S. Boll, S. Maaß & R. Malaka (Hrsg.): Mensch & Computer 2013
München: Oldenbourg Verlag, 2013, S. 67–77

Gestaltungskonflikte in der Softwareergonomie

Reinhard Keil, Christian Schild

Heinz Nixdorf Institut, Universität Paderborn

Zusammenfassung

Eine wichtige Herausforderung im Bereich der Softwareergonomie besteht darin, Gestaltungswissen so zu kodifizieren, dass dieses systematisiert und als Grundlage für eine hypothesengeleitete Technikgestaltung verwendet werden kann. Analytische Kriterien, Raster und Checklisten sind dazu wenig geeignet, weshalb in den letzten Jahren verstärkt das Konzept der Mustersprachen (pattern languages) für diesen Zweck erschlossen wird. Die bislang verfolgten Ansätze ermöglichen es, vorrangig tradierte Lösungsansätze aufzuzeichnen und zu systematisieren. Mit dem in diesem Beitrag vorgeschlagenen Konzept der Gestaltungskonflikte soll eine wichtige Erweiterung von Mustersprachen erreicht werden, um auch Lösungsansätze für eine prospektive Gestaltung erfassen zu können.

1 Einleitung

Gestaltungskriterien, wie zum Beispiel die Grundsätze zur Dialoggestaltung (DIN EN ISO 9241 Teil 110) oder auch Heuristiken (zum Beispiel Nielsen 1994), können zwar wichtige Hinweise zur Gestaltung geben, gestatten es aber nicht, eine Systematik auf Basis theoretischer Konzepte und Grundlagen zu erstellen. Sie geben zwar vielfältige Hinweise auf gute und schlechte Beispiele, bieten allerdings keinen Ansatz für die Entwicklung von Grundlagen für eine hypothesengeleitete Technikgestaltung aus der Sicht der Informatik (Keil 2011). Insbesondere fällt auf, dass Gestaltungskonflikte, deren Austarierung maßgeblich für eine gute Gestaltung ist, nicht thematisiert werden. In Anlehnung an den Architekten Christopher Alexander sind deshalb in den letzten Jahren viele Ansätze zur Entwicklung von Mustersprachen entstanden. Im Bereich der Softwareergonomie sei hier auf Bayle et al. (1997) und Tidwell (2005) verwiesen. Das Ziel ist es, Entwicklern bei der Gestaltung von ergonomischen Benutzungsoberflächen Lösungen bzw. Lösungsansätze mit auf den Weg zu geben, die damit auf immer bewährte Lösungsansätze zurückgreifen können.

Allerdings sind speziell die Ansätze aus der Informatik von Alexander kritisiert worden, weil sie das für ihn so wichtige Konzept der „Forces“ (Kräfte) nicht berücksichtigen (Alexander

1978). Solange es vorrangig darum geht, tradierte Lösungsansätze systematisch zu sammeln, kann diese Kritik außer Acht gelassen werden. Allerdings wird damit auch ein Stück weit die konstruktive Kraft aufgegeben, die Christopher Alexander mit seiner „Pattern Language" verbindet, denn die Kräfte sind für ihn ein gestaltungsleitendes Element seiner Mustersprache. Tradierte Gestaltungsformen verkörpern als Lösungsansätze (Muster) eine Fülle von Erfahrungen, wie sich unterschiedliche und teilweise zuwiderlaufende Anforderungen so austarieren lassen, dass eine möglichst praktikable und dauerhafte Lösung im jeweiligen Gestaltungskontext entsteht. So verkörpert der Patio als geschlossener Innenhof in südlichen Ländern für Alexander einen guten Kompromiss zwischen Helligkeit und zugleich Schutz vor Hitze. Die Kenntnis dieser Kräfte ist nach Alexander erforderlich, um eine bestimmte Form als Muster zu erkennen und nutzen zu können, denn nur unter Kenntnis dieser Kräfte ist es möglich, sie an einen neuen Gestaltungskontext anpassen und unterschiedliche Muster zu einer Mustersprache verbinden zu können. Was Alexander als Kräfte bezeichnet, sind alle Wirkungen, die sich aus Umgebungsbedingungen, dem verwendeten Material und den persönlichen und sozialen Anforderungen der Menschen ergeben.

Solange ein Lösungsmuster ohne Kenntnis der damit verbundenen Kräfte genutzt werden soll, läuft man Gefahr, zu schlecht angepassten Lösungen zu kommen. Da ein bloßes Aneinanderreihen von Standardbausteinen, Teillösungen bzw. Schemata auch kein fundiertes Gestaltungsverständnis widerspiegelt, fehlt auch die geeignete Basis, um Mustersprachen anpassen und weiter entwickeln zu können.

