

# Nutzerzentriertes Prototyping

-

## Design, Implementierung, Evaluation, Analyse

Ilhan Aslan, Chi Tai Dang, Björn Bittner, Katrin Janowski,  
Elisabeth André



**Human Centered Multimedia**

Institute of Computer Science

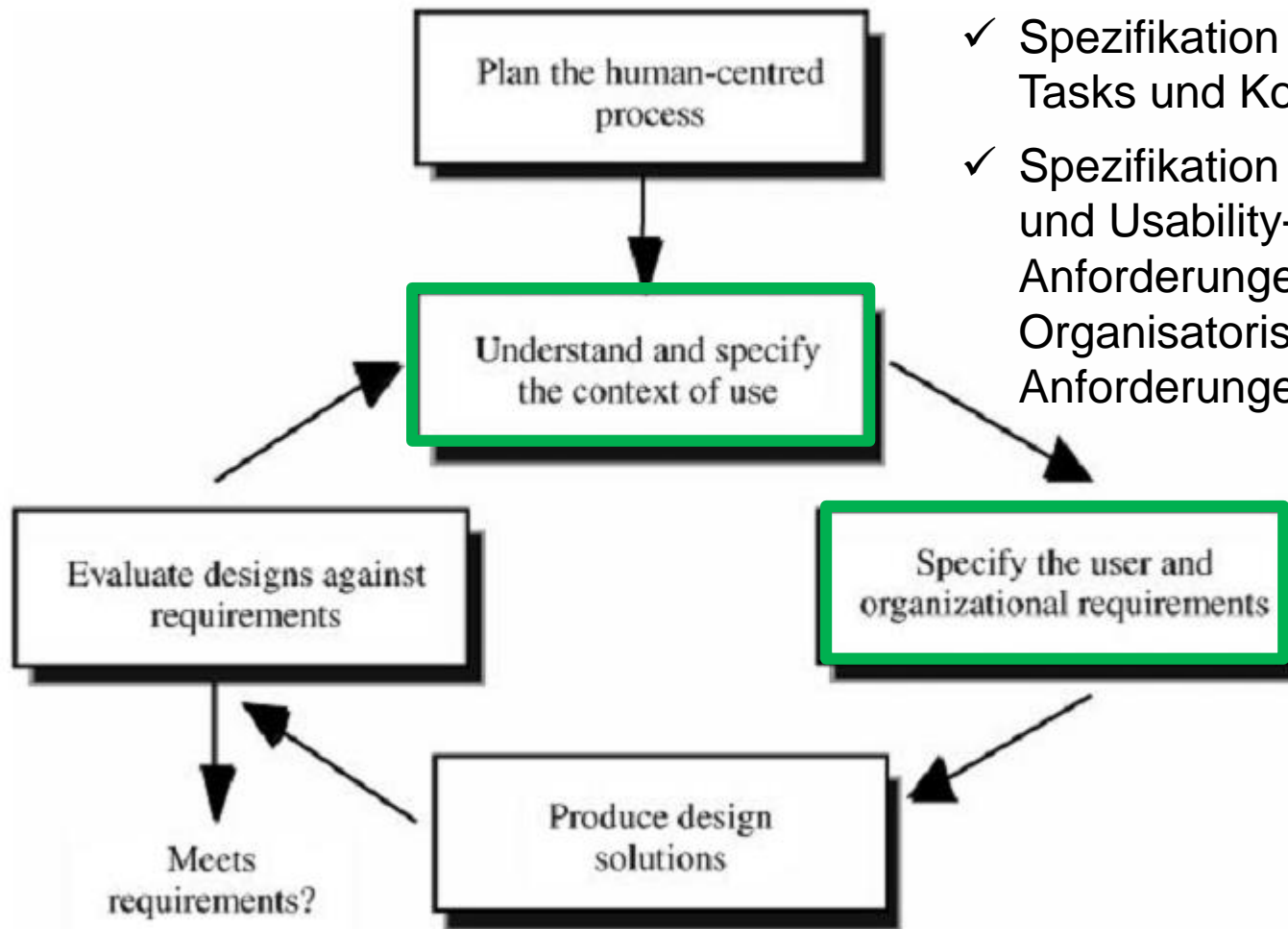
Augsburg University

Universitätsstr. 6a

86159 Augsburg, Germany

WAS ist das Problem?

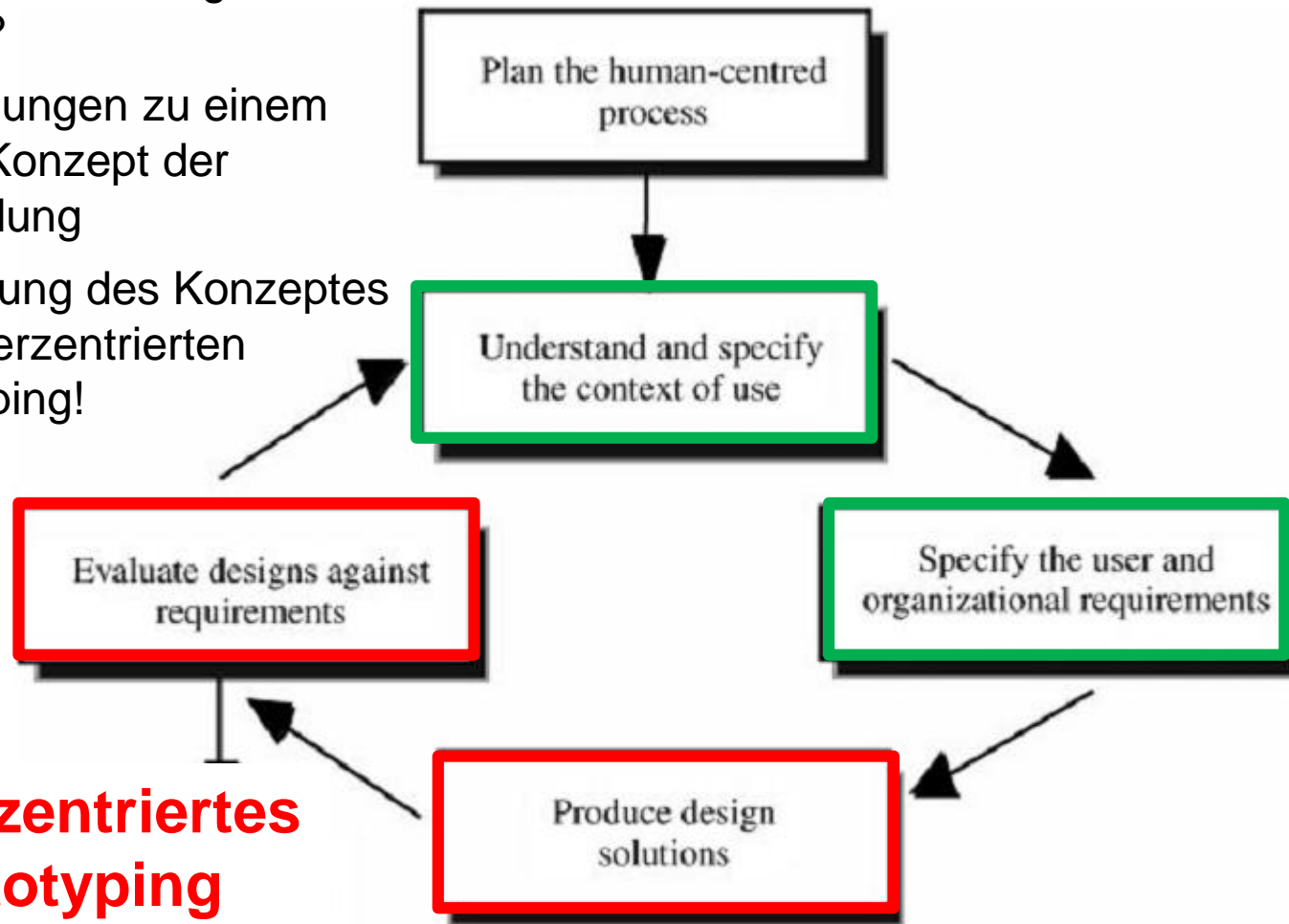
- ✓ Spezifikation von Nutzern, Tasks und Kontext
- ✓ Spezifikation der Nutzer- und Usability-Anforderungen und der Organisatorischen Anforderungen



