Zielkonflikte bei Software-Gestaltungskriterien

Thomas Greutmann, David Ackermann, Zürich

Beim Entwurf von Dialogsystemen können Zielkonflikte zwischen verschiedenen Gestaltungskriterien - insbesondere zwischen denen der Arbeitswissenschaft und denen der Informatik - auftreten. Daher wurde eine Heuristik entwickelt, welche es erlaubt, diese Zielkonflikte beim Entwurf zu berücksichtigen und einseitige Gestaltungsentscheide zu verhindern. Die Anwendung dieses Verfahrens wird an einem realen Entwurfsbeispiel vorgestellt. Die Erweiterung auf ein umfassendes Entwurfsverfahren wird diskutiert.

1. Einleitung

Sowohl die Informatik als auch die Arbeitswissenschaft versuchen die Frage zu beantworten, wann ein Dialogsystem als "gut" zu bezeichnen ist. Beide Disziplinen gehen davon aus, dass die Güte oder Qualität eines Software-Systems ein mehrdimensionales Konstrukt ist, d.h. ein Software-System wird als "gut" bezeichnet, wenn ein Satz von Qualitätskriterien genügend erfüllt ist. Der Schwerpunkt bei der Festlegung der Kriteriensätze ist jedoch in jeder Disziplin anders gesetzt. In der Informatik (Software-Engineering) wird die Güte eines Systems vorwiegend über Kriterien definiert, welche eher den System-Entwickler betreffen (z.B. Wartbarkeit, Portabilität usw.). Diese Kriterien werden als "Qualitätsmerkmale" bezeichnet. Arbeitswissenschafter dagegen definieren die Güte eines Dialogsystems durch Kriterien, die den Benutzer betreffen (z.B. Erlernbarkeit, Transparenz usw.). Entsprechend ihrem Schwerpunkt werden die Ansätze der Informatik im weiteren als "entwicklerorientiert", diejenigen der Arbeitswissenschaft als "benutzerorientiert" bezeichnet.

Diese Ansätze sind häufig in Form einer Auflistung von Kriterien dargestellt. Beispiele für benutzerorientierte Kriterienlisten sind die "Kriterien zur benutzerorientierten Dialoggestaltung" (Ulich, 1986) oder die DIN-Normen 66234 Teil 8 (1988), entwicklerorientierte Kriterienlisten findet man bei Cho (1980) oder Schmitz, Bons und van Megen (1982).

Bei derartigen Auflistungen wird in der Regel nicht auf mögliche Zusammenhänge zwischen den Kriterien hingewiesen. Einige Autoren versuchen daher, die hierarchischen Abhängigkeiten zwischen den Kriterien aufzuzeigen. Es wird von Ober- und Unterkriterien ausgangen, wobei die Erfüllung der Unterkriterien die Voraussetzung für die Erfüllung von Oberkriterien ist. Solche hierarchische Zusammenhänge zwischen be-

nutzerorientierten Kriterien wurden z.B. von Balzert (1987), Spinas (1987) oder auch Triebe, Wittstock und Schiele (1987) aufgelistet. Entsprechende Ansätze für entwicklerorientierte Kriterien findet man bei Boehm, Brown, Kaspar, Lipow, MacLeod und Merrit (1978) oder Gilb (1976).

Die hierarchische Zusammenhänge sind aber nicht die einzigen Beziehungen zwischen den Kriterien. Es gibt auch Kriterien, die nicht ohne weiteres zu vereinbaren sind. Bei der Gestaltung eines Dialogsystems treten daher Zielkonflikte auf, wenn diese Kriterien gleichzeitig erfüllt werden sollen. Norman (1983) z.B. weist darauf hin, dass die Gestaltung von Benutzerschnittstellen stark durch solche Zielkonflikte ("Trade-Offs") geprägt ist. Boehm (1981, p. 3ff) zeigt auf, dass bei der Entwicklung von Software-Systemen ökonomische, soziale und programmiertechnische Aspekte beachtet werden müssen, welche z.T. gegensätzlich sein können. Der Entwickler muss also zwischen verschiedenen gegensätzlichen Zielen abwägen und ein Optimum finden.

