt.

Genau wie das Hilfesystem benötigt auch dieser Teil des Hilfe-Frameworks eine Schnittstelle zur Applikation, über die Teile des Hilfematerials zusammengestellt werden. Soll die Möglichkeit bestehen, daß im Kontexteditor Zustände mit Hilfeknoten verbunden werden und daraus das Kontextmaterial entsteht, so muß die Applikation Auskunft über ihre Zustände geben können. Diese weitere Schnittstelle zwischen Hilfe-Framework und Applikation wird in Abschnitt 8.3. erläutert.

8.2 Handhabungsvisionen der zusätzlichen Werkzeuge

Wie für das Hilfesystem wurden auch für die Werkzeuge zur Erstellung des Hilfematerials Handhabungsvisionen erstellt. Mit Hilfe einiger dieser Handhabungsvisionen wird im folgenden ein Eindruck von der Arbeit mit den Werkzeugen vermittelt, deren Entwurf anschließend erläutert wird. Als Handhabungsvisionen werden vorgestellt:

- Einen neuen Hilfeknoten in den Hilfetext einfügen,
- einen Zustand für das Kontextmaterial mit Hilfe der Applikation erzeugen,
- Merkmale von Zuständen für das Kontextmaterial aus der Applikation übernehmen und
- Hilfeknoten mit Zuständen und Merkmalen von Zuständen verknüpfen.

Die vier exemplarischen Handhabungsvisionen beschreiben nur einige Tätigkeiten, die mit den Werkzeugen des Hilfe-Frameworks erledigt werden können.

8.2.1 Einen neuen Hilfeknoten in den Hilfetext einfügen

Die Entwicklerin arbeitet mit den Werkzeugen des Hilfe-Frameworks, um einen neuen Hilfeknoten zu erstellen. Sie erzeugt im Hilfeknotenauflister einen neuen Knoten, indem sie den Namen des Knotens "Funktionelle Rolle" (s. Abbildung 8-1) eingibt und auf den Knopf "Neu" drückt. Anschließend klickt sie mit Hilfe der Maus doppelt auf den in der Liste der Hilfeknoten eingefügten Namen des Knotens. Der Hilfeknoten erscheint in einem leeren Hilfeknoteneditor.

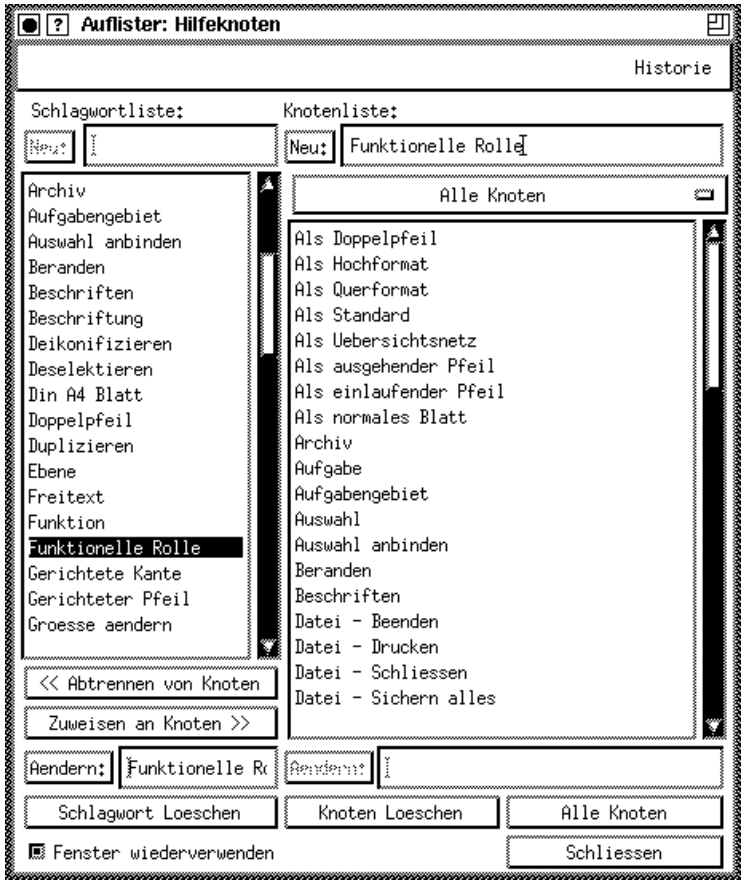


Abb. 8-1: Hilfeknotenauflister

Hier gibt die Entwicklerin den Text des Hilfeknotens ein und fügt in den Text Verweise auf andere Knoten ein (s. Abbildung 8-2). Zu diesem Zweck markiert sie im Text die Stelle, an der der Verweis eingefügt werden soll, wählt den Knotennamen "Verfeinern" aus der Liste neben dem Textfeld aus und drückt auf den Knopf "Verweis einfügen". Der Verweis <<Verfeinern>> wird daraufhin in den Text des Hilfeknotens eingefügt. Die Entwicklerin ordnet den Hilfeknoten in die Klassifikation der Hilfeknoten als Materialknoten ein (s. Abbildung 8-2). Dazu wählt sie den entsprechenden Radioknopf aus der angebotenen Selektion aus.

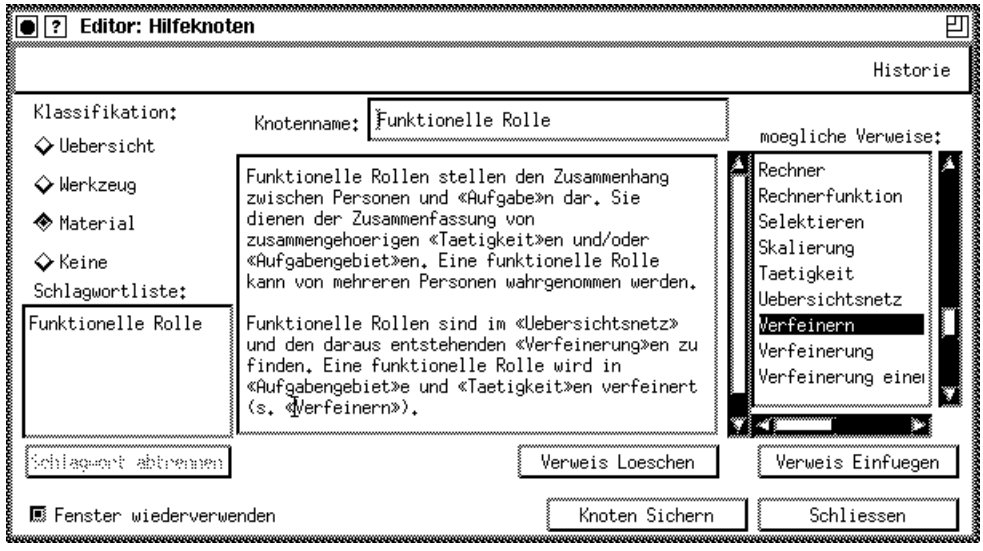


Abb. 8-2: Hilfeknoteneditor

Nachdem sie alle notwendigen Informationen in den Hilfeknoteneditor eingetragen hat, legt sie den Hilfeknoten in den Hilfetext ab, indem sie auf den Knopf "Knoten speichern" drückt. Anschließend schließt sie den Hilfeknoteneditor und verknüpft im Hilfeknotenauflister (s. Abbildung 8-1) den neuen Hilfeknoten mit zugehörigen Schlagwörtern. Zuerst markiert die Entwicklerin das Schlagwort "Funktionelle Rolle" aus der Schlagwortliste (s. Abbildung 8-1) und den Hilfeknoten "Funktionelle Rolle" aus der Liste der Hilfeknoten. Mit Hilfe des Knopfes "Zuweisen an Knoten >>" verknüpft sie das Schlagwort mit dem Hilfeknoten. Um festzustellen, welche Hilfeknoten mit dem Schlagwort verbunden sind, führt sie einen Doppelklick auf das Schlagwort aus. In der Liste der Hilfeknoten werden daraufhin nur noch die Namen der Hilfeknoten angezeigt, die mit diesem Schlagwort verbunden worden sind. Da die Entwicklerin wieder die gesamte Liste der Hilfeknoten aufgelistet sehen will, drückt sie anschließend den Knopf "Alle Knoten".