Key human-centred design activities  
(from ISO 13407)

WIE sollte die Lösung  
aussehen?

- ✓ Überlegungen zu einem  
ersten Konzept der  
Anwendung
- Umsetzung des Konzeptes  
im nutzerzentrierten  
Prototyping!



Key human-centred design activities  
(from ISO 13407)

## Ziele:

- Frühes Einbeziehen der Nutzer (Studien)
- Nutzer sollen früh mit Abbildungen des echten Produktes interagieren
- Schnelles Feedback => Frühes Erkennen und Beheben von Problemen

## Vorteile:

- Interaktion besser vorstellbar also ohne Prototyp
- Prototypen schneller und billiger erstellbar als finales Produkt
- Spart Entwicklungskosten und Entwicklungszeit
- Prototypen dienen als Gesprächsgrundlage (z.B. Storyboard) im interdisziplinären Entwicklungsteam (Programmierer, Designer, Marketing Team...)

- **Schnelle Iterationen**

- Auch kleine Veränderungen werden getestet
- Schnelle Evaluierung neuer Optionen
- Reduziert Risiken von spät entdeckten Problemen (=> Kosten, Zeit)
- Sketches und Papierprototypen sind eine Art Simulation des richtigen Prototypen

Without paper prototyping:

- Idea – sketch – implementation – evaluation

Slow Iteration

With paper prototyping:

- Idea – sketch/paper prototype – evaluation – implementation – evaluation

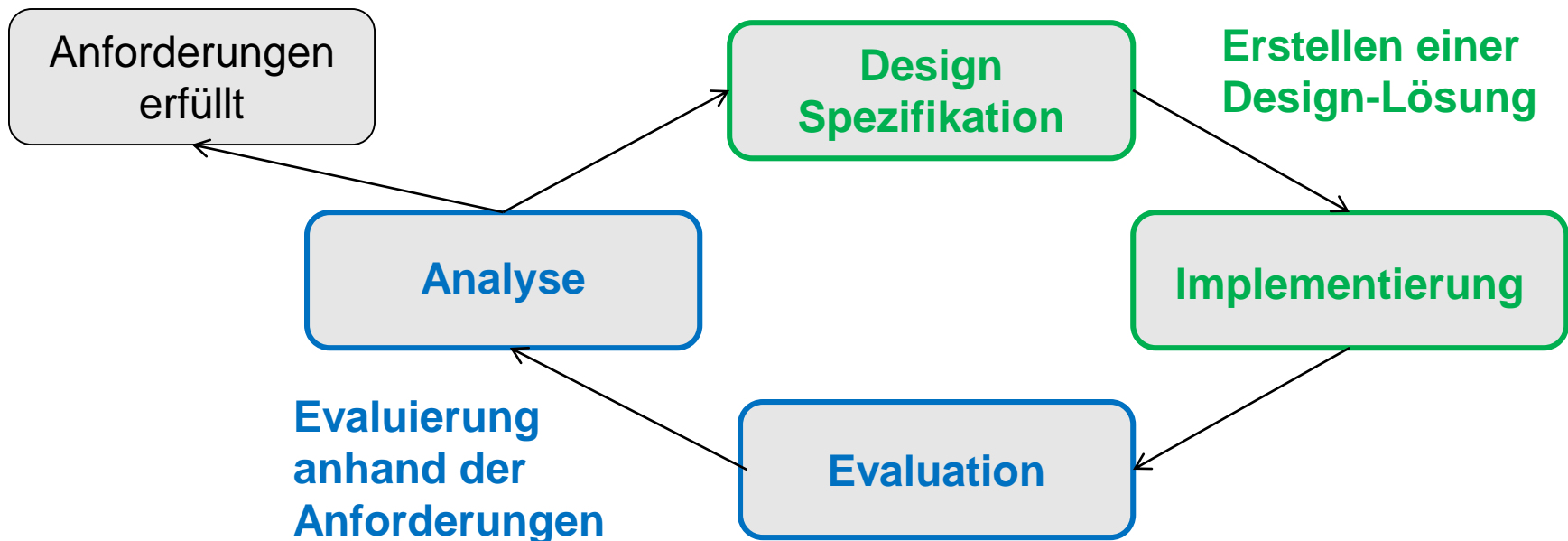
Quick Iteration

Slow Iteration



## Vorgehen:

1. Erstelle eine Design-Lösung (Spezifikation & Implementierung)
2. Evaluiere die Lösung anhand der Anforderungen (Evaluation & Analyse)
3. Wiederhole den Vorgang bis die prototypische Realisierung alle Anforderungen erfüllt.



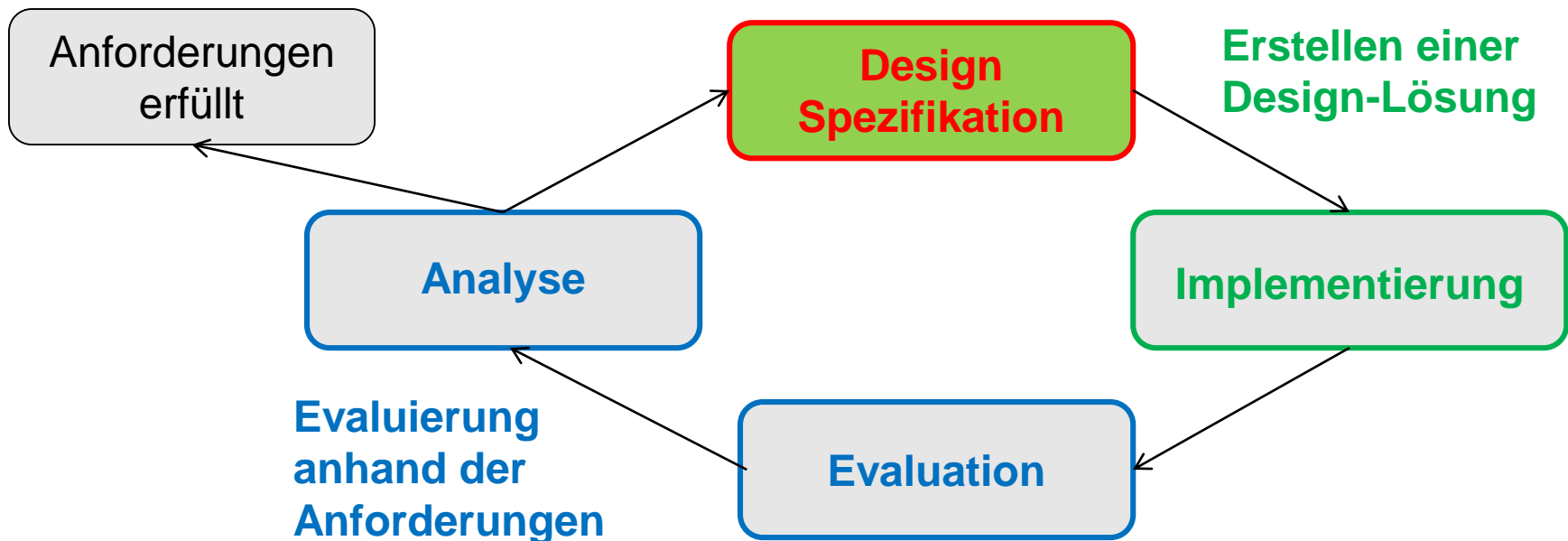
- Nutzerspezifikation: Personas
  - Wer sind meine Nutzer?
  - Was können meine Nutzer?
  - Welches mentale Modell haben meine Nutzer?
- Taskspezifikation: Szenarien und HTA
  - Welche Aufgaben (Tasks) sollen mit der Benutzerschnittstelle erledigt werden können?
  - Wie werden die Aufgaben abgearbeitet?
  - Wie sieht die Interaktion aus?
  - Welche Interaktionen werden mit dem System durchgeführt und welche Objekte haben bei dieser Interaktion eine Bedeutung?