Obwohl erkannt sind diese Zielkonflikte - insbesondere die Konflikte zwischen entwicklerorientierten und benutzerorientierten Kriterien - bisher noch nicht ähnlich systematisch dargestellt worden wie die hierarchischen Zusammenhänge. Lediglich für die entwicklerorientierten Kriterien wurde versucht, alle Zusammenhänge systematisch aufzuzeigen. Ein Beispiel dafür findet man bei Evans und Marciniak (1987): Die vermuteten Zusammenhänge wurden in Form einer Matrix dargestellt (vgl. Abb. 1).

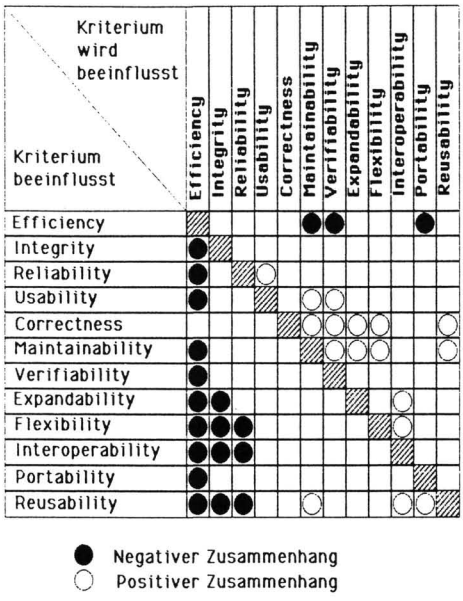

Abb. 1: Zusammenhänge zwischen entwicklerorientierten Kriterien. "Positive" Zusammenhänge deuten auf hierarchische Abhängigkeiten hin, "negative" auf Zielkonflikte. Aus: Evans und Marciniak (1987, p.180).

2. Bessere Einsicht durch Zielkonflikte

Das systematische Aufzeigen von möglichen Zielkonflikten ist für die Gestaltung von Dialogsystemen äusserst wichtig, um die Auswirkungen von Designentscheidungen abschätzen zu können. Wir haben deshalb den Ansatz von Evans und Marciniak (1987) aufgegriffen. In einer Literaturstudie wurden entwicklerorientierte und benutzerorientierte Kriterien zusammengetragen und in einer Matrix paarweise systematisch gegenübergestellt. Die vermuteten Zusammenhänge wurden in der Matrix eingetragen. Empirische Untersuchungen deuten allerdings darauf hin, dass diese Zusammenhänge keineswegs immer eindeutig sind. Aus empirischen Untersuchungen an unterschiedlichen Dialogsystemen werden widersprüchliche Angaben über Zusammenhänge zwischen Kriterien abgeleitet. Dies legt die Annahme nahe, dass die meisten Zielkonflikte bei der Gestaltung von Systemen dadurch entstehen, dass die Kriterien bei der Entwicklung lediglich einzeln und nicht im Zusammenhang mit anderen Kriterien betrachtet werden.
Solche einseitige Gestaltungsentscheide können verhindert werden, wenn gleichzeitig mehrere Kriterien beachtet werden. Durch systematische paarweise Gegenüberstellung von Kriterien können konkrete Gestaltungsregeln hergeleitet werden. Dies soll an einem Beispiel gezeigt werden.

3. Ein Schnittstellen-Manager-System als Beispiel

Die gleichzeitige Beachtung mehrerer Kriterien soll am Beispiel eines Schnittstellen-Manager-Systems für individualisierbare Benutzerschnittstellen dargestellt werden, welches gegenwärtig entwickelt wird. Dieses System soll es dem Entwickler erlauben, eine Schnittstelle zu gestalten, die vom Benutzer im Sinne des Prinzips der differentiellen und dynamischen Arbeitsgestaltung (Ulich, 1978, 1983, im Druck) selber angepasst werden kann. Für dieses System wurden u.a. die folgenden benutzerorientierten Kriterien[1] als zentral erachtet:

Flexibilität/Individualisierbarkeit (Ulich, 1986): Das System soll unterschiedliche Vorgehensweisen sowie Anpassungen an benutzerspezifische Eigenheiten ermöglichen.

