Vorlesung "Auswirkungen der Informatik"

Benutzungsorientierte Entwicklung

Lutz Prechelt
Freie Universität Berlin

- Begriff "Gebrauchstauglichkeit"
- Entwicklungsprinzipien
  - Direkter Kontakt zu Benutzer_innen
  - Fortlaufendes Benutzbarkeitstesten
  - Iterativer Entwurf
  - Integrierter Entwurf
- Benutzungsorientierte Entwicklung
- Fallstudie zur Verwendung der Prinzipien
- Ratschläge an Organisationen
- 2 Beispiele für Fehlschläge
- Ergebnisse von Benutzbarkeitsprüfungen

Wichtigste Eigenschaft??

- Wenn man von einem Informatiksystem nur eine einzige Eigenschaft fordern dürfte, welche wäre das?

Gebrauchstauglichkeit

Gebrauchstauglichkeit

- Gebrauchstauglichkeit umfasst nach DIN/EN/ISO 9241-11:
  - Effektivität zur Lösung einer Aufgabe (insbes. passende Funktionalität)
  - Effizienz der Handhabung des Systems
  - Zufriedenheit der Nutzer einer Software

- *Benutzbarkeit* ist ein wichtiger Teilaspekt davon und bedeutet, dass eine interaktive Software ihre Benutzer_innen gut dabei unterstützt, die gewünschten Arbeitsgänge zu erledigen:
  - Verständlichkeit
  - Erlernbarkeit
  - Bedienbarkeit
    - Bequemlichkeit
    - Geschwindigkeit
    - Fehlervermeidung

These

- These "Prozess":
  Wird Software nicht benutzungsorientiert entwickelt, entstehen oft hochgradig inadäquate technische Systeme.

Gebrauchstauglichkeit und Konstruktionsprozess

- Gute Gebrauchstauglichkeit kann nur erzielt werden, wenn ein geeigneter Software-Entwicklungsprozess verfolgt wird
  - Insbesondere im Hinblick auf Benutzbarkeit; das betrifft vor allem die Bedienschnittstelle (user interface, UI)

- Das Vorgehen dafür ist im Prinzip bekannt,
  wird aber in der Praxis verblüffend selten eingehalten:
  - **Stelle die Endbenutzer_innen ins Zentrum des Entwicklungsprozesses**

- Wir betrachten nun zuerst eine Fallstudie über diesen Effekt
  - "Warum wird so selten ein geeigneter Prozess durchgeführt?"
- und anschließend zwei Fälle, in denen das gründlich schief gegangen ist.

Entwicklungsprinzipien für gute Gebrauchstauglichkeit

1. **Direkter Kontakt zu Benutzer_innen**
2. **Frühes und fortlaufendes Benutzbarkeitstesten**
3. **Iterativer Entwurf**
4. **Integrierter Entwurf**

- Diese Liste stammt von Gould und Lewis
  - J.D. Gould, C.H. Lewis: "*Designing for usability: Key principles and what designers think*",
    Proc. CHI'83 Conf. on Human Factors in Computing Systems, pp.50–83, ACM, 1983, sowie
    Communications of the ACM 28(3):300-311, March 1985
  - Es gibt aber diverse ähnliche andere

- Sehen wir uns kurz die Beschreibungen und Begründungen an:

1. Direkter Kontakt zu Benutzer_innen

Freie Universität Berlin

Beschreibung:

- Entwerfer brauchen von Anfang an direkten Kontakt zu Endbenutzer_innen
  - z.B. mittels Interviews, Beobachtung, Umfragen, direkter Partizipation im Arbeitsprozess

Begründung:

- Nur so können sie deren relevante **Eigenarten verstehen**:
  - Denkgewohnheiten und -fertigkeiten, Verhalten, Einstellungen
- sowie die Eigenschaften der **Aufgaben**, die die Benutzer_innen erledigen wollen

2. Frühes und fortlaufendes Benutzbarkeitstesten

Freie Universität Berlin

Beschreibung:

- Echte Benutzer_innen verwenden die Software
  - oder Prototypen der SW
- Die Entwickler_innen beobachten dies und holen Feedback ein
- Beide entwickeln Verbesserungsvorschläge evtl. gemeinsam

Begründung:

1. Liefert die besten **Einsichten** in Probleme und Lösungen
2. Liefert gute **Motivation**, die Software nochmal umzubauen

- Kein Ansatz zum Entwurf gut tauglicher Software funktioniert zufriedenstellend ohne Benutzbarkeitstests

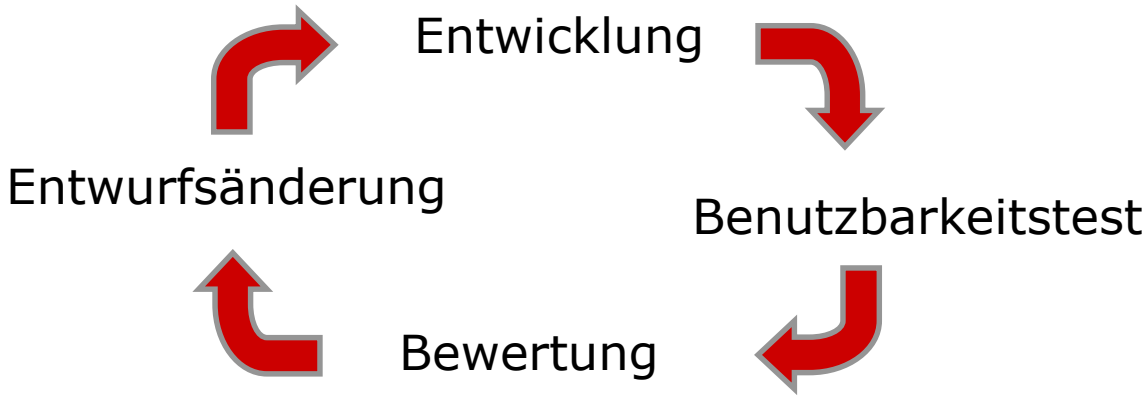
3. Iterativer Entwurf

Beschreibung:

- Die Entwicklung durchläuft viele **Zyklen** von Entwicklung, Benutzbarkeitstest, Bewertung, Entwurfsänderung

Begründung:

1. Die Benutzbarkeitsprobleme kommen nur **nach und nach** zum Vorschein
   - ein Problem *verdeckt* oft viele andere, bis es beseitigt wird
2. Korrekturen können **neue** Probleme hervorrufen
   - und tun dies oft auch



4. Integrierter Entwurf

Freie Universität Berlin

Beschreibung:

- Alle Tauglichkeits-Aspekte müssen zeitlich parallel und inhaltlich gemeinsam (in einer Hand) entwickelt werden
  - 1. Bedienschnittstelle, 2. Lehreinheiten, 3. Hilfe/Dokumentation

Begründung:

1. Andernfalls würden viele Lösungen den Endbenutzer_innen nie wirklich zugute kommen
   - weil sie **erläuterungsbedürftig** sind (➔ Dokumentation)
2. Andernfalls werden viele **Probleme übersehen**
   - z.B. bei der Erlernbarkeit
3. Andernfalls können viele Probleme nicht adäquat gelöst werden
   - z.B. kann ein Tauglichkeitsproblem evtl. nicht in der SW gelöst werden, aber mit einem geeigneten Training abgemildert
   - dafür muss die Erkenntnis aus Benutzbarkeitstests zu den Autor_innen der Trainings übermittelt werden.

Akzeptanz und Verwendung der Prinzipien

- Auf Befragen stufen erfahrene Entwickler_innen diese Prinzipien meist als "offensichtlich" ein
- Sie bilden den Kern diverser etablierter Vorgehensweisen, z.B.
  - Cooperative Design (Skandinavien seit 1970er Jahre)
  - Participatory Design
  - User-Centered Design
    - Diese unterscheiden sich in den konkreten Methoden, verfolgen aber alle das gleiche Ziel "Benutzungsorientierte Entwicklung"
- Tatsächlich werden sie aber nur sehr unvollständig angewandt:
  - Wenig direkte Einbindung von Benutzer_innen
    - schon gar nicht unter Beachtung des kompletten Arbeitsumfelds
  - Zu selten werden genügend (oder überhaupt) Iterationen vorgesehen

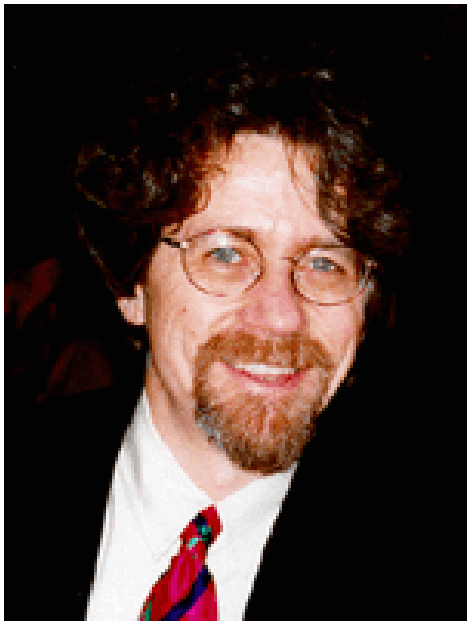
Fallstudie: Poltrock und Grudin

Freie Universität Berlin

- Steven E. Poltrock, Jonathan Grudin:
  "*Organisational obstacles to interface design and development: Two participant observer studies*",
  ACM Transactions on Computer-Human Interaction 1(1):52–80, March 1994

- Kontext und Methode:
  - Große Softwarefirma, Einheit mit 70 Personen
  - Baut neue Version eines wichtigen, international vertriebenen Produkts
  - Forscher wurde für 1 Monat Mitglied einer UI-Gruppe
    - nahm an der Entwicklung teil
    - interviewte 25 Personen innerhalb und außerhalb der Gruppe

User Interface (Bedienschnittstelle, Benutzungsschnittstelle)



Jonathan Grudin

Entwicklungsgeschichte des Softwareprodukts

- Ein CAD-Programm
  - Ursprünglich eine Hochschulentwicklung
  - Seit 5 Jahren erfolgreich am Markt
- Fortentwicklung wurde allmählich immer schwieriger
  - Personalausstattung erhöht
    - einige Verbesserungen erzielt, aber keine langfristige Lösung

