In: Maaß, S. & Oberquelle, H. (Hrsg.): Software-Ergonomie '89: Aufgabenorientierte Systemgestaltung und Funktionalität. Stuttgart : B.G. Teubner. S. 254-263

USIT
Ein Benutzerschnittstellen-Baukasten für ein Interaktionskontinuum

Michael Herczeg, Backnang

Kurzfassung

Die Entwicklung ergonomischer Benutzerschnittstellen ist bislang eine umfangreiche und komplizierte Aufgabe. Herkömmliche Programmierhilfsmittel zur Generierung von Benutzerschnittstellen (z.B. Maskengeneratoren) führen zu unflexiblen und monotonen Bedienoberflächen. Deshalb wurde ein objektorientierter Benutzerschnittstellen-Baukasten entwickelt, der es erlaubt, anwendungs- und benutzerspezifische Interaktionstechniken bereitzustellen. Aus Elementarbausteinen können mit wenig Aufwand komplexe Benutzerschnittstellen-Objekte generiert werden, die als Softwarebausteine in Anwendungsprogrammen oder als Grundlage neuer Bausteine verwendet werden können. Auf diese Weise gelangt man zu beliebig spezifischen Interaktionstechniken.

1 Motivation

Die Entwicklung von Benutzerschnittstellen gestaltet sich für den Systementwickler als umfangreiche und komplexe Aufgabe. Die Gründe dafür sind vor allem:

- Die Anforderungen an Benutzerschnittstellen sind nicht formal spezifizierbar und ändern sich schnell.
- Die potentiellen Benutzer eines Anwendungssystems haben bezüglich der Anwendung unterschiedlichen Kenntnisstand, mehr oder weniger Erfahrung in der Bedienung interaktiver Systeme sowie individuelle sensorische und motorische Fähigkeiten.
- Die Programmkonstrukte zum Bau von Benutzerschnittstellen sind sehr elementar. Sie bestehen oft aus primitiven Ein- und Ausgabeanweisungen einer Programmiersprache oder aus den Sprachelementen eines Graphiksystems.
- Die dezentrale Programmierung von Ein- und Ausgabevorgängen führt zu unübersehbar inkonsistenten Benutzerschnittstellen. Ähnliche Interaktionen werden von verschiedenen Programmteilen abgewickelt.

- Endgeräte variieren sehr stark in ihren Fähigkeiten und Schnittstellen. Standards hinken den technischen Möglichkeiten meist um fünf bis zehn Jahre hinterher. Zur Erhöhung der Portierbarkeit beschränkt man sich auf Minimalstandards.
- Solange Benutzerschnittstellen von den Anwendungsprogrammierern "miterstellt" werden, bildet sich keine Expertise zur Erstellung ergonomischer und effizienter Benutzerschnittstellen. Die Entwicklung der Benutzerschnittstelle wird als eher lästiges und nebensächliches Beiwerk behandelt.

Aus all diesen Problemen resultieren oftmals Implementierungen deren Unzulänglichkeiten vor allem zu Lasten der Endbenutzer gehen.

Eine erste Verbesserung dieser Situation erfolgte durch die Entwicklung von Menü- und Maskengeneratoren zur Erstellung stark standardisierter Benutzerschnittstellen. Der Softwareerstellungsprozeß ließ sich durch diese Programmierhilfsmittel deutlich beschleunigen. Die Konsistenz damit erstellter Benutzerschnittstellen war im wesentlichen garantiert. Der Preis für diese Methodik war, neben dem Aufwand zur Erstellung der Hilfsmittel, die von da ab vorhandene Dominanz unmotiviert uniformer, geradezu monotoner Benutzerschnittstellen. Derartige Hilfsmittel prägen heute das äußere Erscheinungsbild vieler Anwendungssysteme. Die Benutzer plagen sich durch starre und unübersichtliche Menü- und Maskenhierarchien, die nicht durch die Anwendungen, sondern durch die Programmierhilfsmittel geprägt wurden.

