Usability Engineering

Patterns of User Behaviour

Nachfolgende Erklärungen/Beispiele stammen aus dem Buch „Designing Interfaces“ von Jenifer Tidwell

- Safe Exploration: Ausprobieren ohne Dinge zu ruinieren
    - A photographer tries out a few image filters in an image-processing application. He then decides he doesn't like the results and hits "Undo" a few times to get back to where he was. Then he tries another filter, and anothereach time being able to back out of what he did. (The pattern named Multi-Level Undo, in Chapter 5, describes how this works.)
    - A new visitor to a company's home page clicks various links just to see what's there, trusting that the Back button will always get her back to the main page. No extra windows or popups open, and the Back button keeps working predictably. You can imagine that if a web application does something different in response to the Back buttonor if an application offers a button that seems like a Back button, but doesn't behave quite like itthen confusion might ensue. The user can get disoriented while navigating, and may abandon the application altogether.
    - A cell phone user wants to try out some intriguing new online functionality, like getting sports scores for the World Series in real time. But he's hesitant to try it because the last time he used an online service, he was charged an exorbitant amount of money just for experimenting with it for a few minutes.
- Instant Gratification: Ich möchte das jetzt machen, nicht später
    - Der User sollte möglichst schnell Erfolgserlebnisse haben, damit er Vertrauen in die Applikation gewinnt und motivierter ist für die weitere Benutzung
- Satisfaction: Das ist jetzt ok für mich, ich möchte keine weitere Zeit dafür aufwenden.
    - User suchen anfänglich schnell Dinge die gleich funktionieren könnten und probieren es aus ohne alle Alternativen abzuwiegen -> hängt zusammen mit Safe exploration und instant gratification
- Changes in Midstream: Ich habe meine Meinung geändert, ich möchte etwas anderes machen
    - Beispiel: Ein Mensch geht in einem Raum um dort nach etwas zu suchen, zB Schlüsseln, findet aber eine Zeitung und beginnt diese zu lesen. Solche Zieländerungen sollen auch im Programm möglich sein (Global Navigation, connection to other pages,…)
- Spatial Memory: Der Button war doch gerade noch hier
    - Personen sich eher daran wo Dinge waren als wie sie geheißen haben -> zB Organisiertes Chaos auf Schreibtischen,…
- Deferred Choises: Ich möchte mir darüber zur Zeit nicht den Kopf zerbrechen
    - Zum Beispiel möchte ein nicht registrierter Benutzer in einem Forum etwas posten und muss sich registrieren. In erster Linie möchte er seinen Beitrag schreiben und nicht lange für die Registrierung brauchen -> unnötige Dinge wie Avatarbild, Personenbeschreibung,… können auch später ausgefüllt werden. Manchmal sind

diese Informationen noch gar nicht vorhanden (auf welchem Server soll ein neues Dreamweaver Projekt gehostet werden)

- Incremental Construction: möchte das gerne Ändern. Jetzt möchte ich jetzt nochmal ändern, ja jetzt sieht es gut aus.
    - So gut wie keiner Arbeit wird am Anfang begonnen und in einem bis zum Ende gemacht -> oft werden Dinge überarbeitet oder verbessert, teilweise sogar neu gemacht.
- Habituation: Das funktioniert über anders auch so, warum hier nicht
    - Ctrl + A, Ctrl +C, Ctrl+V,…
- Prospective Memory: Ich werde das hier ablegen, damit ich mich erinnere das nachher zu machen
- Streamlined Repetition: Wie oft muss ich das noch machen
    - Search & Replace (viele Ersetzungen aber nur jeweils ein Klick auf „Replace“ oder Batchverarbeitung oder Makros)
- Keyboard Only: ich möchte die Maus nicht verwenden
- Other’s people advice: Was denken andere Leute darüber

Diagnose: Ungenügend Usability

- Nutzer arbeiten nicht so schnell wie erhofft mit dem System
- Einarbeitung benötigt viel Zeit und Aufwand
- Qualität der Arbeit sinkt
- Nutzer minimieren die Arbeit am System und es werden Workarounds gebildet (Arbeitsschritte werden auf andere Art und Weise gelöst)
- Prozessvorgaben werden weggelassen und Sicherheitsmaßnahmen ignoriert
- Benutzerfehler führen zu Schäden

