

Erfahrungen und Thesen zur Entwicklung von Benutzeroberflächen

Auf Projekterfahrungen und aktuellen HCI-Forschungsthemen basierende Thesen von Paul Chlebek und Markus Hillebrand.

These 1: Mit einer Maschine kommuniziert man nicht, sondern man benutzt sie.

Mensch-Maschine-Kommunikation ist keine Kommunikation. Denn selbst wenn die Maschine zum Kommunikationsmedium wird, kommuniziert man immer noch nicht MIT der Maschine, sondern MITTELS der Maschine. Deshalb ist es zutreffender, von einer Benutzeroberfläche statt von Mensch-Maschine-Kommunikation zu sprechen.

Schon immer haben Menschen versucht Maschinen zu bauen, die soweit mit eigenem Wesen ausgestattet sind, dass man mit ihnen kommunizieren kann. Die Anwendungsentwicklung beschäftigt sich nicht mit diesem Bestreben und sollte dies auch in Zukunft nicht tun. Tischlein deck dich, Esel streck dich, Knüppel aus dem Sack sind Beispiele für Anwendungen – nicht für Maschinenwesen. Sie sind Werkzeuge, die auf Kommandos reagieren, der dialektischen Kommunikation unfähig. Diese ist im Anwendungsbau auch gar nicht wünschenswert. Mit einer Anwendungssoftware wollen wir einen Homunkulus und keinen künstlichen Mensch bauen.

HCI (Human-Computer Interface) ist keine Kommunikation, sondern die Bedienung von Funktionen. Wollen wir wirklich mit einem Radio, einem Navigationssystem, einem Lohnsteuerrechner kommunizieren? Diskutieren? Wozu? Wir könnten diese Systeme nicht überzeugen, sie zu etwas anderem überreden, sie würden uns nicht unterhalten, einen Witz machen, etwas ironisch formulieren, uns schmeicheln, uns Komplimente machen oder mit uns zu streiten beginnen. Sie würden auch nicht merken wenn wir sie belügen und betrügen.

Wir schließen zur Begründung an das Gedankenspiel den Turing-Test an: Erst wenn wir nicht mehr erkennen, ob es sich bei der Interaktion um einen menschlichen oder einen maschinellen Gegenüber handelt, wollen wir von Kommunikation sprechen.

Benutzeroberflächen sind hochkomplexe Bedienertafeln, nicht mehr und nicht weniger. Deren Bau orientiert sich an Kommunikation, weil Kommunikation die komplexeste Interaktionsdefinition liefert, zu der Menschen fähig sind. Aber will ich deshalb eine Benutzeroberfläche als Person anerkennen oder vor ihr Achtung haben? Soll ich von ihr verlangen, dass sie sich in meinen Kontext einfühlt? Möglicherweise ist deshalb z.B. die Adaptivität von Benutzerschnittstellen ein Problem - wir trauen der Maschine und ihrer Schnittstelle so ein Verhalten doch gar nicht zu und wahrscheinlich wollen wir das auch gar nicht! Wir erwarten vor allem, dass uns Funktionen schnell und effizient zur Verfügung stehen, egal in welchem mikropolitischen Zustand wir uns mit dem HCI befinden, ob es uns gerade toll findet, oder geschweige denn, in welcher Laune es gerade ist.

Da diese Definitionsdiskussion nun weder technisch noch philosophisch als abgeschlossen gilt, darf man sich in aller Freiheit streiten, inwiefern Mensch-Maschine-Interaktion Kommunikation ist oder nicht. Wir hoffen darauf, dass die Streitigkeiten Kriterien hervorbringen, die uns helfen besser zu verstehen, was wir mit dem Bau von Benutzeroberflächen eigentlich erreichen wollen und was nicht.

These 2: Für die Bedienung interaktiver Systeme genügt das EVA-Prinzip nicht.

