



5 User-Centered Design

5.1 Der Entwurfszyklus

User-Centered Design (UCD)

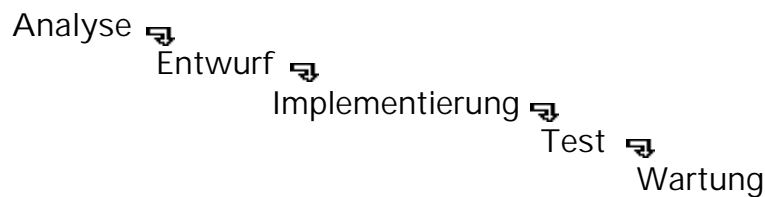


- Problem:
Der normale Software-Entwicklungsprozess kann leicht an den Bedürfnissen der Benutzer vorbeigehen
- Idee:
Benutzer von Beginn bis Ende der Entwicklung mit einbinden
- Mittel:
Fragebögen, Interviews, kommentierte Durchläufe, Benutzbarkeitstests, ...
- Ziel:
Benutzbarere, also erfolgreichere Systeme

„Normale“ Softwareentwicklung



- Um UCD zu verstehen, muss man den normalen Entwicklungsprozess kennen
 - Wasserfallmodell, seit 80er Jahren etabliert
- Phasen (normalerweise nacheinander):



- Jede Phase liefert Meilenstein
 - Dokument, SW



Folie 197

Wasserfallmodell: Analysephase



- Ergebnisse (Meilensteine):
 - Systemdefinition
 - Problem, Rechtfertigung, Ziele, Rahmenbedingungen, Funktionen, Benutzer, Systemumgebung, Lösungsstrategie, Prioritäten, Akzeptanzkriterien, Referenzen, Glossar
 - Projektplan
 - Entwicklungsmodell, Meilensteine, Arbeitsteilung, Kostenschätzung, Werkzeuge, Programmiersprachen, Testanforderungen, Dokumentliste, Lieferungsform, Installation, Training, Wartung, Bezahlung



Folie 198

Wasserfallmodell: Entwurfsphase



- Ergebnisse (Meilensteine):
 - Anforderungsspezifikation
 - Produktübersicht, Entwicklungs- und Laufzeitumgebung, externe Schnittstellen und Datenfluss, Benutzeranzeigen+ Kommandos, Funktionen, Leistung, Fehlerbehandlung, Untermengen+ Prioritäten, absehbare Erweiterungen, Funktions- und Leistungstests, Dokumentationsstandards, Entwurfsrichtlinien, Referenzen, Glossar
 - (Vorläufiges) Benutzerhandbuch
 - Einleitung (Produktübersicht, Zweck, Terminologie, Funktionen, Anzeigen), Einstieg (Programmstart, Hilfe, Beispieldurchlauf), Programmzustände (Kommandos, Dialoge, Ausgaben), weitere Funktionen, Syntax+ Optionen



Folie 199

Implementierungs-, Test- & Wartungsphasen



- Ergebnisse der Implementierungsphase:
 - Programmcode
 - Testplan
- Ergebnisse der Testphase:
 - Inspizierter Programmcode
 - Dokumentation der Einzel- und Systemtests
 - Auslieferung
- Ergebnisse der Wartungsphase (open end):
 - Verbesserungen, Anpassungen und Korrekturen von Programmcode und Dokumentation