Genau hier soll das Konzept der Gestaltungskonflikte helfen, dieses von Alexander an bestehenden Ansätzen zur Entwicklung von Mustersprachen in der Informatik (Gamma et al. 2004; Tidwell 2005) kritisierte Defizit zu beseitigen und damit letztlich bestehende Mustersprachen so anzureichern, dass sie auch Perspektiven für eine prospektive Gestaltung oder gar eine theoretische Fundierung ergeben, die vielleicht langfristig eine hypothesengeleitete Gestaltung interaktiver Systeme ermöglicht (Keil 2011).

2 Zur Rolle von Konflikten

Das Konzept der Gestaltungskonflikte wurde ursprünglich entwickelt, um im Prozess der Gestaltung das Augenmerk der Entwickler auf die Fülle widersprüchlicher Anforderungen zu lenken, die zudem nicht als binäre Variablen (erfüllt, nicht erfüllt) betrachtet werden können. Wesentlich bei Gestaltungskonflikten ist, dass eine oder mehrere berechtigte Anforderungen nur auf Kosten anderer, gleichermaßen berechtigte Anforderungen umgesetzt werden können. Solche Konflikte können grundsätzlich nur in Bezug auf den jeweiligen Kontext entschieden werden (Keil-Slawik 1990; Brennecke, Keil-Slawik 1995). Dieser Ansatz ist verbunden mit der Einsicht, dass gute Gestaltung darin besteht, die Fülle der im Entwicklungsprozess auftretenden Gestaltungskonflikte möglichst situationsgerecht auszutarieren.

Ein klassisches Beispiel für einen Gestaltungskonflikt in der Informatik ist das bekannte Problem *Laufzeit vs. Speicherplatzbedarf*: Bei der Entwicklung von Software muss entschieden werden, ob das Programm schneller läuft, dafür aber mehr Speicher verbraucht, oder ob die Speicherauslastung gesenkt wird, dafür aber die Laufzeiteigenschaften verschlechtert werden.

Wie Keil-Slawik (1990) beschreibt, kann man unter Zuhilfenahme von Alexanders Ideen der widersprechenden und interagierenden Kräfte „das Verständnis des Designers für die Art von Problemen, mit denen er es zu tun hat“ fördern (Alexander 1964, 35).

Allerdings ist das Konzept der Gestaltungskonflikte bislang nicht als Mittel zur Konstruktion von Mustersprachen genutzt worden. Nachfolgend soll deshalb ein erster Ansatz skizziert werden, wie das Konzept der Gestaltungskonflikte genutzt werden kann, um einerseits relevantes Gestaltungswissen und dessen theoretischen Begründungszusammenhang zu erschließen und andererseits in Form von Gestaltungsmustern zu kodifizieren. Der Anspruch dabei ist nicht, bislang unbekanntes Gestaltungswissen zu erzeugen, sondern vielmehr, bekanntes Wissen gestaltungsorientiert zu systematisieren. Dazu gilt es die jeweiligen Kräfte so zu beschreiben, dass flexibel unterschiedliche Lösungsmuster gestaltet und kontextgerecht angepasst werden können.

In der Literatur werden die hier besprochenen Konflikte bei der Gestaltung von Oberflächen als Designkonflikte, als Gestaltungskonflikte oder auch als Zielkonflikte bezeichnet. Im weiteren Verlauf soll hier der Begriff Gestaltungskonflikt als Obergriff verwendet werden.

Im Sinne der Mustersprachen kann für die Erfassung von Gestaltungskonflikten das folgende Schema verwendet werden:

- **Name des Konflikts:** Der Konflikt wird über die gegensätzlichen Anforderungen (Kräfte) beschrieben.
- **Beschreibung der Anforderungen:** Die sich widersprechenden Anforderungen werden beschrieben und mit den möglichen Auswirkungen auf die gegenteilige Anforderung erfasst. Die entsprechenden Einflussfaktoren sollten hierbei berücksichtigt werden.
- **Lösungsmöglichkeiten:** Für die Lösung des Konflikts werden mögliche Lösungen genannt. Wenn dies nicht eindeutig möglich ist, sollten von den am Entwicklungsprojekt beteiligten Personen unter Verwendung bereits bekannter (empirischer) Untersuchungen entsprechende Lösungswege angesprochen werden.