---

[1] Der Sprachgebrauch bei der Bezeichnung der benutzerorientierten Kriterien ist in der Literatur sehr uneinheitlich. Häufig werden verschiedene Bezeichnungen für das gleiche Kriterium gewählt. So findet man für "Transparenz" auch die Ausdrücke "Kommunikationsfreundlichkeit" (Maier, 1983) oder "Erwartungskonformität" (Triebe u.a., 1987). Diese stimmen mit der obigen Definition von "Transparenz" weitgehend (aber nicht vollständig) überein. Umgekehrt kann es auch vorkommen, dass der gleiche Begriff von verschiedenen Autoren unterschiedlich definiert wird. Bei den entwicklerorientierten Kriterien ist der Sprachgebrauch wesentlich einheitlicher.

Transparenz (Dehning, Essig & Maass, 1981; Ulich, 1986): Das Systemverhalten soll durchschaubar sein und dem Benutzer ermöglichen, sich ein inneres Modell des Dialogsystems aufzubauen.

Konsistenz (Ulich, 1986): Das Systemverhalten soll einheitlich und für den Benutzer kalkulierbar sein.

Von den entwicklerorientierten Kriterien sind für das Schnittstellen-Manager-System u.a. die folgenden besonders wichtig:

Allgemeingültigkeit (Schmitz u.a., 1982): Das System soll für einen breiten Anwendungsbereich geeignet sein.

Portabilität (Gilb, 1976; Cho, 1980; Schmitz u.a., 1982): Das System lässt sich einfach auf andere Systemumgebungen übertragen.

4. Gegenüberstellung der Kriterien

Die in der Literaturstudie zusammengetragenen Kriterien wurden - in Anlehnung an die Matrix von Evans und Marciniak (1987) - paarweise einander gegenübergestellt und aufgrund von empirischen Fallbeispielen zur Schnittstellengestaltung auf ihre Zusammenhänge hin untersucht. Im Vordergrund standen die möglichen Zielkonflikte, die beim Entwurf des individualisierbaren Schnittstellen-Managers eine Rolle spielen könnten. Unter Beachtung dieser Konfliktmöglichkeiten wurden konkrete Gestaltungsregeln für das zu entwickelnde System formuliert. Einige davon sollen hier beispielhaft vorgestellt werden:

4.1 Flexibilität/Individualisierbarkeit vs. Transparenz

a) Bei existierenden Dialogsystemen, welche Individualisierungs- und Modifikationsmöglichkeiten bieten, tritt häufig das Problem auf, dass die Modifikationen eines Benutzers auf andere Benutzer ebenfalls Auswirkungen haben. Solche (Fremd-) Modifikationen sind aber für die anderen Benutzer nicht mehr durchschaubar. Dies führt zu folgender Gestaltungsregel für das zu entwickelnde System:

G1: Individuelle Modifikationen dürfen nur für den modifizierenden Benutzer selber, aber nicht für andere Benutzer Auswirkungen haben.

Daraus lässt sich eine weitere Gestaltungsregel direkt ableiten:

G2: Gemeinsam verwendete Datenstrukturen dürfen nicht individualisierbar sein. Lediglich die Präsentation der Daten für den Benutzer darf individuell verändert werden.

b) Die Modifikationen können aber auch für den modifizierenden Benutzer selber negative Konsequenzen haben, wenn er (z.B. durch eine irrtümlich vorgenommene Modifikation) seine eigene Konfiguration nicht mehr durchschauen kann. Deshalb:

G3: Es muss ein Rückstell-Befehl angeboten werden, der es dem Benutzer erlaubt, in einen definierten Systemzustand ohne individuelle Modifikationen zurückzugelangen.

c) Häufig werden Modifikationsmöglichkeiten mit Hilfe von Mode-"Schaltern" realisiert, die in der Menu- oder Befehlsstruktur kontextabhängig verteilt sind. Dies kann die Transparenz beeinträchtigen, wenn eine gewünschte Aktion durch einen dem Benutzer (zum Zeitpunkt der Aktion) nicht gegenwärtigen Mode-"Schalter" beeinflusst wird. Daraus resultiert:

G4: Bei Verwendung von Mode-"Schaltern" muss eine zentrale "Schalttafel" vorgesehen werden, wo sämtliche Modifikationsmöglichkeiten mit dem aktuellen Zustand ersichtlich sind und auf Wunsch geändert werden können.

