1 Einführung

1.1 Unterscheidung Anwender ↔ Benutzer

Anwender sind Entitäten, welche direkt oder auch indirekt ein Hilfsmittel zur Erzielung eines Vorteils verwendeen

Benutzer sind natürliche Personen, die direkt mit einem Hilfsmittel arbeiten, um einen Vorteil zu erzielen

Beispiel: Ein Direktor (Anwender) lässt sich von seinem Chauffeur (Benutzer) mit einem Auto (Hilfsmittel) zu seinem nächsten Termin bringen

1.2 Definition Interaktives System

Ein **interaktives System** gibt dem Benutzer die Möglichkeit, mittels einer Benutzerschnittstelle die laufende Bearbeitung einer Aufgabe durch ein technisches System zu beeinflussen.

1.3 Formen von Interaktion

- **Anweisung**
  - Eingabe von Kommandos
- **Dialog**
  - Ablauf von Eingabe und Ausgabe
- **Manipulation (von Kommandos, Inhalten)**
  - Auswählen („Choice“)
  - Ansteuern („Target Acquisition“)
  - Verändern
- **Erkunden (einer Menge von Kommandos, Inhalten)**
  - Suchanfragen
  - Browsing

1.4 Arten von Benutzerschnittstellen

- **T**ext **U**ser **I**nterface
- **T**angible **U**ser **I**nterface
- **G**raphical **U**ser **I**nterface
- **V**oice **U**ser **I**nterface
- **H**aptic **U**ser **I**nterface
- **N**atural **U**ser **I**nterface
  (möglichst wenig zu erlernende Eingabegeräte)
- **I**ntuitive **U**ser **I**nterface
  (möglichst leicht erlernbare Eingabegeräte)
- **B**rain **C**omputer **I**nterface
- **C**ommand **L**ine **I**nterpreter

1.5 Fallstudien

Sketchpad
Programm entwickelt von Ivan Sutherland im Rahmen einer Doktorarbeit.
Mittels Lichtgriffel: Zeichnen und Deformieren geometrischer Objekte; Vektorgraphiken; Erste objektorientierte Ansätze zur Bedienung

Xerox Alto
Kommandozeileninterpreter Alto Executive: Graphische Oberfläche; Mehrere Applikationen; Mehrere Fenster

Apple Macintosh
Features u.a. Desktop-Metapher, Drop-Down-Menüs, Ordner, Mülleimer sowie überlappende Fenster

Windows 1.0
Zeigegerät, Menüs, Statuszeile, Symbole, Zwischenablage, Taskleiste. Wegen Vektorgrafik-Fähigkeit bereits als GUI (und nicht als TUI) klassifiziert.

2 Zeigegeräte

2.1 Einteilung von Interaktionstechnologien

Nach *Krauß*

- Koordinatengebend
  - Dimensionalität
    → eindimensional, zweidimensional, mehrdimensional
  - Verhältnis Lage zu Wirkort
    → Indirekt wirkende vs. direkt wirkende
- Nicht koordinatengebend
  - Tasten
  - Sprache
  - Gesten

2.2 Koordinatengebende

2.2.1 Direkte Zeigegeräte

- Es wird unmittelbar auf die Ausgabe gezeigt
- Merkmale
  - Lernaufwand Bedienung Zeigegeräts minimal
  - Verdeckung der Ausgabe durch Zeigegerät
  - Ermüdungserscheinungen
- *Beispiel:* Touchscreen

2.2.2 Indirekte Zeigegeräte

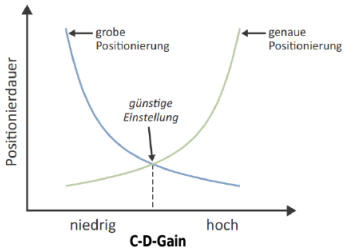
- Nicht im direkten Kontakt mit der Ausgabe
- Bewegung an anderer Stelle und Transformation der Koordinaten an den Bildschirm
- Merkmale
  - Lernaufwand Bedienung des Zeigegeräts z.B. Hand-Augen-Koordination bei Computermaus
  - Kopplung zwischen Zeigegerät und Ausgabe als modulierbarer Parameter
- *Beispiel:* Computermaus

2.2.3 Positionierung

Absolute Positionierung Eindeutige Zuordnung der Position des Zeigegerätes zur Position auf dem Bildschirm. Der Bezug zum Ursprung kann gegeben sein.

Relative Positionierung Es wird von aktuell gespeicherter Position ausgegangen und die Veränderung der Koordinaten umgesetzt.

Control-Display-Gain «MacKenzie» Der durch die Bewegung eines indirekten Zeigegerätes ("Control") resultierende Effekt („Gain“) am Bildschirm.
Nach MacKenzie balanciert eine günstige Einstellung des Gain Schnelligkeit und Präzision der Positionierung.