3 Beispiele für Gestaltungskonflikte

Um zu sehen, ob und inwieweit Gestaltungskonflikte geeignet sind, die für Alexander entscheidenden Kräfte aufzuspüren und in konstruktive Lösungsmuster zu transformieren, sollen hier beispielhaft die beiden Konflikte Ikonizität vs. Einfachheit und Breite vs. Tiefe skizziert und nach den oben beschriebenen Konventionen aufgeschrieben werden. Dabei ist zu beachten, dass das Auffinden solcher Konflikte selbst ein wesentlicher Teil der Forschung ist, da sie auf theoretischen Begründungen und empirischen Befunden beruhen.

Ikonizität vs. Einfachheit

Wandmacher bezeichnet Piktogramme als „bildhafte Darstellungen, die Objekte, Funktionen, Aktionen oder Prozesse repräsentieren“ (Wandmacher 1993, 383). Dementsprechend sollten Piktogramme es also den Nutzern ermöglichen, die Objekte, Funktionen, Aktionen oder Prozesse, die ein Piktogramm auslöst, aufgrund der Darstellung wiederzuerkennen. Da sich für solche Piktogramme bei Benutzungsoberflächen der Begriff Icon eingebürgert hat, soll

dieser nachfolgend verwendet werden. Die Frage, wie Icons optimal gestaltet sein sollten, geht mit der Entwicklung graphischer Benutzungsoberflächen einher, verkörpern sie doch ein wesentliches Interaktionselement. Entsprechend vielfältige Hinweise, Anleitungen und Kriterien gibt es für die Gestaltung von Icons, sodass es für Entwickler schwierig ist, die relevanten Aspekte der Gestaltung zu erkennen. Deshalb sollen die verschiedenen Einsichten nachfolgend unter dem Gestaltungskonflikt „Ikonizität vs. Einfachheit“ charakterisiert werden. Dieser Konflikt ergibt sich, wenn die folgende Betrachtung angestellt wird: Ein optisches Gebilde muss durch eine Folge von Augenbewegungen (Sakkaden) erschlossen werden. Je komplexer die wahrzunehmende Form ist, desto größer ist der Aufwand zum Erkennen. Neben der Formwahrnehmung gilt es aber auch das Gedächtnis zu entlasten, denn ein Bild muss auch wiedererkannt werden. Die Wiedererkennbarkeit sollte nicht mit der Erkennbarkeit verwechselt werden. Die Bedeutung eines Icons muss immer erst erlernt werden und kann nicht direkt aus dem erstmaligen Betrachten erkannt werden. Dieser Lerneffekt ist von vielen unterschiedlichen Faktoren abhängig (Eco 2002, 204 ff.).

Zum Wiedererkennen sind nun zwei Schritte erforderlich: zunächst die Konstruktion von entsprechenden Hinweisreizen, wofür das Icon stehen könnte, und dann der tatsächliche

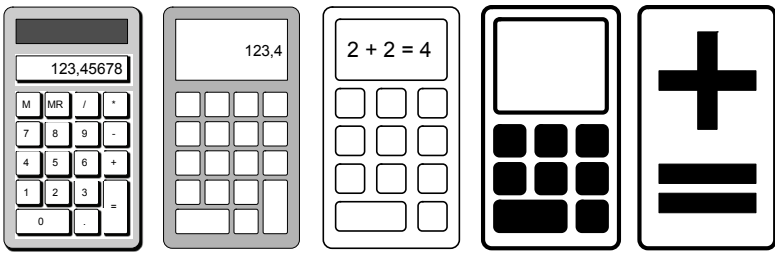

Abbildung 1: Abstraktionsgrade eines Icons nach Mullet & Sano 1994 (dort angelehnt an Nadin 1988, 284)

Abruf aus dem Gedächtnis. Bei ikonischen Darstellungen kann der erste Schritt entfallen, da sie bereits die entsprechenden Hinweisreize in sich tragen. Ballstaedt beschreibt ikonische Zeichen als Abbilder, die visuelle Merkmale mit dem Bezeichneten gemeinsam haben (Ballstaedt 1997, 207). Icons sollten demnach visuelle Merkmale mit dem Abbild bzw. der Funktion gemeinsam haben, die sie repräsentieren. Dies bringt das Problem mit sich, dass, je mehr visuelle Merkmale ein Icon mit dem Referenzobjekt bzw. der Funktion gemeinsam hat, desto größer die Gedächtnisentlastung ist, aber zugleich damit der Erkennungsaufwand für den Nutzer steigt, da das Auge alle Einzelheiten erfassen muss.