4.2 Konsistenz vs. Portabilität

Dies ist ein Beispiel dafür, wie durch gleichzeitige Beachtung zweier Kriterien grundsätzlich völlig andere Implikationen für die Systemgestaltung nahegelegt werden als bei blosser Beachtung eines einzelnen Kriteriums. Konzentriert man sich bei der Entwicklung eines Dialogsystems lediglich auf die Portabilität, so impliziert dies die Verwendung eines geräteabhängigen Moduls, auf dem das eigentliche Anwendungs-System aufbaut (vgl. Abb. 2). Diese Systemarchitektur kann aber negative Auswirkungen auf die Konsistenz haben. Das System wird sich in allen Umgebungen, auf die es portiert wird, identisch verhalten. Das Systemverhalten solcher Systeme steht häufig in krassem Widerspruch mit der jeweiligen Systemumgebung und anderen Anwendungsprogrammen. Als Extrembeispiel kann ein konventionelles Command-Line-Anwendungssystem dienen, das auf einen modernen Arbeitsplatzrechner mit Fenstersystem (z.B. Macintosh) portiert wird. Dieses System wird - sofern nur das geräteabhängige Modul angepasst wird - völlig inkonsistent mit der übrigen Systemumgebung sein.

Um Portabilität **und** Konsistenz gleichzeitig zu garantieren, muss daher eine spezifische Systemarchitektur gewählt werden. Es müssen mehrere wohldefinierte Schichten vorgesehen werden, wobei die in den Schichten definierten Systemobjekte und -operationen nach oben zunehmend höhere Abstraktion aufweisen: In den untersten Schichten werden z.B. Events (Tastendrücke, Zeigen mit der Maus) definiert, in den nachfolgenden höheren Schichten elementare Dialogobjekte wie Zahleneingabe-Felder, Ja/Nein-Antwortmöglichkeiten und in noch höheren Schichten abstrakte Dialogobjekte wie Dialogboxen oder Fenster, welche verschiedene elementare Dialogelemente enthalten.

Um das System dann zu portieren und konsistent zu halten, müssen entsprechend der Zielumgebung wenige oder viele Schichten angepasst werden: Ist die Zielumgebung

der Ursprungsumgebung sehr ähnlich oder stellt sie Elemente aus höheren Schichten zur Verfügung, so müssen nur wenige Schichten angepasst werden; handelt es sich dagegen wie im obigen Extrembeispiel um sehr verschiedene Systeme, so muss die Mehrzahl der Schichten angepasst werden, um wieder Konsistenz zu garantieren. Die beiden unterschiedlichen Systemarchitekturen sind in Abb. 2 dargestellt.

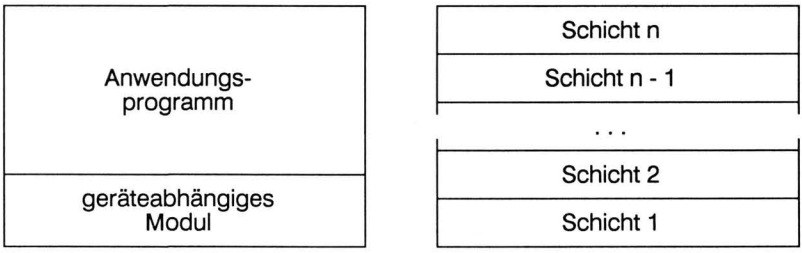

Abb. 2: Systemarchitektur bei alleiniger Beachtung von Portabilität (links) und bei gleichzeitiger Beachtung von Konsistenz und Portabilität (rechts).

Ein bekanntes Beispiel - nicht aus dem Gebiet der Dialogschnittstellen - für eine solche Mehrschichten-Architektur ist der ISO OSI-Standard für Datenkommunikation (z.B. beschrieben in Tanenbaum, 1981).

5. Diskussion

Die genannten Beispiele belegen, dass bei der Entwicklung von Dialogsystemen Zielkonflikte zwischen verschiedenen Gestaltungskriterien auftreten können, deren systematische Beachtung und Ueberprüfung die Herleitung von Gestaltungsregeln erlaubt, die wiederum als Spezifikationen verwendet werden können.