8.2.2 Zustand für das Kontextmaterial mit Hilfe der Applikation erzeugen

Die Entwicklerin will einen Zustand aus SANE in das Kontextmaterial aufnehmen, um ihn anschließend mit mehreren Hilfeknoten zu verbinden. Dafür ruft die Entwicklerin den Zustandsauflister auf und drückt auf den Knopf "Applikation starten". SANE wird erzeugt, und die Entwicklerin bringt SANE in den Zustand, den sie in das Kontextmaterial aufnehmen will. Zu diesem Zweck zeichnet sie ein Übersichtsnetz und selektiert ein Symbol für eine funktionelle Rolle. Nachdem sie in SANE den gewünschten Zustand erreicht hat, klickt sie im Zustandsauflister auf den Knopf "Zustand übernehmen". Der Zustandsauflister läßt sich von SANE den aktuellen Zustand geben und trägt ihn als neuen Zustand "Zustand 1" in seine Liste ein. Die Entwicklerin ändert den Namen des Zustands in "F. Rolle bearbeiten" ab. Daraufhin öffnet sie mit einem Doppelklick auf dem Namen des Zustands den Zustandseditor und trägt zu dem Zustand einen Kommentar ein (s. Abb. 8-4). Anschließend arbeitet

Abb. 8-3: Zustandsauflister

sie mit SANE weiter und übernimmt einen weiteren Zustand in den Zustandsauflister.

8.2.3 Merkmale von Zuständen für das Kontextmaterial aus der Applikation übernehmen

Abb. 8-4: Zustandseditor

Die Entwicklerin will verschiedene Zustände nicht mit Hilfe von SANE einzeln nach spielen (vgl. 8.2.2), sondern die Merkmale der Zustände (aktive Operationen, Modus, selektiertes Symbol) direkt aus der Applikation extrahieren und gesondert zu Zuständen zusammenstellen. Dazu wählt die Entwicklerin den Zustandsauflister an und drückt den Knopf "Merkmale übernehmen" (s. Abb. 8-3). Der Zustandsauflister übernimmt aus SANE alle Werte, die die einzelnen Merkmale annehmen können. Anschließend erzeugt die Entwicklerin im Zustandsauflister einen neuen Zustand, indem sie den Namen des Zustands "Info. bearbeiten" eingibt und den Knopf "Neu" betätigt. Mit einem Doppelklick auf den Namen des Zustands in der Liste der Zustände öffnet sie den Zustandseditor.

Hier wählt sie mit Hilfe von zwei 1-aus-n Selektionen für den Modus den Wert "Objekte manipulieren" und für das selektierte Symbol "Informationsstruktur" aus. Als Beschreibung zu dem Zustand gibt sie einen Kommentar in das zu diesem Zweck vorhandene Kommentarfeld ein. Anschließend speichert die Entwicklerin den so zusammengestellten Zustand ab und beginnt einen weiteren Zustand zu erzeugen.

8.2.4 Hilfeknoten mit Zuständen und Merkmalen von Zuständen verknüpfen

Die Entwicklerin will mehrere Zustände, die sie mit Hilfe von SANE nach gespielt hat (s. Handhabungsvision 8.2.2), aber auch Zustände und Merkmale von Zuständen, die sie direkt aus der Applikation übernommen hat (s. Handhabungsvision 8.2.3), mit Hilfeknoten verbinden. Zu diesem Zweck ruft die Entwicklerin den Kontexteditor auf.

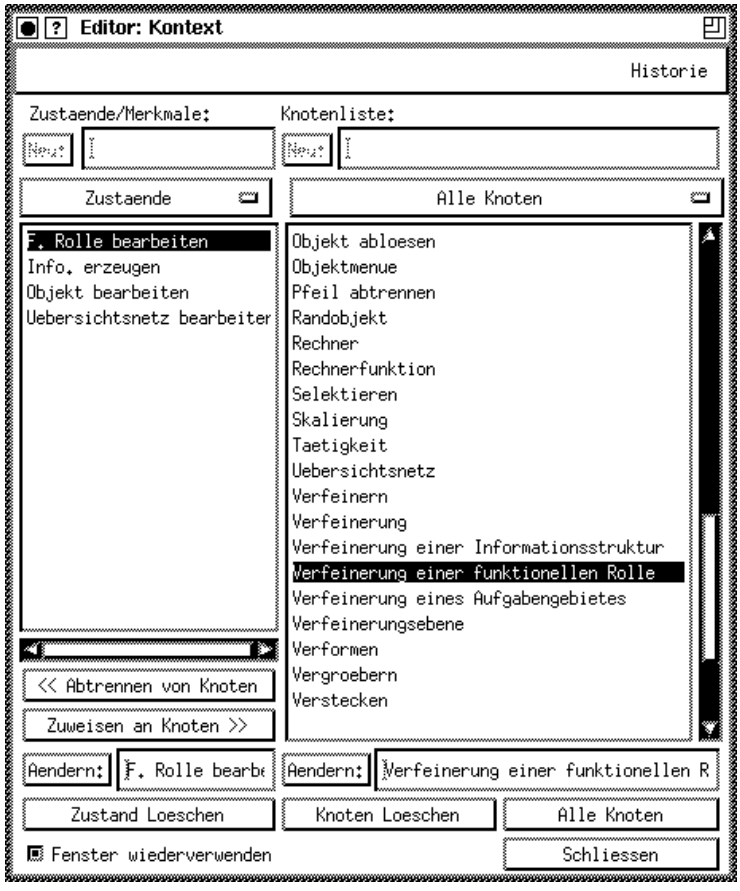


Abb. 8-5: Kontexteditor

Im Kontexteditor selektiert sie mit Hilfe eines 1-aus-n Knopfes in der rechten Liste der Hilfeknoten "Alle Knoten". In der linken Liste selektiert sie mit dem dort vorhandenen 1-aus-n Knopf alle Zustände. Nachdem die Entwicklerin in der linken Liste den Zustand "F. Rolle bearbeiten" und in der rechten Liste den Knoten "Verfeinerung einer funktionellen Rolle" selektiert hat (s. Abbildung 8-5), verbindet die Entwicklerin den Zustand mit den Hilfeknoten, indem sie auf den Knopf "Zuweisen an Knoten" drückt. Anschließend verbindet sie den Zustand noch mit weiteren Hilfeknoten. Nachdem sie alle nötigen Verbindungen hergestellt hat, schließt sie den Kontexteditor wieder.

Die in diesen Visionen vorgestellte Handhabung muß ihre Umsetzung im Entwurf der Werkzeuge zur Erstellung des Hilfematerials finden. Die Konstruktion der einzelnen Werkzeuge wurde entsprechend dem Hilfewerkzeug nach den Mustern der Werkzeug-Material Metapher vorgenommen. Da der Entwurf der einzelnen Werkzeuge keine Besonderheiten aufweist, wird auf eine detailliertere Beschreibung verzichtet. Zwei Entwurfsentscheidungen, wie die Werkzeuge mit der Applikation verbunden und auf welche Weise sie koordiniert werden können, bieten mehr Stoff zur Diskussion, so daß sie in den nächsten beiden Abschnitten vorgestellt werden.

8.3 Verbindung zur Applikation

Um im Kontextmaterial Hilfeknoten mit Zuständen oder deren Merkmalen zu verbinden, müssen im Kontexteditor die Zustände und deren Merkmale bekannt sein. Es bieten sich verschiedene Möglichkeiten an, diese Information verfügbar zu machen.

Zum Einen kann der Zustandseditor derart konzipiert werden, daß die Zustände und deren Merkmale insgesamt neu definiert werden müssen. In diesem Fall ist keine softwaretechnische Verbindung zur Applikation notwendig. Die Entwicklerin muß "lediglich" in der Lage sein, die Verbindung zwischen Zuständen der Applikation und den im Zustandseditor definierten Zuständen in Gedanken aufrecht zu erhalten. Diese Lösung verlangt von der Entwicklerin viel unnötige Arbeit wie das Zusammenstellen von Zuständen, obwohl das Wissen über die Zustände in der Applikation bereits vorhanden ist.

