DIS I Fragenkatalog

Dieser Katalog ist ohne Richtigkeitsanspruch, wer Rechtschreibfehler findet darf sie behalten, oder besser von mir das Originaldokument anfordern und sie selber korrigieren!

Viel Spass beim Lesen!

Quizfragen zu Fitts Law

Keine Gewähr für richtige Antworten. (Ich schreibe hier das auf, woran ich mich in der Vorlesung erinnere):

1) *Microsoft Toolbars haben Beschriftung unter den Icons. Wieso können beschriftete Tools besser angewählt werden?*

- die Fläche der Tools ist durch die Beschriftung größer, nach Fitts' Law ist dies ein Vorteil für die Auswählbarkeit

2) *Eine Grafik-Anwendung habe eine Tool Palette, bestehend aus 16x16 Icons die in einer 2x8 Reihe am linken Rand des Displays ausgelegt sind. Wie kann man die Auswählzeit verbessern, ohne die Icons zu vergrößern oder die Tools vom linken Bildschirmrand zu entfernen?*

- lege die Tools in einer 1x16 reihe am linken Bildschirmrand aus. Dann haben die Tool Icons eine unendliche Fitts' Law Größe (da sie nun alle am Rand liegen)

3) *Ein rechtshänder ist 10 Pixel von der Mitte eines 1600x1200 Schirms entfernt. Er soll ein Ziel von einem Pixel treffen. Welche 5 Pixel auf dem Bildschrim kann er am besten treffen? Was ist die Reihenfolge?*

- Die 4 Pixel am Bildschirmrand haben eine unendliche Fitts' Law Größe, der Pixel genau unter dem Mauszeiger hat eine Distanz von 0. Reihenfolge (vermutlich, für Rechtshänder): Pixel am Mauszeiger, Pixel am linken Bildschirmrand, Pixel am Rechten Bildschirmrand

4) *Microsoft hat eine Taskbar die in die 4 Bildschirmecken gelegt werden kann. Die Taskbar kann auf Auto-Hide gestellt werden oder immer angezeigt werden.*

- Überlappungen mit Aktiven Elementen in Fenstern am unterem Bildschirmrand (die gleichen Probleme gibt’s mit dem Dock in OS X)
- Verschwendung des Fitts' Law Vorteils, da die Leiste an den Bildschirmrändern sehr schnell angewählt werden kann, mit Auto-Hide muss man jedoch das Aufklappen abwarten

5) *Warum kann ein Mac-Pulldownmenü 5x Schneller angewählt werden als ein Windows Menü? Warum hat M$ eine anscheinend so ungünstige Entscheidung gemacht?*

- Das Mac-Menü ist am Obersten Bildschirmrand, hat also einen Fitts' Law bonus, die Windows Menus sind hingegen in den Fenstern, die meistens nicht am Bildschirmrand sich befinden. Ein Entscheidungsgrund könnte sein, dass es sich Applikationen unterschiedlicher Frameworks einfacher innerhalb eines dedizierten Fensters erstellen lassen.

6) *Wo ist der Flaschenhals in Hierarchischen Menüs. Was ist die Lösung bei Macs?*

- Hierarchische Menüs werden unbequem wenn es zu viele Einträge bzw. zu viele Ebenen gibt. Mac-Lösung: keine ahnung, die kommen mir genauso vor. Andere Lösung: Fisheye View für viele einträge. Andere Lösung: WinXP, nur häufigst genützte Einträge antzeigen

7) *Was ist der Vorteil von kreisrunden Pop-Up Menüs gegenüber linearen Pop-Up Menüs?*

- Fitts' Law Vorteil: Die distanz aller Einträge ist immer gleich.

8) *Wie könnte man lineare Popup Menüs besser ausbalancieren, damit alle Einträge gleich schnell angewählt werden können?*

- mögliche Lösung: weiter entfernte Einträge größer erscheinen lassen. Andere Möglichkeit: fisheye view

9) *Die Imac designer entwarfen eine runde Maus und haben die Kommandotasten auf der Tastatur schmäler gemacht. Wieso ist das ungünstig?*

- Maus: Wo ist Vorne? Tastatur: Laut Fitts' Law ist es nun schwieriger die Kommandotasten zu treffen.