Die Annahme liegt allerdings nahe, dass einige der möglichen Zielkonflikte sich nicht lösen lassen. Gerade zwischen Gestaltungszielen und Charakteristika, die sich auf den Entwicklungsprozess beziehen (Zeitdauer, Kenntnisstand der Entwickler usw.) und Kriterien, welche sich auf die Systemeigenschaften (z.B. Robustheit/Fehlertoleranz) beziehen, sind unlösbare Konflikte zu vermuten, wie dies auch durch die empirischen Befunde des COCOMO (COnstructive COst MOdel) von Boehm (1981) nahegelegt wird.

Die systematische Ueberprüfung der Konfliktmöglichkeiten ist kein deterministisches Verfahren, welches ein eindeutig definiertes Resultat liefert. Es kann auch nicht garantiert werden, dass die erarbeiteten Gestaltungsregeln vollständig und umfassend sind. Das Verfahren ist aber ein gutes Hilfsmittel zum Aufzeigen von möglichen Zielkonflikten und zur Unterstützung der Diskussion von Lösungsmöglichkeiten.

Die Nicht-Determiniertheit und die nicht garantierte Vollständigkeit sind u.a. darauf zurückzuführen, dass die Anwendung dieser Vorgehensweise stark vom Kenntnisstand der Durchführenden abhängt. Es ist daher zu prüfen, ob diese systematische Gegenüberstellung der Kriterien auch allgemein (d.h. systemunabhängig) durchgeführt werden kann. Dadurch würden sich allgemeine Gestaltungsregeln herleiten lassen, welche dann direkt bei der Entwicklung verwendet werden könnten.
In jedem Fall weist diese systematische Kriterien-Diskussion einige Vorteile auf, die sich aus dem beschriebenen Fallbeispiel ableiten lassen:

1) Sie ermöglicht die Herleitung von **konkreten Gestaltungsregeln**, die über die beschriebenen Beispiele weit hinausgehen.
2) Diese Gestaltungsregeln **verhindern einseitige Gestaltungsentscheide**, weil die Kriterien nicht isoliert, sondern im Zusammenhang mit anderen Kriterien betrachtet werden.
3) Das systematische Vorgehen bewirkt, dass die Gestaltungsentscheide **bewusst** gefällt werden und **explizit** gemacht werden.
4) Dadurch sind die Gestaltungsentscheide für andere Entwickler **nachvollziehbar**.

8. Ausblick

Die dargestellte Vorgehensweise ist ein heuristisches Hilfsmittel, welches beim Entwurf von Dialogschnittstellen eingesetzt werden kann. Sie muss jedoch noch durch weitere Heuristiken ergänzt werden, welche zusammen ein umfassendes Entwurfsverfahren für Dialogschnittstellen in 5 Schritten ergeben. Diese Erweiterungen sollen im folgenden kurz dargestellt werden.

(1) Erstellen einer Liste von benutzerorientierten und entwicklerorientierten **Kriterien**.
(2) **Bestimmung der relevanten Kriterien**. Unter Berücksichtigung der vorgesehenen Benutzergruppe und der zu bearbeitenden Aufgabe muss festgelegt werden, welche Kriterien bei der Systementwicklung in welchem Masse zu berücksichtigen sind. Damit ergibt sich eine erste Grobspezifikation des Dialogsystems.
(3) **Systematisches Gegenüberstellen der Kriterien und Erarbeiten von Gestaltungsregeln**. Die relevanten Kriterien werden paarweise einander gegenübergestellt und die Gestaltungsregeln erarbeitet, welche eine präzisere Spezifikation darstellen.
(4) **Analyse des Handlungsspielraums und Erarbeiten von Alternativen**. In diesem Schritt sollen mögliche Lösungen für die Gestaltung der Schnittstelle erarbeitet werden.

(5) **Bewertung der Alternativen.** Aufgrund der in den Schritten (2) und (3) erarbeiteten Spezifikationen lassen sich die in Schritt (4) erarbeiteten Alternativen bewerten.