Entwicklungsgeschichte: Der Super-Entwerfer

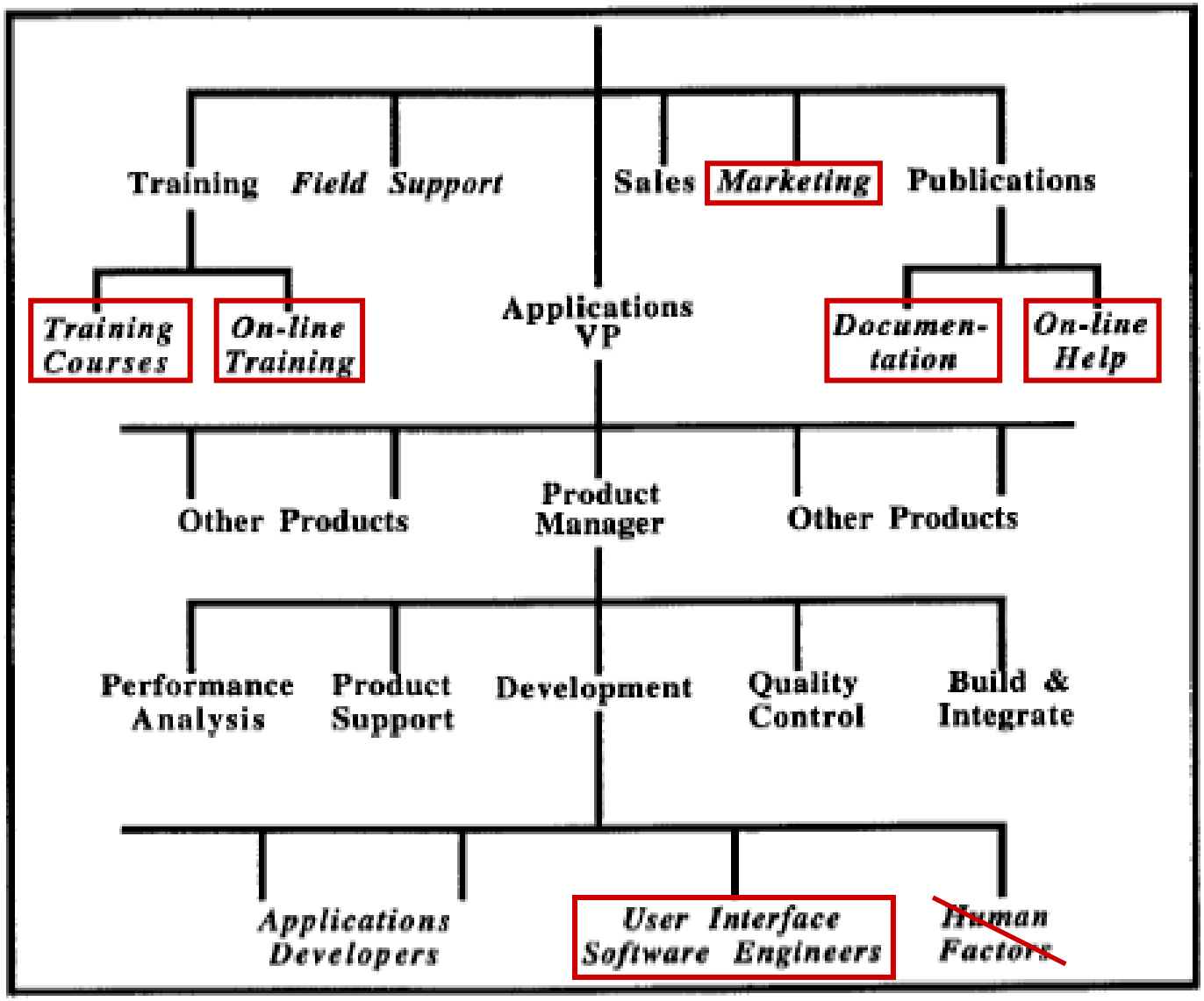
- Dann wurde ein "Superentwerfer" ins Team gebracht:
  - aus der Supportgruppe eines anderen Standorts
  - hervorragende technische Fähigkeiten
  - hatte Erfahrung mit der Anpassung des Produkts für einen großen Kunden; viel Kontakt mit Manager_innen und Endbenutzer_innen
- Er überzeugte einen Vizepräsidenten davon, binnen 1 Jahr eine sehr gute neue Version bauen zu können
  - Hatte eine klare Vision.
  - Begann, sich einfach Mitarbeiter_innen zu suchen und ohne Zustimmung ihrer Manager_innen Aufgaben zu geben.
  - Volle Rückendeckung vom Vizepräsidenten half gegen die Beschwerden.
- Ergebnis: Sehr erfolgreiche neue Version entwickelt

Entwicklungsgeschichte: Fortgang

- Dieses Verfahren wurde jedoch nicht aufrecht erhalten
- Zustand bei Beginn der Studie:
  - Der Vizepräsident war weg
  - Der Superentwerfer ging gerade weg
  - Neueingestellte Manager hatten Fokus auf Standard-Entwicklungspraktiken mit maximaler Überwachung des Entwurfs und viel Koordination
    - bewerteten Zuverlässigkeits- und Planungsziele höher als Innovation
- Neues Team zur Entwicklung der Benutzungsschnittstelle
  - Keine Produktbenutzer_innen unter den Entwickler_innen
  - Auch keine Erfahrung in der Zusammenarbeit mit Kund_innen
  - Keine Möglichkeit zur Einbeziehung von Endbenutzer_innen
  - Keine Benutzbarkeitstests waren geplant

Organisationsstruktur

- ungefähr so vorgefunden
  - mit gewissen Abweichungen
- Prinzip 4 (integrierte Entwicklung) ist direkt verletzt!

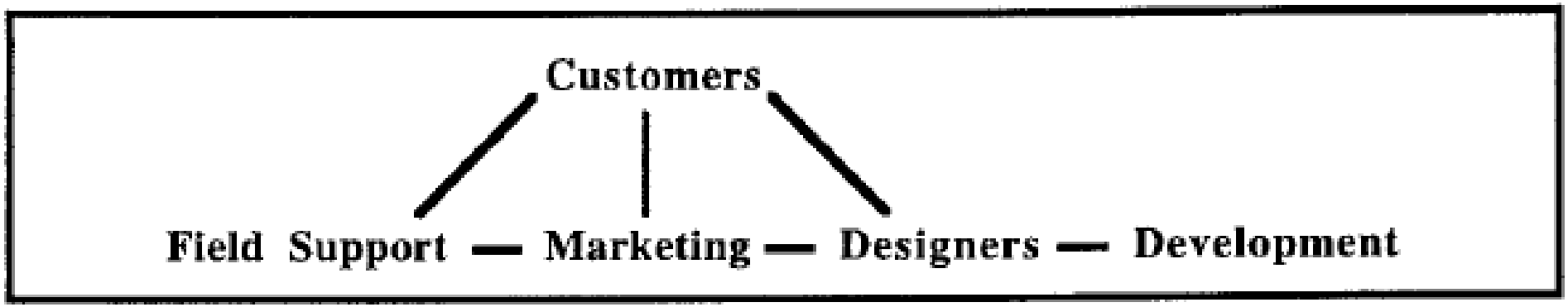


Theorie der UI-Entwicklung dieser Organisation

Freie Universität Berlin

Schlechte Idee!