- Kontextspezifikation: Szenarien
  - Im welchem Umfeld wird die Benutzerschnittstelle genutzt?
  - Wie sollte die Benutzerschnittstelle auf die Kontexten reagieren?
- Anforderungsspezifikationen
  - Funktionale Anforderungen:
    - Tasks, die der Nutzer mit dem System erfüllen möchte
  - Nicht-Funktionale Anforderungen:
    - Usability
    - Performance
    - User Experience

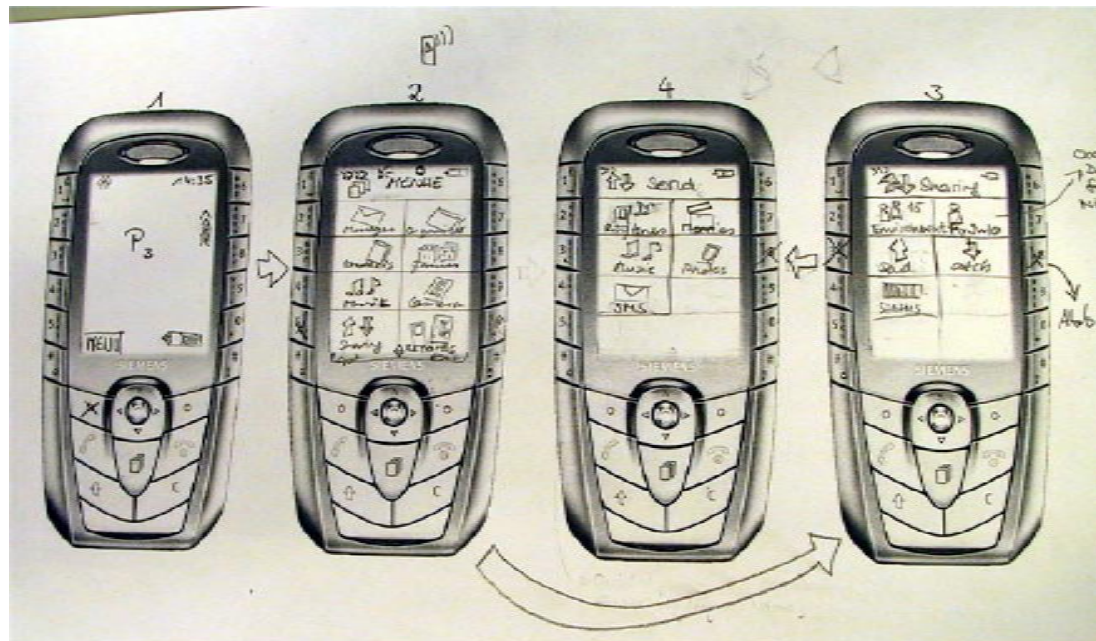


Vorgehen:

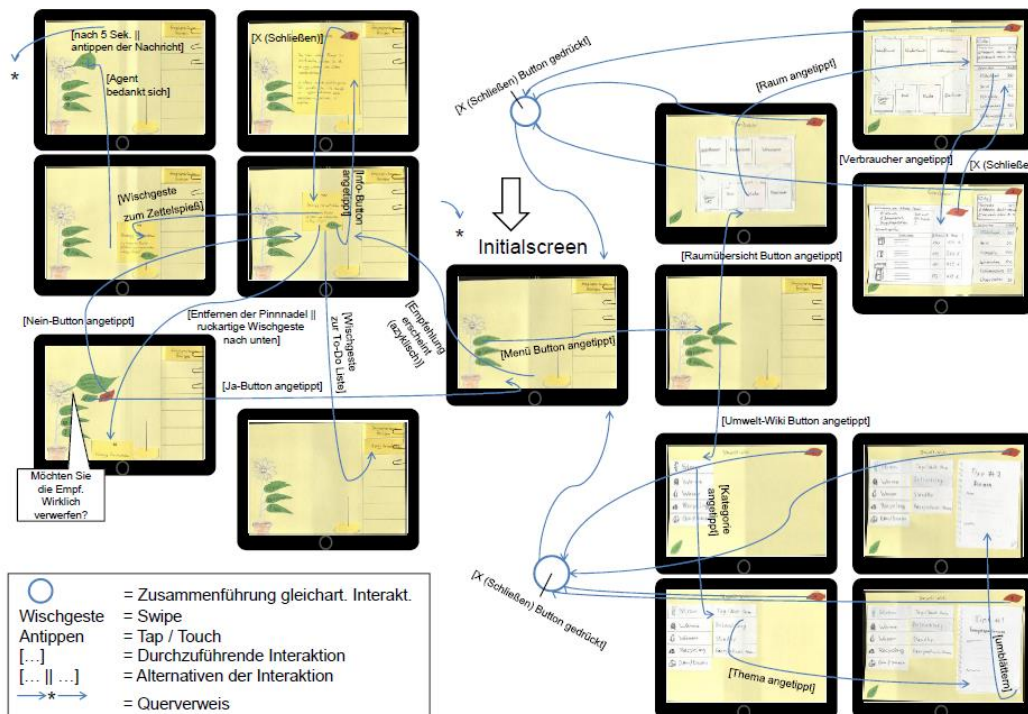
1. **Erstelle eine Design-Lösung (Design & Implementierung)**
2. **Evaluiere die Lösung anhand der Anforderungen (Evaluation & Analyse)**
3. **Wiederhole den Vorgang bis die prototypische Realisierung alle Anforderungen erfüllt.**