10) *Was haben alle Lösungen hier gemeinsam?*

- Lösungsvorschläge nehmen als Grundlage Aussagen des Fitts' Law

Review Fragen:

zu HCI, Fitts' Law

- *Was sind die vier großen Bereiche von HCI*
  - User and Context, Development, Human, Computer

- *Was sind die Hauptbestandteile des CMN Modells?Schlüsselzahlen?*
  - Der Human Model Processor, mit drei Prozessoren:
    - Perception: ~100ms
    - Cognitive: ~70ms
    - Motor: ~70ms
      (Verarbeitungszeiten pro Informationschunk)

- *Was ist Fitts' Law?*
  - Fitts' Law beschreibt den Zusammenhang zwischen der Zeit, die ein Benutzer braucht um einen Knopf zu drücken und dessen Größe (S) und Distanz (D) zum Benutzer
  - Allgemeine Formuliereung: T = a + b (ID) mit ID = log_2(2D/S)
- *Wo hat Tico seinen Abschluss gemacht?*
  - Georgia Tech bzw. Stanford

zu Norman #1

- *Was sind die 7 Stages of Action? Gulfs? Designkonsequenzen?*
  - (Goal ->) Intention -> Action Sequence -> Execution -> Perception -> Interpretation -> Comparison (-> Goal)
  - Gulfs können bie jedem der Punkte entstehen, z.B. bei falscher Aktionssequenz oder einer falschen Perzeption kann die Interaktion schief laufen

- *Was sind Mappings, natürliche Mappings? Arten?*
  - Mappings sind Abbildungen von Bedienelementen zu ihrer Auswirkung das zu bedienende Objekt.
  - Natürliche Mappings folgen den natürlichen Charakteristiken des zu Bedienenden Objekts (z.B. Sitzlehensteuerung sieht aus wie Sitz), oder folgen auch anerkannt kulturellen Analogien. Arten: Relativ, Absolut, etc...

- *Was sind Constraints? Wie unterscheiden sich diese von Affordances? Arten?*
  - Constraints sind Eigenschaften physischer Objekte die dessen Bedienmöglichkeiten einschränken. Somit sind Constraints das Gegenteil von Affordances
  - *Physische Constraints:* Constraints durch die physische Beschaffenheit des Objekts (z.B. Netzstecker)
  - *Logische Constraints:* gewisse Endergebnisse sollen aus logischen Gründen ausgeschlossen werden (z.B. Modellflieger kommt am Ende raus, wenn man Baukasten richtig zusammensetzt)
  - *Semantische Constraints:* Durch Wissen in der Welt, kann die Bedeutung der speziellen Situation erfasst werden (z.B. Pilot im Modellflieger sitzt mit Gesicht in Flugrichtung...)
  - *Kulturelle Constraints:* Gewisse Bedienmöglichkeiten/Konventionen durch Kulturelle gegebenheiten / Konventionen ausgeschlossen, z.B. Rot für „Gefahr“

zu Norman #2

- *Was ist besonders an Audio-Feedback?*
  - Es lenkt die Aufmerksamkeit des Benutzers stark ab
  - Es kann unterstützend aber auch störend sein

- *Was ist der Unterschied zwischen Mistakes und Slips (Fehler und Versehen)?*
  - Mistake: Falsche Aktion durch falsche Zielsetzung (bewusster Fehler)
  - Slip: Zielsetzung richtig, jedoch Aktionssequenz kaputt (unbewusster Fehler)

- *Was für verschiedene Slips gibt es?*

Slip - Typen

Slip Typ	***Beschreibung***	***Beispiele***
Capture Error	Entsteht wenn zwei Aktionssequenzen mit ähnlichem Anfang aber unterscheidlichem Ende vorliegen. Die besser erlernte Aktionssequenz kann die andere „Einfangen“ (Capture), obwohl diese nicht ausgeführt werden sollte.	- Sonntags zur Arbeit fahren - ausgeliehenen Stift „einsacken“
Description Error	Die Intention wird nicht hinreichend detailliert formuliert und somit kann mehr als eine Aktionssequenz zutreffen. Findet oft statt wenn zwei ähnliche Objekte physisch nah an einender platziert sind (z.B. Schalter).	- T-Shirt in den Mülleimer anstatt Wäschekorb werfen - Deckel auf falschen Behälter machen - Orangensaft in die Kaffekanne füllen
Data-Driven Error	Ankommende Sinneswahrnehmungen stören die sich gerade in Ausführung befindliche Aktionssequenz, was zu unerwünschten Aktionen führt.	- Zimmernummer anstatt Telefonnummer wählen, weil man vor sich das Nummernschild sieht.
Associative	Interne Assoziation löst falsche Aktion aus.	- Beim Abnehmen des Telfons „Herein“