Human-Computer Interaction

- Learnability: Wie schnell gelingt es dem Benutzer einfache Tasks auszuführen, wenn er das Design noch nicht kennt.
- Efficiency: Wie schnell kann der Benutzer Aufgaben ausführen, wenn er das Design verstanden hat
- Memorability: Wie gut erinnert sich der Benutzer an Aufgaben, wenn er das System schon lange nicht mehr benutzt hat
- Errors: Wie viele Fehler macht der User bei der Verwendung des Systems. Wie schlimm sind diese Fehler und wie leicht kann er die Fehler wieder rückgängig machen bzw. kommt er wieder zurück zum Normalzustand
- Satisfaction: Wie angenehm/zufriedenstellend ist das Design für die Verwendung

UI-Patterns (Benutzerschnittstellen)

Organising the Content

- Two Panel Selector: Thunderburd, oben Auswahl unten Details
- One Window-Drilldown: Ipod → „Unterprogramme“
- Alternative Views: Darstellung mit und ohne Grafiken, Explorer Detailansicht vs. Große Symbole

- Wizards
- Extras on Demand: Colorchooser ->Define Customs-→ RGB Werte

Getting around (Navigation, Sign-posts, wayfinding)

- Clear Entry Points
- Global Navigation: Ein Teil der UI bleibt immer gleich und bringt den Benutzer wieder zum Beginn der wichtigsten Sektionen
- Hub and Spoke: siehe Handy. Vom Menü in Unterpunkte und wieder zurück, keine Querverweise
- Pyramid: Hauptmenü in Unterpunkte, Querverweise zu anderen Unterpunkten (zB Tutorial mit Next/Prev und Inhaltsverzeichnis)
- Modal Panel: zB.: Speichern bevor beenden: Ja, Nein, Abbrechen
- Sequence Map: Fortschritt der Vorgangs, was hat man schon erledigt, wo ist man, was kommt noch (Amazon Bestellung)
- Breadcrumbs: Home -> Products -> …
- Color Coded Sections: Hintergrundfarbe ändert sich pro Sektion
- Escape Hatch: Möglichkeit zum Abbruch, wenn nur wenige Navigationsoptionen vorhanden sind, (Ja, Nein und Abbruch)

Organising the Page (Layout of the page elements)

- Visual Framework: Einheitliches Grundlayout (Farben, Formen)
- Center Stage: Der wichtigste UI-Teil in die Mitte
- Titled Sections: Definierte Sektionen im Inhalt mit aussagekräftigem Titel (Products, Support, Shop,…)
- Card Stack: (Tabs, Reiter,…) Gruppierung von vielen Inhalten
- Right/Left Alignment: Zweispaltige Tabellen: linke mit rechter Ausrichtung, Rechte mit linker
- Responsive Disclosure: minimales UI, nach jedem Schritt wird ein bisschen mehr angezeigt (Formular: Nächste Frage von Eingabe der vorigen abhängig)
- Liquid Layout: Bei Größenänderung der Seite soll die Seite immer „gefüllt“ mit Inhalten sein und alles sichtbar

Doing things (Action and Commands)

- Button Groups: Verwandte Funktionen zusammenfassen (Add, Edit, Remove oder OK, Cancel)
- Action Panel: Anstatt nur Funktionen in Menüs, Liste mit möglichen Funktionen
- Prominent „Done“ Button
- Smart Menu Item: Anpassen der Menüs um zu zeigen was sie machen wenn sie ausgeführt werden
- Preview
- Progress Indicator
- Cancelability: Möglichkeit zeitlich-lange Aktionen abzubrechen (Browser -> rotes X)
- Mulit-Level Undo: (Photoshop)
- Command History (Matlab)

Showing Complex Data (Trees, tables …)