- Spezifikation des Transitions- und Präsentationsmodells
  - Grundlage für alle prototypischen Realisierungen
  - Modellierung des Ablaufes der Anwendung
  - Typischerweise für alle Tasks (funktionale Anforderungen):
    - Präsentation: Bildschirme, Dialoge, Menüs, Formulare etc.
    - Transition: Ablaufstruktur durch (Nutzer-)Aktionen



- Spezifikation des Transitions- und Präsentationsmodell
  - Ähnlich einem Zustandsdiagramm
    - Zustände sind Sichten auf die Anwendung (Präsentation)
    - Übergänge (Transitionen) sind Interaktionsmöglichkeiten, die Nutzer mit der Anwendung ausführen können

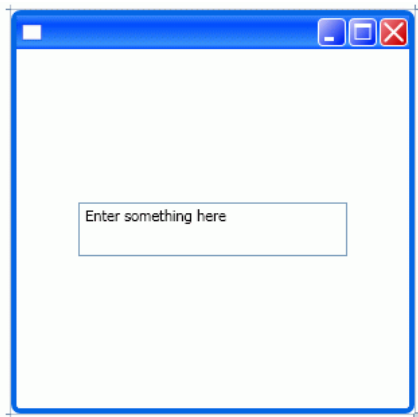


## Schrittweiser Aufbau in kleinen Iterationsschritten

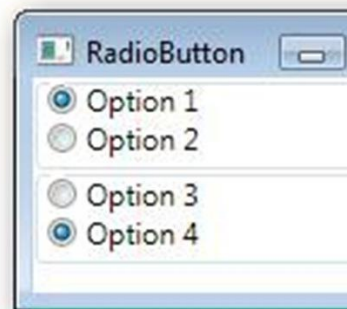
- In den ersten Iterationen:
  - Verzicht auf Details wie Farben und Schriftgröße
  - Nutzung von Papier und Bleistift
  - Nur Schlüsselbildschirme erstellen oder komplett auf die Präsentation verzichten (siehe Beziehungsdiagramme)
- Erweiterung basierend auf Nutzer- und Taskspezifikationen:
  - Passt die Design-Spezifikation zu den definierten Personas?
  - Wurden alle Tasks in der Design-Spezifikation abgedeckt?
  - Hinweis: Sehr hilfreich ist die HTA, um alle Tasks und deren Aktionen und das benötigte Feedback zu berücksichtigen!
  - **WICHTIG!!! Beachtung der Einhaltung von Design-Prinzipien und Richtlinien! (z.B. Aktionen leicht rückgängig machen)**

- Generell wichtig für Präsentationen:
  - **Minimale Anzahl an Screens!**
    - Nur die Screens, die auch wirklich nötig sind!!
    - Sonst ist der Verlust der Orientierung möglich
  - **Erster Screen = Hauptmenü bzw. Hauptansicht!**
    - Zentraler Einstiegspunkt für jeden Haupttask
    - Bei Fehlern immer ein Neuanfang möglich

- Generell wichtig für Präsentationen:
  - **Beachte die Kluft der Ausführung (Norman's 7 Handlungsschritte)!**
    - Welche Aktionen sind nötig bzw. werden erwartet?
    - Alle unnötigen Aktionen müssen weggelassen werden!
    - Beispiele für Interaktionselemente:

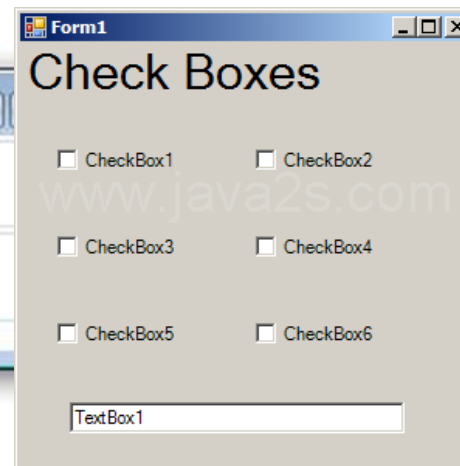


Enter something here



RadioButton

☒ Option 1  
☐ Option 2  
☐ Option 3  
☒ Option 4

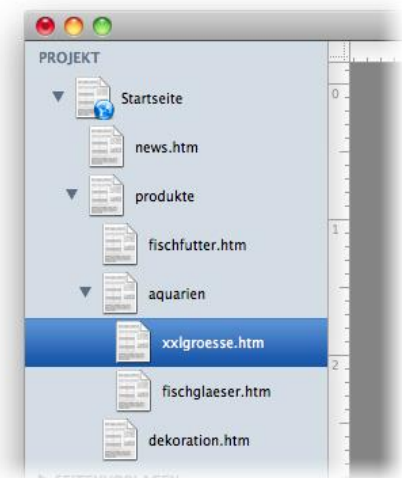


Form1

## Check Boxes

☐ CheckBox1 ☐ CheckBox2  
☐ CheckBox3 ☐ CheckBox4  
☐ CheckBox5 ☐ CheckBox6

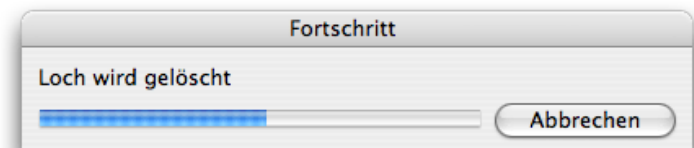
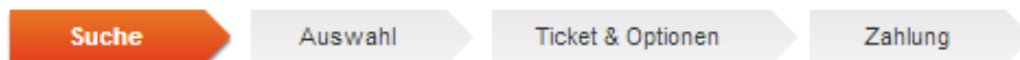
TextBox1



PROJEKT

- Startseite
- news.htm
- produkte
- fischfutter.htm
- aquarien
- xxlgrösse.htm**
- fischglaeser.htm
- dekoration.htm

- Generell wichtig für Präsentationen:
  - **Beachte die Kluft der Evaluation (Norman's 7 Handlungsschritte)**
    - Nach jeder Aktion ein passendes Feedback anzeigen.
    - Visuell (Anpassung der GUI)



- Eventuell auch auditiv und/oder haptisch (z.B. bei Gefahr)
- **Achtung: Übermäßiges Feedback / Kontrolle vom System kann auch nerven!**



- Generell wichtig für Transitionen:
  - Welche Aktionen führen den Nutzer zu welchen Screens?
  - Für jede (relevante) Aktionsmöglichkeit sollte es einen Weg (Transition) zu einem anderen Screen geben!
  - Übergänge im Modell mit ausgeführten Aktionen beschriften (z.B. Speichern)
  - Eine geeignete Interaktionsstruktur wählen. (tief oder flach, Baum oder Graph)
  - **Achtung:**
    - **Immer schnelle und intuitive Transitionen zum Hauptmenü erlauben!**
    - **Nutzer darf sich in der Interaktionsstruktur nicht verlieren!!**



Transitionen sind Interaktionen des Nutzers mit der Nutzerschnittstelle, die von einer Präsentation zu einer anderen führen.