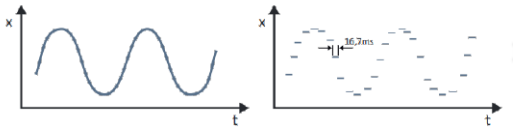


- *Gain niedrig:* Große Bewegung Controller bewirkt durchschnittlichen Effekt; Vorteilhaft für Feinpositionierung
- *Gain hoch:* Kleine Bewegung Controller bewirkt durchschnittlichen Effekt; Vorteilhaft für Grobpositionierung

2.2.4 Physikalische Charakterisierung

Abtastrate

- Erfassung der Position und Tasten des Zeigegerätes zu diskreten Zeitabständen
- Definiert die zur Positionsbestimmung verfügbare Zeit



Verzögerung

- Zeitabstand zwischen Abtastung und Änderung am Bildschirm
- Verzögerung sollte geringer als die Hälfte der Abtastrate sein
- Schon geringe Verzögerungen bringen höhere Fehlerrate bei Selektionen mit sich

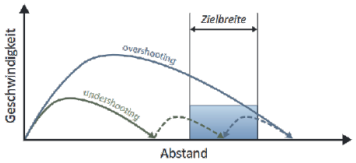
Genauigkeit

- Genauigkeit: Abweichung der durch einen Sensor gemessenen Position von der tatsächlichen Position
- Gemeinsame Optimierung von Genauigkeit und Abtastrate nicht möglich
  - Obergrenze bildet die räumliche Auflösung des Sensors
  - Hohe Abtastrate geht zu Lasten der Genauigkeit

Abhängigkeit von der individuellen Anwendung

- Bei Interaktion in komplexen virtuellen Welten in modernen Computerspielen oder 3D-Modellierungstools können hohe Anforderungen an Zeigegeräte gestellt werden

2.2.5 Zeige Aufgaben



Das Experiment nach Fitts: Wie lange benötigt ein Mensch MT, um mit einm Stift ausgehend von einem Startpunkt ein Ziel zu erreichen? Das Experiment steht dabei in der Abhängigkeit von der Distanz D des Ziels und der Breite W.

Fitts

$$MT = a + b \cdot ID$$

$$ID = \log_2\left(\frac{2D}{W}\right)$$

MacKenzie
Verbesserung der Formel von Fitts

$$ID = \log_2\left(\frac{D}{W} + 1\right)$$

Accot & Zhai
Berücksichtigt auch Höhe und y-Koordinate des Ziels

$$MT = a + b \cdot \log_2\left(\sqrt{\left(\frac{D}{W}\right)^2 + \eta\left(\frac{D}{H}\right)^2} + 1\right)$$

2.2.6 Ergebnisse des Praktikumsversuchs

MacKenzie > Accot/Zhai > Fitts

- In Fitts' Experiment wird ein direktes Zeigegerät benutzt
- *Precuing* macht eine Zeigeaufgabe immer einfacher
- *Reset* macht eine Zeigeaufgabe zunächst schwerer, jedoch kann es schnell erlernt werd

2.2.7 Unterstützung von Zeigeaufgaben

Anpassung Bedienelemente Eine Anpassung der Größe, Anordnung, Form.
Verkleinern ist nach Fitts schwieriger und die Unterscheidbarkeit nimmt ab.

Temporäre Modifikation des Interface Eine Anpassung des User Interface.
Nach Parker kann man das Ziel oder den Cursor vergrößern, sowie den Cursor zum Ziel bewegen oder auch das Ziel zum Cursor bewegen.
Nachteile: Eine Veränderung des User Interface kann den Benutzer irritiren; Aussagen über eine Selektionsinteraktion sind schwer

Mehrere Mauszeiger Nachteile: Potentiell mehrdeutige Zielauswahl

2.2.8 Direct Manipulation

Direkt manipulative Interfaces zeichnen sich aus nach *Hutchins et al.* durch:

- Geringe Distanz
  - von Benutzerzielen
  - zu deren Erreichung nötigen Interaktionen
- Hohes Engagement
  - direkte Auseinandersetzung des Benutzers
  - mit dem Objekt von Interesse
- *Beispiel:*
  - Skript schreiben:
    Distanz hoch, Engagement niedrig
  - Drag & Drop:
    Distanz niedrig, Engagement hoch

Prinzipien nach Shneiderman

- Kontinuierliche Repräsentation des Objekts von Interesse
- Eingabe auf Basis einfacher physikalische Aktionen (z.B. Bewegung der Maus) anstelle sprachbasierter Kommandos
- Schnelle, schrittweise, umkehrbare Operationen, deren Auswirkung auf das Objekt von Interesse unmittelbar erkennbar wird
- Schrittweises Erlernen der Interaktion ausgehend von minimalem Vorwissen