Eine andere Art, die Zustände im Kontexteditor bekannt zu machen, besteht darin, die Zustände direkt aus der Applikation zu extrahieren. Um das zu ermöglichen, muß die Schnittstelle der Applikation so erweitert werden, daß die Applikation Auskunft über alle Zustände geben kann, in die sie gelangen kann. Bei der Untersuchung verschiedener Applikationen hat sich herausgestellt, daß die Zusammenstellung aller Zustände in einer Applikation schwierig ist, weil eine Applikation i.d.R. nicht so entworfen wird, daß sie Auskunft über die Gesamtheit ihrer Zustände geben kann. Einfacher und leichter zu verwirklichen ist, daß die Applikation Auskunft über die verschiedenen Belegungen der einzelnen Merkmale von Zuständen geben kann. Die Applikation könnte Informationen darüber bereitstellen, welche Operationen sie besitzt, die für die Benutzerin ausführbar sind, über welche Modi sie verfügt und welche Symbole sie der Benutzerin zur Selektion anbietet. Die Entwicklerin hat nun die Möglichkeit, diese Merkmale im Zustandseditor zu Zuständen zu verknüpfen und sie im Kontexteditor entweder einzeln oder als Zustand mit Hilfeknoten zu verbinden. Der Vorteil dieser Lösung liegt darin, daß einige Verknüpfungen zwischen einem Merkmal eines Zustands und einem bestimmten Hilfeknoten nicht für jeden Zustand neu definiert werden müßten (s. Abschnitt 7.3.2). Der Befehl "Löschen" zum Beispiel sollte, egal in welchem Zustand er gerade aktiv ist, immer zu dem Hilfeknoten führen, der eine Erklärung für diesen Befehl enthält. Kann diese Verknüpfung als für alle Zustände geltend vorgenommen werden, so reduziert sich die Anzahl der neu zu erstellenden Verknüpfungen pro Zustand beträchtlich.

Als dritte Möglichkeit bietet es sich an, daß die Entwicklerin bestimmte Zustände mit der Applikation nachstellt und den Arbeitsumgebung für Hilfe die Applikation auf ihren Zustand hin analysieren läßt. Auf diese Weise kann die Entwicklerin wie in Vision 8.2.2 mit Hilfe des Zustandsauflisters Zustände definieren, während sie interaktiv mit der Applikation arbeitet. Anschließend kann sie diese Zustände im Kontexteditor mit Hilfeknoten verknüpfen. Diese Lösung ist wegen ihrer direkten Verknüpfung zwischen der Anwendung der Applikation und dem zu erstellenden Kontextmaterial sehr attraktiv. Außerdem erlaubt es diese Lösung, daß eine Operation des Aspekts `erklärbar`, mit dem das Hilfewerkzeug arbeitet, wiederverwendet werden kann.

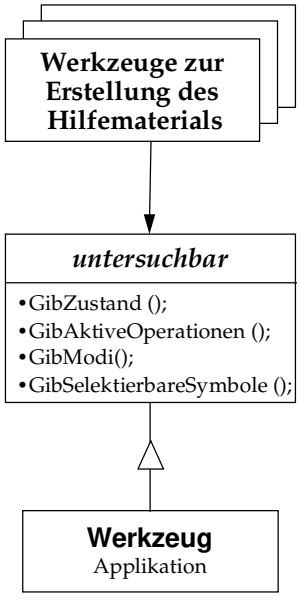


Abb. 8-6: Schnittstelle

Im Hilfe-Framework wurden die zweite und die dritte Lösung in Kombination umgesetzt, um die Vorteile beider Vorgehensweisen auszunutzen. Entwicklerinnen können sowohl mit dem Zustandseditor Zustände aus den von der Applikation gelieferten Merkmalen zusammensetzen als auch im Zustandsauflister neue Zustände mit Hilfe der Applikation erzeugen. Sollen Zustände aus Merkmalen zusammengesetzt werden, so werden im Zustandseditor die mögliche Belegungen der Merkmale in der Applikation erfragt und diese zu Zuständen zusammengestellt. Soll ein Zustand aus der Applikation übernommen werden, so muß die Applikation aus dem Zustandsauflister heraus gestartet und ein bestimmter Zustand in der Applikation herbeigeführt werden. Anschließend kann der Zustandsauflister aufgefordert werden, den Zustand bei der Applikation zu erfragen und als neuen Zustand aufzunehmen.

Um diese Anforderungen zu erfüllen muß eine Schnittstelle zwischen der Applikation und den Werkzeugen zur Erstel-

lung der Hilfematerialien formuliert werden. Die Methode, welche dem Hilfesystem Auskunft über den in der Applikation vorhandenen Zustand gibt, kann wiederverwendet werden, wenn die Entwicklerin Zustände bei der exemplarischen Arbeit mit der Applikation definiert. Die drei anderen Operationen werden benötigt, um die möglichen Belegungen der einzelnen Merkmale von Zuständen bei der Applikation zu erfragen. Aus den einzelnen Merkmalen kann die Entwicklerin Zustände definieren und Verknüpfungen mit Hilfeknoten vornehmen.

Wird das Hilfe-Framework mit einer Applikation eingesetzt, so ist zu bedenken, daß Applikationen häufig weiterentwickelt werden und von neueren Versionen abgelöst werden. Soll das Hilfe-Framework während des Entwicklungsprozesses einer Applikation eingesetzt werden, so sind die Änderungen die eine Applikation erfährt, noch tiefgreifender. Die Entwicklerin muß jede Art von Änderungen im Hilfetext und im Kontextmaterial nachführen können, um die Konsistenz zwischen Applikation und Hilfe jederzeit aufrecht zu erhalten. Das Hilfe-Framework unterstützt die Entwicklerin bei dieser Arbeit, indem es ihnen die Freiheit läßt, Hilfeknoten und Zustände zu verändern, neu zu definieren und zu löschen. Auch die Verknüpfungen zwischen Hilfeknoten und Zuständen können jederzeit verändert oder neu definiert werden.

8.4 Koordination von Werkzeugen mit mehreren Materialien

Die Werkzeuge, die in diesem Kapitel zur Erstellung von Hilfematerialien vorgestellt wurden, sind mit Hilfe eines eigenen Werkzeugkoodinators zusammengefaßt. Das in Abschnitt 6.4 geschilderte Prinzip der Koordination wiederverwendbarer Subwerkzeuge wurde auch für die Koordination der Werkzeuge zur Erstellung des Hilfematerials verwendet. Um hier erneut eingesetzt werden zu können, müßte das Prinzip der Koordination erweitert werden, weil es unter den Werkzeugen zur Erstellung des Hilfematerials kein ausgezeichnetes Werkzeug gibt, das die Koordination übernehmen kann.

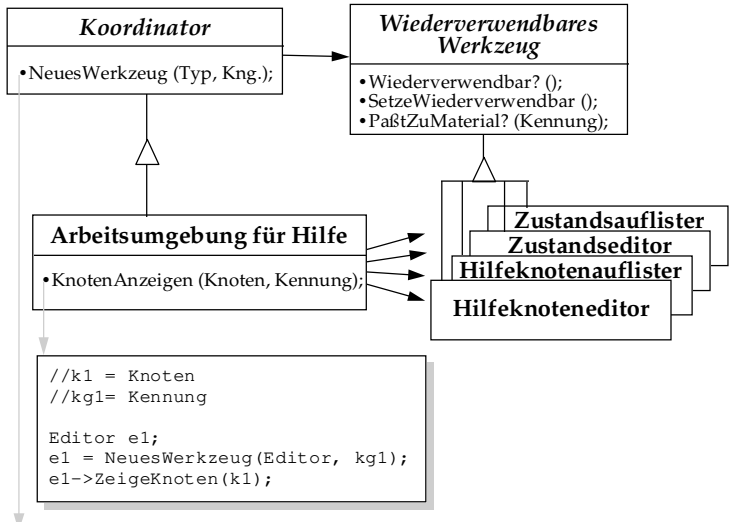
Bei dem Versuch, die verschiedenen Werkzeuge für die Erstellung der Hilfematerialien zu koordinieren, hat es sich gezeigt, daß es nicht wie beim Hilfesystem ein Kontextwerkzeug gibt, das den anderen (Sub-)Werkzeugen übergeordnet ist. Vielmehr stehen alle oben aufgezählten Werkzeuge gleichberechtigt nebeneinander. Trotzdem sollte die Entwicklerin bei ihrer Arbeit mit den Werkzeugen in der Lage sein, das Hilfematerial mit einer Gruppe von Werkzeugen zu bearbeiten.