## Aber:

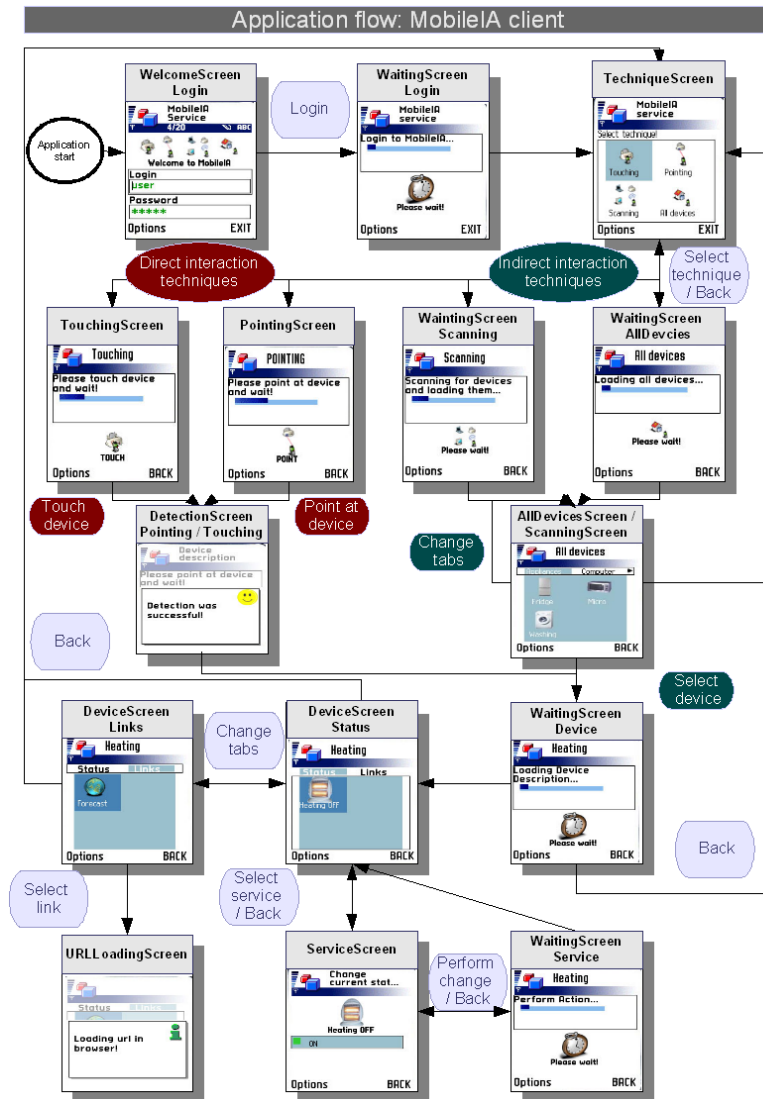
- Wie werden diese Aktionen ausgeführt? Wie würden Nutzer möglichst natürlich die jeweilige Interaktion ausführen?
    - Das Modell **kann** Informationen über das eingesetzte Interaktionsparadigma enthalten (z.B. Sprechen, Gestik, Maus, Tastatur)
  - Wie wird das System die Interaktionen unterstützen?
  - Welche möglichst naheliegenden Objekte (GUI oder reale Welt) können bei dieser Interaktion helfen?
    - Beispiel: Tangibles (z.B. Würfel, Steine, Karten)
- Szenarios enthalten oft wichtige Informationen zu natürlichen Interaktionen und Interaktionsobjekten!

## Beispiele:

- Beziehungsdigramme
  - Visualisierung der Beziehung von Konzepten (z.B. Interaktionsstruktur von Webseiten)
- Card Sorting (ähnlich zu Beziehungsdigrammen)
  - Auf Karten werden Konzepte notiert und deren Beziehung zueinander hergeleitet
  - Oft eingesetzt für hierarchische Strukturen (z.B. Webseiten)

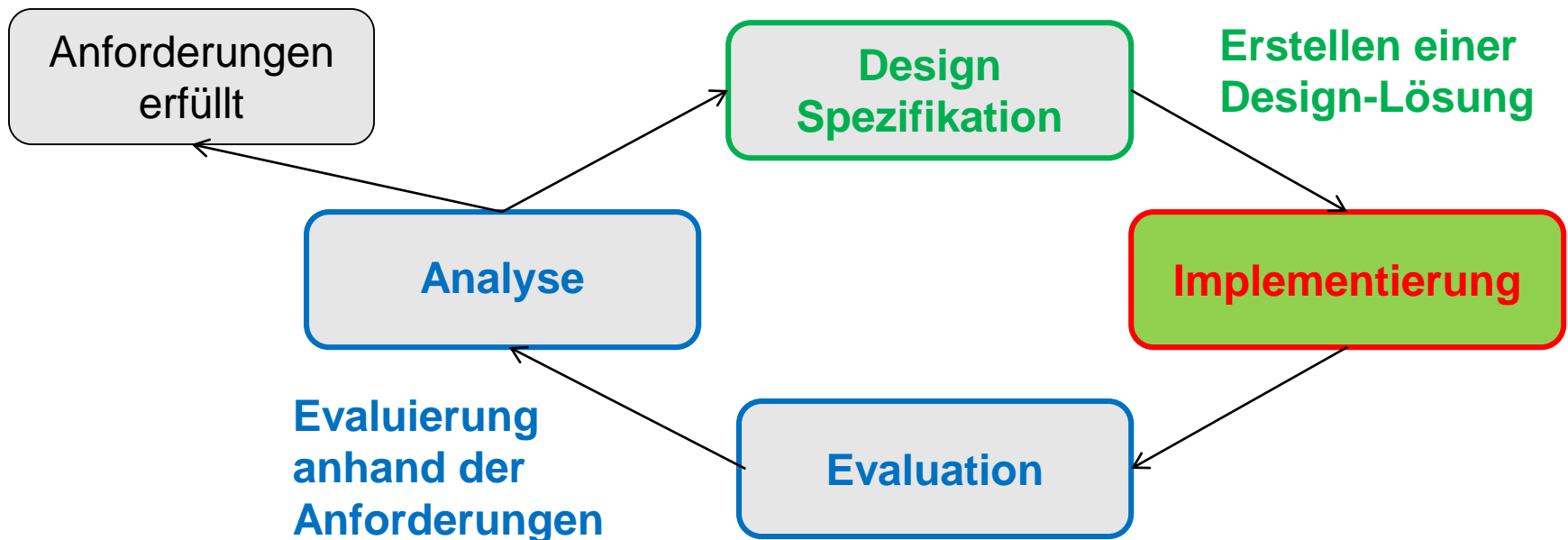


## Beispiele:



Vorgehen:

1. **Erstelle eine Design-Lösung (Design & Implementierung)**
2. Evaluiere die Lösung anhand der Anforderungen (Evaluation & Analyse)
3. Wiederhole den Vorgang bis die prototypische Realisierung alle Anforderungen erfüllt.