Folie 200

Probleme des Wasserfallmodells

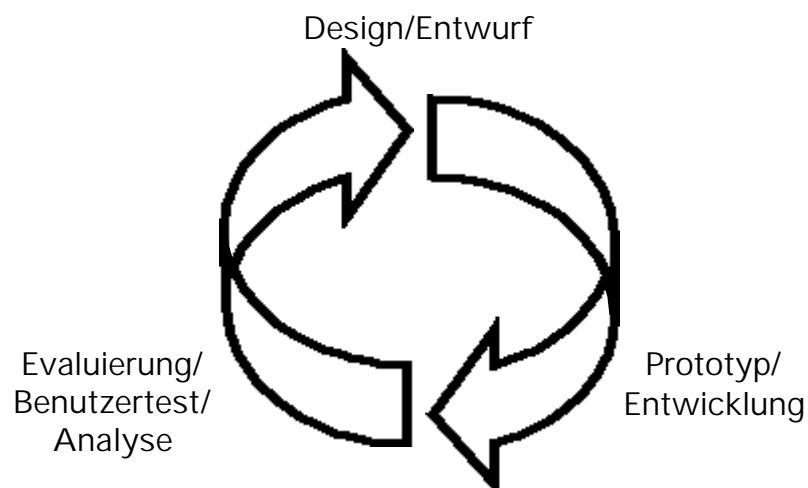


- Phasen nur im Idealfall so klar trennbar
- Praxis erfordert oft Rückschritte
 - Anforderungen ändern sich
- Durch Festhalten an der Phasenstruktur werden falsche Annahmen oft zu spät erkannt oder sind nicht mehr gut korrigierbar
- Die gewählten Benutzungsszenarien sind oft zu abstrakt, nicht konkret/realistisch genug



Folie 201

Besseres Modell für UCD: Der Entwurfszyklus



Folie 202

Der Entwurfszyklus



- Reale Projekte durchlaufen in Entwicklung mehrere solche Zyklen, bis das Design „stimmt“
- Mit jeder Iteration wird
 - Das Design präziser und konkreter
 - Der Prototyp technisch anspruchsvoller und detaillierter
 - Das Benutzerfeedback auf kleinere Probleme konzentriert
- Die ersten Iterationen sollten grobe Fehler ausbügeln, spätere kleinere korrigieren



Folie 203

UCD-Techniken in der Analyse



- Benutzer in alle Phasen miteinbeziehen, mit Schwerpunkt am Projektanfang in Analyse:
 - Zielbenutzergruppe herausfinden
 - Wer sind die typischen Benutzer? (Denken)
 - Funktionsbedarf herausfinden
 - Was wollen diese typischen Benutzer vom System? (Fragen)
 - Plagiarisieren
 - Von existierenden Lösungen lernen (Abgucken)
 - Taskanalyse
 - Typische Aufgaben modellieren, z.B. mit GOMS (Fragen)
 - Bearbeitungszeiten, Lernzeiten abschätzen (Denken)



Folie 204

UCD-Techniken im Entwurf



- Prototyp bauen
 - Auf Basis des ersten Systementwurfs
 - Zweck:
frühes Feedback von Benutzern
 - Schwerpunkt auf Interaktion, nicht auf Funktionalität (Wizard-of-Oz-Technik)
 - Ggf. mehrmals aufgrund des Feedbacks überarbeiten (iteratives Design)
- Benutzbarkeit testen (Usability Studies)
 - am Prototypen und fertigen Produkt



Folie 205

Nochmal ganz deutlich:



Die ersten 2 Fragen, die man sich beim Entwurf einer Benutzerschnittstelle stellen muss, sind:

1. Wer sind die Benutzer?
2. Was wollen sie vom System?

Viele Projekte scheitern, weil diese Fragen ignoriert wurden!



Folie 206



5 User-Centered Design

5.2 Benutzerbefragungen

Die Benutzer-Zielgruppe



- Ein System muss nützliche Funktionen erfüllen
 - sonst will es keiner
- Aber:
Die Funktionen müssen sich nahtlos in den Arbeitsprozess und das sonstige Tätigkeitsumfeld der meisten Benutzer einfügen
 - sonst macht sich keiner die Mühe, sie zu benutzen – Kosten/Nutzen!
- Also:
Man muss den „typischen Benutzer“ kennen: **Know the User!**



Folie 208

Die Benutzer-Zielgruppe



- Im UCD muss am Anfang versucht werden, einige reale Menschen zu finden, die Interesse daran haben, das geplante System zu benutzen
- Wenn das nicht gelingt, ist etwas faul! (Denn wer sollte es dann erst kaufen?) – Und wer will schon etwas entwickeln, das keiner haben will?



Folie 209

Wie findet man Zielbenutzer?



- Viele „haben keine Zeit“
- Problem:
Dann werden sie auch keine Zeit haben, das fertige System zu verwenden!
- Also:
Benutzer motivieren und von der Nützlichkeit des Systems überzeugen
- Technik-Freaks sind oft williger
- Incentives (Bestechung) helfen auch
 - Gummibärchen, T-Shirts, Kaffeetassen – oft besser als Geld



Folie 210

Beliebte Ausreden



- „Mein System ist für jeden nützlich!“
 - Erfahrung:
Leider sind solche Systeme oft für niemanden nützlich
 - Also:
Warum nicht wenigstens ein paar Leute finden, für die es nützlich ist?
- „Ich bin auch jemand, das reicht doch!“
 - Würden Sie es WIRKLICH selbst täglich benutzen? (Erschreckend oft: Nein!)
 - Außerdem:
Entwickler sieht Nutzen nach langem Nachdenken, aber Benutzer wird ihn so schnell nicht erkennen



Folie 211

Wie unterscheiden sich Benutzer?