Abgesehen von der Problematik des fehlenden Kontextwerkzeugs sollen in diesem Teil des Hilfe-Frameworks mehrere Hilfematerialien gleichzeitig bearbeitet werden können. In Gegensatz dazu stehen im Hilfesystem immer nur ein Hilfetext und eine Kontextmaterial zur Zeit zur Verfügung - nämlich das Hilfematerial, das zu einer bestimmten Applikation paßt. Um der Entwicklerin die Möglichkeit zu bieten, verschiedene Hilfematerialien gleichzeitig zu bearbeiten, muß das Hilfe-Framework in der Lage sein, mehrere Hilfematerialien zur gleichen Zeit anzubieten. Bei der ursprünglich für das Hilfewerkzeug entwickelten Koordination von Subwerkzeugen wählt der Koordinator aus der Liste der vorhandenen Subwerkzeuge jeweils ein Subwerkzeug aus, das wiederverwendbar ist und dem Typ von Werkzeug entspricht, das die Benutzerin durch ihre Aktion angefordert hat. Bearbeitet eine Entwicklerin mehrere Hilfematerialien gleichzeitig, so ist es von Vorteil, wenn Werkzeuge nur dann wiederverwendet werden, wenn sie zu dem selben Hilfematerial gehören. Ein Werkzeug muß bei dieser erweiterten Art der Koordination nicht nur wiederverwendbar sein und dem Typ des angeforderten Werkzeugs entsprechen, sondern auch zu demselben

Abb. 8-7: Die Arbeitsumgebung für Hilfe als Koordinator

Hilfematerial gehören, wie das Werkzeug, von dem aus die Benutzerin das neue Werkzeug angefordert hat.

Um diese Problematik aufzulösen, wurde das Konzept der Subwerkzeugkoordination um ein separates Kontextwerkzeug erweitert (s. Abbildung 8-7), das die in Bearbeitung befindlichen Hilfematerialien verwaltet und die verschiedenen auf ihnen einsetzbaren Werkzeuge koordiniert. Dieses Kontextwerkzeug wird an der Benutzungsoberfläche angezeigt und gibt Auskunft über die geöffneten Materialien und die vorhandenen Werkzeuge.




```
//k1 = Knoten
//kg1= Kennung

Editor e1;
e1 = NeuesWerkzeug(Editor, kg1);
e1->ZeigeKnoten(k1);
```

```
//t1 = Typ des gesuchten Werkzeugs
//kg1= Kennung des Materials

FKList->First();
while not (( FKList->Current() vom Typ t1            )
       and ( FKList->Current()->PasstZuMaterial(Kg1) )
       and ( FKList->Current()->Wiederverwendbar?()  ))
  FKList->Next();

if not (FKList->IsLast())
  return FKList->Current();
else
  return tFKBase::SpätErzeugen(t1);
```

Abb. 8-8: Werkzeugkoordination

Technisch verwirklicht das Kontextwerkzeug die Werkzeugkoordination, indem es für jedes Material eine Kennung bereitstellt und diese den einzelnen Werkzeugen bei der Erzeugung übergibt. Wird ein neues Werkzeug angefordert, so fragt der Koordinator alle Werkzeuge danach, ob sie zu dem gerade bearbeiteten Projekt gehören, ob sie vom gewünschten Typ sind und ob sie wiederverwendbar sind (s. Abbildung 8-8). Werden alle Fragen positiv beantwortet, so kann das untersuchte Werkzeug wiederverwendet werden. Vergleicht man das Konzept der Werkzeugkoordination wie es in Kapitel 6 beschreiben wurde mit seine hier erläuterte Erweiterung, so kann man feststellen, daß die hier beschriebene Variante nur um die Kennung ergänzt wurde.

8.5 Das Hilfe-Framework im Entwicklungsprozeß

Während der Entwicklung des Hilfe-Frameworks kam der Gedanke auf, daß es möglich wäre, das Hilfewerkzeug im Entwicklungsprozeß als Lern- und Kommunikationsmittel einzusetzen. Während der Entwicklung einer Software entstehen i.d.R. eine Vielzahl von Dokumenten, die das zukünftige System beschreiben. Ein Großteil dieser Dokumente wird als Hypertext angelegt, weil die einzelnen, verschiedenen Dokumente stark miteinander verknüpft sind. Beispielsweise werden häufig Glossare benutzt, die Definitionen für wich-

tige Begriffe enthalten. Aus allen anderen Dokumenten wird auf die Einträge im Glossar verwiesen, so daß Definitionen zentral abgelegt werden können. Abgesehen von der hyperartigen Struktur vieler Dokumente, die ihre Nähe zum Hilfetext bereits deutlich macht, beschreiben die Dokumente das Zielsystem, seine Handhabung und seine Einbettung in die Anwendungswelt. Genau diese Informationen sollen später in einem Hilfetext abgelegt werden. All diese Überlegungen weisen daraufhin, daß Hilfetexte auf der Basis von Entwicklungsdokumenten erstellt werden sollten. Diese Idee ist nicht neu, sie wird in allen einschlägigen Anleitungen zur Projektplanung und Durchführung aufgeführt. An dieser Stelle soll aber noch etwas weiter gedacht werden.

Eine häufig im Softwareentwicklungsprozeß eingesetzte Technik, um die Kommunikation zwischen Entwicklerinnen und zukünftigen Benutzerinnen zu verbessern, ist das Prototyping[23]. Bei Prototyping werden die Benutzerinnen mit einem Softwaresystem konfrontiert, das einen bestimmten Teil der zukünftigen Software umsetzt. Mit Hilfe dieses konkreten Artefakts in Form eines Prototyps sollen die Vorstellungen der Entwicklerinnen für die Benutzerinnen greifbar gemacht werden. Denkbar wäre nun, daß der Prototyp so gestaltet wird, daß er bereits in einem frühen Stadium mit dem hier vorgestellten Hilfesystem zusammenarbeiten kann. Ist das Hilfesystem in der Lage, die oben diskutierten Dokumente zu dem zukünftigen System anzuzeigen, so können die Benutzerinnen die Entwicklungsdokumente direkt auf ihre Tragfähigkeit für den Prototyp überprüfen. Gleichzeitig können die Benutzerinnen evaluieren, ob der Prototyp überhaupt mit den zwischen Entwicklerinnen und Benutzerinnen ausgehandelten Visionen überein stimmt. Die Benutzerinnen können die verschiedenen Dokumente selbst dann als Erklärungen benutzen, wenn sie ohne Entwicklerinnen mit einem Prototyp im Piloteinsatz umgehen. Ein wichtiger Vorteil des Hilfe-Frameworks ist hier, daß Benutzerinnen Kommentare im Hilfesystem zu den Dokumenten eintragen können. Auf diese Weise kann das Hilfesystem bereits im Entwicklungsprozeß als Lern- und besonders als Kommunikationsmittel eingesetzt werden.

Abgesehen von den Vorteilen für den gesamten Entwicklungsprozeß können auch die Entwicklerinnen vom frühen Einsatz profitieren. Eine Software, bei der schon in frühen Phasen der Entwicklung über Zustände und den Anschluß an das Hilfe-Framework nachgedacht wird, braucht später nicht um Teile erweitert zu werden, die für das Hilfe-Framework bei einer Zusammenarbeit unverzichtbar sind.

[23] s. [BKKZ92, BBLSSZ94, Floyd94b]

Zusammenfassung und Ausblick

Ziel dieser Arbeit war, ein Hilfesystem zu entwickeln, das Benutzerinnen bei ihrer Arbeit mit Applikationen unterstützt. Darüber hinaus sollte das Hilfesystem zu den Konzepten der Werkzeug-Material Metapher paßt und den symbolischen Charakter von Computerartefakten berücksichtigt. Diese drei Ausgangspunkte führten zu der im Grundlagenteil erarbeiteten Anforderung an Hilfesysteme (vgl. Anforderung 3-3), anhand derer ein Hilfesystem entworfen und konstruiert werden konnte. Um den Gang der Argumentation und die wichtigsten Aussagen zusammenzufassen, wird im folgenden entlang der Anforderung auf die einzelnen Aspekte des entstandenen Hilfe-Frameworks zurückgeblickt. Dabei wird jeweils ein Teil der Anforderung in bezug auf seinen Beitrag zu den drei Ausgangspunkten betrachtet.

Anforderung 3-3:

> Die Aufgaben eines Hilfesystems bestehen darin, Informationen anzuzeigen, das Auffinden von Informationen zu ermöglichen und den Benutzerinnen Orientierungsmöglichkeiten in der Fülle der Hilfeinformationen zu geben.

Die in der Anforderung genannten Aufgaben konnten im Hilfesystem so umgesetzt werden, daß die Benutzerin des Hilfesystems bei der Arbeit mit einer Applikation unterstützt wird. Die Hilfeinformation wird vom Hilfesystem in Hilfeknotenbrowsern *angezeigt*, die durch das Prinzip der Koordination von wiederverwendbaren Subwerkzeugen beliebig an der Benutzungsschnittstelle angeordnet werden können.