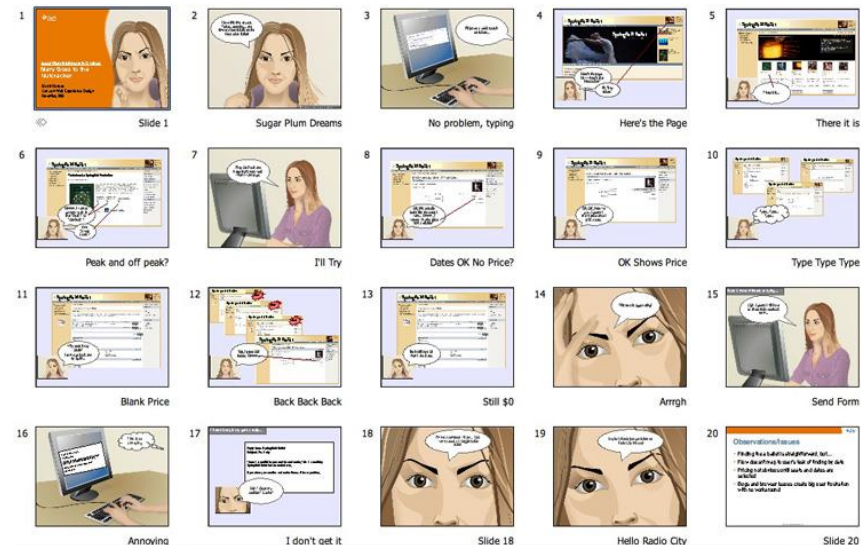
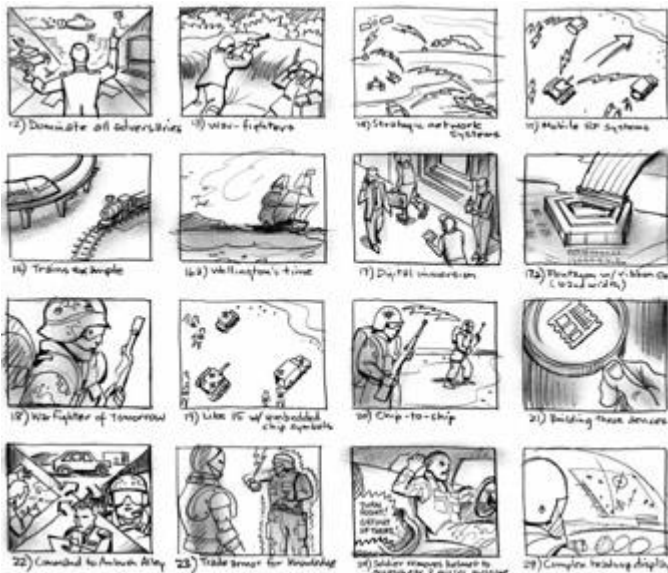
- Was ist ein Prototyp?
  - Erstellung basierend auf dem Transitions- und Präsentationsmodell
  - Hat im Vergleich zum finalen Produkt Einschränkungen in unterschiedlichen Formen
- Welche Einschränkungen sind möglich?
  - Look & Feel
  - Funktionalität der Tasks
  - Umfang der unterstützten Tasks
- Warum?
  - Untersuchung spezieller Aspekte erfordert kein vollständiges Produkt
  - Spart Zeit und Geld!
  - Ermöglicht schnelles Feedback von Nutzern und Projektteam zu jeder Phase im Entwicklungsprozess!



## Beispiele:

### Storyboard

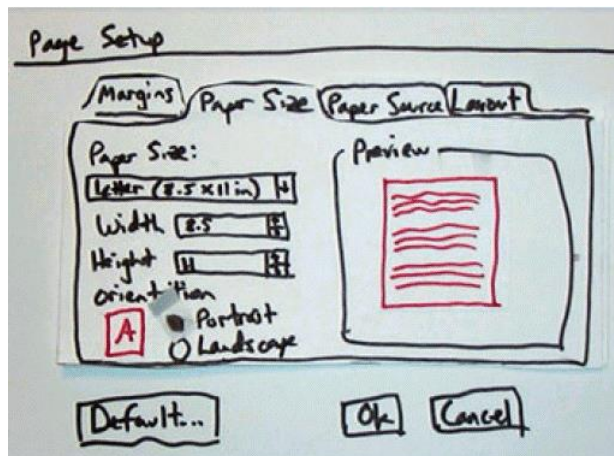
- Sequenzen von Bildern, die die typische Nutzung der Anwendung schildern sollen.
- Beinhaltet Aktionen und Input für das System
- Beschreibt Bilder mit möglichen Menüs, Dialogboxen und Fenstern



## Beispiele:

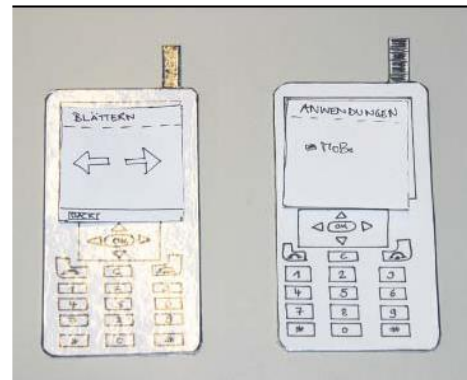
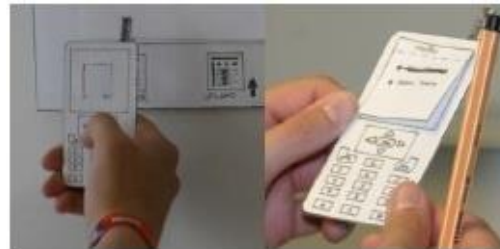
### Papier Prototypen (Pen-and-Paper Prototype)

- Simulation des echten Softwareproduktes auf Papier
- Bei Studien genutzt, um die definierten Aufgaben zu erledigen.
- Eine Person simuliert das System indem nach einer Nutzeraktion die passende Systemreaktion gezeigt wird (z.B. neuer Bildschirm)



Beispiele:

**Papier Prototypen**





## Beispiele:

### Software Prototypen

- Computersimulation, die realistischer ist sich aber noch in der Entwicklung befindet.
- Sonderform: **Wizard-of-Oz Prototyp**  
Simulation einiger Systemfunktionen (z.B. Sprach- oder Gestenerkennung)