Dieser Konflikt ist von Minov unter dem Gesichtspunkt der Abstraktion anhand eines Icons für Taschenrechner beschrieben (Minov 2004, 43). Dabei greift er auf die Beschreibungen von Nadin (1988) zurück. Nach Arnheim sollte eine „vernünftige Abstraktion generativ sein, womit gemeint ist, dass ein Begriff es uns ermöglichen soll, ein vollständigeres Bild des Gegenstandes zu entwickeln, als der Begriff selbst enthält.“ (Arnheim 1972, 167 f.) Arend und Wandmacher haben in ihren Untersuchungen (Arend, Wandmacher 1989) vier Klassen von Icons betrachtet.

Wort-Icons transportieren die Bedeutung des Icons über den Text. Arbiträre Icons sind willkürliche Symbole, die keine direkte oder indirekte Ähnlichkeit mit der Erscheinungsform des Referenzobjekts haben (Wandmacher 1993, 383). Ein Desktop-Icon oder ein globales Icon hat „eine visuelle Ähnlichkeit mit der Erscheinungsform seines Referenzobjekts oder eines

charakteristischen Teils oder Merkmals des Referenzobjekts.“ (Wandmacher 1992) Für das Beispiel Taschenrechner aus *Abbildung 1* zeigt *Abbildung 2* die vier oben angesprochenen Icon-Klassen.

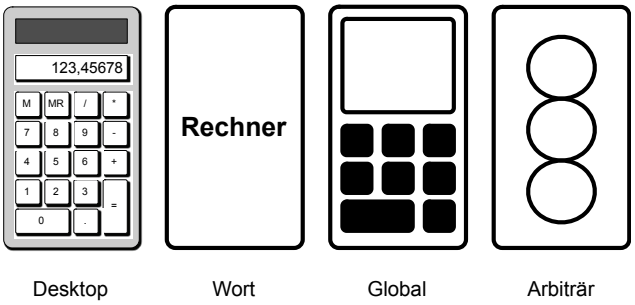

Abbildung 2: Klassifizierung von Icons

Der oben beschriebene Konflikt bezüglich des Abstraktionsgrads eines Icons lässt sich unter Zuhilfenahme der Schlussfolgerungen von Arend und Wandmacher auflösen (Arend, Wandmacher 1989, 15 f.), da sich die globalen Icons hinsichtlich der Wiedererkennbarkeit und des motorischen Aufwands zur Erfassung sehr gut eignen. Für die Gestaltung dieser Icons gibt Wandmacher (Wandmacher 1993, 392) folgende auf den Untersuchungen von Easterby (1970) basierende Gestaltungsempfehlungen:

- geschlossene Figuren,
- möglichst solide Figuren,
- solide Figuren in einer Umrissfigur, wenn das Icon aus beiden bestehen soll,
- dunkle Figuren oder Umrisslinien auf hellem Hintergrund,
- konvexer Umriss,
- vertikale Hauptachse, Symmetrie um die vertikale Hauptachse, Einheitlichkeit der Größen und Proportionen, insbesondere von graphischen Elementen, die in verschiedenen Icons vorkommen.

Diese Betrachtungen zeigen, dass mithilfe des Gestaltungskonflikts Ikonizität vs. Einfachheit ein etwas präziseres Verständnis des Gestaltungsproblems verbunden ist, als wenn es nur über die Frage des Abstraktionsgrads beschrieben wird, weil hier grundlegende Kräfte der Wahrnehmung (schnelle Erkennung vs. Gedächtnisentlastung) thematisiert werden und damit auch Randbedingungen systematisch betrachtet werden können, in denen unter Umständen eher die schnelle Erkennbarkeit als die einfache Wiedererkennbarkeit zählen. Ersteres kann beispielsweise der Fall sein, wenn man es mit einer dynamischen Steuerung zu tun hat, die weniger Icons umfasst, die dafür aber schnell erkannt werden müssen. Die Gedächtnisentlastung dürfte dagegen ein gewichtiger Faktor bei gängigen Standardanwendungen sein, da es sich hier in der Regel um viele hundert Artikel handelt.

Grundsätzlich ist zu berücksichtigen, dass zum einen weitere Gestaltungskonflikte eine Rolle spielen können und zum anderen das Lösungsmuster – in diesem Fall globale Icons – mit anderen Anforderungen in Konflikt tritt. Beispielsweise empfiehlt der Apple Styleguide für die Dock-Bar-Icons komplexe Icons (Apple 2012, 112 ff.). Hier muss entschieden werden, ob eher den Grundsätzen der Dialoggestaltung (DIN EN ISO 9241) gefolgt und die System-

konformität priorisiert wird, der wissenschaftlichen Empfehlung, globale Icons zu verwenden, gefolgt oder aber eine dritte Lösung als geeigneter Kompromiss der beiden Empfehlungen entwickelt wird.