Activation Error	Auch bekannt als *Freudsche Slips*. [denke Freudsche Versprecher].	sagen.
Loss-of-Activation Error	Vergessen des Ziels während Aktionssequenz abläuft. Spezielle Version davon, zu vergessen etwas zu machen. Aktionssequenz kann durch Wiederholen des ursprünglichen Stimulus wieder aktiviert werden.	- ins Schlafzimmer laufen und vergessen was man machen wollte - reaktivierung: wieder ins Wohnzimmer laufen und etwas sehen dass einen an den Grund errinert, weshalb man ins Schlafzimmer gelaufen ist
Premature Conclusion Error	Vergessen, eine Aktionssequenz zu ende zu führen, weil der Hauptteil des Ziels schon erreicht ist.	- Bankkarte im Automaten vergessen - Originale im Koperier lassen
Mode Error	Die falsche Aktion auslösen, da sich das Gerät in einem unerwartetem Modus befindet. Passiert immer, wenn Geräte mehr Funktionen haben als Bedienelemente. Prominentes Problem in vielen Software Benutzeroberflächen.	- `:wq` - programmieren eines Videorekorders

- *Wie korrigieren wir Slips?*
  - Enge und Flache Entscheidungsstrukturen
  - Fehler beim Design in Betracht ziehen
  - Forcing Functions

- *Was sind Forcing Functions?*
  - Forcing Functions verhindern das Durchführen einer fehlerhaften / gefährlichen Aktionssequenz
  - *Lockout:* Gewisse Handlungsmöglichkeiten werden verhindert (z.B. deaktivierte Schaltflächen)
  - *Lockin:* Benutzer muss eine Gewisse Aktionssequenz fertigstellen, bevor er was anderes machen kann (z.B. Wizard beim Einrichten eines Email-Faches)
  - *Interlock:* erzwingt Einhaltung der korrekten Aktionssequenz, z.B. (beim Motorrad) Ausschallten der Zündung beim Einlegen des Ganges während der Kickstand noch ausgklappt ist

- *Wie können mittels UI Design Fehler verhindert werden?Was sind Normans „7 Principles of Design“?*

  Für Benutzer Entwerfen

  - Form sollte Funktion Folgen: d.h. Funktionalität geht vor ausgefallenem Design
  - Designer sind meistens keine Benutzer. Sie sollten aber ihre eigenen Produkte auch mal verwenden
  - Es gibt keinen „Durchschnittsbenutzer“. 99% iges Design lässt schon Millionen benutzer „im Trockenen“, Alterserscheinungen fangen mitte 20 an!

  Sieben Prinzipien des Designs

  - Wissen in der Welt und im Kopf verwenden
  - Aufgabenstrukturen vereinfachen
  - Sichtbarkeit, Gulfs überbrücken
  - Natürliche Mappings verwenden
  - Natürliche und künstliche Constraints verwenden

- Bei Entwicklung immer Fehler annehmen (*design for error).*
- Wenn alles andere scheitert, standardisieren.

zu HCI Geschichte #1

- *Was war die Programmierschnittstelle von Rechnern vor der v. Neumann / Zuse Architektur?*

Per Hand Kabel richtig zusammenstecken.

- *Wie sehen 0-D / 1-D / 2-D UIs aus?*

0-D: Lochkarten 1-D: Terminals (Konsolen) 2-D: Grafische Benutzeroberfläche, Maus

- *HCI Innovationen im Sketchpad?*

  - Vektorgrafik
  - direkte Manipulation
  - Constraints
  - objektorientierte Aspekte

- *HCI Innovationen in NLS / Augment?*

  - Mauszeiger
  - Hypertext
  - Email
  - Simultananwendungen
  - WYSIWYG Texteditor

- *Was hat den Apple II erfolgreich gemacht?*

Die Tabellenkalkulation VisiCalc führte dazu, dass der Apple II von vielen Unternehmen gekauft wurde, und somit ihm zu seinem kommerziellen Erfolg führten.