Das *Auffinden von Information* wird im Hilfesystem durch eine Vielzahl von Maßnahmen unterstützt: Verweise im Hypertext, Hilfeknotenklassifikation, Schlagwörter, Kontextsensitivität, Lesezeichen und Kommentare von Benutzerinnen.

Ein Teil der bisher aufgeführten Mittel, nämlich Lesezeichen, Kommentare von Benutzerinnen, die Hilfeknotenklassifikation aber auch die Koordination wiederverwendbarer Subwerkzeuge, werden gleichfalls als *Orientierungsmöglichkeiten* verstanden, die die Fülle der Hilfeinformationen übersichtlicher werden lassen. Als weiteres Mittel, die Orientierung der Benutzerin zu erleichtern, wurde das Hilfesystem mit Historien versehen. Anhand dieser Historie kann die Benutzerin ihren Weg im Hilfetext zurückverfolgen und zu vorher aufgesuchten Hilfeknoten zurückkehren.

> Ein Hilfesystem ist programmtechnisch unabhängig von einer Applikation und bietet dennoch kontextsensitive Hilfe an.

Um das Hilfesystem *programmtechnisch unabhängig* von einer bestimmten Applikation zu entwickeln und trotzdem *dynamische* Hilfe geben zu können, wurde eine abstrakte Schnittstelle, der Aspekt `erklärbar`, für die Zusammenarbeit zwischen Hilfesystem und Applikation eingeführt. Die *dynamische* Hilfe zu einer Applikation konnte realisiert werden, indem der Begriff des Werkzeugzustands für Hilfesituationen gebildet wurde. Mit Hilfe der Schnittstelle und des Werkzeugzustands ist das Hilfesystem in der Lage, kontextsensitiv auf den Zustand der Applikation zu reagieren, ohne mehr als die anonyme Schnittstelle `erklärbar` zu kennen. Die Auswertung des Zustands konnte in den Zustandsinterpreter ausgelagert werden, so daß das Hilfesystem lediglich mit dem Anzeigen von Hilfetexten, mit der Unterstützung beim Auffinden von Information und mit der Bereitstellung von Orientierungsmöglichkeiten beschäftigt ist.

> Ein Hilfesystem besteht aus zwei Komponenten: einem Hilfewerkzeug und einem Hilfematerial. Das Hilfewerkzeug ermöglicht der Benutzerin den Zugriff auf das Hilfematerial, welches Hilfe zu einer Applikation enthält.
>
> Das Hilfewerkzeug bietet passive Hilfe an, da Hilfesituationen nie eindeutig bestimmbar sind. Es stellt dynamische Hilfe zur Verfügung, die aufgrund des Zustands der Anwendung kontextsensitiv ist, und es stellt insofern individuelle Hilfe bereit, als daß es den Benutzerinnen Hilfe zur explorativen Selbsthilfe gibt.

Das Hilfesystem konnte nach den Mustern der *Werkzeug-Material Metapher* entwickelt werden und folgt dem Leitbild des Arbeitsplatzes für qualifizierte menschliche Arbeitstätigkeit. Ausgehend von diesem Leitbild wurde das Hilfewerkzeug derart konzipiert, daß es *passive*, *dynamische* und *individuelle* Hilfe anbietet.

Passivität ist ein charakteristisches Verhalten von Werkzeugen an einem Arbeitsplatz. Werkzeuge drängen sich nicht selbst zum Gebrauch auf, sondern werden von den Benutzerinnen zur Hand genommen und zur Arbeit eingesetzt.

Die *dynamische* Hilfe läßt sich in die Werkzeug-Material Metapher insofern einordnen, daß das Hilfewerkzeug als Werkzeug 2. Ordnung auf einem anderen Werkzeug arbeitet. Mit Hilfe von sondierenden Operationen verschafft sich das Hilfewerkzeug beim Werkzeug der Applikation die benötigten Angaben, um entsprechend seiner Aufgabe zu der Applikation dynamisch Hilfeinformationen anzeigen zu können.

Auch *individuelle* Hilfe ist ein Konzept, das dem Leitbild der Werkzeug-Material Metapher folgt. Genau wie Menschen sich ihren Arbeitsplatz individuell einrichten und besonders das Material auf ihre individuelle Art und Weise bearbeiten, so kann auch die Benutzerin das Hilfewerkzeug individuell einsetzen. Zu diesem Zweck stehen der Benutzerin verschiedene Mechanismen zur Verfügung: Mit Hilfe von Lesezeichen kann eine Benutzerin bestimmte Texteinheiten im Hilfetext markieren, damit die entsprechende Stelle später wiedergefunden werden kann. Des weiteren kann sie die Hilfetexte mit eigenen Kommentare versehen, um so ihr persönliches Verständnis und ihre Erfahrungen mit der Handhabung der Applikation festzuhalten. Schließlich verschafft das Prinzip der Koordination wiederverwendbarer Subwerkzeuge der Benutzerin die Möglichkeit, die einzelnen Subwerkzeuge des Hilfewerkzeug beim Gebrauch individuell zu gruppieren.

> Das Hilfematerial besteht aus kurzen textuellen und visuellen Einheiten, die als Hypermedia organisiert sind. Diese Einheiten erläutern die an der Benutzungsschnittstelle verwendeten Symbole für Gegenstände und Handlungen und stellen sie in eine konsistente Beziehung zum Anwendungsgebiet. Außerdem erklärt das Hilfematerial die in der Applikation verwendeten Metaphern.

Der *symbolische Charakter von Computerartefakten* wurde bei der Entwicklung des Hilfesystems speziell für die Gestaltung der Hilfetexte im Hilfematerial berücksichtigt. Die Hilfetexte erläutern die an der Benutzungsschnittstellen verwendeten *Symbole* für *Gegenstände* und *Handlungen*.

Um diese Vorgabe in den Hilfetexten umzusetzen, wurde für die Hilfeknoten eine Klassifikation in Übersichtsknoten, Werkzeugknoten und Materialknoten eingeführt. Mit Hilfe dieser Klassifikation können Symbole für Gegenstände und Symbole für Handlungen in

getrennten Hilfeknoten erklärt werden und so eine bessere Übersichtlichkeit für die Benutzerin erreicht werden. Weiterhin wurde für Werkzeug- und Materialknoten jeweils ein Schema entwickelt, nach dem der Text der entsprechenden Hilfeknoten gegliedert wird.

Die wirkliche Ausgestaltung der Hilfetexte muß auch zukünftig von den Autorinnen der Hilfetexte selbst vorgenommen werden. Sie haben darauf zu achten, daß die Symbole an der Benutzungsschnittstelle in eine *konsistente* Beziehung zum Anwendungsgebiet gestellt werden und daß die in der Applikation verwendeten *Metaphern* erläutert werden.

Um die Entwicklerinnen bei der Zusammenstellung der Hilfematerialien zu unterstützen, wurden dem Hilfesystem weitere Werkzeuge zur Seite gestellt, die in der *Arbeitsumgebung für Hilfe* zusammengefaßt werden konnten. Alle Komponenten - das Hilfesystem und die Arbeitsumgebung für Hilfe - wurden in dieser Arbeit als *Hilfe-Framework* bezeichnet. Das Hilfe-Framework konnte so entwickelt werden, daß es einer Anwendungsentwicklerin alle hilferelevanten Komponenten zur Verfügung stellt, die benötigt werden, um eine Applikation mit einem Hilfesystem zu verbinden.

Aus der Entwicklung des Hilfe-Frameworks haben sich einige Fragen ergeben, die in dieser Arbeit teilweise beantwortet werden konnten. Als noch offene Fragen lassen sich identifizieren:

- *Arbeit mit den Hilfetexten*: Die Hilfetexte wurden aufgrund von sprachlichen Annahmen nach einem bestimmten Schema geschrieben. Gegenstände auf dem Bildschirm und Handlungen wurden voneinander getrennt erklärt und die einzelnen Hilfeknoten nach einem Grundgerüst (s. Abschnitt 6.2.3) aufgebaut. Ob diese Art, Hilfetexte zu schreiben, für Benutzerinnen Vorteile bringt, läßt sich erst im wirklichen Einsatz des Hilfe-Frameworks feststellen.
- *Hilfetexte aufspalten in Werkzeug- und Materialhilfetext*: Um der Tatsache Rechnung zu tragen, daß Werkzeuge auf verschiedenen zu ihnen passenden Materialien arbeiten und Materialien von verschiedenen Werkzeugen bearbeitet werden, sollten Hilfetexte für Werkzeug und Material getrennt erstellt werden (s. Abschnitt 6.2.4). Ungeklärt ist, wie diese Verbindungen zwischen den Hilfetexten spezifiziert werden sollten, so daß Hilfetexte zusammenpassen und bei Gebrauch einer bestimmten Werkzeug-Material-Kombination entsprechend zusammengesetzt werden können.
- *Zustände von Werkzeugen und Materialien*: Kontextsensitivität bzw. dynamische Hilfe erfordert, daß zwischen Hilfesystem und Applikation eine Verbindung hergestellt werden kann. Da das Hilfesystem als unabhängiges Werkzeug konzipiert wurde, wird eine Abstraktion benötigt, die das Wissen über die Benutzungssituation in der Applikation transportieren kann. Das Hilfe-Framework arbeitet zu diesem Zweck mit dem Zustand der Applikation (s. Abschnitt 7.3.1). Die gefundene Verallgemeinerung für den Werkzeugzustand muß ihren allgemeinen Charakter bei der Zusammenarbeit des Hilfe-Frameworks mit anderen Applikationen als SANE erst noch beweisen.