Breite vs. Tiefe

Der Gestaltungskonflikt Breite vs. Tiefe kann als Abwägung zwischen räumlicher Präsenz und schrittweiser Erschließbarkeit verstanden werden (Keil-Slawik 1990, 40). Dieser Konflikt tritt beispielsweise bei der Gestaltung von Menüs auf. Wenn 30 Menüeinträge in einer Menüleiste verteilt werden sollen, können diese in 5 Menüpunkte mit jeweils 6 Unterpunkten aufgeteilt werden. In diesem Fall wird Platz für 5 Punkte benötigt. Bei 10 Menüpunkten, mit jeweils 3 Unterpunkten, wird Platz für mindestens 10 Menüpunkte benötigt.

Abbildung 3: Menüstruktur - Tiefe 3

Bei der Verteilung einer großen Anzahl von Funktionen muss also ein Kompromiss gefunden werden zwischen den beiden Extrempositionen *Alle Funktionen in ein Menü ohne Untermenü* (Breite n, Tiefe 1) und *jeder Eintrag in ein eigenes Untermenü* (Breite 1, Tiefe n). Natürlich ergeben die beiden Extrempositionen des Konflikts in den meisten Szenarien keinen Sinn, da sie für den Nutzer keine Orientierungshilfe mit sich bringen. Sollen nur fünf Funktionen in einem Menü untergebracht werden, ist die Verwendung von Untermenüs ebenfalls nicht sinnvoll. Wie in Abbildung 3 zu sehen ist, sollten die Funktionen zunächst gruppiert werden (Format, Anordnen, Darstellung etc.).

Shneiderman und Plaisant geben die Obergrenze für die Tiefe von Menüs mit 3 an (Shneiderman, Plaisant 2004, 283). Dabei verweisen sie auf die Untersuchung von Jacko und Salvendy, die folgendes nachweisen: „Increased depth of a menu as an attribute that significantly affected users' perceptions of the complexity of the task." (Jacko, Salvendy 1996, 1198)

Die Lösungen können zwar teilweise auf wissenschaftliche Erkenntnisse zurückgreifen, jedoch repräsentieren sie keine universell anwendbare Lösung, denn in vielen praktischen Anwendungen muss man von solchen einfachen numerischen Vorgaben zwangsläufig abweichen.

Unter der Berücksichtigung, dass ein numerischer Wert keinen Begründungszusammenhang darstellt, ist es sinnvoller, die Frage methodisch aufzulösen, indem durch die Analyse des Einsatzkontextes versucht wird zu ermitteln, welcher der Kräfte in diesem Konflikt vorrangig nachzugeben ist. Zugleich ist ein Gestaltungskonflikt immer auch Anlass darüber nachzudenken, ob nicht auch in anderen Situationen dieselben Kräfte austariert werden müssen. Beispielsweise kostet ein gut strukturierter Aufbau eines Bildschirminhalts Platz. Sollen aber viele Aspekte schnell und zusammen betrachtet werden, so bietet sich eine sequenzielle Erschließung über mehrere Bildschirmseiten nicht an. Ein Aussteller von Fahrkarten oder Tickets, der den ganzen Tag mit demselben System viele Ausgaben tätigen muss, wird bis zu einem gewissen Grad hier die Unübersichtlichkeit des Bildschirmaufbaus einem zusätzlichen Hin- und Herschalten zwischen verschiedenen Bildschirmen vorziehen, da er mit dem Bildschirmaufbau sehr vertraut ist. Ein gelegentlicher Nutzer dagegen, der sich schnell eine Fahrkarte am Automaten beschaffen muss, dürfte in der Regel eine übersichtliche Führung mit mehreren Interaktionsschritten bevorzugen.

Diese Beispiele zeigen zugleich, dass Gestaltungskonflikte ein geeignetes Konzept sein können, um Mustersprachen aufzubauen, da beispielsweise eine hierarchische Struktur angelegt werden kann, bei der die beiden Gestaltungskonflikte „Breite vs. Tiefe“ und „Übersichtlichkeit vs. stufenweise Erschließung“ in dem Designkonflikt „Räumlichkeit vs. sequenzielle Erschließbarkeit“ zusammengefasst werden.