- *HCI Innovationen im Xerox Alto / Star?*

  - Grafische Benutzeroberfläche
  - Benutzerzentrierter Entwicklungsprozess (DIA Cycle)
  - Schaffen der „Desktop“, „Drag & Drop“ und „Fenster“ Metaphern
  - Ethernet, Email
  - Smalltalk, Objektorientierung

- *Erfahrungen aus dem Entwicklungsprozess des Xerox Star?*

  - iterativ Entwickeln
  - für und mit dem Benutzer entwickeln (Psychologen, Benutzertests)
  - für Spezialbereiche Spezialisten anheuern (z.B. Grafikdesigner für Optik)
  - sorgfältiger Entwurf vor dem ersten coden

- *Innovationen von Mac / Windows / Motif?*

    - Mac: Erste erfolgreiche Kommerzialisierung einer GUI, Konsistenzrichtlinien für GUI Design
    - Windows: Erste erfolgreiche GUI der PC Welt, massenverbreitung
    - Motif: einfachere Entwicklung von GUIs in der Unix Welt, Objektorientierung

zu HCI Geschichte #2

- *Wichtigsten Fortschritte von:*
    - *Put that there*
    - *Apple knowledge navigator*

    - Put that there:
        - erkennen von Eingabegesten
        - kombination von Spracherkennung und anderer Eingabemodi
            - erstes multi-modal interface

    - Apple knowledge navigator:
        - stellt das Konzept des „Persönlichen Agenten“ vor
        - Schaukasten für diverse Multimediaapplikationen (z.B. Videokonferenz)
        - Zusammenspiel von Sprach Ein- / Ausgabe und klassischer Ein- / Ausgabemethoden interessant

- *Unterschiede zwischen Ubicomp und VR?*

    - durch ubicomp verschwindet der PC aus der welt, er wird unsichtbar aber immer präsent. Die Wahrnehmung der Umgebung wird dann höchstens vom computer unterstützt, nicht ersetzt („Augmented Reality“)

    - VR versetzt den Benutzer in einer virtuelle, für ihn in der Realität nicht erreichbare oder zu gefährliche Welt

- *PARC Ubicomp Geräteklassen?*

PARC orientierte sich an Objekten in einer reellen Büroumgebung:

- **Tab**: handheld device, skala in inches, x*100 pro Raum, elektronisches Post-it-Note
- **Pad**: Tablet-PC äquivalent: skala fuß, x*10 pro Raum, elektronischer Schreibblock
- **Board**: Großer Wandbildschirm als Ersatz für Whiteboards oder Tafeln. Skala yard, x*1 pro Raum

- *Was lernte man aus dem SUN Starfire Video?*

Das Video stellte wichtige Richtlinien zum Erstellen von Videoprototypen auf:

- die Frage muss immer gestellt werden, ob die Annahmen in 10 Jahren auch gültig sind
- der Videoprototyp sollte genauso wie andere Prototypen iterativ entwickelt werden
- für die Glaubwürdigkeit hilft es immer auch Szenarien einzubringen, wo etwas schiefgeht
- unmögliche Hardwareentwürfe sollte vermieden werden (kein Science-Fiction)
- Interface zuerst entwickeln, dann Film mit Restbudget drehen (z.B. Maus, Sprache, „reverse

angle“, preiswerter als Gestenerkennung, Stiftbasierte Eingebe und Simulation der UI anfertigen)

zu DIA Cycle

- *DIA Cycle:*
  - *welche Stufen? Unterschiede vom Wasserfallmodell?*
  - *Was passiert mit jeder Iteration?*

Der DIA Cycle hat drei Stufen, **Design, Implement, Analyse**

- Design: Konzeption und Spezifikation eines Designs
- Implement: Hier wird das Design implementiert
- Analyze: Qualitätssicherungsschritt. Hier werden (Benutzer-)Tests durchgeführt und fehler gefunden bzw. Verbesserungsvorschläge für die nachfolgende Iterationen ausgearbeitet

Im Unterschied zum Wasserfallmodell ist der DIA Cycle iterativ und die Arbeitsschritte sind nicht fix sondern nur sehr locker vorausgeplant. Dass man mal ein paar Schritte zurückgeht und einen Neuansatz wagt kann bei den ersten Iterarationen durchaus passieren. Im allgemeinen werden die Iterationen aber immer spezifischer und behandeln immer kleinere Teilbereiche. Man geht also vom Groben Konzept mit Prototypen über zu fertiger Implementation und Detailverbesserungen.