Mit bloßer EVA (Eingabe-Verarbeitung-Ausgabe) Logik lässt sich eine interaktive Benutzeroberfläche nicht beschreiben. Dies ist deswegen nicht möglich, weil eine moderne Benutzeroberfläche Erstens aus einer Vielzahl von Eingabe-Verarbeitung-Ausgabe Sequenzen besteht, Zweitens diese Sequenzen untereinander interferieren und Drittens Eingaben wie Ausgaben unvollständig sein können.

Die zu klärenden Fragen sind:

- Was ist ein interaktives System?
- Und wie erfolgt die Interaktion zwischen Nutzer und Anwendung aus der HCI-Perspektive?

Ein interaktives System ist nach unserer Modellvorstellung ein Gebilde aus Eingabeempfängern (**Rezeptoren**), Interpretations- und Verarbeitungsregeln (**Prozessoren**) und Ausgabemechanismen (**Manifestoren**).

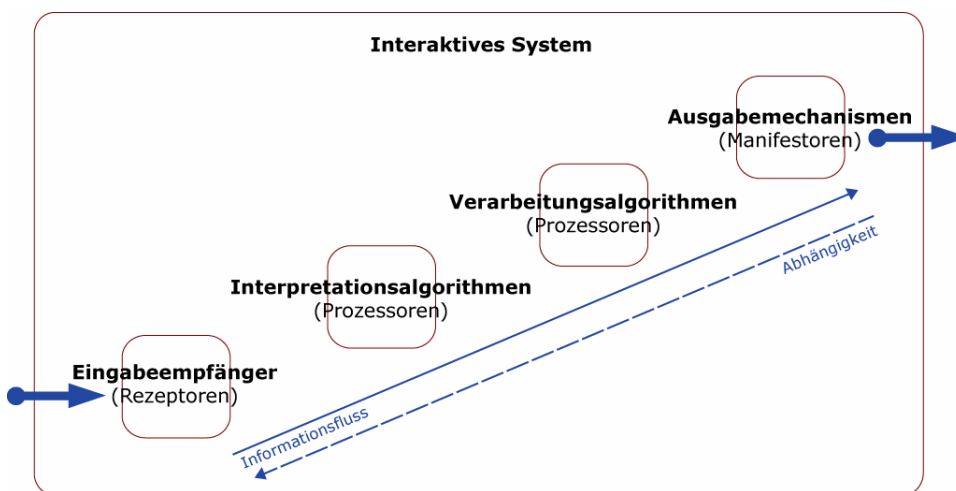


Abbildung 1: Kommunikationsmodell für das interaktive System „Anwendungssoftware“

Entsprechend dem vereinbarten Systemzweck ist das interaktive System zu einer geordneten Manifestation des Empfangs, zur Interpretation und zur Verarbeitung von Eingaben in der Lage, ohne dabei beschädigt oder in seiner Funktionalität verändert zu werden.

Ein interaktives System reagiert (bei zweckgemäßer Nutzung) auf eine Eingabe mit einer für den Nutzer wahrnehmbaren Veränderung seiner Erscheinung. Die Interpretations- und Verarbeitungsregeln sowie die Ausgabeinhalte eines interaktiven Systems sind nur **mittelbar** über die Eingabeempfänger erreichbar.

Um diese Lücke zu schließen wird ausgehend von Mensch-Mensch-Kommunikation das EVA-Prinzip um ein Rahmenwerk für die Interaktion von Softwaresystemen mit den Nutzern erweitert.

Das Rahmenwerk führt die HCI-Perspektive der Interaktion durch Erweiterung der EVA-Perspektive von Eingabe und Ausgabe ein. Die Erweiterung erfolgt durch „**InMitteilung**“ in Richtung der Benutzeraktion und durch „**OutMitteilung**“ in Richtung der vom Benutzer erwarteten Reaktion des interaktiven Systems.