Durch Systeme wie dem Xerox STAR Bürosystem [Seybold 81,Smith/Irby et al. 82] wurden neue Designmöglichkeiten demonstriert, die heute in vielen – wenn auch nicht allen – Details als richtungsweisende Lösungen bei der Gestaltung ergonomischer Anwendungssysteme anzusehen sind. Hierbei wurden Menü- und Maskendialoge durch vielfältige, direkt manipulative, graphische Oberflächenmodellierungen angereichert, die bis hin zur Nachbildung manueller Arbeitsumgebungen (z.B. Desktop-Metapher) reichen [Shneiderman 83,Hutchins/Hollan/Norman 86].

Leider bedeuteten diese direkt manipulativen Systeme im Vergleich zu den Menü- und Maskengeneratoren zunächst einen softwaretechnischen Rückschritt. Sie erforderten einen äußerst hohen Programmieraufwand. Die Konstruktion des Xerox STAR Systems erforderte mehr als 100 Personenjahre Entwicklungs- und Programmieraufwand [Lipkie/Evans et al. 82]. Inzwischen hat man durch die Bereitstellung von Programmierhilfsmitteln den Erstellungsaufwand für derartige Systeme wieder deutlich reduzieren können. Beispiele für solche Hilfsmittel sind **Benutzerschnittstellen-Baukästen**, von denen im folgenden die Rede sein soll.

Aber auch diese Baukästen sind noch immer mit Problemen behaftet. Ihre Bausteine schränken die Gestaltung von anwendungsgerechten Benutzerschnittstellen noch zu stark ein. Immer wieder erzwingen sie den Kompromiß zwischen umständlichen Interaktionstechniken oder dem aufwendigen Bau neuer, anwendungs- und benutzerspezifischer Bausteine. Mit dem im folgenden beschriebenen Baukasten *USIT* lassen sich solche spezifischen Bausteine aus einigen Elementarkonstrukten recht einfach erstellen. Ein solches Konzept macht von standardisierten Bausteinen unabhängig und erlaubt die wirtschaftliche, anwendungs- und benutzergerechte Erstellung von nahezu beliebig spezifischen Interaktionstechniken.

2 Der Benutzerschnittstellen-Baukasten *USIT*

Im folgenden wird der im Rahmen des WISDOM-Projekts[1] erstellte Benutzerschnittstellen-Baukasten *USIT (User Interface Toolkit)* vorgestellt [Herczeg 88]. Der Baukasten diente bereits zur Realisierung einer Reihe von Prototypen. Das Spektrum der mit *USIT* erstellbaren Systeme reicht von tradionellen menü- und formularorientierten Lösungen bis zu anwendungsspezifischen, direkt manipulativen, graphischen Bedienoberflächen.

2.1 Definition von Interaktionstechniken

Eine Interaktionstechnik beschreibt zwei grundlegende, komplementäre Abläufe. Zum einen müssen auf den Ausgabegeräten Daten und Funktionen des Anwendungssystems präsentiert werden. Die Ausgaben müssen der Anwendung und dem Benutzer angemessen aufbereitet sein. Dies können Darstellungen auf Bildschirmen sein, wie Texte, Bitmuster, Linien, Flächen und räumliche Graphiken oder auch akustische oder tastbare Ausgaben. Zum anderen müssen Eingaben des Benutzers erfaßt werden, die ebenfalls über unterschiedliche periphere Geräte aufgenommen werden können.

Eine **Interaktionstechnik** definiert, welche Ausgaben erzeugt und welche Eingaben erfaßt werden und wie diese sequenziert und einander zugeordnet werden. Ein **Benutzerschnittstellen-Baustein** ist ein Softwaremodul zur Implementierung einer solchen Interaktionstechnik. Ein Benutzerschnittstellen-Baukasten muß die für die vorgesehenen Anwendungen benötigten Bausteine enthalten. Es hat sich in unserer Arbeit bei der Entwicklung von mehr als 50 hochinteraktiven Prototypen in den vergangenen ca. 6 Jahren herausgestellt, daß für die ergomische Oberflächengestaltung neuer Anwendungen erfahrungsgemäß immer wieder neue, spezielle Interaktionstechniken benötigt werden. Der Baukasten muß also möglichst einfach um völlig neue Bausteine erweiterbar sein; existierende Bausteine sollten zu neuen Bausteinen verfeinert werden können.