Vorteile nach Shneiderman

- Anfänger können die grundlegende Funktionen schnell erlernen, beispielsweise nach Vorführung durch einen erfahreneren Benutzer
- Interaktionskonzepte werden nach längerer Nichtanwendung leicht erinnert
- Benutzer können unmittelbar erkennen, ob sie sich durch Aktionen ihren Zielen annähern – und gegebenenfalls ihre Aktionen anders ausrichten
- Benutzer sind weniger zögerlich im Umgang mit dem System, da Aktionen leicht verständlich und umkehrbar sind
- Benutzer erleben Kontrolle über das System und können dessen Reaktionen direkt beobachten

2.3 Nicht koordinatengebend

2.3.1 Sprachbasierte Interaktionsformen

Nach *Preim & Dachselt*

- Kommandosprachen
- Textuelle Suche
- Natürlich-sprachliche Systeme

Phoneme & Prosodie

Phonem: "Phoneme sind die kleinsten bedeutungsunterscheidenden Einheiten einer Sprache."
Der Austausch eines Phonems ändert die Bedeutung eines Wortes. Prosodie: "Prosodie ist die Gesamtheit derjenigen lautlichen Eigenschaften der Sprache, die nicht [...] ans Phonem als minimales Segment, sondern an umfassendere lautliche Einheiten gebunden sind"
Dient u.a. der Kennzeichnung des Satztyps (z.B. Frage, Aussage), Gewichtung oder Gliederung.

Spracherkennung Audiosignal → Text

- Vorverarbeitung (Filterung um Störgeräusche zu extrahieren, Frequenzspektrum berechnen, Sprachrelevante Merkmale extrahieren)
- Extraktion von Phonemen
- Abbildung von Phonemfolgen auf Wörter

Semantische Analyse Bedeutung der Aussage auf struktureller Ebene (Agenten, Handlungen, Intentionen, Objekte), Ergebnis unter Umständen nicht eindeutig

Pragmatische Analyse Bedeutung der Aussage im Kontext, z.B. Ort, Zeit, Fähigkeiten, Intention; Hinzuziehen weiterer Sprachmerkmale (Prosodie); auch hier Ergebnis unter Umständen nicht eindeutig

Sprachsynthese Text → Audiosignal

- NLP (Natural Language Processing) transkribiert Text anhand eines Phonem-Alphabets und einer Prosodiebeschreibung
- DSP (Digital Speech Processing) synthetisiert aus der NLP-Ausgabe das Audio-Signal gesprochener Sprache

Ansätze für die Umsetztung:

- Modellbasierte Synthese: erzeugt ein Audio-Signal auf Basis eines Modells
- Konkatenative Synthese: verknüpft Ausschnitte aus Aufzeichnungen gesprochener Sprache um zu einem neuen Audiosignal

2.3.2 Tastatur

Die Effizienz hängt ab von ...

... Größe, Form und Position der Tasten

... Art der Eingabe (Symbolhäufigkeiten und Symbolfolgen), im Fall natürlicher Sprache also von der Sprache, in der der Text verfasst wurde

... Technologischen Randbedingungen

Die **QWERTY** Tastenbelegung orientierte sich danach das sich in der Englischen Sprache die Typenhebel einer Schreibmaschiene nicht kollidierten.

Virtuelle Tastaturen

- Platzersparnis: z.B. Temporäres Einblenden bei Touch-Interaktion genau dann, wenn alphanumerische Eingaben benötigt werden
- Konfigurierbarkeit: z.B. Tastaturen, deren Layout sich anpasst

Probleme: Fat Finger Problem, Fehlender Druckpunkt (Tasten können versehentlich ausgelöst werden)

3 Graphische Dialogsysteme

3.1 WIMP-Prinzipien

WIMP: **W**indows **I**cons **M**enus **P**ointer

Metaphors Interaktionsmechanismen werden auf bekannte Arbeitsabläufe abgebildet

Direct Manipulation Der Benutzer soll direkt mit Objekten der Benutzerschnittstelle interagieren können

See and Point Die Interaktion mit der Maschine soll auf den gerade sichtbaren Objekten basieren

Consistency Objekte mit ähnlicher Funktion sollen ähnlich aussehen und sich ähnlich verhalten

User Control Nur der Benutzer soll Aktionen der Maschine initiieren und deren Ablauf kontrollieren

Feedback and Dialog Alle ausgeführten Aktionen des Benutzers und deren Effekte sollen ohne Verzögerung erkennbar sein

Forgiveness Aktionen des Benutzers sollen rückgängig gemacht werden können, bei kritischen Aktionen soll gewarnt werden

Aesthetic Integrity Das Erscheinungsbild der Benutzeroberfläche soll einfach, aufgeräumt und konsistent sein

Modelessness Der Benutzer soll alle möglichen Aktionen zu jedem Zeitpunkt ausführen können. (Aufwendig zu implementiren)