- Erfahrung: wichtigstes Kriterium
- Komplette Neulinge
 - kein Wissen über Aufgaben oder Interface
- oder Erstbenutzer
 - kennen Aufgabengebiet, aber nicht das Interface
- Evtl. Benutzungssängste, erfordern umfangreiche Hilfe, kleines, konsistentes Vokabular, einfaches Interface, wenig Funktionen, viel Feedback (auch bei Fehlern), viel Schulungsmaterial



Folie 212

Wie unterscheiden sich Benutzer?



- Durchschnittlich erfahrene Benutzer
 - kennen Aufgabe gut, Interface mäßig, vergessen Funktionen
- Benötigen klare Menüstrukturen, Konsistenz, Sehen und auswählen statt erinnern und eingeben, weiterhin Fehlerschutz



Folie 213

Wie unterscheiden sich Benutzer?



- Benutzerexperten, regelmäßige Benutzer
 - kennen Aufgabe & Interface gut
- Geschwindigkeit & Effizienz, kurzes, nichtstörendes Feedback, Shortcuts, Makros, individuelle Konfiguration und Erweiterbarkeit



Folie 214

Weitere wichtige Informationen über Benutzer



- Hintergrund
 - Name, Alter, Geschlecht, Nationalität, Ausbildung, Einkommen
- Computererfahrung
 - spezielle Applikationen, Dauer, Wissenstiefe - siehe vorige Seite, Vertrautheit mit speziellen Funktionen (Drucken, ...)
- Arbeitsaufgaben
 - Führungsaufgaben, Entscheidungsbereich, Motivation



Folie 215

Weitere wichtige Informationen über Benutzer



- Persönlichkeit
 - intro-/extrovertiert, Risikobereitschaft, „early/late adopter“, systematisch/spontan
- Gefühl nach Benutzung
 - verwirrt/klar, frustriert/kontrolliert, gelangweilt/angeregt, ggf. Gründe für Systemablehnung (unpassend, zu komplex, zu langsam, ...)



Folie 216

Wie stellt man Fragen?



- Fragen müssen klar verständlich sein
- Antwortmöglichkeiten müssen logisch sein
- Also: nicht zu allgemein fragen
 - „Wie zufrieden sind Sie mit dem Interface?“
- Antworten sollten quantifizierbar sein
- Schlechte Fragen liefern unbrauchbare oder sogar falsche Aussagen



Folie 217

Wahre Antworten?



- Viele Fragen werden von Benutzern nicht wahrheitsgemäß beantwortet
 - Grund:
Unkenntnis, Verschätzen, Peinlichkeit
 - Beispiel:
Englische Umfrage ergab bei Männern im Mittel drei mal so viele unterschiedliche Sexualpartner wie bei Frauen...
 - Folge:
Fragen müssen sorgfältig, manchmal indirekt formuliert werden



Folie 218

Beispiele für sinnvolle Fragestellungen



- Die Systemkommandos sind einfach zu benutzen.
- Ich habe das Gefühl, die Kommandos gut zu kennen und kompetent zu benutzen.
- Die Fehlermeldungen helfen mir, das Problem zu lokalisieren.
- Es gibt zu viele Optionen und Spezialfälle.



Folie 219

Beispiele für sinnvolle Fragen



- Die Kommandos könnten wesentlich vereinfacht werden.
- Ich kann mir die Kommandos schlecht merken und muss häufig im Handbuch nachsehen.
- Bei Problemen frage ich jemanden, der sich auskennt.



Folie 220

Beispiele für sinnvolle Antworten



- Für obige Fragen:
 - 1: Stimme voll zu.
 - 2: Stimme weitgehend zu.
 - 3: Neutral.
 - 4: Stimme eher nicht zu.
 - 5: Stimme überhaupt nicht zu.
- Quantifizierung erlaubt statistische Auswertung
 - Mittelwert, Standardabweichung



Folie 221

Was soll Befragung liefern?



- Profil des Benutzers
- Profil der Tätigkeiten
- Problembereiche im System
- Ideen für Verbesserungen
- In unserem Fall:
Ein Bild der Aufgabe,
die das eigene System erfüllen soll.