- *Zwei Fragen die man sich am Anfang eines Projektes stellen muss?*

  1) ***Wer sind die Benutzer?***
  2) ***Was wollen sie mit dem System machen?***

- *Brainstorming Techniken?*

  - Zeit: 5-10 Minuten
  - ein Protokollant zum Mitschreiben
  - man sollte sich entspannen, spaß haben und gute Brainstormer einladen
  - noch nicht über die Ideen urteilen, keine Kritik ausüben
  - stattdessen auf den Ideen der Anderen aufbauen
  - es geht um Quantität, nicht Qualität: verrückte ideen sollten zugelassen werden
  - eine bestimmte Anzahl (c.a. 100) von Ideen sollte als Richtwert dienen
  - die Fragen „who-what-where“ sind praktisch um Ansätze für neue Ideen zu bekommen (z.B. Was machen ein Polizist, Krankenschwester, Lokführer in einem Bahnhof...)

- *Concept Mapping Techniken?*

- Mindmaps
- Storyboarding

- *Wozu dienen Storyboards im UI Design?*

Storyboards sind Grafische Beschreibungen einer Interaktionssequenz. Sie dienen als Konzeptioneller Entwurf eines Systems und helfen auch Benutzerfeedback zu allgemeinen Designideen zu bekommen, die sie leicht zu verstehen sind, und schnell gezeigt werden können.

- *Starman?*

- *Top-Ausreden, keine Benutzer involviert zu haben?*

  - „Mein System ist für jeden Nützlich!“
    - Wenn das System so toll ist, sollte das Finden von Teskandidaten doch kein Problem sein
    - wenn nicht, dann heisst „jeder“ „niemand“
  - „Bin selbst ein durchschnittlicher Benutzer!“
    - Würdest du es dann auch *jeden Tag* benutzen?
    - Die Tollen Features die sich ein Designer ausgedacht hat müssen erstmal von anderen verstanden werden

- *Dimensionen, um Benutzer zu klassifizieren?*

  - Anfänger, Fortgeschrittener, Experte
  - Auch demographische Informationen: Alter, Einkommen, Geschlecht, Staatsangehörigkeit, etc...

zu Prototyping

- *Wann wird geprototypt?*

  - In der „Implement“ Phase in frühen Iterationen des DIA Cycle s

- *Was sind Papier Prototypen?*

  - Es gibt Storyboards, Flipbooks, Post-it Notes und auch Hardwareprototypen aus Pappe
  - Papierprototypen sind meist Handzeichungen des Interfaces auf Pappe die verschiedene Bildschirmausschnitte zeigen
  - Papier Prototypen sind Low-Fidelity und dienen dazu Feedback über die allgemeinen Qualitäten des Interfaces zu erhalten

- *Software Prototypen?*

Softwareprototypen sind alle in Software erstellten Prototypen.

  - *Wie kann man sie limitieren?*

    - Horizontal: Alle Bedienelemente aufführen, Funktionalität ist aber nicht implementiert
    - Vertikal: exemplarisch wird eine Funktion vollständig (in die Tiefe hin) implementiert

  - *Was sind die Gefahren?*

    - Werden für das Endprodukt gehalten
    - Feedback von Benutzern wird verfälscht, da zu viel Feedback auf Detailebene oder Angst ein „fast fertiges“ Produkt zu kritisieren

- *Typen von SW. Prototypen / Tools?*
    - Flash / Director, Skriptsprachen, einfache Hochsrachen, Interface Builder, Web, etc..