- Marketingabteilung arbeitet mit Vertriebsabteilung ("sales") und Kunden, um die Marktanforderungen zu verstehen.
  - Außerdem mit Kundendienst ("field support")
- Entwurfsabteilung ("designers") empfängt diese und arbeitet mit Kunden, um die technischen Details der Anforderungen zu verstehen
- Entwicklung ("development") bekommt eine komplette, aber abstrakte Funktionsspezifikation von Design
  - entwickelt konkrete UI-Spezifikation (dann: Begutachtung, Bau)
  - muss (und kann) nicht selbst mit Endbenutzer_innen arbeiten



Praxis der UI-Entwicklung

- *Wahrnehmung der Insassen: "Wir sind uns hier einig, dass unsere Marketinggruppe so gut wie wertlos ist."*
- Aussage eines Endbenutzers
  - ein Entwickler traf ihn auf einer Messe
  - *"Wie ihr über Produktfeatures entscheidet ist wirklich eine Schande:*
    Ich kann nur mit Marketing reden.
    Marketing versteht nicht, was ich will.
    Design versteht nicht, was Marketing aufgeschrieben hat.
    Dann kriegt es Entwicklung, weiß nicht, was es bedeuten soll,
    weiß nicht, wo es herkommt,
    und kann mich nicht kontaktieren, um zu fragen."

- Die Entwickler_innen versuchen das Problem zu umgehen:
  - Sie reden mit dem Kundendienst (Entwurfsdurchsichten etc.)
  - Leider wurde der dann in die Marketing-Abteilung verlagert und die sperrte diese Interaktionen

Problem 1: Rollenkonflikt für Marketing

Freie Universität Berlin

- Die Aufgabe der Marketingabteilung ist es, zu definieren, was gebraucht wird, um das Produkt zu verkaufen
  - Dazu reden Sie *unter anderem* mit Kunden
  - Aber meist mit den Entscheider_innen, nicht mit den Benutzer_innen
- Das ist jedoch weitaus zu grob, um eine im Detail nützliche Systemgestaltung vorzugeben

Problem 2: Kein Kontakt zwischen Entwicklung und Benutzer_innen

Freie Universität Berlin

- Marketingabteilung über Entwicklungsabteilung:
  - *"Die Entwicklung kann keine benutzerfreundliche Software bauen, denn die wissen gar nicht, was ein Benutzer ist."*
- Entwicklungsabteilung über Kontakte zu Kundendienst:
  - *"Gerade vorhin habe ich mit einem über ein Feature gesprochen; was er denkt, wie das aussehen sollte. '*
    Jetzt weiß ich, dass ich alles wegwerfen und von vorn anfangen muss."
- Lösungsversuche der Entwicklungsabteilung:
  - Unterhielt informelle Kontakte zu Kunden
    - aber meist wiederum nur zu Entscheider_innen

Folgen für die Anwendung der 4 Prinzipien

Freie Universität Berlin

1. Direkter Kontakt zu Benutzer_innen
   - Fehlt. Das ist das Ausgangsproblem
2. Frühes und fortlaufendes Benutzbarkeitstesten
   - Ist folglich allenfalls mit unechten Benutzer_innen möglich
3. Iterativer Entwurf
   - Ist nur mäßig nützlich, da man kein qualifiziertes Feedback hat
4. Integrierter Entwurf
   - Wird von der Organisationsstruktur extrem erschwert

- Diskussion siehe nächste Folien

Frühes und fortlaufendes Benutzbarkeitstesten

Freie Universität Berlin

- Entwicklungsabteilung präsentierte ihre Benutzungsschnittstelle vor Vertreter_innen der anderen Abteilungen
  - *"Wenn man ein Architekturmodell vorstellt,*
    hat kaum jemand dazu was zu sagen.
    Aber wenn man Bildschirmentwürfe zeigt,
    bekommt man so viele Meinungen wie Leute im Raum sind."

- Am nützlichsten war noch, als zwei Mitarbeiter_innen des Kundendienstes das Produkt so ausprobierten, wie sie glaubten, dass Endbenutzer_innen es verwenden würden
    - (Sie fanden dabei außerdem mehr Defekte als die Qualitätssicherung es tat!)
  - Dies war aber <u>kein</u> Bestandteil des regulären Entwicklungsprozesses

Integrierter Entwurf

- Diese Organisation hatte eine lange Tradition der getrennten Entwicklung von HW, SW, Dokumentation und Training

- Die ganze Kommunikation sollte eigentlich über Spezifikationen stattfinden
  - Das ist für Benutzungsschnittstellen jedoch sehr schwierig:
    "Man kann nicht in Worten ausdrücken, was man vorhat, wenn man ein bestimmtes UI entwirft.
    Man kann zwar eine Spezifikation davon aufschreiben, aber keiner weiß, ob die richtig ist, bevor man sie benutzt hat."

- Diese Organisationsform erzeugt auch Widerstände gegen Iterationen
  - oder führt dazu, dass Dokumentation und Lehrmaterial erst sehr spät erstellt werden

Lösungsvorschläge eines Entwicklers

Freie Universität Berlin

- Rolle von Marketing:
  - *"Marketing sollte eine Richtung vorgeben;*
    Gebiete, die zu verbessern sind.
    Und dann ein paar Kunden nennen, mit denen wir
    direkt zusammen arbeiten können."

- Art und Rolle von Benutzerkontakt:
  - *"Man muss zuerst mal verstehen, was Benutzer überhaupt erreichen wollen. Wie sie es angehen, ihren Tag strukturieren.*
    Mit einem solchen Überblick bekommt man auch Ideen, wie man ihnen am besten helfen kann."

Ende der Fallstudie.