Dieses erweiterte Entwurfs-Vorgehen in fünf Schritten wird bei der Entwicklung des Schnittstellen-Manager-Systems versuchsweise angewendet. Die Schritte (1) bis (3) wurden oben vorgestellt, die beiden anderen Schritte müssen noch weiter entwickelt werden.

9. Neue Fragestellungen

Das systematische Aufzeigen von Zusammenhängen verweist auf eine Fülle interessanter neuer Fragestellungen, welche interdisziplinär angegangen werden müssen. Die umfassende empirische Abklärung der Zusammenhänge zwischen den Gestaltungskriterien ist auch für den Entwickler von Bedeutung, können diese doch die Fortpflanzung der Auswirkungen und damit die Tragweite von Gestaltungsentscheiden aufzeigen. Die Gegenüberstellung der Kriterien in einer Matrix kann so zur Hypothesengeneration eingesetzt werden.

10. Literatur

Balzert, H. (1987). Gestaltungsziele der Software-Ergonomie. In: W. Schönpflug & M. Wittstock (Hrsg.). Software-Ergonomie '87. Stuttgart, Teubner.

Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., MacLeod, G.J. & Merrit, M.J. (1978). Characteristics of Software Quality. Amsterdam, North-Holland.

Boehm, B.W. (1981). Software Engineering Economics. Englewood Cliffs NJ, Prentice-Hall.

Cho, C.-K. (1980). An Introduction to Software Quality Control. New York, Wiley.

Dehning, W., Essig, H. & Maass, S. (1981). The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogs. Berlin, Springer.

DIN 66234 Teil 8 (1988). Bildschirmarbeitsplätze - Grundsätze ergonomischer Dialoggestaltung. Berlin, Beuth.

Evans, M.W. & Marciniak, J.J. (1987). Software Quality Assurance and Management. New York, Wiley.

Gilb, T. (1976). Software Metrics. Lund, Studentlitteratur.

Maier, H.H. (1983). Die Prüfung des Software-Qualitätsmerkmals Benutzungsfreundlichkeit. Management der Qualitätssicherung 1983, Heft 1, S. 23-26.

Norman, D.A. (1983). Design Principles for Human-Computer Interfaces. In: A. Janda. Human Factors in Computing Systems, Proceedings of the CHI'83 Conference. Amsterdam, North-Holland.

Schmitz, P., Bons, H. & van Megen, R. (1982). Software-Qualitätssicherung - Testen im Software-Lebenszyklus. Braunschweig/Wiesbaden, Vieweg & Sohn.

Spinas, P. (1987). Zur Benutzerfreundlichkeit von Bildschirmsystemen. In: W. Schönpflug & M. Wittstock (Hrsg.). Software-Ergonomie '87. Stuttgart, Teubner.

Tanenbaum, A.S. (1981). Computer Networks. Englewood Cliffs, Prentice-Hall.

Triebe, J.K., Wittstock, M. & Schiele, F. (1987). Arbeitswissenschaftliche Grundlagen der Software-Ergonomie. Schriftenreihe der Bundesanstalt für Arbeitsschutz, Sonderschrift S24.

Ulich, E. (1978). Ueber das Prinzip der differentiellen Arbeitsgestaltung. Industrielle Organisation 1978, 47, S. 566-568.

Ulich, E. (1983). Differentielle Arbeitsgestaltung - ein Diskussionsbeitrag. Zeitschrift für Arbeitswissenschaft 1983, 37, S. 12-15.

Ulich, E. (1986). Aspekte der Benutzerfreundlichkeit. In: W.Remmele & M. Sommer (Hrsg.). Arbeitsplätze morgen. German Chapter of the ACM, Bericht Nr. 27. Teubner, Stuttgart.

Ulich, E. (im Druck). Individualisierung und Differentielle Arbeitsgestaltung. In: B. Zimolong & C. Graf Hoyos. Enzyklopädie der Psychologie, Band Ingenieurpsychologie. Hogrefe, Göttingen (in Vorbereitung).

Thomas Greutmann
Lehrstuhl für Arbeits- und
Organisationspsychologie
ETH Zürich
Nelkenstr. 11
CH - 8092 Zürich

David Ackermann
IBM
T.J. Watson Research Center
PO Box 218
Yorktown Heights
NY 10598 USA