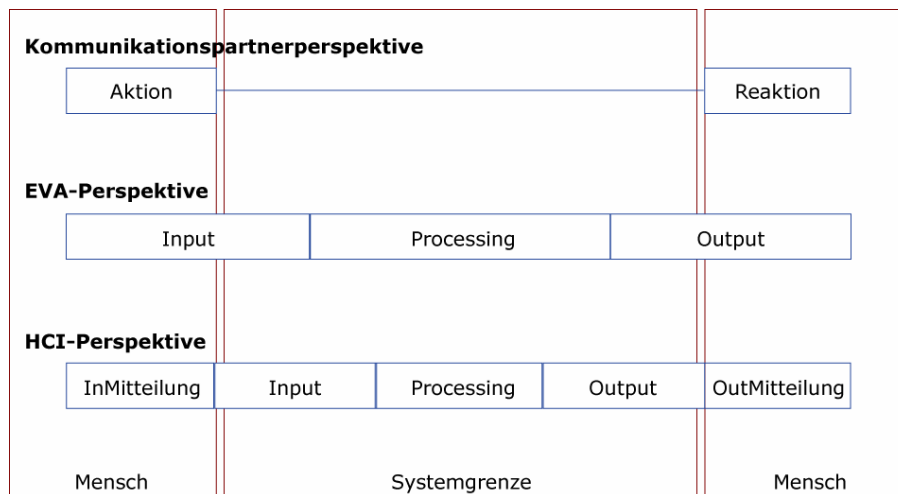


Abbildung 2: Rahmenwerk für die Mensch-Maschine Interaktion

Die tatsächliche Funktion einer interaktiven Anwendung ist nur mittelbar über „InMitteilung“ und „OutMitteilung“ erreichbar und wahrnehmbar. Ein unmittelbarer „Input“ und „Output“ zwischen Anwendungsfunktionen und Anwender ist nicht möglich.

Daraus folgt, dass Teile der Anwendung die Aufgabe haben, von InMitteilung nach Input und von Output nach OutMitteilung zu übersetzen. Diese interpretierenden Übersetzungsalgorithmen sind das HCI der Anwendung.

Die Transition InMitteilung→Input ist die Interpretation des Versuchs einer Eingabe des Nutzers an die Anwendung. Das Processing erfolgt auf Basis des mittelbar durch Interpretation der InMitteilung gewonnenen Inputs.

OutMitteilung ist der auf InMitteilung oder auch nicht auf InMitteilung beruhende Versuch der Manifestation des Processing-Outputs in irgendeiner für den Nutzer wahrnehmbaren Form.

Die vage Definition dieser Begriffe macht deutlich, dass das Verhalten interaktiver Anwendungen nur dann vom Nutzer leicht zu interpretieren und zu steuern ist, wenn die Interpretationsregeln der Anwendung ihrerseits eindeutig und einfach nachvollziehbar sind.

Die Bedienbarkeit einer interaktiven Anwendung hängt von der Interpretierbarkeit ihres Verhaltens durch den Menschen ab. Die Schnittstellen zwischen den Interaktionspartnern (Nutzer und Anwendung) bestehen nicht auf funktionaler Ebene, sondern erfolgen ausschließlich mit Hilfe der Wechselwirkung von InMitteilung und OutMitteilung.

Deshalb müssen diese Regeln für die InMitteilungToInput und OutputToOutMitteilung Transitionen eindeutig formuliert werden können. Aus diesem Bedarf leitet sich die Notwendigkeit einer normierten, formalen **HCI-Spezifikationssprache** ab.

These 3: Die Informationsmodelle der HCI-Spezifikation und der HCI- Programmierung sind verschieden.

Die Information in einer HCI-Spezifikation muss schnell änderbar formuliert sein, um auf wechselnde Anforderungen und Kundenwünsche einzugehen. Der Unterschied zwischen Oberflächen-Spezifikation und Oberflächen-Programmierung ist, dass die Spezifikation skizzenhaft und (weitgehend) implementierungsunabhängig bleiben will, während die Oberflächenprogrammierung die Klärung der technischen Implementierungsdetails anstrebt.

Feinspezifizieren ist der Übergang zum programmieren. Beim programmieren von Benutzeroberfläche kann man sich der beim spezifizieren gesammelten Informationen zum generieren von Programmcode oder zur Verwendung als ladbare Steuerungsdaten bedienen.