4 Schlussfolgerungen

Wie eingangs beschrieben, sollen uns Gestaltungskonflikte helfen, Muster zu beschreiben, um den Entwicklern die Auflösung von Gestaltungskonflikten zu erleichtern und zu einer besseren Gestaltung von Benutzungsoberflächen beizutragen.

Der erste beschriebene Gestaltungskonflikt, Ikonizität vs. Einfachheit, lässt sich mithilfe der aufgezeigten Prinzipien meist kontextfrei lösen, denn globale Icons sind in der Regel die bestmögliche Wahl. Das am Ende des Beispiels erwähnte Problem, das sich aus der Verwendung von der Lösung widersprechenden Styleguides ergibt, lässt den Schluss zu, dass sich die Gestaltungskonflikte aber auf unterschiedlichen Ebenen lösen lassen. Insofern ist es wichtig, dass Gestaltungskonflikte das Potenzial haben, den Entwicklern eine Gestaltungsorientierung zu vermitteln, in welcher Richtung bzw. mit welchen Konsequenzen sie nach Anpassungen und Erweiterungen suchen könnten. Das zweite Beispiel lässt sich jedoch fast nie kontextfrei lösen und bedarf je nach Einsatzort einer genauen Untersuchung durch die Entwickler bzw. Designer. Die Lösungshinweise geben jedoch den Entwicklern im Rahmen der beiden Extrempositionen einen guten Lösungsraum mit auf den Weg. Da Entwickler häufig Gestaltungskonflikte lösen müssen, liegt es nahe, dass durch die systematische Erfassung solcher Konflikte in einer Mustersprache nicht nur feste Lösungen vorgegeben werden, sondern auch ein erweiterbarer Lösungsraum, der zudem auch an den jeweiligen Kontext anpassbar ist. Dadurch können bei der Betrachtung der entsprechenden Konflikte Designer nicht nur analytisch, sondern auch prospektiv an die Lösungen ihrer Aufgaben herangehen.

Die Analyse der beiden Beispiele hat gezeigt, dass Konflikte auf unterschiedlichen Ebenen auftreten. Bisher lassen sich zwei dieser Ebenen identifizieren. Um diese zu unterscheiden, wenden wir folgende Sprachkonvention an:

Ebene 1 – Designkonflikte

Konflikte, die sich, ohne Berücksichtigung des jeweiligen Nutzungskontextes, lösen lassen, sollen als Designkonflikte bezeichnet werden und sind in der ersten Kategorie zusammengefasst. Das Beispiel zur Icon-Gestaltung gehört in diese Klasse, da die widerstrebenden Kräfte sich rein aus den Besonderheiten kognitiver Verarbeitung begründen und damit unabhängig von explizit von außen gesetzten Anforderungen sind. Eine Lösung für diese Konflikte lässt sich wissenschaftlich finden. Das bedeutet aber nicht, dass es sich um eine Vorgabe ohne Anpassungsspielraum handelt. In der Gesamtheit der Konflikte ist dies voraussichtlich der kleinste Teil, da eine optimale Lösung, die losgelöst vom jeweiligen Kontext für den Konflikt gilt, gefunden werden kann.

Ebene 2 – Zielkonflikte

Die zweite Ebene, die hier als Zielkonflikte bezeichnet wird, ist dadurch gekennzeichnet, dass die Lösung methodisch kontextbasiert gefunden werden kann. Je nach Priorität in dem jeweiligen Kontext können unterschiedliche Ansätze in den verschiedenen Szenarien die jeweils bessere Lösung darstellen. Das zweite Beispiel, „Räumlichkeit vs. Sequenzialität“, gehört in diesen Bereich. Die Herausforderung in dieser Kategorie besteht also aus dem Austarieren der jeweiligen Anforderungen im jeweiligen Kontext. Auch wenn Konflikte der zweiten Ebene nicht über eine optimale Lösung verfügen, so lassen sich doch mithilfe wissenschaftlicher Erkenntnisse Lösungsvorschläge bzw. -wege erarbeiten.

5 Ausblick

Gestaltung ist ein komplexer Prozess mit vielfältigen Variablen, die zur Anpassung gebracht werden müssen. Dabei beeinflussen sich die Wirkungen der jeweiligen Lösungsansätze und widersprechen sich in vielen Fällen sogar. Das von Christopher Alexander begründete Konzept der Mustersprachen beinhaltet das Konzept der Forces (Kräfte) als ein wesentliches Mittel, um Lösungsmuster zu finden, sie als solche zu verstehen und sie flexibel an neue und veränderte Bedingungen anpassen zu können. Die bisher entwickelten Mustersprachen in der Informatik haben bislang weitgehend auf dieses wichtige Konzept verzichtet. Damit ist es zwar zum ersten Mal gelungen, eine gewisse Systematik in Bezug auf die Kodifizierung von Gestaltungswissen zu erreichen, doch fehlt die auf eine konstruktive Anpassung und Weiterentwicklung ausgerichtete Grundlage.