**Player**



**Embodied  
characters**



**Operator**

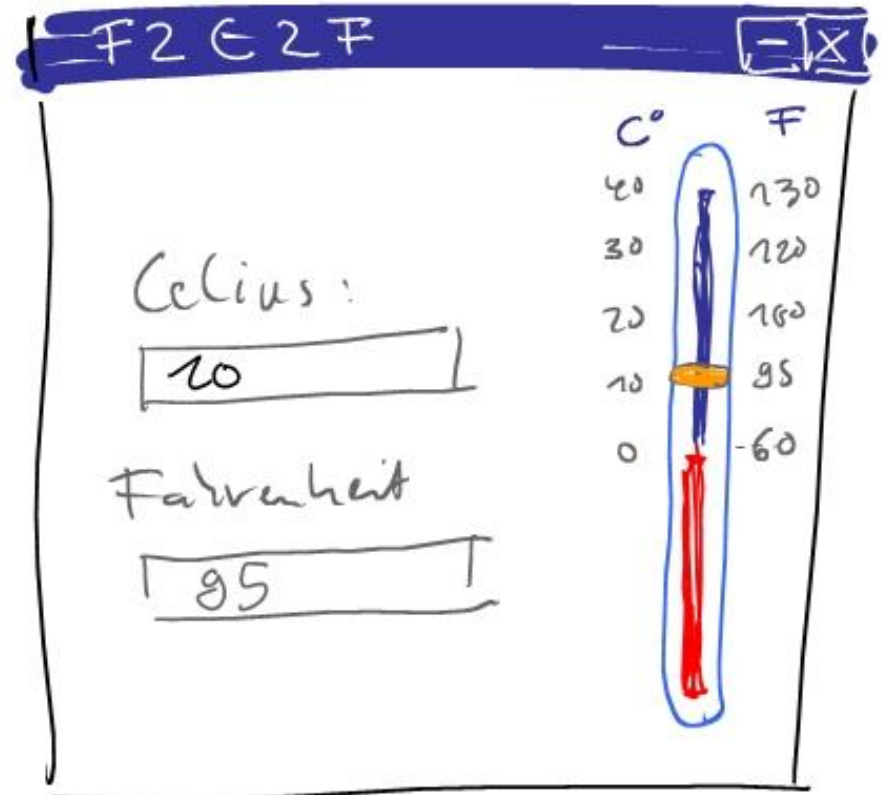


## Unterscheidungsmerkmale für Prototypen:

- Low vs. High-Fidelity
  - Wie niedrig oder hoch ist der Detailgrad des Prototypen im Vergleich zum finalen Produkt?
- Horizontal vs. Vertical
  - Zu welchem Umfang (horizontal) und mit welcher Funktionalität (vertikal) werden die Tasks mit einem Prototypen unterstützt?
- Throwaway vs. Evolutionary vs. Incremental vs. Extreme
  - Wie wird ein Prototyp nach einer Iteration verändert?

## Low-Fidelity Prototypen:

- Statisch
- Nicht computerisiert
- Nicht funktionsfähiger Mock-up
- Tools und Methoden:
  - Sketches und Storyboards
  - Papierprototypen
  - Nutzung von GUI-Buildern für Prototypen
  - Simulatoren mit limitierter Funktionalität



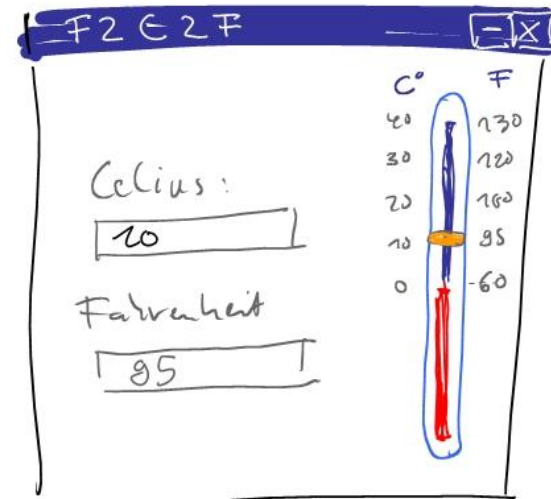
## Low-Fidelity Prototypen:

- Vorteile:
  - Schnell und billig zu erstellen
  - Früher Einsatz ohne eine Codezeile  
(Fehler auf Papier haben weniger Konsequenzen als im Code)
  - Schnelles Finden genereller Probleme
  - Schnelles Erzeugen eines besseren Designs
  - Keine technischen Barrieren! Auch nicht-technische Leute können Einfluss nehmen!
- Nachteile:
  - Kann Nutzern fremd vorkommen
  - beschränkte Systemfeatures

## Hinweis:

- Realistische und ästhetisch ansprechende Prototypen
  - schränken Kommentare bei der Evaluation ein
  - lenken vom grundlegenden Konzept ab
  - führen hauptsächlich zu Feedback zum Design und der Interaktion
- Realistische, aber ästhetisch unausgereifte Prototypen
  - können dazu führen, dass Nutzer Fragen zum Konzept stellen
  - können einfach verbessert werden

➤ **KEEP IT UGLY!** 😊



## High-Fidelity Prototypen:

- Software Prototypen mit Bildschirmen der Anwendung
- Dynamisch
- Computerisiert
- Funktionsfähiges Modell
- Hoher Level of Detail von Darstellung und Technik
- Tools und Methoden:
  - HTML, JavaScript
  - Flash, Director
  - GUI Bilder (z.B. Netbeans Mobility Pack)
  - Java, C++.....

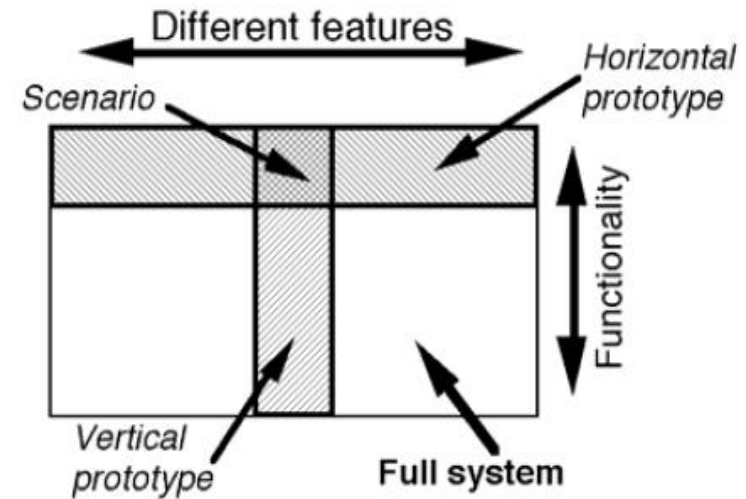


## High-Fidelity:

- Vorteile:
  - Realistischer
  - Detailliertes Feedback
  - tatsächliche Interaktion
- Nachteile:
  - Teuer
  - Funktionalität muss beschränkt werden
  - Kann den Nutzer einschränken

- Horizontale Prototypen

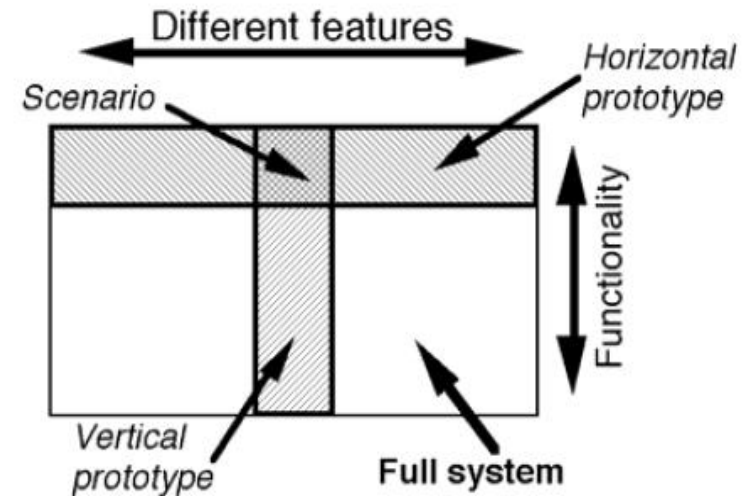
- Viele Tasks (viele Features)
- Geringer Detailgrad der Tasks (Funktionalität in die Tiefe)
- „Von-Oben-Ansicht“ auf das Produkt möglich
- Zeigt Spektrum an Features
- Erlaubt es dem Nutzer durch das System zu navigieren
- Tests für Navigation, grundsätzliches Interface-Konzept, Zugänglichkeit, Nutzerpräferenzen
- Einsatz oft zu einem sehr frühen Entwicklungszeitpunkt
- Einsatz oft zum Testen der grundlegenden Interaktion
- Angemessen für Low- und High-Fidelity Prototypen