- Overview plus Details: Übersicht in Photoshop
- Datatips: wie Tooltips nur genauere Infos
- Row Stripping: Abwechselnde Farben in Tabellen -> ITunes
- Sortable Table
- Jump to Item: durch tippen des Beginn eines Titels kommt man zum ersten Element das der Zeichenfolge entspricht
- Cascading Lists: Hierarchie in Tabellen (Spalten)
- Tree Table

Getting input from user (Forms and Controls)

- Forgiving Format: User kann Daten in verschiedenen Formaten eingeben
- Structured Format: zB Serial Number in mehrere unterteilte Textfelder eingeben
- Fill-in-the-blanks: Eingabe der Daten in Satz in Prosaform eingeben
- Input Hints: Beispiele für Art der einzugebenden Daten bereitstellen
- Input Promt: Eingabefeld mit Hinweisen befüllen
- Autocompletion
- Good Defauts
- Same Page Error Messages

Builders and Editors

- Edit in Place: Direkt an der angezeigten Stelle Änderungen durchführen (Dateiname direkt im Explorer)
- Smart Selection: Markierung von Texten durch Doppelklick
- Magnetism: Andocken von einzelnen Fenstern in Programmen
- Guides: Hilfslinien für die Orientierung/Ausrichtung

Making it look good (Visual styles and aesthetics)

- Deep Background
- Few Hues, Many Values: Max 2-3 prägnante Farben, aber viele dezente Abstufungen (Brightness)
- Coner Treatments: Einheitliche Ecken, nicht nur normale Ecken
- Borders that echo fonts: Rändern und Rahmen sollen der Schriftarten entsprechen
- Hairlines: Zur Unterteilung, optischen Auflockerung
- Contrasting Font Weights: Unterscheidungs zwischen Überschriften / Texten: eine dicke und dunkle andere dünn leicht
- Skins

Aufgabenbereiche von UE

Analyse

Es werden die Geschäftsmodelle modelliert und analysiert in denen das neue Produkt eingebettet werden soll. Dabei können sich Prozesse ändern.

Verständnis der Benutzer und des Anwendungszweckes relevant. Das soll durch Interviews und Beobachtung der User vor Ort analysiert, dokumentiert und ausgewertet werden.

Zum Beispiel durch

- Contextual Inquiry
- Beobachtungen
- Interviews
- Moderierte Gesprächsgruppen
- Strukturierte Aufgabenanalysen

Modellierung

Das System soll für möglichst viele Benutzer die damit arbeiten optimal design sein -> Modellierung aus verschiedenen Sichten. Aus Analyse sollen Einbettung des Systems in Geschäftsprozesse, die Arbeitsweise der Benutzer und die Funktionalität sowie das Verhalten modelliert werden. Erste Entwürfe sehr früh und in regelmäßigen Zyklen mit Feedback von den Benutzern. Erkenntnisse in Personas (Benutzerprofile) und Szenarien (Arbeit mit dem neuen System aus Benutzersicht) umgesetzt.

Personas und Szenarien als Basis für Use-Case. Storyboards sollen Lösungsideen veranschaulichen um Feedback von den Benutzern zu erhalten → detaillierte Benutzersicht auf Schnittstellen mit dem System. Daraus dann erste User Interface Prototypen mit möglichst einfachen Mitteln.

Spezifikation

Inhalt für Vertragserfüllung und Erwartungen an das System von beiden Seiten. Use Case Spezifikation, Use Case Modelle, Ablaufdiagramm beschreiben funktionale Anforderungen, Rahmenbedingungen werden aufgenommen ebenso wie nicht-funktionale Anforderungen. Szenarien, Storyboards und UI Prototypen optional.

Realisierung

Software Architektur wird entworfen und Spezifikation wird in ein technisches Design umgewandelt. Realisierung wird von Usability Guidelines und Styleguides begleitet -> konsistentes „regelkonformes“ UI

Evaluierung

Erstellte Resultate werden durch Benutzer überprüft, getestet und danach optimiert (Prototypen / realisiertes System).