Bei der Überlegung, wie das Konzept der Kräfte in dem Bereich der Gestaltung von Benutzungsoberflächen übertragen werden kann, liegt es nahe, auf bereits bestehende Einsichten und Begriffe zurückzugreifen. Dabei erscheint uns das Konzept der Gestaltungskonflikte ein potentieller Kandidat zu sein, um darüber Kräfte aufzuspüren und Gestaltungsmuster zu formulieren. Konflikte der ersten Ebene, die durch nachgewiesene, optimale Lösungen gekennzeichnet sind, geben Entwicklern einen guten Leitfaden zur Lösung an die Hand. Die zweite Ebene kann Entwicklern zwar keine konkreten optimalen Lösungen bieten, jedoch zeigen die beispielhaft vorgestellten Lösungsansätze einen Weg auf, um die Konflikte bzw. die gegensätzlichen Anforderungen auszutarieren.

Wie sich die softwareergonomischen Gestaltungskonflikte zahlenmäßig auf die einzelnen Ebenen verteilen bzw. ob es noch weitere Ebenen gibt, lässt sich noch nicht vorhersagen und

muss noch untersucht werden. Auf jeden Fall aber lassen sich mit den beiden Ebenen sowohl Erkenntnisse aus der Forschung als auch aus der alltagstauglichen Umsetzung kodifizieren.

Ein weiterer Punkt, der noch nicht angesprochen wurde, ist die Verbindung der einzelnen Konflikte untereinander. Alexander legt großen Wert auf die Beziehungen zwischen den einzelnen Mustern, da sich erst durch die Verbindung der Einzelteile eine vollständige Lösung erzielen lässt. Von daher gilt es weiterhin zu untersuchen, in welcher Beziehung die Konflikte zueinander stehen und ob bzw. wie die Einbeziehung weiterer Gestaltungskonflikte die Lösungsstrategien der bereits betrachteten Konflikte beeinflusst.

Mit Blick auf das zu gestaltende Artefakt und die Charakteristika des Gestaltungsproblems hat der vorliegende Beitrag deutlich gemacht, dass Gestaltungskonflikte notwendigerweise berücksichtigt werden müssen, möglicherweise aber auch für die Fundierung eines systematischen Gestaltungsansatzes hilfreich sind. Der Ansatz zu einer hypothesengeleiteten Technikgestaltung basiert auf der Einsicht, dass dies notwendig ist, weil allein durch Erhebungen bei den späteren Nutzern und ein methodisches Vorgehen bei der Umsetzung viele Aspekte einer guten Gestaltung nicht oder nur unzureichend erfasst werden können. Expertenwissen ist in jedem Fall erforderlich. Die Frage dabei ist, auf welcher Grundlage Informatiker hier einen Beitrag liefern können, ohne in das von Terry Winograd angesprochene Problem des amateurhaften Erschließens fremder Fachdisziplinen zu geraten: „But it is misleading to see the problem as one of offering an ‚interdisciplinary' education. We will not succeed at developing competence in design by turning computer students into amateur sociologists, amateur anthropologists, amateur psychologists and amateur organization theorists. Although it is certainly valuable to introduce them to the key insights that each of these disciplines has generated, there needs to be an integration – a way of turning a multi-disciplinary goulash into a background that makes sense in the context of the design tasks our students will encounter in the exercise of their profession." (Winograd 1990, 445)

Natürlich lässt sich mit den hier vorgestellten Beispielen nur begründen, dass es prinzipiell möglich scheint, auf der Basis des hier vorgestellten Ansatzes Musterbeschreibung auch systematisch anzulegen. Dafür sprechen zudem die Möglichkeit der Bildung von Hierarchien und das Aufzeigen von Abhängigkeiten zwischen verschiedenen Gestaltungsmustern. Dies allein ist aber nicht ausreichend. Insbesondere muss in weiteren Arbeiten noch gezeigt werden, dass es wirklich möglich ist, über die relevanten Gestaltungsbereiche hinweg diese Systematik voranzutreiben. Dabei ist die Frage zu klären, welche weiteren Konzepte tatsächlich zur Formulierung und Ausgestaltung von Gestaltungsmustern und den mit ihnen zusammenhängenden Kräften geeignet sind. Wir denken aber, dass die Beispiele zeigen, dass die Weiterentwicklung und die Suche nach neuen Möglichkeiten Erfolg versprechend sind.