- *Einsatz des Hilfe-Frameworks im Entwicklungsprozeß*: In der Arbeit wurde die Vermutung ausgesprochen, daß sich das Hilfe-Framework bereits im Entwicklungsprozeß einsetzen läßt (s. Abschnitt 8.5). Bei den im Entwicklungsprozeß erstellten Dokumenten, wie Szenarien, Glossare und Visionen, handelt es sich i.d.R. um Hypertextdokumente. Das Hilfe-Framework arbeitet ebenfalls auf der Basis von Hypertextdokumenten, so daß es die Entwicklungsdokumente darstellen könnte. Das Hilfe-Framework könnte darüber hinaus insbesondere während des Entwicklungsprozesses eingesetzt werden. Die Benutzerinnen kommen bei Prototyping mit einer neu entwickelten Applikation in Kontakt und können bei der Arbeit mit der Applikation Einsicht in die ihnen bekannten Dokumente nehmen. Auf diese Weise können die Entwicklungsdokumente und die Applikation verglichen und auf ihre Stimmigkeit hin evaluiert werden.

Neben diesen offenen Fragen bleibt zu klären, ob sich das Hilfe-Framework, wie es hier vorgestellt wurde, bewähren wird. Der erste Schritt auf diesem Weg wird im Herbst 1995 getan, wenn erneut Studierende mit SANE arbeiten und das Hilfesystem benutzen sollen. Dieser Einsatz in der universitären Praxis wird Einsicht in die Brauchbarkeit der entwickelten Konzepte geben und einige der aufgeworfenen Fragen klären können. Ein weiterer Schritt sollte darin bestehen, das Hilfe-Framework mit anderen Applikationen als SANE zusammenarbeiten zu lassen. Bei diesem Einsatz wird sich zeigen, inwieweit das Hilfe-Framework von SANE unabhängig und allgemein verwendbar ist.

Anhang

Literaturverzeichnis

[AarCar86a] A. Aaronson, J.M. Carroll: *The Answer is in the Question: A Protocol Study of Intelligent Help*, IBM Research Report, IBM Watson Research Center, Yorktown Heights, N.Y. 1986

[AarCar86b] A. Aaronson, J.M. Carroll: *Intelligent Help in a One-Shot Dialog, A Protocol Study*, IBM Research Report, IBM Watson Research Center, Yorktown Heights, N.Y. 1986

[AarCar87] A. Aaronson, J.M. Carroll: *Understanding Intelligent Help: A Protocol Analysis of Advisory Interactions*, IBM Research Report, IBM Watson Research Center, Yorktown Heights, N.Y. 1987

[AarCar89] A. Aaronson, J.M. Carroll: *Learning by Doing with Simulated Intelligent Help*, in: E. Barrett (Hrsg.): The Society of text, The MIT Press, Cambridge Massachusetts 1989

[BBLSSZ94] D. Bäumer, W.R. Bischofsberger, H. Lichter, M. Schneider-Hufschmidt, V. Sedlmeier-Scholz, H. Züllighoven: *Prototyping von Benutzungsoberflächen*, SWT-Schriftenreihe Nr. 2, 1994

[BKKZ92] R. Budde, K. Kautz, K. Kuhlenkamp, H. Züllighoven: *Prototyping*, Springer-Verlag, Berlin, Heidelberg, New York, Tokio 1992

[BäGrZü95] D. Bäumer, G. Gryczan, H. Züllighoven: *Objektorientierte Softwareentwicklung für Banken - Methodik und Erfahrungen aus einer mehrjährigen Projektpraxis*, OBJEKTspektum 3/95, Mai, Juni 1995, S. 45-53

[Bauer88] J. Bauer: *Konzepte und Prototypen interaktiver Hilfesysteme*, Universität Stuttgart, Fakultät Mathematik und Informatik, Dissertation, Januar 1988

[BauSch88] J. Bauer, T. Schwab: *Anforderungen an Hilfesysteme*, in: H. Balzert (Hrsg.): Einführung in die Software-Ergonomie, W. de Gruyter, Berlin, New York 1988, S. 197-214

[BudZül90] R. Budde, H. Züllighoven: *Software-Werkzeuge in einer Programmierwerkstatt*, Bericht der GMD, Nr. 182, Oldenbourg 1990

[BüGrZü92] U. Bürkle, G. Gryczan, H. Züllighoven: *Erfahrungen mit der objektorientierten Vorgehensweise bei einem Bankenprojekt*, Informatik-Spektrum Nr. 15, Springer-Verlag 1992, S. 273-281

[CaMaKe87] J.M. Carroll, R.L. Mack, W.A. Kellogg: *Interface Metaphors and User Interface Design*, IBM Research Report, IBM Watson Research Center, Yorktown Heights, N.Y. 1987

[CarMac82] J.M. Carroll, R.L. Mack: *Metaphors, Computing Systems, and Active Learning*, IBM Research Report, IBM Watson Research Center, Yorktown Heights, N.Y. 1982

[CGLMR85] J.M. Carroll, N.L. Grischkowsky, C.H. Lewis, R.L. Mack, S.R. Robertson: *Exploring a word processor*, in: Human-Computer Interaction Vol. 1, Lawrence Erlbaum Associates 1985

[CoHaBe92] D. Coleman, F. Hayes, S. Bear: *Introducing Objectcharts or How to Use Statecharts in Object-Oriented Design*, IEEE Transactions on Software Engineering, Vol. 18, Nr. 1, Januar 1992, S. 9-18

[DuPaMe92] T.M. Duffy, J.E. Palmer, B. Mehlenbacher: *On Line Help, Design and Evalution*, Adlex Publishing Corporation, Norwood, New Jersey 1992

[Dutke94] S. Dutke: *Mentale Modelle: Konstrukte des Wissens und Verstehens, Kognitionspsychologische Grundlagen für die Software-Ergonomie*, Verlag für Angewandte Psychologie, Göttingen, Stuttgart 1994

[FloGeb91] K. Floegel, M. Gebbert: *Benutzergerechte Gestaltung von Kurzanleitungen am Beispiel des Textverarbeitungssystems MS-WORD*, Universität Hamburg, Fachbereich Informatik, Studienarbeit, Februar 1991

[Floyd87] C. Floyd: *Outline of a Paradigm Change in Software Engineering*, in: G. Bjerknes, P. Ehn, M. Kyng: Computer and Democracy, Avebury, Grower Publishing Company Limited, Aldershot, Brookfield USA, Hong Kong, Singapore, Sydney 1987

[Floyd92] C. Floyd: *STEPS-Projekthandbuch*, Universität Hamburg, Fachbereich Informatik, Hamburg 1992

[Floyd94a] C. Floyd: *Evolutionäre Systementwicklung und Wandel in Organisationen*, Der GMD-Spiegel 3/94, September 1994, S. 36-40

[Floyd94b] C. Floyd: *Software-Engineering - und dann?*, Informatik Spektrum, Band 17, Heft 1, 1994, S. 29-37

[FlReSc89] C. Floyd, F.-M. Reisin, G. Schmidt: *STEPS to Software Development with Users*, in: C. Ghezzi, J.A. McDermid: ESEC'89, Lecture Notes in Computer Science No. 387, Springer Berlin Heidelberg 1992, S. 48-64

[FlZüBuKS92] C. Floyd, H. Züllighoven, R. Budde, R. Keil-Slawik (Hrsg.): *Software Development and Reality Construction*, Springer Verlag, Berlin 1992

[GaHeJoVl95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison-Wesley Publishing Company, Reading, Massachusetts 1995