WYSIWYG Dokumente sollen bei der Erstellung so angezeigt werden, wie sie als Ausdruck ausgegeben werden. (**Nur Drucker!**)

3.2 Fenster

- Rechteckige Bereiche, die Interaktionen von Benutzern verarbeiten können
- Präsentation und Verhalten angelehnt an Metapher „Papier“
  - stapelbar, verschiebbar
  - Passend zur Metapher „Schreibtisch“
  - Historisch in Büroarbeiten begründet
- Begriff verdeutlicht die Funktion einer Sicht („View“) auf Objekte einer Anwendung

3.2.1 Varianten

Applikationsfenster Beinhaltet (logisch) alle Bedienelemente einer Anwendung

Dialogfenster Informieren oder Aufforderung zur Eingabe

Elementares Fenster (Widget) Die jenigen Bedienelemente, die eine Einheit aus Erscheinung und zugehörigem Verhalten bilden

3.2.2 Verwaltung von Fenstern

- Window Manager sind entweder im Fenstersystem integriert, oder können auch austauschbar sein
- Realisieren das Verschieben, Skalieren, Öffnen und Schließen der Fenster
- Steuern das Aussehen der Fenster, d.h. die Gestaltung der Titelzeile und der Scrollbars
- Realisieren Fensterplatzierungsstrategien
- Unterstützen Navigation in Fenstern

3.2.3 Koordinatensysteme

Das Koordinatensystem eines Fenstersystems bestimmt Positionierung und Größe präsentierter Information

Geräteabhängige Koordinaten Die Koordinaten auf einem gegebenen Ausgabegerät

Geräteunabhängige Koordinaten Die Entkopplung der Präsentation von geräteabhängigen Parametern (insbesondere: Auflösung) der Ausgabegeräte
Beispiel: Eine 1 cm Linie soll auf einem Bildschirm mit 150 dpi genauso lang ausgegeben werden wie auf einem Drucker mit 2400 dpi

Weltkoordinaten Entkopplung der Präsentation von der Repräsentation der zugrundeliegenden Daten
Beispiel: Sowohl Nanometer für Moleküle wie auch astronomische Koordinaten sollen vom Modell abgebildet werden können

3.3 Piktogramme

- Symbole, die Komponenten eines Fenstersystems graphisch repräsentieren
- Orientierung an bildhaften Darstellungen in anderen Bereichen der natürlichen Umgebung
  - Ziel: Ausnutzen der ausgeprägten Stärke des Menschen, erlernte Muster (wieder) zu erkennen
- Unabhängigkeit von einer bestimmten Sprache
  - Können damit zur Internationalisierung beitragen

3.3.1 Varianten

Repräsentative Piktogramme Stellen einen abstrakten Sachverhalt bzw. ein Konzept dar
Beispiel: Symbole für Mülleimer, Disketten, Drucker

Abstrakte Piktogramme Stellen einen abstrakten Sachverhalt bzw. ein Konzept dar und müssen somit erst erlernt werden
Beispiel: Symbole für Pfeile, Wiederholung, Rückgängig

Hybride Formen Stellen eine Mischform da

3.3.2 Richtlinien

- Einfachheit und Klarheit

- Verständlichkeit
- Einprägsamkeit
- Platzsparende Gestaltung
- Klarer Kontrast zu Hintergrund
- Hohe Unterscheidbarkeit vs. Konsistenz
- Selektierter Zustand sollte klar ersichtlich sein

3.4 Menüs

- Basieren auf Hierarchien und Taxonomien
- Dienen dem Auslösen atomarer Aktionen
- Komplexere Interaktionen wie die Eingabe mehrerer Werte etc. durch die Verbindung mit Dialogen

Motivation

- Es soll unter einer endlichen Menge von Aktionen gewählt werden
- Man soll die Menge der Aktionen nicht erlernen müssen
- Man soll im Vermeiden syntaktischer Fehler unterstützt werden

3.4.1 Hick & Hyman

Versuchsaufbau „choice-reaction time"

- 2-10 Lampen angeordnet in einem irregulären Kreis
- Jeder Lampe ist eine Taste zugeordnet
- Alle 5 Sekunden leuchtet zufällig eine Lampe auf
- Testperson soll zugeordnete Taste drücken
- Vorfeld: Testpersonen erlernen Zuordnung

Hick-Hyman Gesetz

- Informationsgehalt der Aufgabe („choice")
  $H = \log_2(n)$
- Reaktionszeit („reaction time")
  $RT = a + b \cdot \log_2(n)$
- Reaktionszeit in hierarchischen Menüs folgt dem Hick-Hyman-Gesetz laut Landauer & Nachbar, 1985
- Reaktionszeit bei häufig verwendeten Aktionen folgt dem Hick-Hyman-Gesetz ist aber linear bei selten verwendeten Aktionen laut Sears & Shneiderman, 1994
- Bei Anfängern ist die RT in einem Menü linear, bei Experten folgt sie dem Hick-Hyman-Gesetz (laut Cockburn & Gutwin, 2008)
  Beachte: Bei plötzlichen Menü-Änderungen sind auch Experten wieder Anfänger und müssen das Menü neu erlernen!