Literaturverzeichnis

Alexander, C. (1964). Notes on the Synthesis of Form: Harvard University Press.

Alexander, C.; Ishikawa, S.; Silverstein, M. (1978). A pattern language: Oxford University Press.

Apple (2012). OS X Human Interface Guidelines.http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html#//apple_ref/doc/uid/20000957. Version: Juli 2012.

Arend, U.; Wandmacher, J. (1989). Gestaltungsprinzipien für Piktogramme und ihr Einfluß auf die Menüauswahl. In: Notizen zu Interaktiven Systemen 17:17.

Arnheim, R. (1972). Anschauliches Denken: DuMont.

Ballstaedt, S. (1997). Wissensvermittlung: Die Gestaltung von Lernmaterial: Beltz, PsychologieVerlagsUnion.

Bayle, E. et al. (1998). Putting it all together: towards a pattern language for interaction design: A CHI 97 workshop. ACM SIGCHI Bulletin 30(1):17–23.

Brennecke, A.; Keil-Slawik, R. (1995). Alltagspraxis der Hypermediagestaltung – Erfahrungen beim Einsatz des World Wide Web und Mosaic in der Lehre. Fachtagung „Software-Ergonomie´95 – Mensch-Computer-Interaktion – Anwendungsbereiche lernen voneinander“. Darmstadt, 20.–23. Februar.

Easterby, R. S. (1970). The Perception of Symbols for Machine Displays. In: Ergonomics 1(1):149–158.

Eco, U. (2002). Einführung in die Semiotik. Bd. 105: Utb.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (2004). Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software: Addison Wesley.

Jacko, J.; Salvendy, G. (1996). Hierarchical Menu Design: Breadth, Depth and Task Complexity. In: Perceptual and Motor Skills 82:1187–1201.

Keil-Slawik, R. (1990). Konstruktives Design: Ein ökologischer Ansatz zur Gestaltung interaktiver Systeme. Habilitation. Technische Universität Berlin.

Keil-Slawik, R. (1993). Software-Entwicklung als Lernprozeß. In: Arbeitsrecht im Betrieb. Frankfurt a. M., S. 507–514.

Keil, R.; Fleigl, L.; Geißler, S. (2006). MObiDig – Manipulierbare Objekte in digitalen Systemen.

Keil, R. (2011). Hypothesengeleitete Technikgestaltung als Grundlage einer kontextuellen Informatik. In: Breiter, A.; Wind, M. (Hrsg.): Informationstechnik und ihre Organisationslücken. Soziale, politische und rechtliche Dimensionen aus der Sicht von Wissenschaft und Praxis. Berlin: LIT-Verlag.

Minov, C. (2004). Design ergonomischer Benutzeroberflächen für Computeranwendungen in der Medizin. Technische Universität München, Universitätsbibliothek, Dissertation.

Mullet, K.; Sano, D. (1994). Designing visual interfaces: Communication oriented techniques: Prentice Hall PTR.

Nadin, M. (1988). Interface design: A semiotic paradigm. In: Semiotica 69(3-4):269–302.

Nielsen, J. (1994). Usability Engineering. Boston: Academic press.

Shneiderman, B.; Plaisant, C. (2004). Designing the User Interface: Strategies for Effective Human-Computer Interaction: Pearson Education.

Tidwell, J. (2005). Designing Interfaces – Patterns for Effective Interaction Design: O'Reilly Media, Inc.

Wandmacher, J. (1993). Software-Ergonomie. Bd. 2: Walter de Gruyter.

Winograd, T. 1990: What can we teach about human-computer interaction? In: Proceedings SIGCHI Conference on Human Factors in Computing Systems: Empowering People. Seattle (WA), S. 443–449.

Kontaktinformationen

Prof. Dr.-Ing. Reinhard Keil
Heinz Nixdorf Institut
Universität Paderborn
Kontextuelle Informatik
Fürstenallee 11
33102 Paderborn

+49 5251 60-6411
reinhard.keil@hni.uni-paderborn.de
koi.uni-paderborn.de/rks

Christian Schild
Heinz Nixdorf Institut
Universität Paderborn
Kontextuelle Informatik
Fürstenallee 11
33102 Paderborn

+49 5251 60-6416
christian.schild@uni-paderborn.de
koi.uni-paderborn.de/schild