Auf diese Weise entstehende anwendungsspezifische Interaktionstechniken sind oft nicht mehr einfach klassifizierbar. Es gibt viele Mischformen, beliebig hinzu- oder wegnehmbare Details und eine praktisch beliebige Anzahl von einstellbaren Eigenschaften. Wir haben es daher nicht mehr mit klar abgrenzbaren Interaktionstechniken zu tun, sondern mit einem vieldimensionalen, nicht diskretisierbaren Designraum, für den ich den Begriff **Interaktionskontinuum** vorschlage.

Vollzieht man diesen konzeptionellen Schritt von standardisierten zu beliebig spezifischen Interaktionstechniken, so stellt man auch die bislang intensiv verfolgten Klassifizierungsschemata für Dialoge in Frage. Daß man in den letzten zwei Jahrzehnten bei diesen Kategorisierungsbemühungen zu keinem einigermaßen befriedigenden und stabilen Resultat gekommen ist, scheint diese neue Sichtweise zusätzlich zu rechtfertigen.

[1]Die beschriebene Arbeit ist im Rahmen der vom Bundesminister für Forschung und Technologie sowie von der Triumph Adler AG geförderten Forschungsgruppe INFORM am Institut für Informatik der Universität Stuttart entstanden.

2.2 Repräsentation von Interaktionstechniken

Bei der Realisierung eines Benutzerschnittstellen-Baukastens ist insbesondere auf die folgenden Anforderungen zu achten:

Modularisierbarkeit: Die Softwarebausteine sollten nicht nur konzeptionell sondern auch programmtechnisch als Einheiten betrachtet werden können, die einfach änderbare Attribute und klare Schnittstellen nach außen besitzen. Sie werden im folgenden auch **Interaktionsobjekte** genannt.

Spezialisierbarkeit: Vorhandene Bausteine sollten zur begrenzten Änderung ihrer Präsentation oder ihres Verhaltens einfach zu neuen Bausteinen spezialisierbar sein.

Aggregierbarkeit: Vorhandene Bausteine sollten zu komplexeren Bausteinen zusammengesetzt werden können. Diese neuen, komplexen Bausteine sollten wieder als Einzelelemente im Sinne der oben genannten Modularisierbarkeit betrachtet und benutzt werden können.

Erweiterbarkeit: Ein Benutzerschnittstellen-Baukasten sollte als offenes System konzipiert werden, das Hilfsmittel bereitstellt, neue Bausteine herzustellen, die nicht durch Spezialisierung oder Aggregation vorhandener Bausteine erzeugt werden können.

Zur Implementierung von Benutzerschnittstellen-Baukästen bietet sich die Verwendung einer **objektorientierten Programmiersprache** an [Herczeg 86]. Nahezu alle fortgeschrittenen Baukastensysteme wurden mittels objektorientierter Programmiersprachen implementiert oder, wenn eine solche nicht zur Verfügung stand, streng objektorientiert konzipiert [Lipkie/Evans et al. 82,Goldberg/Robson 83,Sym 86].

Die Gründe dafür liegen in den folgenden Eigenschaften objektorientierter Modellierungstechniken:

- Objekte sind Einheiten mit änderbaren Attributen (Slots) und funktionalen Verhaltensbeschreibungen (Methoden). Es ist sehr natürlich, Bausteine durch Objekte zu beschreiben.
- Die Kommunikation mit Objekten erfolgt ausschließlich mittels Botschaften. Die zulässigen Botschaften (die Schnittstellendefinitionen) werden durch die Festlegung von Methodenfiltern beschrieben. Eine derartige Schnittstellendefinition ist gut lokalisiert und damit bei Bedarf leicht einsehbar und änderbar.
- Klassen beschreiben Mengen gleichartiger Objekte durch die Definition ihrer Attribute und Funktionalität. Sie dienen gleichzeitig als Generatoren von Bausteinen (Instantiierung). Durch die Änderung von Klassenattributen (Klassenslots) kann das Verhalten einer ganzen Menge von existierenden Bausteinen dynamisch, d.h. auch zur Laufzeit, geändert werden.