Anders gesagt: Die Spezifikation dient dazu, die Vorstellungen und Wünsche der verschiedenen Stakeholder zu bündeln und verfügbar zu machen. Ziel ist die Vereinigung dieser Angaben, um den nachfolgenden Prozess der Umsetzung zu stabilisieren. Einigkeit der Stakeholder und detaillierte Umsetzung der Anforderungen sind zwei verschiedene Ziele, daher gibt es auch die Berechtigung zweier verschiedener Informationsmodelle.

Ziel einer modellgetriebenen Entwicklung soll sein, diese beiden Informationsmodelle zueinander kompatibel zu machen, um die Transformationen der Informationen von einem zum anderen Modell weniger aufwendig, weniger konfliktreich und vor allem nachvollziehbarer zu gestalten. Durch diese konsequente Anforderungsumsetzung steigt der Erfüllungsgrad der Erwartungen an die Ergebnisse.

Natürlich gilt für diese Prozessverbesserung „Garbage in - Garbage out“. Doch durch einen im Informationsfluss verbesserten, weil durchgängigeren Entwicklungsprozess, haben auch die guten Anforderungen eine höhere Chance im Endprodukt umgesetzt zu werden.

These 4: Unterspezifikation ist in Ordnung.

Man muss ein gewisses Vertrauen in den Leser der Spezifikation haben. Beispiel: Ein Architekt zeichnet die Skizze eines Gebäudes mit Hintergrund, Schattierungen etc., um dieses Modell mit dem Kunden abzustimmen. Dann zeichnet er die notwendigen Detailpläne des Gebäudes. Diese Detailpläne müssen so genau sein, dass der Bauingenieur und die Leute auf dem Bau das Gebäude bauen können. Wenn Architekt und Bautrupp bereits miteinander gearbeitet haben, muss vielleicht weniger spezifiziert werden, weil die Lücken durch korrekte Annahmen ausgefüllt werden.

Soll der Bauauftrag ausgeschrieben werden, z.B. an einen Bautrupp aus Indien, muss die Spezifikation entsprechend inderpretationssicher :-) gestaltet sein.

Zurück zum Softwarebau: Eine totale Detaillierung (und was wäre das auch?) ist speziell im Software-Bereich ein Grenzfall zur Implementierung und daher ökonomisch nicht sinnvoll. Programmier-Quellcode ist ja bereits eine ausdetaillierte Spezifikation für eine Ablaufmaschine. Daher muss man sich fragen: Wozu zwei Spezifikationen erstellen und pflegen?

Wie tief spezifiziert werden muss, hängt also vom Projektkontext und dem Interpretationsvertrauen in den oder die Umsetzer ab und muss unter diesen Gesichtspunkten vom HCI-Architekten abgeschätzt werden. Außerdem gilt: Vertrauen ist gut, Testen ist besser.

These 5: State-Charts sind für die Modellierung von HCIs nicht genug.

Im Bereich der UIST (User-Interface-Software-Technologies) wird zur Modellierung der Ablauflogik überwiegend auf State-Charts verwiesen. Historisch betrachtet sind State-Charts eine Erweiterung der allgemeinen Zustandsmaschinen. Sie bilden einen festen Bestandteil der allgemeinen und populären Modellierungssprache UML 2.0.

In der Praxis der HCI-Modellierung stoßen wir bei der Verwendung von State-Charts schnell auf Schwierigkeiten. Trotz der Möglichkeit Zustandshierarchien zu bilden, werden State-Charts im HCI-Bereich schnell groß und unübersichtlich.

Wir sehen hier Optimierungs- und Erweiterungsbedarf und haben einige Methoden entwickelt, wie sich State-Charts als Modellierungssprache verbessern lassen:

- Einführung von **Eingangs- und Ausgangsbedingungen** bei den Zuständen. Dies erlaubt auch abstrakte Zustände bzw. Dispatch-Zustände und minimiert die Anzahl der Transitionen.
- Unterstützung von Standardabläufen durch **Bildung von geordneten Sequenzen**. Dies unterstützt auch relative Transitionen (springe zum nächsten Zustand oder zum vorherigen Zustand etc.)
- Unterstützung von **Invarianten** zur besseren Abbildung von Software- bzw. Produkt-Familien.
- **Verweisende Vererbung** von Zuständen und Event-Typen über die Hierarchie-Bildung hinaus.
- Detaillierung der **Schnittstellen** zwischen State-Charts und ihrer Umwelt.
- Einführung von Name-Lookups zur **relativen Namens-Adressierung** in den State-Charts („Präpositionen bilden“).
- **Selbstbezügliche Aktionen** (Stop-Transition, Berechnung des Zielzustandes).
- Bereitstellung von Bedingungen, Texten, Grafiken, Listen, Formaten, Vokabelsätzen und Schnittstellen als **Ressourcen- und Kontextpool**, um die Ablauflogik von dieser Definitionsarbeit zu entschlacken und deren Wiederverwendung anzubieten.

Für den HCI-Bereich sehen wir weitere spezifische Verbesserungsmöglichkeiten:

- Bessere Unterstützung von Multimodalität durch qualifizierte HCI-States (GUI-State, Voice-State etc.)
- Vererbung von States muss nicht nur eine Verhaltensvererbung sein, sondern kann gleichzeitig auch die Widgetschachtelung vererben und spezialisieren.
- Unterstützung eines HCI-spezifischen Ablaufstacks. Wir denken hier an Service-Zustände mit optionaler Rücksprungmöglichkeit.
- Im HCI-Bereich gibt es über Enter/Exit/Effekt weitere spezifische Standard-Aktivitätstypen z.B. OnCommand, OnFocus, OnDefocus etc.
- Gliederung des Gesamtmodells in die typischen Perspektiven der Modellierer, die sich durch die Arbeitsteilung ergeben: Ressourcen, Umwelt-Schnittstellen, Strukturgliederung, Ablauflogik und Präsentation.
- Der HCI-typische architektonische Schnitt zwischen Kontext, Maske und Eingabefeld kann in der Modellierungssprache erfasst werden. Er verbessert die Orientierung im Modell und macht es anschaulicher.

Die genannten Verbesserungen dienen in erster Linie dazu, die Komplexität zu senken um HCI-Modelle bzw. Spezifikationen überschaubarer und handhabbarer zu gestalten.

Wir erwarten nicht, dass diese Vorschläge morgen in die UML-Version x.y einfließen. Aber wir wollen aufzeigen, dass es genügend Bedarf und Möglichkeiten gibt, den jetzigen State of the Art weiterzuentwickeln. Unter Praktikern gesagt wären wir jetzt erst einmal froh, wenn sich die State-Charts im UST-Bereich als Standardbasis weiträumig durchsetzen würden.

Eine Forderung bei der Entwicklung der genannten Verbesserungen ist, dass sie so gestaltet sein müssen, dass wir sie mit Hilfe der modellgetriebenen Software-Entwicklung heute schon nutzen können, ohne mit den State-Charts brechen zu müssen.

These 6: Steps statt States.

Es interessiert nicht, wie es einer Benutzerschnittstelle geht und in welcher Verfassung sie sich befindet (kontra Statemaschinen). Vielmehr interessiert,

welchen nächsten Schritt man anwählen muss um an ein Ziel zu kommen (pro Workflowschritte).

Eine wichtige Grundüberlegung zur Ergonomie eines Systems hat mit der Art der Erwartungshaltung des Benutzers dem System gegenüber zu tun: Will er ein Spiel oder ein Wissensportal exploratorisch durchstreifen. Oder möchte er eine Funktion auslösen, hat er also ein festes Ziel das er anstrebt?

Diese Überlegung hilft uns, mindestens zwei unterschiedliche Charakteristika in der Art der Benutzerführung zu erkennen. Was beide Charakteristika wieder verbindet, ist die Wichtigkeit der Entscheidung, wie man den Benutzer durch das System führen möchte. Diese Entscheidung beruht darauf, dass man die wahrscheinlichen von den weniger wahrscheinlichen, aber dennoch möglichen, Benutzeraktionen unterscheiden kann. Ein so durchdachtes System reduziert für die meisten Benutzer die Anzahl der Klicks die sie brauchen, um an ihr Ziel zu gelangen.