4 Post-WIMP

WIMP nimmt eine Anwendungssituation an:

- Benutzer kennt Grundkonzepte von Computern
- Benutzer ist konzentriert auf den Computer
- Bearbeitung alleine und vom Nutzer getrieben
- Hardware =
  Desktop-PC, Maus, Tastatur, Bildschirm
- Büroumgebung
  (d.h. gute Beleuchtung, wenig Störeinflüsse)

Diese Voraussetzungen sind in heutigen Anwendungssituationen teilweise nicht mehr gegeben daher wandelte sich das WIMP Konzept zu dem heutigem Post-WIMP Konzept.

4.1 WIMP vs. Post-WIMP

WIMP	Post-WIMP
Metaphors	Interaktion in der realen Welt
Direct Manipulation	Delegierung von Aufgaben an das System
See and Point	Describe and input Command
Consistency	Diversity; angepasst an den Einsatztort (kann nicht immer angewendet werden)
WYSIWYG	Erkennung des Wunsches des Benutzers und der Darstellung einer gleichwertigen Ausgabe
User Control	teilweise gegeben oder Shared Control
Feedback and Dialog	nicht zum Zeitpunkt der Interaktion
Forgiveness	User Model; System weiß, was der Nutzer als Ergebnis haben möchte
Aesthetic Integrity	oft verletzen von See and Point durch Ausblenden von Inhalten
Modelessness	User Model vorhanden, um Verhalten vorherzusagen; mehrere Funktionen für z.B. einen Button

4.2 Touchscreens

Technische Varianten: Kapazitiv, resistiv; Singletouch, Multitouch; Druckstärke

4.2.1 Kapazitive Touchscreens

Funktioniert mit einem elektrischen Feld, in fast allen Smartphones
Vorteil: lässt sich mit starkem Glas gut vor äußeren Einwirkungen schützen

4.2.2 Resistive Touchscreens

Funktioniert mit zwei Schichten welche aneinander gedrückt werden
Vorteil:

- Bedienung mit jedem Eingabestift möglich
- Mit Handschuhen und Prothesen bedienbar
- Genauer als kapazitive Touchscreens
- Geringe Fertigungskosten

Nachteile:

- Nur eingeschränktes Multitouch (Two-touch)
- Ist schlecht lesbar bei Sonneneinstrahlung durch Zusatzschicht
- Die Gestenbedienung, aufgrund des notwendigen Drucks, ist erschwert.
- Verschleiß durch die mechanische Belastung beim Betätigen
- Unerwünschtes Auslösen beim Transport durch Kontakt mit anderen Gegenständen möglich

5 Entwicklung interaktiver Systeme

5.1 DIN EN ISO 9241-110

DIN EN ISO	WIMP
Aufgabenangemessenheit	Aesthetic Integrity, See and Point
Selbstbeschreibungsfähigkeit	Direct Manipulation, Metaphors
Lernförderlichkeit	Metaphors, Feedback & Dialog, Forgiveness
Steuerbarkeit	User Control
Erwartungskonformität	Consistency, WYSIWYG
Individualisierbarkeit	Modelessness
Fehlertoleranz	Forgiveness

5.2 Prinzipien der Entwicklung

- Konzentration auf Benutzer
  - Benutzeranalyse
  - Umfeldanalyse
  - Tätigkeitsanalyse
- Frühes und kontinuierliches Testen
  - Erlebbarkeit der Benutzerschnittstelle in Entwurfsphase fördern
  - Integriertes Design
    - Erheben von Feedback zum Zusammenspiel von Interaktionselementen bei der Durchführung typischer Aufgaben der Benutzer
    - Solche Test erfordern die volle Verfügbarkeit aller involvierten Interaktionselemente die bei herkömmlicher Implementierung erst spät im Entwicklungsprozess gegeben sind.
    - Ganzheitliche Betrachtung der Anwendung
    - Jedes Teil der Anwendung sollte entworfen und in seinem Interaktionsverhalten umgesetzt werden
  - Iteratives Design
    - Bearbeitung kontinuierlicher Änderungsaufträge als Nebenwirkung von «Frühes und kontinuierliches Testen»
    - Streng lineare Entwicklungsmodelle (z.B. Wasserfallmodell) sehen Veränderungen an abgeschlossenen Abschnitten nicht vor
    - Prototyping: Erlebbarkeit mit reduzierten Implementierungsaufwand verbinden (Prototyping)
    - Dokumentation: Gründe von Änderungen müssen rückverfolgbar sein
  - Empirische Daten zur Bewertung der Qualität einer Benutzerschnittstelle erforderlich