Folie 222

Zusammenfassung



- Traditionelles Wasserfallmodell:
Analyse-Entwurf-Implementierung-Test-Wartung
- User-Centered Design:
Benutzer in allen Phasen involvieren, um
Benutzbarkeit zu optimieren,
iterativer Entwurfszyklus
- Erster Schritt:
Wer sind Benutzer, was wollen sie?
- Fragebögen liefern Benutzer- und
Aufgabenprofil sowie Problemstellen
- Sorgfältiges Fragebogendesign erforderlich



Folie 223



5 User-Centered Design

5.3 Entwurf und Design-Guidelines

Entwurfsentscheidungen



- Entscheide:
Wer (Benutzer) soll mit dem System was (Tasks/Funktionen) tun?
- Von Benutzer-Feedback beeinflusst
- Identifiziere zunächst typische Beispieltasks, die anfallen (anfangs noch ohne das neue System)
- Basiere dann Funktionen und Design des (neuen) Systems auf diesen Beispieltasks
- Berücksichtige dabei Design-Guidelines



Folie 225

Guidelines



- Hinweise zur Gestaltung interaktiver Systeme
- Zweck:
Dem Benutzerschnittstellen-Designer Richtlinien & Checklisten an die Hand geben
- Behandeln nur Entwurfsentscheidungen;
User-Centered Design wird als Methode vorausgesetzt



Folie 226

Formen von Guidelines



- Allgemeine Designrichtlinien
- Spezifische „Wenn-Dann“-Szenariensammlungen
- Style Guides
 - Definition eines „Look&Feel“-Standards
 - z.B. Apple Macintosh Human Interface Guidelines, IBM CUA, OSF/Motif Style Guide
- Implementierte Guidelines (Software-Toolkits)
 - z.B. OSF/Motif Widget Set, Java „Metal L&F“
- Design Patterns (kommen noch)



Folie 227

Die 9 Goldenen Regeln des Interface-Designs



1. Halte die Schnittstelle einfach!
2. Verwende die Sprache der Benutzer!
3. Achte auf Konsistenz und Vorhersagbarkeit!
4. Liefere Rückmeldungen!
5. Minimiere die Gedächtnisbelastung!
6. Vermeide Benutzerfehler, behandle sie hilfreich, erlaube Undo!
7. Biete klare Ausgänge und geschlossene Dialoge!
8. Stelle Hilfe und Dokumentation zur Verfügung!
9. Biete Experten Abkürzungen an!



Folie 228

1. Halte die Schnittstelle einfach!



- Wichtigste Regel
- Erster Entwurf meist zu kompliziert/
umständlich gedacht;
Einfachheit erst durch Redesign
- Vermeide „Creeping Featurism“:
 - System soll immer mehr Funktionen bieten
 - Bedienbarkeit darf darunter nicht leiden!
 - Erfahrung: 80% der Benutzer nutzen nur 20%
der Funktionalität (Beispiel: Word)
 - Formuliere Ziel:
Nächste Version nicht noch mehr Funktionen,
sondern einfacher zu bedienen
 - gliedere notfalls Funktionsmengen in
Unterdialoge aus



Folie 229

2. Verwende die Sprache der Benutzer!



- Begriffe und Konzepte sollten aus der
Anwendungsdomäne, nicht aus der
Informatikdomäne stammen
- Terminologie sollte bei anfänglicher
Benutzerbefragung und Taskanalyse erfasst
werden
- Beispiel:
„Datei “ sagt einem Architekten und
Computerneuling nichts, „Zeichnung“
hingegen schon
- Betrifft nicht nur Worte für einzelne
Objekte, sondern auch für Arbeitsvorgänge
(„Auftrag“) etc.