- Klassen können in Vererbungsverbände eingeordnet werden. Dadurch ererben Klassen die Attribute und die Funktionalität ihrer Superklassen. Diese ererbten Eigenschaften können von der erbenden Klasse erweitert oder redefiniert werden können. Auf diese Weise lassen sich redundanzarm mehr oder weniger spezialisierte Bausteine oder Bausteinfamilien definieren.

Der Benutzerschnittstellen-Baukasten *USIT* wurde unter Ausnutzung der genannten Mechanismen in der objektorientierten Programmiersprache *ObjTalk* basierend auf *Common Lisp* [Steele Jr. 84] implementiert [Rathke 86,Girgensohn/Rathke 88].

2.3 Uniforme Repräsentation von Interaktionstechniken

Bei der Entwicklung unterschiedlicher Benutzerschnittstellen-Bausteine für *USIT* über einen Zeitraum von mehr als fünf Jahren haben sich immer wieder Gemeinsamkeiten in der Attributierung und der Struktur von Interaktionstechniken abgezeichnet. Es zeigte sich desweiteren die Notwendigkeit, vielerlei Mischformen von Interaktionstechniken bereitzustellen, wie zum Beispiel Formulare, die anstatt eines oder mehrerer Texteingabefelder graphische Menüs anbieten.

Zur Lösung dieser Anforderungen wurde unter den Benutzerschnittstellen-Baukasten *USIT* ein weiteres Baukastensystem gelegt, das Elementarbausteine zum Aufbau von Interaktionsobjekten bereitstellt. Die wichtigsten Elementarbausteine sind Präsentationsobjekte, Fensterobjekte, Eingabeparser, Layouter, Syntaxcontroller und Dialogprotokolle. Durch die einheitliche Verwendung solcher Elementarbausteine sowie durch deren vordefiniertes Standardverhalten (Defaults) wird eine hohe Grundkonsistenz der damit aufgebauten Benutzerschnittstelle erreicht. Bei der Realisierung unkonventioneller Interaktionstechniken wirkt dies allerdings nicht als Barriere, da alle Standardeinstellungen durch die Angabe alternativer Beschreibungen einfach geändert oder verfeinert werden können.

Durch diese feinkörnige Architektur ist es möglich, sehr spezielle Interaktionstechniken zu spezifizieren. Mit einem objektorientierten Beschreibungsformalismus lassen sich die gewünschten Interaktionsobjekte auf uniforme Weise zu einer Klasse konfigurieren. Durch die Instantiierung einer solchen Klasse entstehen dann aktivierbare Inkarnationen der neuen Interaktionstechnik. Eine derartige Konfigurierbarkeit von Interaktionstechniken verhindert auf vordefinierte Interaktionstechniken festgelegt zu sein und öffnet gewissermaßen den Weg zu einem **Interaktionskontinuum**, ohne die Möglichkeit des Vordefinierens häufig benötigter, singulärer Interaktionstechniken zu nehmen. Gerade die Menge der Standardinteraktionstechniken wie beispielsweise Textmenüs oder einfache Formulare werden den Kern eines Benutzerschnittstellen-Baukastens bilden, der je nach Anwendungsbereich und Endbenutzer durch spezielle Interaktionsobjekte sukzessive erweitert wird.

Die derzeitige Version von *USIT* enthält Familien von Interaktionsobjekten für Standardinteraktionstechniken wie editierbare Textfelder, Icons, textuelle und graphische Menüs, multimodale Schalter, Formulare und Tabellen.