[Gamma92] E. Gamma: *Objektorientierte Software-Entwicklung am Beispiel von ET++, Design-Muster, Klassenbibliothek, Werkzeuge*, Springer-Verlag, Berlin, Heidelberg 1992

[GryZül92] G. Gryczan, H. Züllighoven: *Objektorientierte Systementwicklung: Leitbild und Entwicklungsdokumente*, Informatik Spektrum, Band 15, Heft 5, Oktober 1992, S. 264-272

[Harel87] D. Harel: *A Visual Formalism For Complex Systems*, Science of Computer Programming 8, 1987, S. 231-274

[HarSmi88] J.R. Hartley, M. Smith: *Question answering and explanation giving in on-line help systems*, in: J. Self (Hrsg.): Artifical intelligence and human learning:

Intelligent computer-aided instruction, Chapman and Hall, London 1988, S. 338-360

[HenCar86] D.A. Henderson, S.K. Card: *Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface,* ACM Transactions on Graphics, Vol. 5, Nr. 3, Juli 1986, S. 211-243

[HenRod95] C. Henkel, B. Rodeck: *Erstellung eines Hypermedia-Ausstellungsführer*, Universität Hamburg, Fachbereich Informatik, Diplomarbeit, Mai 1995

[Heß93] H. Heß: *Grundsteine für eine STEPS-Werkzeugumgebung*, Universität Hamburg, Fachbereich Informatik, Diplomarbeit, Juni 1993

[Herrmann86] T. Herrmann: *Zur Gestaltung der Mensch-Computer-Interaktion: Systemerklärung als kommunikatives Problem*, Max Niemeyer Verlag, Sprache und Information Bd. 14, Tübingen 1986

[Herrmann88] T. Herrmann: *Probleme bei der Konstruktion und beim Einsatz von Hilfesystemen*, in: H. Balzert (Hrsg.): Einführung in die Software-Ergonomie, W. de Gruyter, Berlin, New York 1988, S. 215-228

[Herrmann92] T. Herrmann: *Verständigungsprobleme und ihre Bedeutung für die computergestützte Vermittlung von Erklärungen*, in: H. Stoyan (Hrsg.): Erklärung im Gespräch - Erklärung im Mensch-Maschine-Dialog, Proceedings, Springer-Verlag, Berlin, Heidelberg 1992

[Hoffmann88] S. Hoffmann: *Adaptierbarkeit und Adaptivität von Hilfesystemen*, Universität Hamburg, Fachbereich Informatik, Diplomarbeit, Februar 1988

[Horton90] W.K. Horton: *Designing and Writing Online Documentation: Help files to hypertext*, John Wiley and Sons, Inc., New York 1990

[HügLüb91] A. Hügli, P. Lübcke (Hrsg.): *Philosophie Lexikon*, Rowohlt Verlag GmbH, Reinbek bei Hamburg 1991

[Jacobson92] I. Jacobson: *Object-Oriented Software Engineering - A Use Case Driven Approach*, Addison-Wesley, 1992

[Kearsley88] G. Kearsley: *Online help systems: design and implementation*, Adlex Publishing Corporation, Norwood 1988

[Keil-Slawik89] R. Keil-Slawik: *Systemgestaltung mit Aufgabennetzen*, S. Maaß, H. Oberquelle (Hrsg.): Software Ergonomie '89, Teubner, Stuttgart 1989, S. 123-133

[Kellogg87] W.A. Kellogg: *Conceptual Consistency in the User Interface. Effects on User Performance*, in: H.-J. Bullinger, B. Shackel (Hrsg.): Proc. Human-Computer Interaction, INTERACT '87, Elsevier Science (North-Holland), Amsterdam 1987, S. 289-349

[KiGrZü94] K. Kilberth, G. Gryczan, H. Züllighoven: *Objektorientierte Anwendungsentwicklung - Konzepte, Strategien, Erfahrungen*, Vieweg, Braunschweig/Wiesbaden, 2. überarbeitete Auflage 1994

[KleLyy92] H. Klein, K. Lyythinen: *Toward a New Understanding of Data Modelling*, in: [FlZüBuKS92]

[Krämer92] S. Krämer: *Symbolische Maschinen, Computer und der Verlust des Ethischen im geistigen Tun*, in: W. Coy, F. Nake, J.-M. Pflüger, A. Rolf, J. Seetzen, D. Siefkes, R. Stransfeld (Hrsg.): Sichtweisen der Informatik, Theorie der Informatik, Vieweg, Braunschweig/Wiesbaden 1992, S. 335-341

[Krämer93] S. Krämer: *Operative Schriften als Geistestechnik. Zur Vorgeschichte der Informatik*, in: P. Schefe, H. Hastedt, Y. Dittrich, G. Keil (Hrsg.): Informatik und Philosophie, BI-Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich 1993, S. 69-83

[LakJoh80] G. Lakoff, M. Johnson: *Metaphors we live by*, The University of Chicago Press, Chicago und London 1980

[Lehman87] M.M. Lehman: *Programs, Life Cycles, and Laws of Software Evolution*, in: Proceedings of the IEEE, Vol. 68, No. 9, September 1980, S. 1060-1076

[MaaObe92] S. Maaß, H. Oberquelle: *Perspektives and Metaphors for Human-Computer Interaction*, in: [FlZüBuKS92]

[Maaß94] S. Maaß: *Transparenz - eine zentrale Software-ergonomische Forderung*, Universität Hamburg, Fachbereich Informatik, Bericht Nr. 170

[MacCormac85] E.R. MacCormac: *A Cognitive Theory of Metaphor*, A Bradford Book, The MIT Press, Cambridge, Massachusetts, London 1985, Third Printing 1990

[Meyer91] B. Meyer: *Design by Contract*, in: D. Mandrioli, B. Meyer (Hrsg.): Advances in Object-Oriented Software Engineering, Prentice-Hall, London 1991, S. 1-50

[Nake92] F. Nake: *Informatik und die Maschinisierung von Kopfarbeit*, in: W. Coy, F. Nake, J.-M. Pflüger, A. Rolf, J. Seetzen, D. Siefkes, R. Stransfeld (Hrsg.): Sichtweisen der Informatik, Theorie der Informatik, Vieweg, Braunschweig/Wiesbaden 1992, S. 181-201

[NakSch93] F. Nake, H. Schelhowe: *Vom instrumentalen Medium. Kooperation in der Software-Entwicklung unter konfliktierenden Leitbildern*, Abschlußbericht des Projektes "Kooperative Gestaltung informationstechnischer Arbeitsmittel für kooperatives Arbeiten: Neuartige Zugänge zu Literaturbeständen", artec-paper 26. Juli 1993

[Naur85] P. Naur: *Programming as Theory Building*, Microprocessing and Microprogramming, Nr. 15, North-Holland Publishing Company 1985, S. 253-261

[Nievergelt83] J. Nievergelt: *Die Gestaltung der Mensch-Maschine-Schnittstelle*, in: I. Kupka (Hrsg.): GI - 13. Jahrestagung, Hamburg 3.-7. Oktober 1983, Proceedings, Springer, Berlin 1983, S. 41-50

[Norman89] D. Norman: *The Psychology of Everyday Things*, Basic Book 1988

[Oberquelle94] H. Oberquelle: *Situationsbedingte und benutzerorientierte Anpaßbarkeit von Groupware*, in: A. Hartmann, T. Herrmann, M. Rohde, V. Wulf (Hrsg.): Menschengerechte Groupware - Software-ergonomische Gestaltung und partizipative Umsetzung, Bericht des German Chapter of the ACM, 42, Teubner, Stuttgart 1994, S. 31-49

[Piepenburg94] U. Piepenburg: *Semantikerstellung in der Softwareentwicklung, Qualitätszirkel in der Softwareentwicklung*, Universität Hamburg, Fachbereich Informatik, Dissertation 1995

[Price89] J. Price: *Creating a Style for Online Help*, in: E. Barrett (Hrsg.): Text, ConText, and HyperText, Writing with and for the Computer, The MIT Press, Cambridge, Massachusetts, London, England 1989, 3. Auflage

[RaeVel95] A. Raeithel, B.M. Velichkovsky: *Joint Attention and Co-Construction*, in: B. Nardi (Hrsg.): Context and Consciousness, Activity Theory and Human Computer Interaction, MIT Press, Boston 1995

[Riehle93] D. Riehle: *Objektorientierte Architektur nach der Werkzeug-Material Metapher am Beispiel eines grafischen Aufgabennetzeditors*, Universität Hamburg, Fachbereich Informatik, Studienarbeit, August 1993