Ratschläge an Softwareorganisationen

Freie Universität Berlin

1. Trage den Mitarbeitern möglichst nur Dinge auf, die sie gut können
   - Viele Tätigkeiten können nur von einem *Team* mit verschiedenen Rollen bewältigt werden
   - Sie einer Person zuzuweisen führt zu schlechten Ergebnissen und zu Reibungsverlusten
2. Sorge für kurze Kommunikationspfade
3. Sorge für eine zentrale Entscheidungsinstanz
4. Hindere die Mitarbeiter_innen nicht, das zu tun, was nötig ist!

Folgen der Nichtbeachtung

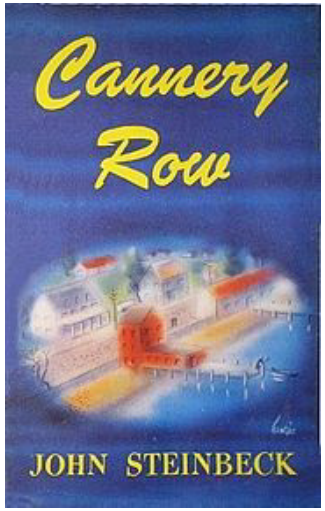
- Verletzung einer benutzerzentrierten Vorgehensweise kann frappierend einschneidende Folgen für die Gebrauchstauglichkeit haben

- Es folgen zwei Beispiele dafür:
  - Bibliothekssoftware-Entwurf ohne Bibliothekare
  - Krankenhaussoftware-Entwurf ohne Medizinpersonal

Katastrophe 1: Bibliothekssoftware

Freie Universität Berlin

- Bibliotheks-SW der East Lake County Library (Florida)



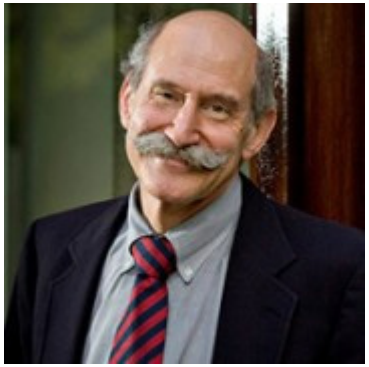
- Sondert Bücher aus, die 2 Jahre nicht ausgeliehen wurden.
- Die Bibliothekare wissen, dass das oft nicht sinnvoll ist
  - und solche Bücher z.T. später neu beschafft werden.
- Sie erfanden den Leser "Chuck Finley", der binnen 9 Monaten 2361 Bücher "auslieh", um sie zu bewahren.
  - Er bekam eine Adresse und eine Führerscheinnummer
- ➔ Mehraufwand, verfälschte Statistiken
- Andere Bibliotheken machen das wohl auch,
  - obwohl sie zum Teil nach Zahl der Ausleihen budgetiert werden.

Ende von Beispiel 1.

Quelle: Orlando Sentinel, 2016-12-30

Katastrophe 2: Krankenhaussoftware

- Ross Koppel et al.: "Workarounds to computer access in healthcare organizations: you want my password or a dead patient?", Information Technology and Communications in Health 2015
  - Basis: Interviews und Feldbeobachtungen mit Medizinpersonal in zahlreichen Krankenhäusern



- Haupterkenntnisse:
  - Die Zugriffsschutz-Mechanismen in Medizinsoftware sind oft so praxisuntauglich, dass sie massenhaft umgangen werden.
  - Auch andere Arten von Fehlentwürfen beeinträchtigen die Krankenhausarbeit massiv.

Krankenhaussoftware-Fehlschläge: Login

- Zu viele verschiedene Systeme und Passwörter
  ➔ Alle benutzen das gleiche Passwort

- Häufige erzwungene Passwortänderungen
  ➔ Passwörter auf gelben Zetteln

- Extremer Auswuchs:

"endemic circumvention of password-based authentication"

Krankenhaussoftware-Fehlschläge: Logout

Freie Universität Berlin

- Logout ist wichtig, um
  - unberechtigten Zugriff zu verhindern, aber vor allem um
  - versehentliche Eintragungen in einen falschen/fremden Datensatz zu vermeiden
- Logout-Automatik mit Nähesensor
  ➔ ausgetrickst mit Kaffeetasse auf dem Sensor
- Logout-Automatik per kurzem Timeout
  ➔ dienstjüngstes Teammitglied muss alle 10s Leertaste drücken
- Oder: Man kann sich nicht anmelden, wenn man noch irgendwo anders angemeldet ist
  - das System sagte aber nicht, wo die alte Anmeldung besteht

Eine Intensivstation

Krankenhaussoftware-Fehlschläge: Hinderliche Konsistenzprüfungen

Freie Universität Berlin

- OP-Vorbereitung muss minutengenau Zeit eintragen für Erreichen des OPs.
  - OP-Raum ist aber zwei Minuten entfernt und
  - System erlaubt keine Zeiten in der Zukunft.
  - Also: Eingeloggt bleiben, Bildschirm aus, Patient zum OP bringen, zurückkommen, Zeiteintrag machen

- Ein System für Blutverdünnungs-Prophylaxe zwingt den Arzt, einen Blutverdünner zu verordnen.
  - Auch, wenn der Patient schon welchen bekommt.
  - Arzt muss sich abmelden, neu anmelden, zweiten Blutverdünner wieder abbestellen.
  - Vergessen ist für den Patienten lebensgefährlich

Krankenhaussoftware-Fehlschläge: Falscher Dokumentationsgrad

Freie Universität Berlin

- Systeme zu umständlich;
  Systeme beschränken möglichen Detailgrad;
  Systeme erlauben keine inoffiziellen Einträge
  ➔ Personal führt zusätzliche Schattenakten auf Papier

- Hoher Dokumentationsgrad wird erzwungen
  auch wo medizinisch nicht nötig
  ➔ Mündliche Kommunikation (effizienter und detaillierter) nimmt ab

Krankenhaussoftware-Fehlschläge: Fazit

- Unklar, wie man das Medizinpersonal optimal unterstützt.
- Viele absichtliche technische Zwänge und Beschränkungen schaden aber mehr als sie nützen
  - vor allem Sicherheitsmaßnahmen nach dem Enthaltsamkeitsprinzip "Du darfst nicht…" funktionieren miserabel.
    - (Erfolgreiche Sexualerziehung bringt den Jugendlichen bei, wie sie gesund und kinderlos bleiben, *obwohl* sie Sex haben.
      Ähnliches braucht auch die Krankenhaus-Computersicherheit.)
    - Quelle 1, Quelle 2
- Die Probleme sind inhärent schwierig.
  Ohne benutzungsorientierte Entwicklung kommen deshalb oft desaströse Informatiksysteme heraus.
  - (Mit ihr bekommt man auch nicht alles perfekt hin,
    aber zumindest kann man die Schwächen stark reduzieren.)