Folie 230

3. Achte auf Konsistenz und Vorhersagbarkeit!



- Konsistenz muss auf vielen Ebenen gelten
 - Ähnliche Kommandos für ähnliche Situationen
 - Einheitliche Terminologie in Menüs, Dialogen und Hilfeseiten
 - Einheitliche Zeichensätze, Layout, Farben, Groß-/Kleinschreibung etc. im gesamten System
 - Nur wenige, verständliche Ausnahmen, z.B.
 - Kein Echo bei Passworteingabe
 - Extraabfrage vor Löschen



Folie 231

Vorhersagbarkeit



- Folge dem „Principle of Least Surprise“:
 - Das System sollte immer so reagieren, wie es den Benutzer am wenigsten überrascht (und in der Folge verwirrt, verärgert, ...)
- Das System sollte nichts Unerwartetes tun
 - ... oder Aktionen unerwartet schwer machen („Man muss das doch irgendwie doppelseitig ausdrucken können!“)
- Benutzer (v.a. Experten) wollen stets das Gefühl der „Kontrolle“ über das System haben:
 - Sie initiieren Aktionen, und das System reagiert



Folie 232

4. Liefere Rückmeldungen!



- Bekannt aus Vorlesung (Aktionszyklus):
 - Komplettes und kontinuierliches Feedback überbrückt Bruchstelle „Aktion“//„Bewertung“
- Jede Benutzeraktion braucht Rückmeldung
 - Unaufdringlich für kleine/kurze/häufige Aktionen
 - Tastendruck, Menüauswahl
 - Auffälliger für wesentliche/lange/seltene Aktionen
 - Datei speichern/kopieren
 - Graphische Objektrepräsentationen (Icons) erleichtern Visualisierung von Aktionen & Zuständen (Direktmanipulation)
- Nichts mehr frustrierend als „Wo bin ich?“ oder „Was tut es jetzt?!?“



Folie 233

5. Minimiere die Gedächtnisbelastung!



- Kurzzeitgedächtnis: begrenzte Kapazität
 - Etwa 7 +/- 2 chunks
- ➔ Vermeide Situationen, in denen vorherige Dialoginformationen aus dem Gedächtnis reproduziert werden müssen
 - Bsp.: Man sollte nichts 2x eintippen müssen
- Biete überschaubare Anzeigen, Zugang zu Hilfsseiten (für Abkürzungen, Codes etc.)
- GUIs besser geeignet als Kommandozeilen
 - Lesen&Auswählen statt Erinnern&Eingeben



Folie 234

6a. Vermeide Benutzerfehler, behandle sie hilfreich!



- Entwirf System so, dass Fehler möglichst gar nicht erst gemacht werden können
 - Beispiele:
Auswahl statt (falsch) Eintippen, keine Buchstaben in Zahlenfeldern, automatische Korrektur bei MacOS-Dateinamen, Videospieleautomaten (haben praktisch keine Fehlermeldungen!)
- Bei Fehlern: Stresssituation
 - Biete einfache, konstruktive, konkrete, hilfreiche und komfortable Anleitungen zur Behebung
 - Systemzustand sollte sich durch Fehleingaben nicht verändern, oder einfach restaurierbar sein



Folie 235

6b. Biete Undo!



- Soviele Aktionen wie möglich sollten reversibel sein
- Senkt Angst bei Bedienung, da Benutzer weiß, dass Fehler behebbar sind
- Ermutigt Benutzer zum Ausprobieren neuer Funktionen
- Ideal:
Mehrfaches Undo, und auf mehreren Ebenen



Folie 236

7. Biete klare Ausgänge und geschlossene Dialoge!




- 3 häufigsten Fragen des Benutzers im Dialog
 - Wo bin ich?
 - Was kann ich hier tun?
 - Wie komme ich zurück dorthin, wo ich herkam?
- Klare Ausgänge („Zurück“ oder „Ende“) helfen bei Frage 3
- Geschlossenheit der Dialoge:
 - Gefühl vermitteln, Arbeitsschritt beendet zu haben
 - Ermöglicht Entspannung, „Durchatmen“, Kopf frei für nächsten Schritt



Folie 237

8. Stelle Hilfe und Dokumentation zur Verfügung!



- Hierarchie von Hilfesysteme (steigender Umfang, sinkende Zugreifbarkeit)
 - Automatische Stichworteinblendungen (Beispiele: MacOS Balloon Help, Windows Tool Tips) – abschaltbar!
 - Online-Tutorials und -Referenzen
 - Gedruckte Dokumentation aber: 
- Gezieltere Unterstützung durch aktive Hilfe
 - MacOS Guide, Windows Wizards+ Assistenten
 - Gefahr: System wird initiativ (verletzt Vorhersehbarkeitsregel)



Folie 238

9. Biete Experten Abkürzungen an!



- Mit häufigem Gebrauch steigt Bedarf nach weniger und schnelleren Interaktionsschritten
- Erfahrene Benutzer schätzen Abkürzungen, Tastaturkürzel, Makrofähigkeiten, Programmierbarkeit und kurze Antwortzeiten ohne (für sie) überflüssiges Feedback
- Konflikt:
Im Zweifelsfall Einfachheitsregel wichtiger