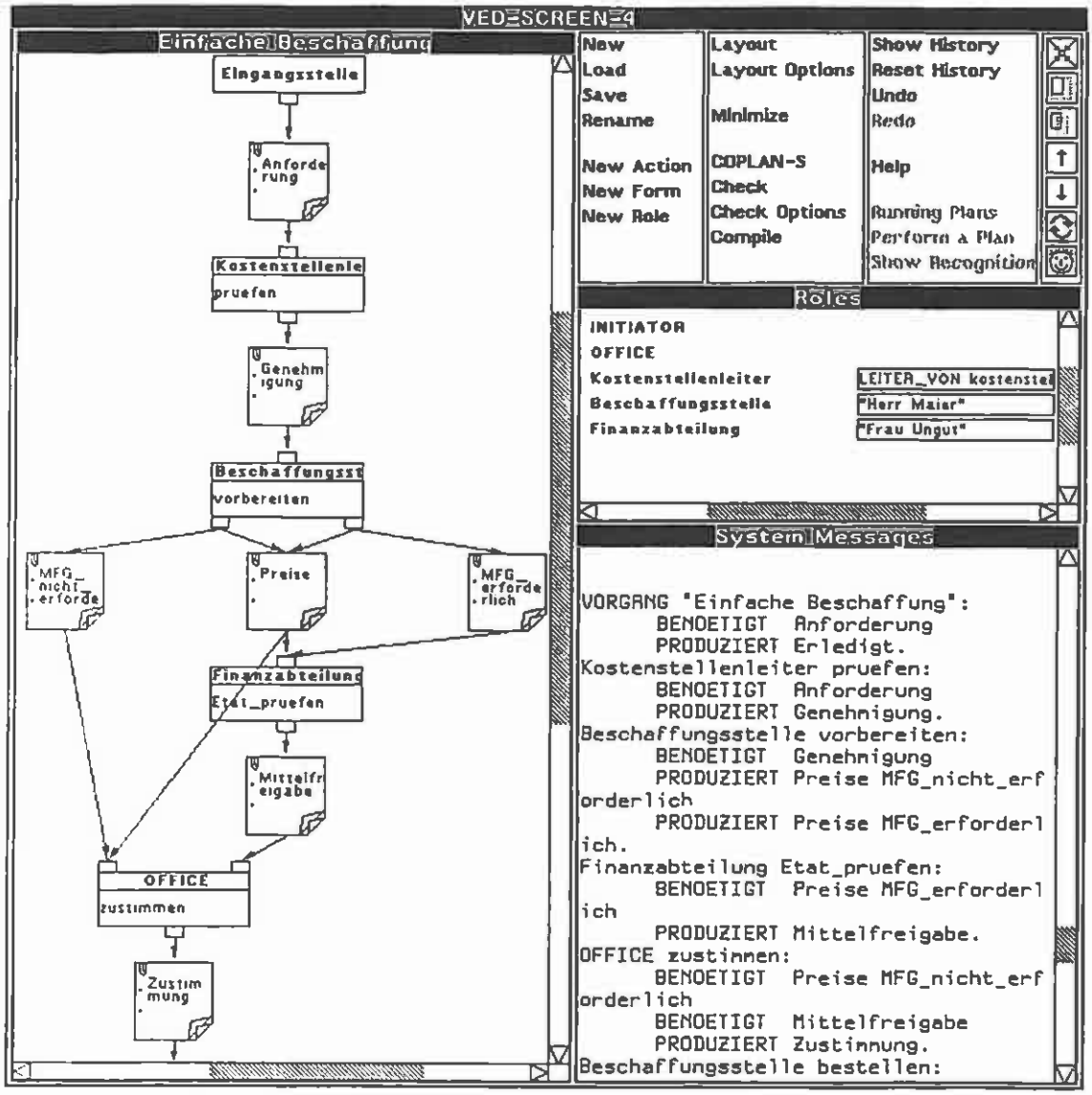

Bild 1: Spezialisierte Interaktionsobjekte im Editor *VED* für Bürovorgänge

Diese Standardtechniken können vielfältig in Inhalt, Layout und Verhalten variiert und untereinander kombiniert werden. Darüberhinaus existiert inzwischen ein Vielfaches an für verschiedene Anwendungen spezialisierten Interaktionsobjekten. Teilweise wurden diese nur in jeweils einem System benötigt, um eine bestimmte Anwendungssituation geeignet unterstützen zu können. Ein solches Beispiel findet sich in Bild 1, wo spezialisierte, vernetzte Icons mit einer variablen Anzahl von Eingangs- und Ausgangskonnektoren sowie mehreren teils editierbaren, teils festen Textfeldern dargestellt werden.

3 Ein Beispiel

Im folgenden findet sich ein Beispiel, wie in *USIT* neue Interaktionsobjekte definiert werden. Das Beispiel ist die Definition der Aktionsknoten im Vorgangseditor *VED*, der im Bild 1 zu sehen ist. Die *ObjTalk*-Syntax wurde im Beispiel geringfügig vereinfacht, um das Lesen der Definition zu erleichtern. Die hervorgehobenen Texte sind Referenzen auf Objekte, meist Klassen.

```
(define-class ved-action-icon
   (superclass icon)
   (slots
     (in-connections
       (part connector))
     (out-connections
       (part connector))
     (actor-name
       (part text-area
         (size = (96 12))
         (border? = t)
         (font = helvetica-8-bold)
         (adjust-text-vertical = center)
         (adjust-text-horizontal = center)))
     (action-name
       (part text-area
         (size = (96 24))
         (border? = t)
         (reactivity = (((mouse left) . start-editor)
                        ((keyboard end) . tell-view-of))))
         (shading-when-inactive = nil)
         (mouse-feedback-when-active = blinker)
         (active-mouse-blinker = cross)
         (interline-spacing = 1.0)
         (font = helvetica-8)
         (adjust-text-vertical = center)
         (adjust-text-horizontal = left)))
     (layouter
       (layout distance-cluster
         (orientation = down)
         (distance = 1)))
     (layout-sequence
       (default (in-connections actor-name action-name out-connections)))
     (pop-up-part
       (default net-action-menu))
     (reactivity = (((mouse left) . tell-view-of)
                    ((mouse right) . pop-up-part)))
     (view-of
       (class coplan-action))))
```