These

- These "Prozess":
  Wird Software nicht benutzungsorientiert entwickelt, entstehen oft hochgradig inadäquate technische Systeme

Also bitte:
Die Prinzipien einhalten

Freie Universität Berlin

1. **Direkter Kontakt zu Benutzer_innen**
   - Dies ist nicht zu ersetzen und grundlegend für alles andere
2. **Frühes und fortlaufendes Benutzbarkeitstesten**
   - Liefert den Antrieb für Verbesserungen
3. **Iterativer Entwurf**
   - Anerkennen, dass man nicht alles auf Anhieb richtig machen wird und auch die zweite Idee bisweilen noch falsch ist
4. **Integrierter Entwurf**

- Doch wie soll man sich die Ergebnisse von Benutzbarkeitstests vorstellen?

Ergebnisse von Benutzbarkeitsprüfungen

Freie Universität Berlin

Fallen überwiegend in vier Kategorien:

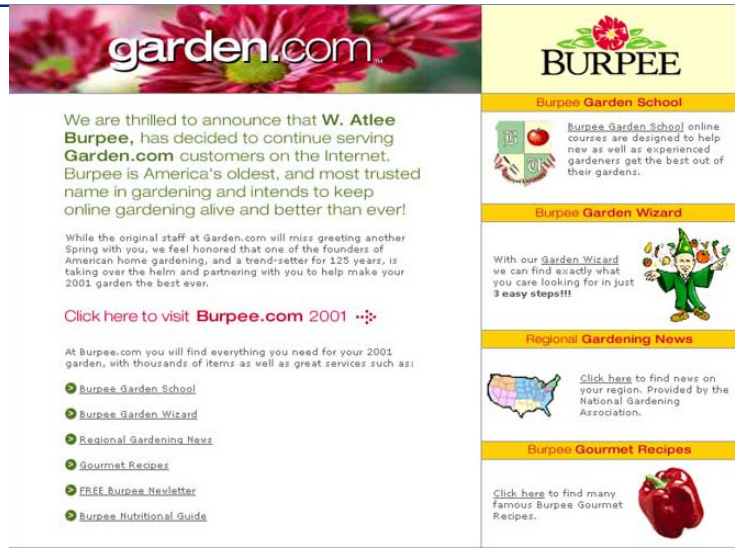
- **Defekte**:
  - Eine Funktion tut eindeutig etwas anderes als vorgesehen
- **Funktionslücken/Untauglichkeit** :
  - Eine eindeutig benötigte Funktionalität ist gar nicht vorhanden
  - oder passt nicht zum Zweck [kommt im nachfolgenden Beispiel aber nicht vor]
- **Verständnisprobleme**:
  - Lernproblem: Das Konzept einer Funktion oder Bedienweise ist schwierig zu erfassen
  - Benutzungsproblem: Eine Darstellung ist auch für geübte Benutzer oft mehrdeutig und provoziert Fehler; eine Eingabeweise provoziert Fehler
- **Bedienschwierigkeiten**:
  - Eine Funktion ist zu umständlich zu benutzen
  - Eine Bedienweise provoziert Fehler

(Die Zuordnung kann mehrdeutig sein)

Ergebnisse: Beispiel

Benutzbarkeitsprüfung eines
Web-Ladens für Gartenartikel

http://ucs.ist.psu.edu ,
Case study "Garden-com"



Bereich: Produktkatalog

- Defekte, Funktionslücken: --
- Verständnisprobleme (Erlernen):
  - Mehrdeutigkeit zwischen "Community" und "Meine Sachen"
  - Benutzer verwenden nicht Link "Varianten" wenn ein angezeigtes Produkt ihnen nicht direkt passt
- Verständnisprobleme (Benutzen):
  - Man sucht oft im falschen Teil des Produktbaums
  - Kategorienamen im Baum sind anders als dann angezeigt
  - Sinnarme Kategorienamen (z.B. "Geschenke")
- Bedienschwierigkeiten: --

Beispiel-Ergebnisse: Bezahlen

Bereich: Bezahl-Funktion

- Defekte: --
- Funktionslücken:
  - Wenn ein Benutzer versehentlich seine Bestellung löscht, gibt es keinen Weg, sie zurück zu bekommen
- Verständnisprobleme (Erlernen):
  - Benutzer klicken oft ohne Wirkung auf Kreditkarten-Symbol
- Verständnisprobleme (Benutzen): --
- Bedienschwierigkeiten: --