Folie 239

Beispiel: Cheap Shop

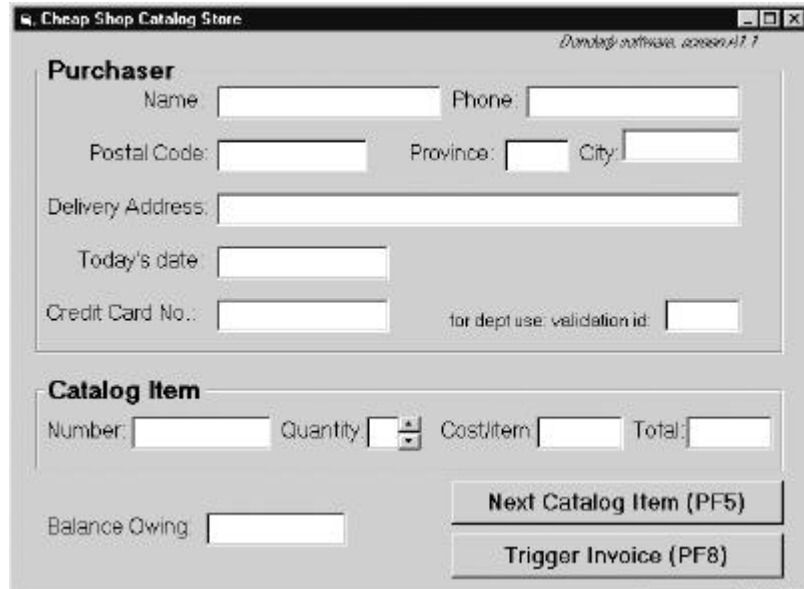


- Ein Spielwarengeschäft, bei dem Kunden normalerweise einen Katalog durchsehen, um dann an der Theke zu bestellen, möchte einen elektronischen Katalog einführen
- Aufgabe:
 - Bildschirme & Spezifikation ansehen
 - Wo werden 9 Goldene Regeln verletzt?



Folie 240

Bildschirm 1



Cheap Shop Catalog Store Dundley Software, screen A1.1

Purchaser

Name: Phone:

Postal Code: Province: City:

Delivery Address:

Today's date:

Credit Card No.: for dept use: validation id:

Catalog Item

Number: Quantity: Cost/Item: Total:

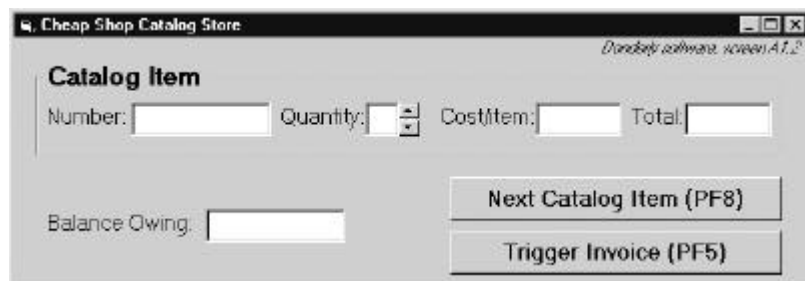
Balance Owing:

Next Catalog Item (PF5)

Trigger Invoice (PF8)

Folie 241

Bildschirm 2



Cheap Shop Catalog Store Dundley Software, screen A1.2

Catalog Item

Number: Quantity: Cost/Item: Total:

Balance Owing:

Next Catalog Item (PF8)

Trigger Invoice (PF5)

Folie 242

Spezifikation



- Bestellung erzeugen:
 - Auf Schirm 1 persönliche Informationen und 1. Bestellung eingeben
 - Text über Tastatur eingeben, Maus wechselt zwischen Feldern
- Für weitere Bestellungen:
 - Next Catalog Item Button drücken
- Bestellung abschließen:
 - Trigger Invoice drücken, System veranlasst Lieferung + Rechnung und springt zurück zu leerem Schirm 1



Folie 243

Spezifikation



- Bestellung abbrechen:
 - 30s nichts eingeben (als wäre man weggegangen)
 - System löscht dann alle Eingaben und springt zu Schirm 1
- Eingabeprüfung:
 - Alle Felder werden bei Auswahl eines der Knöpfe geprüft
 - Falsche Eingaben werden gelöscht, können neu eingegeben werden



Folie 244

Fortsetzung des Beispiels als Übung



- Szenarien durchspielen
- Erst dadurch werden weitere Probleme ersichtlich!