- *Wizard of Oz?*
    - Ein Mensch simuliert die Ausgabe des Interfaces indem er es z.B. per Hand steuert
    - gut für futuristische Projekte

- *Harware Prototypen?*
    - Papp- /Holz- /Styroporattrappen der Hardware auf der die Software am Ende läuft
    - nützlich bei Handys, PDAs, wo die physische Interaktion auch sehr wichtig ist

- *Was machtg man mit einem Prototypen?*
    - Wurde der Prototyp schnell und billig entworfen wird er meistens weggeschmissen
    - Wenn er aber gut ist und potenzial zeigt, kann er in den nächsten Iterationen weiternentwickelt werden

zu 9 Golden Rules

1) Keep the Interface simple
2) Speak the users's language
3) Be consistent and predictable
4) Provide feedback
5) Minimize memory load
6) Avoid errors, help to recover, offer undo
7) Design clear exits and closed dialogs
8) Include help and documentation
9) Offer shortcuts for experts

zu Evaluation

- *Wann evaluieren?*
    - Beim „Analyze“ Schritt im DIA Cycle

- *Unterschied zwischen Analysen im Labor und im Feld?*
    - Im Labor hat man mehr Analysemöglichkeiten, durch bessere Ausrüstung an Geräten und eine kontrollierte Umgebung
    - Die Laborumgebung kann jedoch das Verhalten der Benutzer verändern
    - Im Feld (natürliche Umgebung des Benutzers) ist das Testen realistischer da die Situationen und das Benutzerverhalten natürlicher ist
    - Die natürliche Umgebung ist besser für Langzeitstudien geeignet
    - Nachteile sind z.B. Geräusche und Ablenkungen
    - Auch Versuche im Feld fühlen sich wie ein „Test“ an, was das Verhalten beeinflusst

- *Was versteht man unter „Participatory Design“?*

- im Participatory Design werden die Benutzer in den gesamten Entwicklungsprozess eingebunden, um ein besser an den Benutzer angepasstes Produkt zu erzeugen
- dies erfordert Techniken für Teamkommunikation wie Brainstorming, Storyboarding, Workshops, Interviews, etc..
- Probleme sind ein hoher Aufwand und ein Konflikt mit Kundenhierarchien

- *Techniken zur Evaluation ohne Benutzer?*
  - **Cognitive Walkthrough:** Kognitiver Spezialist geht Anwendung Schritt für Schritt durch und überlegt sich die Kognitiven Vorgänge des Benutzers
  - **Heuristic Evaluation:** Interface wird anhand bekannter Heuristiken (z.B. 9 Golden Rules) analysiert und bewertet
  - **Model Based Evaluation:** Analyse der Schnittstelle anhand eines fundierten Interaktionsmodells, wie z.B. Key-Level GOMS, Patterns, oder Design Rationalen
    - **GOMS**: **G**oals **O**perators **M**ethods **S**election Rules
      - Ausführungs -und Lernzeiten eines Interfaces anschätzen bevor sie implementiert wird
      - Verschiedene Stufen: Keystroke-Level, CMN-GOMS, NGOMSL, CPM-GOMS...
      - *Raskin*: GOMS liefert obere Schranke für Interaktionsdauer eines relativ geübten Benutzers

- *Techniken zur Evaluation mit Benutzern?*
  - Qualitativ:
    - **Model Extraction:** Experte Zeigt dem Benutzer Bildschirmausschnitte, Benutzer muss erklären, wie er denkt dass die Benutzeroberfläche funktioniert, d.h. sein mentales Modell der Sache schildern -> schlecht um zu erfahren, wie sich das Modell mit der Zeit verändert
    - **Silent Observation:** Tester schaut dem Benutzer in seiner natürlichen Umgebung **still** zu, wenn er die Schnittstelle bedient -> hilft große Probleme zu entdecken, man erfährt aber wenig über den Entscheidungsprozess des Benutzers
    - **Think Aloud:** Wie Silent Observation, aber Benutzer muss jedoch jede Entschedung / Aktion laut beschreiben (Zustand, Ziele, Aktionen). -> Problem: fühlt sich komisch an, laut denken kann das Verhalten der Benutzer verändern
    - **Constructive Interaction:** Zwei Leute arbeiten zusammen an einem Problem, meist ein erfahrenerer Benutzre und ein Anfänger. Die normale Unterhaltung der Benutzer wird aufgezeichnet. -> Weniger unangenehm als Think Aloud. Man bekommt einblick in Mentales Modell des Anfängers sowie des Fortgeschrittenen
    - **Retrospective Testing:** Zusätzliche Aktivität nach einer Untersuchung. Benutzer und Tester schauen sich Video des Tests noch einmal an, Benutzer kommentiert seine Aktionen -> Gut für ein Anschließendes Interview, Falsche Erinnerungen werden beseitigt
  - *Quantitativ:*
    - **Experimente:**
      - Hypothese -> Benutzerwahl -> Variablen festlegen -> Durchführung -> Korrelationen bestimmen -> Hypothese stärken oder verwerfen
    - **Variablen**:
      - **Unabhängige:** werden vom Tester festgelegt
      - **Abhängigen:** werden beim Test gemessen
    - **Durchführung:**
      - **In Groups:** jeder Testkandidat führt *alle Varianten* des Experiments durch -> Lerneffekte, individuelle Eigenschaften heben sich auf
      - **Between Groups:** jeder Testkandidat füjrt nur *eine Variante* des Experiments durch