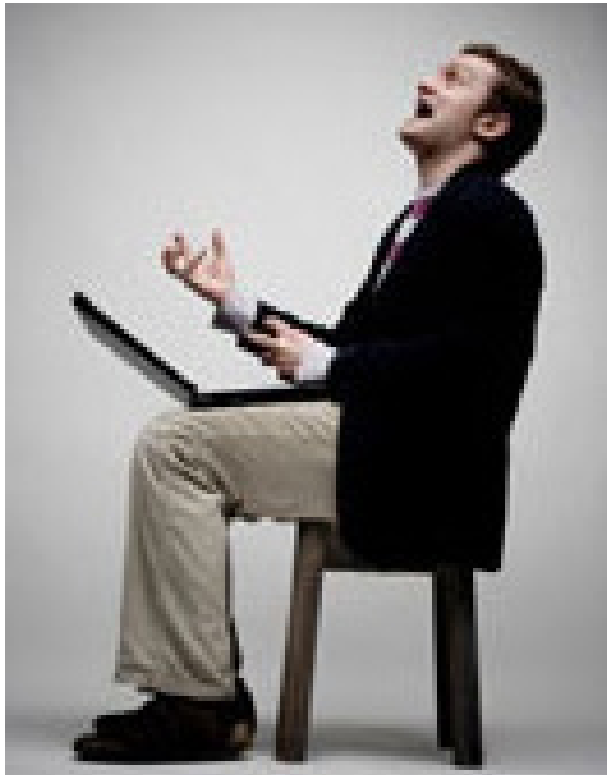
weitere Beispiele siehe Zusatzfolien am Ende

Zusammenfassung

- Software mit guter Gebrauchstauglichkeit zu bauen verlangt
  - direkten Kontakt zu Benutzer_innen sowie
  - iterative Entwicklung mit fortlaufenden Benutzbarkeitsprüfungen

- Leider ist das in konkreten SW-Organisationen oft schwer zu gewährleisten
  - das führt in der Praxis zu grauenhaft untauglichen technischen Systemen

- Die Ergebnisse von Benutzbarkeitsprüfungen fallen in diverse Kategorien und geben gute Hinweise für Verbesserungen



Benutzungsorientierte Entwicklung: Wie macht man das konkret?

Freie Universität Berlin

V+Ü Human-Computer Interaction I

Prof. Claudia Müller-Birn

Ziele:

- Wissen, was ein positives Benutzungserlebnis ausmacht.
- Wissen wie man Software mit guter Usability entwickelt.
- Usability bestehender Software bewerten können.