Folie 245

Beispiel-Taskbeschreibungen



- Mann kauft Roller
(rot bevorzugt, blau ok), zahlt bar und nimmt ihn gleich mit
- Ältere Frau mit Arthritis vergleicht Kosten eines Kinderzimmers (Tisch, Stuhl, Bett, Matratze, Bettdecke, Kissen), nimmt Beschreibung & Preisinfo mit, kommt 2h später wieder, um alles außer Stuhl zu kaufen



Folie 246

Noch eine Beispiel-Taskbeschreibung



- Einziger Angestellter soll 10 Dinge für Kunden bestellen, der Computer nicht benutzen will.
- Nach Berechnung der Summe verzichtet Kunde auf 4. + 6. Posten und fügt neuen hinzu.
- Er entscheidet sich um von Kreditkarte auf Barzahlung, wünscht Lieferung übermorgen.
- 6 andere Kunden warten derzeit im Laden.



Folie 247



5 User-Centered Design

5.4 Prototyping

Wann macht man einen Prototyp?



- Wenn Analyse der Aufgabe bzw. des Benutzerfeedbacks und Entwurf vorliegen
- Ziel:
„Proof-of-Concept“ des Entwurfs,
Objekt für nächste Evaluierung schaffen
- Mehrere, immer verfeinere Prototypen im Laufe des Softwareprojekts
 - angefangen bei Analysephase
 - bis zur Implementierungs- und Testphase



Folie 249

Formen des Prototyping



- Vom einfachen, schnellen, groben Skizzenentwurf bis zum komplexen, aufwendigen, detaillierten Softwareprototypen
- Wichtig: Kosten/Nutzen beachten!
 - Für erstes grobes Feedback (werden die richtigen Hauptfunktionen in richtiger Reihenfolge angeboten?) braucht man kein aufwendiges Director-Programm!



Folie 250

Skizzen und Storyboards



- Der erste Prototyp
 - schnell und billig
- Grobe Papier-und-Bleistift-Skizzen des Interfaces bzw. der wichtigsten Dialoge
- Dazu Textbeschreibung der Funktion und Zusammenhänge
 - bei mehreren Dialogen → Storyboard
- Zweck:
Brainstorming, erstes User-Feedback



Folie 251

Skizzen und Storyboards

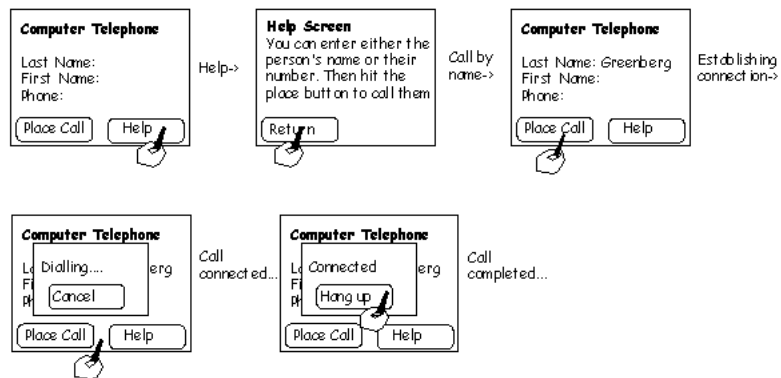


- Vorteil:
Nicht detailliert,
dadurch konzentrieren sich
Designer + Benutzer auf wichtige
grundlegende, abstrakte Dialogkonzepte
- Nachteil:
Dialogablauf schwierig nachzuvollziehen



Folie 252

Beispiel: Storyboard Computertelefon



- Muss nicht mit Computer gezeichnet werden



Folie 253

PostIt-Prototyp



- Eine interaktivere Form des Papier-Prototyps
- Dialoge, Menüs, Fenster auf Klebenotizzettel in mehreren Schichten
- Leere Vordrucke für Objekte machen, dann Ausfüllen
- Erlaubt Simulation sich öffnender Dialoge etc. durch Manipulation der Zettel
- Schnell durch Neuschreiben von Zetteln veränderbar
- Sitzung mit Benutzer kann für spätere Analyse auf Video aufgezeichnet werden



Folie 254

PostIt-Prototyp



- Beispiel:
erste Konzeption eines Informationssystems



Folie 255

Software-Prototyp



- Mehr Details, genauer, interaktiv
- Erst nach initialen, einfacheren Prototypen sinnvoll
- „Mock-Up“ (Modell, Illusion) der endgültigen Benutzerschnittstelle
- Zum Beispiel Director-Animation
- Wichtig:
Interface, nicht die Funktionalität ist entscheidend!