6 Modelle interaktiver Systeme

6.1 Ebenen der Benutzerinteraktion

Aktionssprache

- Konzeptuell:
  Objekte, Beziehungen
- Semantisch:
  Abbildung Eingabefolgen auf Funktionen der Anwendung
- Syntaktisch:
  Folgen atomarer Eingaben
- Lexikalisch:
  Atomare Eingaben (Klick, Texteingabe)

Präsentationssprache

- Konzeptuell:
  Interaktionsmodell
- Semantisch:
  Form der Ausgabe (Menü)
- Syntaktisch:
  Kombinationen von Interaktionselementen
- Lexikalisch:
  atomare Ausgabewelemente (Icons, Text)

Beispiel: Kommandozeile:
Eingabe des Benutzers `cd /usr/share` (Aktionssprache) → Ausgabe des Systems `[/usr/share]$ _` sowie eine Eingabeausforderung (Präsentationssprache)

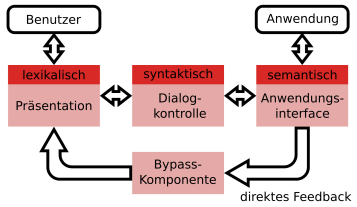
6.2 Architekturmodelle

6.2.1 Seeheim-Modell

Präsentation Lexikalische Schicht der Benutzerschnittstelle (isolierte I/O-Aktionen)

Dialogkontrolle Syntaktische Schicht (bildet Eingaben anhand des Dialogmodells auf Kommandos des Anwendungsinterface ab)

Anwendungsinterface Semantische Schicht der Benutzerschnittstelle (Anwendungslogik)

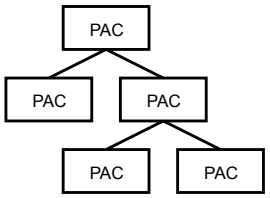


6.2.2 Presentation-Abstraction-Control

Presentation Syntax der Interaktion mit der Anwendung (I/O, wie er vom Nutzer wahrgenommen wird)

Abstraction Semantik der Interaktion

Control Konsistenz von Presentation und Abstraction



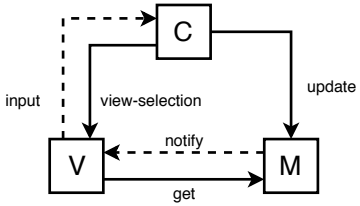
Vorteil: Wartbarkeit; Austausch von Objekten
Nachteil: Dekomponierung der Interaktion fordernd; Verarbeitungsaufwand

6.2.3 Model-View-Controller (MVC)

Model Datenhaltung und Geschäftslogik

View Darstellung der Daten

Controller Steuerung der Anwendung

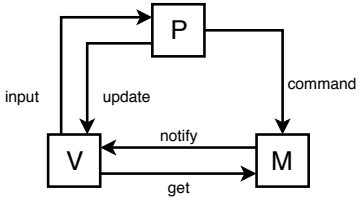


Beobachter-Pattern: View beobachtet Model; Controller kann View austauschen, Kommandos im Modell ausführen, Model ändern

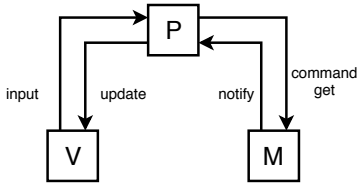
Front Controller *(Fowler)*
Durch die Steigende Komplexität der Web-Anwendungen wird ein seitenspezifischer Controller eingeführt. D.h. Ein Zentraler Zugangspunkt zuständig für Weiterleitungen, welche mit Hilfe von Tabellenerfolgt. Ein Spezifischer Page-Controller generiert im nächsten Schritt die View.

6.2.4 Model-View-Presenter

Potel

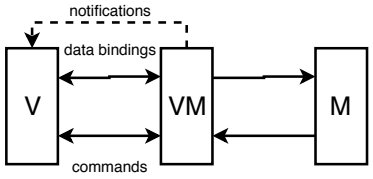


Fowler Das Model-View-Presenter Passive View nach Fowler präsentiert die Aufgaben der mit MVC eingeführten Komponenten, dabei beachtet der Presenter das Modell und der Presenter aktualisiert die View.