- Vertikale Prototypen

- Wenige Tasks (Features)
- Hoher Detailgrad der Tasks (hohe Funktionalität in die Tiefe)
- Tiefenansicht eines Tasks möglich
- Sehr sinnvoll zu einem späteren Entwicklungszeitpunkt
- Einsatz oft zum Testen von Details der Design-Spezifikation
- Nicht-sichtbare Probleme (z.B. Sicherheit) werden nicht berücksichtigt
- Meist mit High-Fidelity Prototypen



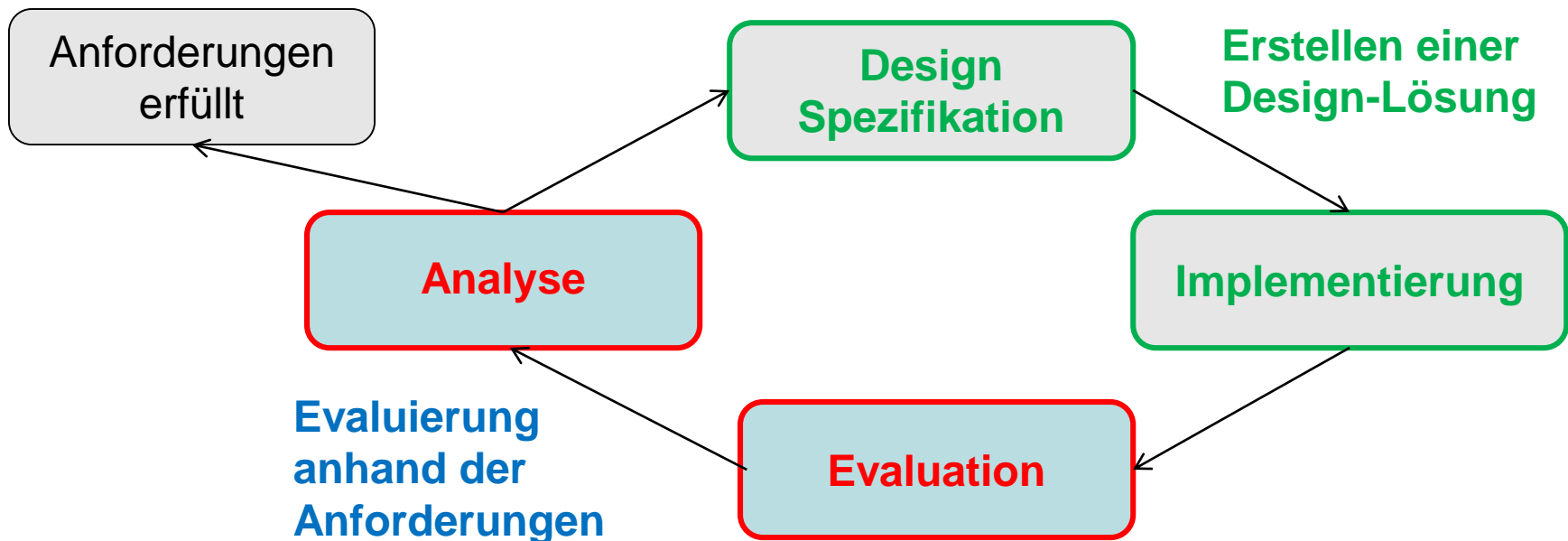
- Throwaway Prototypen
  - Ziel: Anforderungen validieren und ableiten
  - Entwicklung startet mit den Anforderungen, die man am wenigsten verstanden hat.
  - Nach jeder Iteration wird der Prototyp verworfen bzw. weggeworfen und neu realisiert
  - Realisierung schon sehr früh im Entwicklungsstadium auch ohne viel Wissen über den Nutzer und dessen Anforderungen
  - Verfolgt die Idee des Rapid Prototyping
  - Typischerweise Low-Fidelity Prototypen (z.B. Papierprototypen)
  - schnell und kostengünstig

- Evolutionary Prototypen
  - Ziel: Lauffähiges System für Endnutzer
  - Prototyp wird von Iteration zu Iteration verändert bzw. verbessert
  - Entwicklung startet mit den Anforderungen, die man am Besten verstanden hat.
  - Meist am Anfang nicht alle Tasks mit einer hohen Funktionalität
  - Meist von Anfang an robuster High-Fidelity Prototyp als Basis
  - Nach und nach werden weitere Informationen zu Anforderungen erkannt und schrittweise angewandt, um den Prototypen zu verbessern.
  - Das Ergebnis ist dann oft das finale Produkt!

- Incremental Prototypes
  - Entwicklung verschiedener Prototypen, die nach jeder Iteration zusammengefasst werden
- Extreme Prototyping
  - Oft Einsatz bei der Entwicklung von Webseiten
  - Entwicklung in drei Phasen
    - Statischer Prototyp mit Rahmen
    - Bildschirme werden hinzugefügt, aber ohne Dienste
    - Dienste werden zu den Bildschirmen hinzugefügt

## Vorgehen:

1. Erstelle eine Design-Lösung (Design & Implementierung)
2. **Evaluere die Lösung anhand der Anforderungen (Evaluation & Analyse)**
3. Wiederhole den Vorgang bis die prototypische Realisierung alle Anforderungen erfüllt.



- Was wird evaluiert?
  - Sind alle Anforderungen erfüllt?
  - Funktional:
    - Werden alle Tasks unterstützt?
    - Müssen Tasks ergänzt oder entfernt werden?
    - Werden die Tasks angemessen unterstützt?
  - Nicht-Funktional (z.B. Usability):
    - Wo gibt es Problem mit dem jetzigen Design?
    - Gibt es eine Kluft bei der Ausführung der Aktion?
    - Gibt es eine Kluft bei der Ausführung der Evaluation?

- Wie wird evaluiert? (Siehe jeweilige Foliensätze!)
  - mit Endnutzern (Empirische Evaluation)
  - mit Experten in der Domäne und/oder Usability Experten (Analytische Evaluation)
  - Simulationen bei Low-Fidelity Prototypen oder High-Fidelity Prototypen, die noch stark beschränkt sind.
- Werden Probleme entdeckt, wird eine weitere Iteration durchgeführt bis keine Probleme mehr vorhanden sind und alle Anforderungen der Nutzer erfüllt sind.

## Simulation bei Papierprototypen:

- Erzeuge einen Papierprototypen (siehe vorherige Folien)
- Eine Evaluator simuliert das System
- Eine zweiter Evaluator beobachtet und dokumentiert
- Gib dem Nutzer einen konkreten Task und beobachte ihr Verhalten
  1. „Das System“ reicht dem Teilnehmer den jeweiligen Bildschirm bzw. zeigt den aktuellen Systemzustand
  2. Die Testperson interagiert, als ob die Benutzerschnittstelle real wäre (z.B. Klicken, Selektieren, Sprechen)
  3. „Das System“ reagiert auf die Benutzerinteraktion durch ändern des Bildschirms oder ausführen einer Aktion



## Simulation durch Wizard-of-Oz:

- Teile des Prototypen (z.B. Spracherkennung), die noch nicht funktional sind, werden durch Menschen durchgeführt (Simulation im Hintergrund nicht sichtbar für die Testperson)
- Typische Bereiche
  - Multi-Modale Interfaces
  - Simulation eines Spracherkenners
- Bietet dem Nutzer reales Gefühl der Interaktion mit dem System
- Geringerer Implementierungsaufwand nötig