Folie 256

Software-Prototyp: Gefahren



- Benutzer hängen sich an Details auf und übersehen dabei größere Probleme
- Benutzer trauen sich nicht, „hübsches“ Interfacedesign zu kritisieren/verändern
- Management könnte denken, der Prototyp sei bereits die „echte“ Applikation ...



Folie 257

Software-Prototyp: Wie einschränken?



- Vertikaler Prototyp
 - Wenige Funktionen, aber die detailliert real implementieren
 - Allgemeine Entwurfsideen können an einem Beispiel tiefgehend getestet werden
- Horizontaler Prototyp
 - Gesamtes User-Interface sichtbar, aber ohne dahinterliegende Funktionalität
 - Arbeitsschritte werden simuliert, es kann nichts „Wirkliches“ mit dem System getan werden



Folie 258

Software-Prototyp: Wie einschränken?



- Typisches Szenario
 - Kombination vertikaler und horizontaler Beschränkung
 - Simulation einzelner, festgelegter Abläufe im System, keine Abweichungen erlaubt



Folie 259

Was macht man mit dem Prototyp?



- Wegwerfen
 - Dann muss Erstellung schnell & billig sein
- Weiterentwickeln
 - Prototyp wird immer wieder überarbeitet
 - Wird letztlich reales Produkt
 - Dann muss verwendete Technologie stimmen
 - Programmiersprache, Betriebssystem, Zusatzmodule (Datenbank, ...), Hardwareplattform, ...



Folie 260

Womit SW-Prototyp erstellen?



- Bildschirme mit Malprogramm zeichnen
 - Dünner horizontaler Prototyp, keine Interaktion
 - Evtl. besser als handgemaltes Storyboard zu ändern
- Skripts + Animationen:
 - Storyboard im Computer codiert
 - Beispiele: HyperCard, Director
 - Szenenübergänge durch Benutzereingaben
 - Vorteil:
Sieht echt aus
 - Nachteil:
Keine Abweichungen von ausgearbeiteten Pfaden möglich, Weiterverwendung?



Folie 261

Womit SW-Prototyp erstellen?



- User-Interface-Builders
 - Graphische/textuelle Werkzeuge, um Benutzerschnittstelle eines realen Programms zu definieren
 - Vorteil:
Fertiger Entwurf kann für Implementierung direkt verwendet werden, echtes Look&Feel, vertikale Funktionalität nach Belieben dahinter programmierbar
 - Nachteil:
Beschränkung auf 1 Fenstersystem und dessen Dialogbausteine (Fenster, Buttons, ...)



Folie 262

Wizard of Oz



- Methode, um nichtexistentes System zu testen
- Mensch (Wizard) simuliert Reaktionen des Systems (für Benutzer unsichtbar)
- Interagiert mit dem Benutzer über simulierte SW-Benutzeroberfläche
- Gut für Hinzufügen komplexer vertikaler Funktionalität und für futuristische Ideen
- Beispiel: Spracheditor IBM 1984



Folie 263

Hardware-Prototyp



- Bei Systemen, die nicht gut durch Software simuliert werden können, weil physische Gestalt entscheidend für Benutzung ist
 - z.B. bei Entwicklung neuer 3D-Maus
- Entwurf in Holz, Schaumstoff, Plastik, Styropor, Pappe, ...
- Problem:
Hoher Aufwand bei Erstellung und Änderung → Werkstatt erforderlich



Folie 264

Zusammenfassung



- Die 9 Goldenen Regeln des Interface-Designs (Shneiderman)
 - Einfach, Sprache, konsistent, vorhersagbar, Rückmeldungen, Fehler vermeiden/behandeln/Undo, abgeschlossen, Hilfe + Doku, Abkürzungen
- Prototyping: Viele Formen
 - Skizze+ Storyboard, PostIts, Software (horizontal/vertikal), Wizard of Oz, Hardware
 - SW-Prototyp: Bilder, Animationen, UI Builders



Folie 265