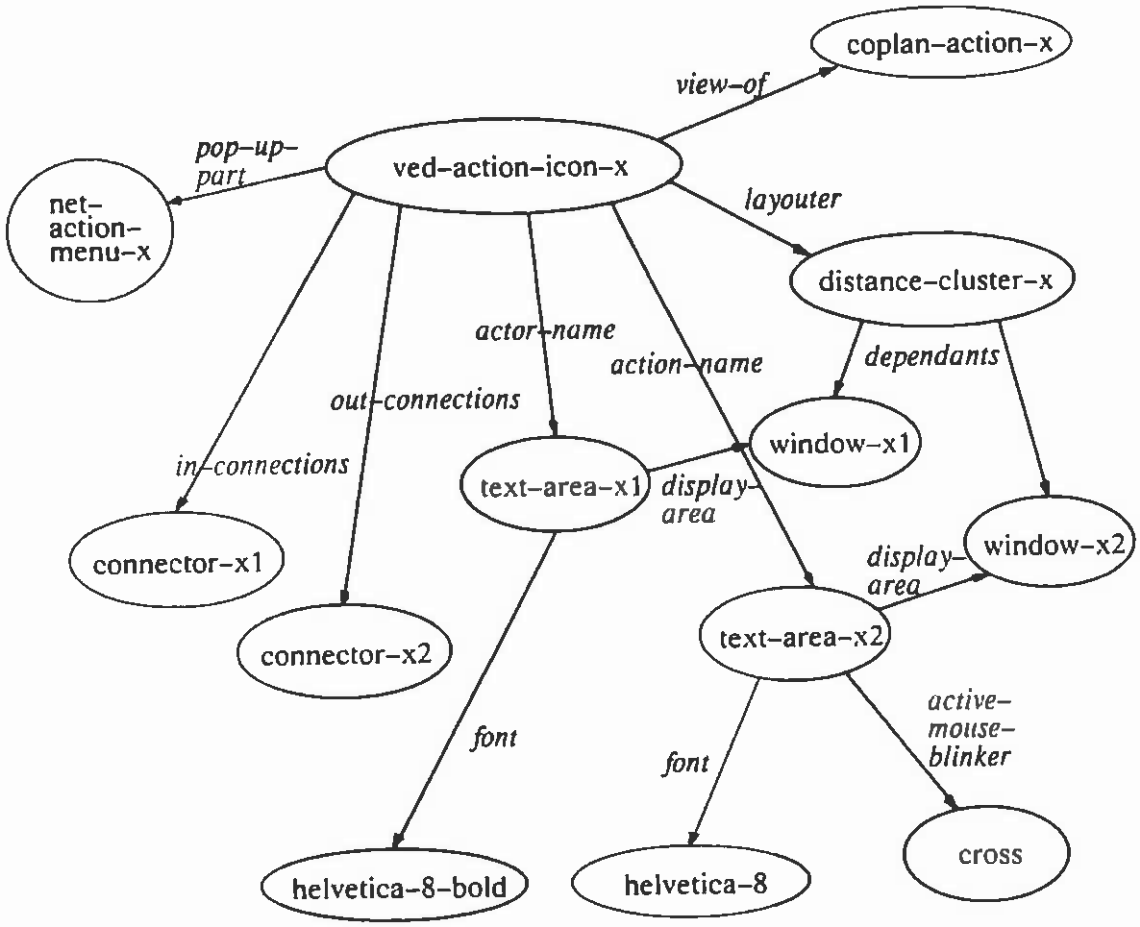

Bild 2: Das Objektnetz eines Interaktionsobjekts

Das Beispiel definiert eine neue Klasse für einen bestimmten, sehr anwendungsspezifischen Typ von Interaktionsobjekt. In dieser Klassendefinition werden mehrfach die Klassen von Elementarbausteinen oder anderen Interaktionsobjekten referenziert. Die `part`-Beschreibungen spezifizieren Darstellungs-Unterobjekte, die `layout`-Beschreibung definiert ein Layoutobjekt zur Anordung der Darstellungs-Unterobjekte und die `view-of`-Beschreibung dient als Zeiger zum assoziierten Anwendungsobjekt. Die verschiedenen `reactivity`-Beschreibungen definieren die Interaktionssyntax sowie die Aktionen auf Benutzereingaben. Durch Instantiierung der neuen Klasse werden auch die referenzierten Klassen instantiiert. Auf diese Weise wird ein ganzes Netz von Objekten generiert. Dieses Objektnetz wurde in Bild 2 skizziert. Verschiedene Objekte, wie beispielsweise einige Fensterobjekte, wurden dabei zur Vereinfachung weggelassen.

4 Zusammenfassung und Ausblick

Die Verfügbarkeit eines **Interaktionskontinuums** anstatt festgelegter Interaktionstechniken stellt sich auf den ersten Blick konträr zur Zielsetzung konsistenter und damit leichter bedienbarer Bedienoberflächen dar. Genauer betrachtet und richtig angewandt bietet die Methodik jedoch die Möglichkeit, anwendungsgerechte Bedienoberflächen zu erstellen, die viele ausgefeilte Details für eine effiziente Bedienung von Anwendungssytemen ermöglichen. Gerade durch die Verwendung einheitlicher Elementarbausteine kann eine sehr grundlegende Konsistenz dargeboten werden. In vielen Fällen wird man bei der Auswahl von Bausteinen auf Standardbausteine zurückgreifen, die ein einheitliches Erscheinungsbild schaffen und gleichartig zu bedienen sind. Änderungen an diesen Bausteinen werden sich in allen Anwendungssystemen konsistent auswirken.