6.2.5 Model-ViewModel-Model (MVVM)

Das MVVM Modell ist eine im Wesentlichen aus MVC und MVP abgeleitete Spezialisierung anhand technischer Merkmale von WPF. (Stärkung der Separation of Concerns durch XAML; Beschreibt die Einbindung von Events und Bindings)



7 Frameworks und Tools

7.1 Frameworks

Programmiergerüst für die Softwareentwicklung (z.B. .NET Framework)

7.2 Klassifizierung anhand Entwicklungsansatz

Black-Box

- Ansatz
  - Vordefinirte Komponenten
  - Komponenten werden durch ihre Schnittstelle beschrieben
  - Verständnis der internen Funktionsweise der Komponenten *nicht* erforderlich
- Anwendung des Frameworks
  - Objektkomposition
- Vorteil
  - Niedrige Einarbeitungszeit

White-Box

- Ansatz
  - Vordefinirte Komponenten und/oder Codebausteine
  - Verständnis der internen Funktionsweise der Komponenten erforderlich
- Anwendung des Frameworks
  - z.B. Erweiterungen des Frameworks durch Ableitungen
  - z.B. Verknüpfen mit Callbacks von Komponenten des Frameworks
- Vorteil
  - Hohe Flexibilität

7.3 Klassifizierung nach Einsatzbereich

- Toolkit
  - Unterstützt die **Erstellung** einer Benutzerschnittstelle durch wiederverwendbare Komponenten (z.B. Menüs, Buttons)
  - Erscheinungsbild und Verhalten der GUI-Komponenten vorgegeben, u.U. anpassbar
- Interface Builder
  - Baut auf einem Toolkit auf
  - Unterstützen (i.A. graphischen) **Entwurf** und **Erstellung** der Benutzerschnittstelle ohne Programmieraufwand
- User Interface Management System (UIMS)
  - Vermitteln zwischen Benutzer und Anwendungsprogrammen
  - Unterstützen **Entwurf**, **Erstellung** und **Pflege** von Benutzerschnittstellen
  - Umfassen daher Toolkit und Interface Builder

7.4 Prototyping

Prototyping: Realisieren ausgewählte Aspekte eines UI mit dem Ziel Feedback der Benutzer vor der detaillierten Umsetzung eines Interaktionskonzepts einzuholen und so frühes und kontinuierliches Testen zu befördern

7.5 Modellbasierte Entwicklung

- Horizontale Prototypen
  - Breite Ausdehnung
  - Enthalten vollständige Benutzerschnittstelle aber keine Funktionalität
- Vertikale Prototypen
  - Tiefe Ausdehnung
  - Enthalten Funktionalität, aber nur einen Teil der Benutzerschnittstelle

7.5.1 Unterscheidung hinsichtlich Übereinstimmung

Low Fidelity

- Zweck
  - Testen von Ideen, Abläufen
- Vorteile
  - Einfaches und schnelles Erstellen
  - Keine Programmierkenntnisse notwendig
  - Änderungen sind schnell herbeizuführen

High Fidelity

- Zweck
  - Testen von UI-Umsetztung nahe dem finalen Produkt
- Vorteil
  - Geringe Abstraktion gegenüber dem finalen Produkt
  - Testen in realer Arbeitsumgebung möglich
  - Zusammenspiel zwischen UI und Anwendung kann getestet werden

Techniken Storyboards und Interface Flow Diagram (Low Fidelity), Mockups (High Fidelity), Wireframes (Low Fidelity)

7.6 Modellbasierte Entwicklung

Das **Ziel** ist die Flexibilisierung von der Design-Phase (Entwicklungsprozess) und der Runtime-Phase (im Wirkbetrieb)

Ansatz

- Aspekte der Benutzerschnittstelle durch Modelle plattformunabhängig beschreiben
- Modelle als Eingabe für eine automatische Generierung der Benutzerschnittstelle verwenden
- Änderungen der Anforderungen im Modell, nicht im Code
- Vermeiden von Inkonsistenzen zwischen Modell und Code als Folge händischer Überführung

7.6.1 Modelltypen im Cameleon-Reference-Framework

- Domain-Modell
  - Concepts: Konzepte die im Anwendungsgebiet vorkommen
  - Tasks: Aufgaben, die der Benutzer durchzuführen hat
- Context-of-Use-Modell
  - User: Benutzer und Rollen im zu unterstützenden Arbeitsablauf
  - Platform: Technische Möglichkeiten und Grenzen involvierter Geräte; Interaktion von Geräten im Anwendungskontext; Verfügbare Interaktionskomponenten (Presentation Model)
  - Environment: Umgebung, in der die Anwendung genutzt wird
- Adaption-Modell
  - Evolution: Zu welcher Benutzerschnittstelle soll gewechselt werden und welches Transition Model soll verwendet werden
  - Transition: Wie soll der Übergang zwischen zwei Benutzerschnittschnellen dem Benutzer kommuniziert werden