[Riehle95] D. Riehle: *Muster am Beispiel der Werkzeug und Material Metapher*, Universität Hamburg, Fachbereich Informatik, Diplomarbeit, März 1995

[RieZül94a] D. Riehle, H. Züllighoven: *A Pattern Language for Tool Construction and Integration Based on the Tools&Material Metaphor*, in: Proceedings of the First Conference on Pattern Languages of Programs(PLOP'94), Monticello, Illinois, August 1994, Paper B

[RieZül94b] D. Riehle, H. Züllighoven: *Späte Erzeugung*, in: E. Köhler (Hrsg.): 39. Internationales Wissenschaftliches Kolloquium 27.-30.9.1994, Band 1, Technische Universität Illmenau

[RoCaMa93] G.G. Robertson, S.K. Card, J.D. Mackinlay: *Information visualization using 3D interactive animation*, ACM, Vol. 36, Nr. 4, April 1993, S. 57-71

[Robinson93] M. Robinson: *Design for unanticipated use.....*, in: G. De Michelis, C. Simone, K. Schmidt (Hrsg.): Proceedings of the Third European Conference on Computer-Supported Cooperative Work, 13.-17. September 1993, Milan, Italien

[Rumbaugh95] J. Rumbaugh: *OMT: The Object Modell*, Journal of Object-Oriented Programming 7, 8 (Januar 1995), S. 21-27

[Rupietta87] W. Rupietta: *Benutzerdokumentation für Softwareprodukte*, BI-Wissenschaftsverlag, Mannheim, Wien, Zürich 1987

[Sniff+94] *User's Guide* zur Entwicklungsumgebung Sniff+, Version 2.0 beta 3, Take-Five Software GmbH, Salzburg 1994

[Suchman87] L.A. Suchman: *Plans and situated actions, The problem of human-machine communication*, Cambridge University Press, Cambridge 1987

[SzeNec94] P. Szekely, R. Neches*: Automatic Generation of Help from Interface Design Models*, in: B. Adelson, S. Dumais, J. Olson (Hrsg.): CHI-94, Conference Proceedings, ACM Press 1994

[Thies94] M.A. Thies: *Planbasierte Hilfeverfahren für direkt-manipulative Systeme: Erkennung, Vervollständigung und Visualisierung von Interaktionsplänen*, Universität Stuttgart, Fakultät Informatik, Dissertation, Dr. Ekkehard Hundt, "infix", Sankt Augustin 1994

[Willms92] S. Willms: *Gestaltung von Hilfesystemen durch Hypertext*, Fachbereich Informatik, Universität Hamburg, Diplomarbeit, Oktober 1992

[Wulf95] M. Wulf: *Konstruktion und Realisierung einer Softwareumgebung zur Koordination rechnergestützter Tätigkeiten in kooperativen Prozessen*, Universität Hamburg, Fachbereich Informatik, voraussichtliche Veröffentlichung im August 1995

[Yetim91] F. Yetim: *Eine Hypertext-Komponente zu einem Expertensystem: Benutzerfragen für Erklärungsdialoge*, in: H. Maurer (Hrsg.): Hypertext/Hypermedia '91, Tagung der GI, SI und OCG, 27./28.5.1991, Proceedings, Springer-Verlag, Berlin 1991

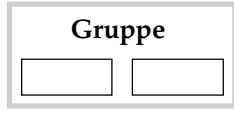
[Zuboff88] S. Zuboff: *In the age of the smart maschine, the future of work and power*, Heinemann Professional Publishing, Oxford 1988

[Züllighoven92] H. Züllighoven*: Umgang mit Software oder Software als Werkzeug und Material*, in: W. Coy, F. Nake, J.-M. Pflüger, A. Rolf, J. Seetzen, D. Siefkes, R. Stransfeld (Hrsg.): Sichtweisen der Informatik, Theorie der Informatik, Vieweg, Braunschweig/Wiesbaden 1992, S. 141-156

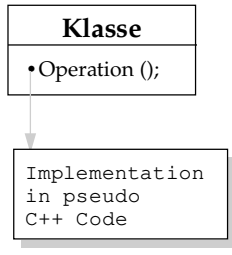
Graphische Notation

Die graphische Notation für objektorientierte Entwurfsdiagramme basiert auf der OMT-Notation [Rumbaugh95] und ist im folgenden dargestellt:

Ein Rechteck steht für eine einzelne Klasse. Ist der Name *kursiv* gesetzt, so ist die Klasse abstrakt.

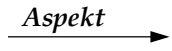


Klassen können logisch zu einer Gruppe von Klassen zusammengefaßt werden. Eine Gruppe wird durch einen grauen Rahmen dargestellt.

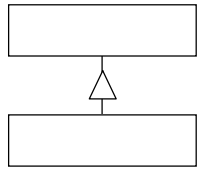


Operationen einer Klasse werden unterhalb des Klassennamens, mit einem Aufzählungszeichen versehen, vermerkt. Die Implementation wird in einem separaten Kasten angezeigt und ist durch einen Pfeil mit der zugehörigen Operation verbunden.

Ein Rechteck mit abgerundeten Ecken verkörpert das Exemplar einer Klasse zur Laufzeit.

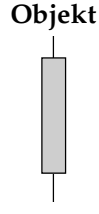


Benutzt-Beziehungen werden durch einen Pfeil repräsentiert. Wenn nötig, wird der Aspekt der Benutzung am Pfeil vermerkt.

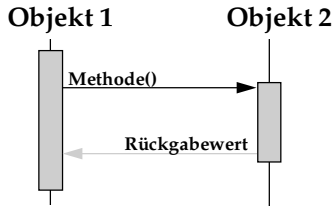


Ein Strich steht für eine Vererbungsbeziehung. Das Dreieck zeigt mit seiner Spitze auf die Oberklasse.

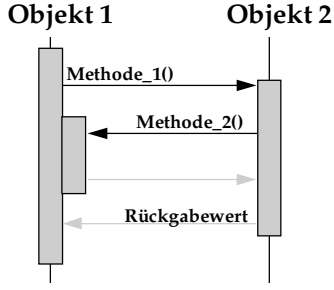
Die graphische Notation, um die Dynamik von objektorientierten Systemen darzustellen, basiert auf den Interaktionsdiagrammen von Jacobson [Jacobson92]. Die Interaktionsdiagramme wurden für die Darstellung objektorientierter Zusammenhänge um einige andere graphische Ausdrucksformen erweitert. Grundsätzlich verläuft die Zeitachse bei Interaktionsdiagrammen von oben nach unten, so daß in einem Interaktionsdiagramm Folgen von "Ereignissen" abgebildet werden können:



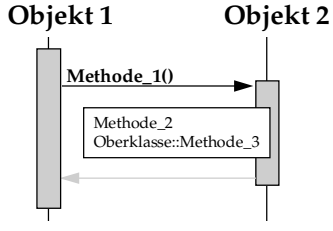
Ein Objekt einer Klasse wird mit seinem Namen über einer Linie gekennzeichnet. Auf und entlang der Linie werden alle Zeitabschnitte durch einen Balken vermerkt, in denen das Objekt mit der Durchführung bestimmter Aufgaben beschäftigt ist.



Interaktion zwischen zwei Objekten wird mit Hilfe von Pfeilen ausgedrückt. Ein schwarzer Pfeil steht für einen Methodenaufruf von Objekt 1 bei Objekt 2. Am Pfeil wird der Methodenname vermerkt. Ist die Methode bei Objekt 2 abgeschlossen, so wird der Kontrollfluß an Objekt 1 zurückgegeben. Der Kontrollfluß wird durch einen grauen gestrichelten Pfeil verdeutlicht. Gibt die Methode einen Wert zurück, wird er an dem entsprechenden Pfeil notiert.



Ruft Objekt 2 Objekt 1 rekrusiv auf (Methode_2), so wird der rekursive Aufruf durch einen zweiten Balken neben dem Balken eingezeichnet, der für die andere, nicht beendete Methode im rekursiv aufgerufenen Objekt steht.



Werden bei der Ausführung einer Methode andere Methoden des Objekts oder Methoden der Oberklassen gerufen, so können diese Methoden zur Verdeutlichung des Ablaufs in einem Kasten an dem entsprechenden Balken vermerkt werden. Statt der Methodennamen können auch Beschreibungen von umfassenderen Abläufen, die aus mehreren Methodenaufrufen bestehen, in dem Kasten aufgeführt werden.