Zur interaktiven Umgestaltung von instantiierten Interaktionstechniken verfügt *USIT* über eine **Meta-Benutzerschnittstelle** [Herczeg/Böcker 87], die derzeit allerdings so ausgebildet ist, daß sie vor allem den Systementwicklern als Hilfsmittel zur Untersuchung von Gestaltungsalternativen dient. Sie könnte leicht auch zur Nutzung durch den Endbenutzer ausgestaltet werden (**adaptierbare Benutzerschnittstelle**). Durch die uniforme Repräsentation aller Interaktionstechniken war die Realisierung eines Metasystems mit geringem Aufwand möglich.

Für die Weiterentwicklung von *USIT* ist die Nutzung eines Standard-Fenstersystems (z.B. *X Windows* [Scheifler/Gettys 86]) als Implementierungsgrundlage geplant. Dies würde die Portierung von *USIT* auf eine Vielzahl von Rechnersystemen ermöglichen.

Literatur

[Girgensohn/Rathke 88] Andreas Girgensohn & Christian Rathke. *ObjTalk - Version 15.15.* INFORM-Manual, Research Group INFORM, Institut für Informatik, Universität Stuttgart, August 1988.

[Goldberg/Robson 83] A. Goldberg & D. Robson. *SMALLTALK-80, The Language and its Implementation.* Addison-Wesley, Reading, Ma., 1983.

[Herczeg 86] Michael Herczeg. *Eine objektorientierte Architektur für wissensbasierte Benutzerschnittstellen.* Dissertation, Fakultät Mathematik und Informatik der Universität Stuttgart, Dezember 1986.

[Herczeg 88] Michael Herczeg. *INFUIMS - The INFORM User Interface Management System.* WISDOM-Forschungsbericht FB-INF-88-25, Institut für Informatik, Universität Stuttgart, 1988.

[Herczeg/Böcker 87] Michael Herczeg & Heinz-Dieter Böcker. DESKTOP: An Adaptable User Interface. In: G. Salvendy (Ed.), *Abridged Proceedings of the HCI International '87. Second International Conference on Human-Computer Interaction*, S. 335, HCI International, Honolulu, Hawaii, August 1987.

[Hutchins/Hollan/Norman 86] E.L. Hutchins, J.D. Hollan, & D.A. Norman. Direct Manipulation Interfaces. In: D.A. Norman & S. Draper (Eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates Ltd., 1986.

[Lipkie/Evans et al. 82] D.E. Lipkie, S.R. Evans, et al. Star Graphics: An Object-Oriented Implementation. *Computer Graphics*, 16(3), July 1982.

[Rathke 86] Christian Rathke. *ObjTalk. Repräsentation von Wissen in einer objektorientierten Sprache.* Dissertation, Fakultät Mathematik und Informatik der Universität Stuttgart, Oktober 1986.

[Scheifler/Gettys 86] R.W. Scheifler & J. Gettys. The X Window System. *ACM Transactions on Grahpics*, 5(2):79-109, April 1986.

[Seybold 81] J. Seybold. Xerox's 'Star'. *Seybold Report Media*, 10(16), 1981.

[Shneiderman 83] Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, August 1983.

[Smith/Irby et al. 82] D.C. Smith, C. Irby, et al. Designing the Star User Interface. *BYTE*, 7(4), April 1982.

[Steele Jr. 84] Guy L. Steele Jr. *Common LISP: The Language.* Digital Press (Digital Equipment Corporation), 1984.

[Sym 86] *Programming the User Interface, Volume A,B.* Symbolics Reference Manual 999025/999029, Symbolics, Inc., Cambridge, Ma., 1986.

Dr. Michael Herczeg
ANT Nachrichtentechnik GmbH
Grundlagenentwicklung
Gerberstraße 33
D-7150 Backnang