Wir meinen nun, dass rein zustandsorientierte HCI-Spezifikationen (State-Charts!) diese Aspekte unterschlagen. Dies ist als Informationsverlust zu werten, da zumindest der Grafiker darauf angewiesen ist zu verstehen, welche Elemente besondere Beachtung finden sollen und welche Elemente eher in den Hintergrund zu rücken sind.

Wir möchten also die Planung auf die präferierten Pfade durch Benutzeroberflächen fokussieren und nicht alle Pfade als gleich gültige Wege darstellen, wie es bei State-Charts der Fall ist. Daher wollen wir lieber über Schritte als über Zustände reden. Wir fordern HCI-Spezifikationssprachen, die dies berücksichtigen.

These 7: Die eine HCI-Spezifikationssprache für alle möglichen HCIs gibt es nicht.

Der Kommunikationsbegriff kann zur Zeit nicht verbindlich definiert werden.

Im Duden der Informatik steht beispielsweise: „Für Menschen ist Kommunikation etwas so Selbstverständliches, dass hierfür früher kein eigenes Wort existierte. Mit der Datenverarbeitung, den Medien, den höheren Rückkopplungen im Arbeitsleben und zunehmenden gesellschaftlichen Diskussionsprozessen entstand etwa im Jahre 1960 das Wort Kommunikation, um diese Vielfalt an Austauschmöglichkeiten und deren Beziehungen und Abhängigkeiten zu beschreiben.“ Wikipedia in der dt. Ausgabe folgt insoweit der allgemeinen Diskussion, indem es unter „Definition und Zusammenhang“ als Erstes feststellt: „Eine allgemein anerkannte Definition des Begriffs existiert nicht.“

Es gibt daher nicht die eine Theorie die uns letztendlich erklärt, was in diesem Bereich alles möglich ist. Daher gibt es auch keine festen Begriffe, aus denen wir die Details „Der Einen Wahrheit“ in Bezug auf die Spezifikation von Kommunikation einfach Top-Down ableiten können.

Den Kommunikationspartnern eines Dialoges geht es meist darum, unbekannte Inhalte gegenseitig verfügbar zu machen. Wozu sollte man sonst Informationen austauschen wollen? Mal die soziale Funktion vertraute Geräusche zu machen ausgenommen.

Wir denken, dass die Orientierung von Mensch-Maschine-Interaktionen an Mensch-Mensch-Kommunikation in etwa soweit zum Erfolg führt, wie Otto Lilienthal aus dem Vorbild von Vogelflügeln Flugzeugtragflächen konstruiert hat. Es ist nicht ganz richtig - aber auch nicht falsch.

Fakt ist: Es gibt weder die eine Theorie der Mensch-Mensch-Interaktionen, sprich Kommunikation, noch die eine Theorie der Mensch-Computer-Schnittstelle.

Wir müssen uns daher Bottom-Up, also empirisch, zu größeren Erkenntnissen hocharbeiten. Wir können hierzu auf Erfahrungen mit vorhandenen Benutzeroberflächen zurückgreifen, wir kennen gebräuchliche Bedienungsparadigmen

und übliche Usability-Phänomene. Wir können auf viel Wissen zurückgreifen um den notwendigen Umfang einer HCI-Spezifikationsprache in einem bestimmten Bereich abschätzen zu können.

Wir sind aber nicht davor gefeit, dass morgen oder übermorgen jemand mit etwas völlig Anderem daherkommt: neue Technologien, neue Bedienkonzepte, neue Anforderungsarten, neue Vorgehensweisen.

Die zukünftige Entwicklung des HCI-Feldes bleibt daher äußerst spannend.

pch, 08/2004
last revised 01/2006

Paul Chlebek
HCI • GUI • UML • OOD • MDA • XSLT

mailto: pc@benutzeroberflaeche.de
phone: +49 173 106 1830

<http://www.projectpeople.net/chlebek/>
<http://www.benutzeroberflaeche.de>

Markus Hillebrand
mailto: himself@markus-hillebrand.de