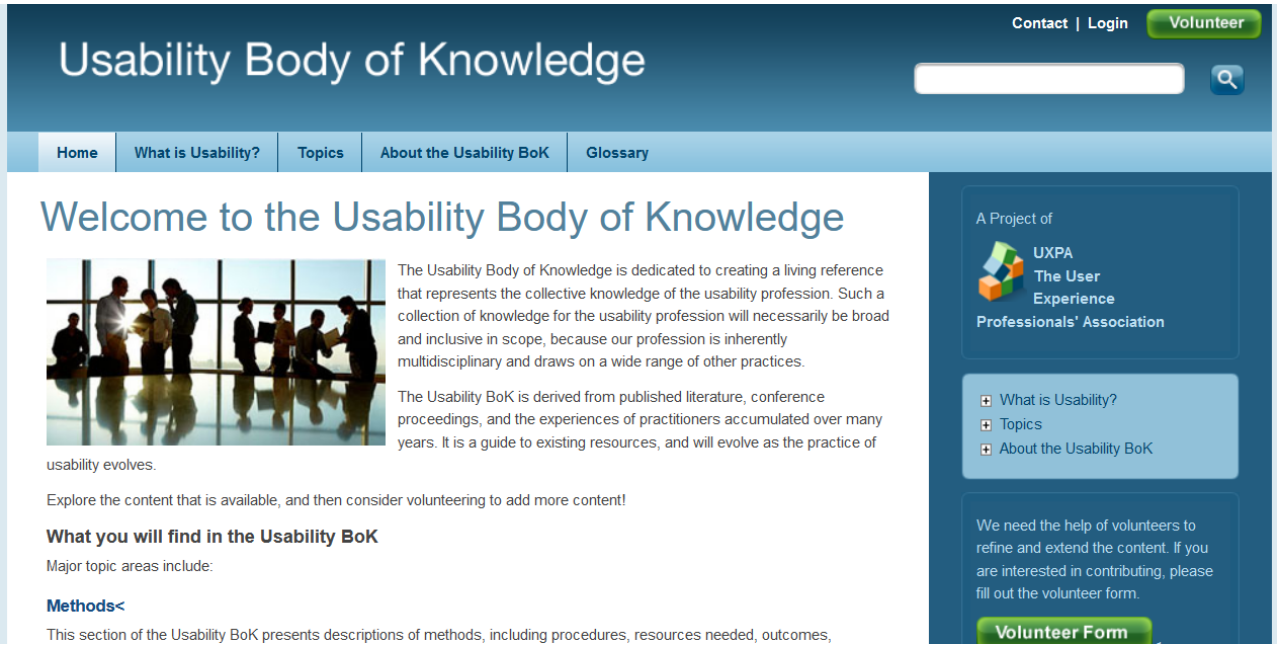
Themen

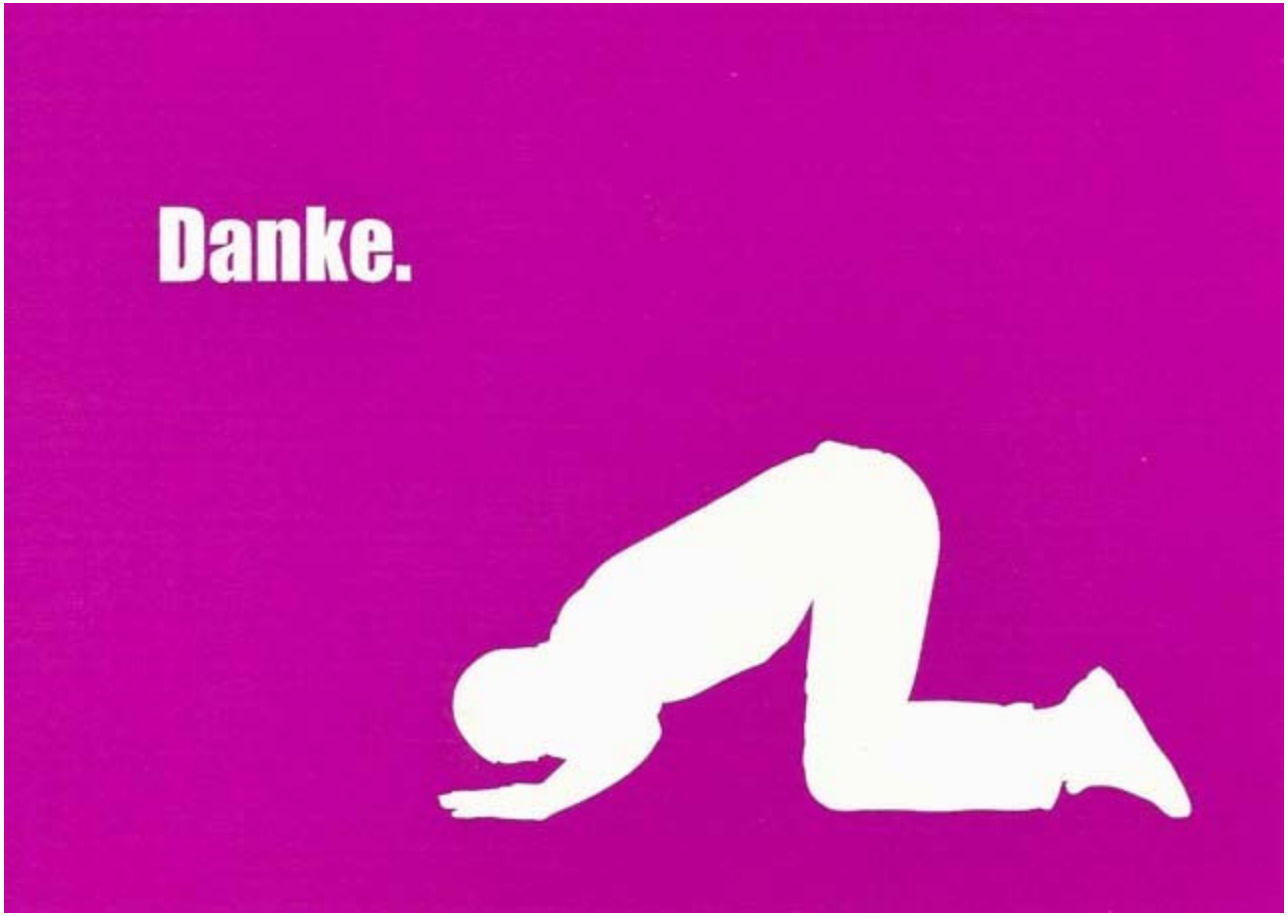
- Den **Benutzer** und dessen Ziele **verstehen:** User Research
- Den **Faktor Mensch** berücksichtigen: Wahrnehmung und Kognition und deren **Relevanz für Interface Design und Interaction Design**
- **Konkrete Anwendungsgestaltung**: Interface Elemente, Interface Design Patterns und deren Einsatzbereiche
- Usability-Prüfung mit Benutzer_innen: **Usability Tests**
- **Usability-Prüfung durch "Experten"**: z.B. Usability Heuristiken

Shneiderman et al. "Designing the user interface: strategies for effective human-computer interaction", Pearson, 2016.

Literatur

- Usability Body of Knowledge
  - http://www.usabilitybok.org
  - Gute und übersichtliche Erläuterung der wichtigen Konzepte:
    - Erhebungs- und Evaluationsmethoden
    - Entwurfsprinzipien
    - Organisations- und Fortbildungsfragen
    - verwandte Wissensgebiete, Institutionen
  - Enthält weitere Literaturverweise




Danke.

Zusatzfolien zu "Ergebnisse von Benutzbarkeitstests"

Beispiel-Ergebnisse: Suchfunktion

Bereich: Suchfunktion

- Defekte: --
- Funktionslücken: --
- Verständnisprobleme (Erlernen):
  - Wenn ein Suchfilter angewendet wurde, wird das nicht direkt angezeigt
- Verständnisprobleme (Benutzen):
  - Es ist unklar, was die Suchfunktion alles durchsucht
  - Suche verlangt oft einen ganz bestimmten Suchbegriff; zu unflexibel
- Bedienschwierigkeiten:
  - Benutzer geben oft falsch geschriebene Suchbegriffe ein (und merken dann evtl. nicht, dass das der Grund für leere Suchergebnisse ist)

Beispiel-Ergebnisse: Einkaufskorb

Bereich: Einkaufskorb-Funktion ("Schubkarre")

- Defekte:
  - Anfänglich ist als Menge für einen Artikel immer "0" gewählt
  - Benutzer können Artikeleigenschaften (z.B. Größe) auf unmögliche Werte ändern
- Funktionslücken: --
- Verständnisprobleme (Erlernen):
  - Benutzer finden die Einkaufskorb-Funktion oft nicht
  - Benutzer finden die Funktion "Entferne Artikel" nicht und löschen statt dessen die ganze Bestellung
- Verständnisprobleme (Benutzen):
  - Die Auswahl eines Stoffes wird immer mit "Kissen-Bezugsstoff" abgefragt, auch wenn der Artikel gar keine Kissen hat
- Bedienschwierigkeiten: --

Beispiel-Ergebnisse: Hilfesystem

Freie Universität Berlin

Bereich: Online-Hilfesystem

- Defekte: --
- Funktionslücken:
  - Es gibt kein Hilfesystem, obwohl Benutzer eines erwarten und es auch brauchen.
- Verständnisprobleme (Erlernen): --
- Verständnisprobleme (Benutzen): --
- Bedienschwierigkeiten: --

Beispiel-Ergebnisse: Sonstiges

Freie Universität Berlin

Bereich: Sonstiges

- Defekte: --
- Funktionslücken: --
- Verständnisprobleme (Erlernen): --
- Verständnisprobleme (Benutzen):
  - Der Farbunterschied zwischen neuen und bereits benutzten Links ist zu gering
- Bedienschwierigkeiten:
  - Es ist umständlich und schwierig, einen Artikel zu finden/zu bestellen, der außerhalb des Einkaufsbereichs erwähnt wird