-> kein Lerneffekt, mehr Testkandidaten benötigt
  - **Korrelationen:** Student's T-Test, ANOVA, Regressionskurve, andere mathematische Verfahren...

- *Wie arbeitet man mit Testkandidaten:*
  - **Einverständniserklärung** unterschreiben lassen
  - **Privatsphäre** des Testkandidatens schützen
  - **nicht** die **Zeit** der Kandidaten **verschwenden**, den Test gut vorbereiten
  - **jede Frage**, die nicht das Ergebnis des Tests beeinflussen könnte **beantworten**
  - Kandidat darauf hinweisen, dass der Test jederzeit **abgebrochen** werden kann
  - angenehme **Atmosphäre** schaffen
  - am Ende des Tests eine kleine **Belohnung** geben, Werbegeschenke, Bonbons meist besser als Geld

zu Notationen

- *Wieso möchte man formal eine UI spezifizieren?*
  - Trennen von UI und Algorithmen beim Programmieren
  - Systementscheidungen und Benutzerentscheidungen schwierig zu trennen
  - Fehlerbehandlung der Benutzereingabe im Programmcode nimmt die Überhand
  - daher: Besser das UI in einer spezialisierteren, separaten Sprache spezifizieren

- *Textnotationen:*
  - *BNF, regExp, Produktionsregeln?*
    - **Grammatiken / BNF /regeXp:** gut für kommandozeilenbasierte Eingabemethoden geeignet, eher nicht für GUIs
    - **Produktionsregeln:**
      - wenn B*edingung* dann *Aktion*
      - gut bei Nebenläufigkeit, schlecht um sequenzielle Abläufe zu beschreiben

- *Graphnotationen:*
  - **STNs, Hierarchische STNs:**
    - Darstelle der Bedienung als Automaten, mit leichten Modifikationen
    - finden Verwendung beim Prototyping
    - Uneffizient wenn viele Informationen pro Zustand gespeichert werden müssen, Kombination von STNs multipliziert die Zustandsmenge -> *Zustandesexplosion*
  - **Petri Netze:**
    - (Trigger und Tokens)
    - schaffen Abhilfe bei der Zustandsexplosion
    - besser bei Nebenläufigkeit
  - **State Charts (Harel):**
    - Erweiterungen von STNs, mit History, Hierarchie, Escapes, Nebenläufige Unterautomaten
  - **Flowcharts, JSD, (roh-)Code:**
    - alles Werkzeuge von Programmierern
    - Flowcharts sehr nützlich, aber sehr formal und aufwändig
    - JSD: von links nach rechts, oben nach unten, * heißt Wiederholung
    - Code: schlimm! (Schwierig zu begreifen für nicht-Programmierer)

- *Eigenschaften überprüfen:*
  - **Vollständigkeit:** Alle Aktionen eine Auswirkung? Alle Pfade begehbar?
  - **Determinismus:** mehrere Pfade für eine Aktion? -> absichtlich oder aus Produktionsregeln

ableitbar (unabsichtlich)
- **Nested Escapes:** (?)
- **Konsistenz:** gleiche Aktion, gleiche Reaktion? Modes und Sichtbarkeit...
- **Zustände:** Erreichbarkeit? Umkehrbarkeit? gefährliche Zustände (will man nicht Erreichen)?