8 Windows Presentation Foundation

```
<Window x:Class = "Zusammenfassung.MainWindow"
   xmlns = "http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x = "http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:d = "http://schemas.micorsoft.com/expression/blend/2008"
   xmlns:local = "clr-namespace:Zusammenfassung"
   Title="MainWindow" Height="350" Width="550"
   DataContext="{Binding RelativeSource={RelativeSource Self}}"/>
  <StackPanel>
    <TextBox Name="Box1"
         Text="{Binding
         ElementName=Box2,
         Path=Text,
         UpdateSourceTrigger=PropertyChanged}" />
    <TextBox Name="Box2">
  </StackPanel>
</Window>
```

```
public partial class MainWindow : INotifyPropertyChanged
{
  public event PropertyChangedEventHandler PropertyChanged;

  private void NotifyPropertyChanged([CallerMemberName] String name = "")
  {
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
  }

  public MainWindow()
  {
    InitializeComponent();
    Binding binding = new Binding("Storage")
    {
      Source = this,
      UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged,
      Mode = BindingMode.TwoWay
    };
    Box2.SetBinding(TextBox.TextProperty, binding);
  }

  private string _storage;
  public string Storage
  {
    get
    {
      return this._storage;
    }

    set
    {
      if (value != this._storage)
      {
        this._storage = value;
        NotifyPropertyChanged();
      }
    }
  }
}
```

8.1 Events

```
<Button Content="Drueck Mich!" Click="OnButtonClick" />
```

```
public void OnButtonClick(object sender, MouseEventArgs args)
{
  // handle the event
}
```

8.1.1 Routed Events

Routed Events pflanzen sich entweder aufwärts oder abwärts in der Hierarchie des *Visual Trees* der GUI fort: Zwischen drin liegende Layout-Manager (`Grid`, `StackPanel`, etc.) bekommen nur dann ein Event, wenn ein enthaltendes Elementen ebenfalls ein Event geschickt bekommt. Bei ≥ 2 sich überlagernden Elementen in der gleichen Ebene wählt das Hit-Testing *immer* das im XAML *zuletzt* deklarierte.

- Direkte Events (werden nicht weitergereicht)
  - Ereignisse werden nur von dem Element verarbeitet, bei dem sie auch aufgetreten sind (konform mit .NET)
- Bubbling-Events
  - Event wird zuerst an den Handler des Quellelements weitergereicht. Dann wird sich dieses Weiterreichen elementweise bis zur Wurzel fortgesetzt (meist Window)
- Tunneling Events
  - Hier ist es umgekehrt wie beim Bubbling. Die Ereigniskette beginnt beim Wurzelelement, d.h. Window oder Page. Nachfolgend der Handler des untergeordneten Elements usw.
- Bubbling/Tunneling stoppen: `args.Handled = true;`

8.2 Attribut-Element-Syntax

```
<Button Name="example" Content="Drueck Mich!" Height="20">
```

```
<Button>
  <Button.Name>example</Button.Name>
  <Button.Content>Drueck Mich!</Button.Content>
  <Button.Height>20</Button.Height>
</Button>
```

8.3 Bindings

Bindungen binden immer nur auf **Properties**, nicht auf Member-Variablen, und der angegebene *Path* muss `public` sein.

8.3.1 Richtungsbestimmung

- *TwoWay*: Zweiwegebindung
- *OneWay*: Von der Quelle zum Ziel
- *OneWayToSource*: Von dem Ziel zur Quelle
- *OneTime*: Eine einmalige *OneWay* Bindung

8.3.2 Data-Context

```
public MainWindow()
{
  InitializeComponent();
  DataContext = this;
}

public string StudentName { get;set; }
```

```
<TextBox Name="Box" Text={Binding Path=StudentName} />
```

8.3.3 INotifyPropertyChanged

```
class DataStorage : INotifyPropertyChanged
{
  public event PropertyChangedEventHandler PropertyChanged;

  private void NotifyPropertyChanged([CallerMemberName] String name = "")
  {
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
  }
  private string _storage;
  public string Storage
  {
    get
    {
      return this._storage;
    }

    set
    {
      if (value != this._storage)
      {
        this._storage = value;
        NotifyPropertyChanged();
      }
    }
  }
}
```

8.4 Style Templates

```
<!--Auch in <App.Resources> moeglich-->
<Window.Resources>
  <!-- x:Key entfernen um Style allgemein zu nutzen-->
  <Style x:Key="MyStyle" TargetType="Button">
    <Setter Property="Background" Value="Black"/>
    <Setter Property="Foreground" Value="White"/>
  </Style>
</Window.Resources>

<Button Style="{StaticResource MyStyle}">Drueck Mich!</Button>
```

8.5 Triggers und Validation

```
<Style x:Key="MyStyle" TargetType="Button">
  <Style.Triggers>
    <Trigger Property="Button.IsMouseOver" Value="True">
      <Setter Property="Button.Background" Value="Yellow"/>
    </Trigger>
  </Style.Triggers>
</Style>
```

```
<TextBox Name="TextBox1">
<TextBox.Text>
<Binding Path="Number"
  UpdateSourceTrigger="PropertyChanged"
  ValidatesOnExceptions="True"
  ValidatesOnDataErrors="True"/>
</TextBox.Text>
</TextBox>
```