Methoden

- Usability Tests: Benutzer werden bei der Benutzung der neuen Anwendung beobachtet, Fehler werden dokumentiert -> daraus werden Verbesserungen erarbeitet
- Usability Walkthrough: Für Evaluierung früher Prototypen
- Usability Fragebögen: Bei großer Anzahl an Benutzern als Feedback. Benutzerschnittstellen durch Checklisten und Experten überprüfen lassen (Experten Review).

Ziele von UE

- Arbeitsgeschwindigkeit maximieren
- Anzahl benötigter Schritte einer bestimmten Funktion und benötigte Zeit minimieren
- Ausbildungs- und Lernaufwand minimieren
- Qualität des Abreitsresultats verbessern
- Anzahl der Fehler der Benutzer minimieren

- Gefährdungspotential durch falsche Interpretationen bzw. Manipulationen reduzieren
- Zufriedenheit der Benutzer mit ihrer Arbeit und dem Produkt verbessern
- Akzeptanz erhöhen

Rahmenbedingungen

- Verfügbarkeit von Benutzern für Beobachtungen/Interviews (Benutzer sind auch wiederum Mitarbeiter und haben Arbeit zu erledigen → Zeit für Interview sind Kosten ohne Resultat für das Kundenunternehmen)
- Interne / Externe Benutzer
- Wie häufig / intensiv arbeiten welche Benutzer mit dem System, wie groß ist ihr Fachwissen
- Besondere Merkmale der Benutzer (Einschränkungen,…)
- Spezielle Hardware

Usability Methoden

Contextual Inquiry

Bei Projektbeginn und in den ersten Phasen des Projektes.

Anayse der Benutzer und des Einsatzumfeldes des Systems. Bedürfnisse an die Anwendung werden durch Beobachtungen (vor Ort während der Arbeitszeit) und Befragungen erhoben. Wissen der Benutzer in Interviews oft schwer erfragbar (Benutzer erzählen nicht gerne andere lügen andere erzählen nicht relevante Sachen), deshalb Kombination aus Befragungen und Beobachtungen -> Erfassung des wirklichen Geschehens und der Gründe dahinter.

Anfänglich ungenaue und breitgefächerte Interviews, da Entwickler den Kontext nicht kennen, später dann genauere Fragen bei Interviews. Auswahl der Befragten sollen möglichste viele verschiedene Benutzer und Meinungen abdecken (Alter, Geschlecht, Position, Arbeitsort, Erfahrung, Fachwissen,…). Ziel ist es Handlungsweisen und Expertenwissen aufzudecken.

Relevant sind Daten wie

- Ziele und Bedürfnisse der befragte Personen sowie Probleme, Werte und Eigenheiten
- Aufgaben, Abläufe und Tätigkeiten -> Grundlage für Abläufe im neuen System
- Probleme, Schwierigkeiten und Lösungsansätze mit aktuellen Werkzeugen (Workarounds, Problemlösungen im aktuellen System) -> wesentliche Bedürfnisse und gewünschte Funktionen
- Taxonomie (Begriffe und Informationen für Datenmodell) -> bestehende Formulare, Dokumente sowie Software

Personas und Szenarien

Personas und Szenarien dienen als Vorlageperson bei der Entwicklung, da die Benutzer nicht immer vorhanden sind. Nur eine geringe Anzahl an Personas und Szenarien sind sinnvoll, sie sind aber nur begrenzt als Modellierungswerkzeug einsetz bar.

Personas

Modellierung der unterschiedlichen Benutzergruppen (in Form von prototypischen Anwendern) und der Anwendung aus Benutzersicht (deren unterschiedliche Ziele und Verhaltensweisen). Erstere sollen die relevanten Eigenschaften der Benutzer exemplarisch darstellen.

Werden anhand von Informationen aus Workshops, Contextual Inquiry, Fragebögen und Usability Walkthroughs anhand bestehender Systeme erstellt.

Ziel ist die Beschreibung folgender Merkmale:

- Ziele der Benutzer
- Beruf, Funktion, Aufgabe, Verantwortlichkeiten
- Fachliche Ausbildung, Wissen und Fähigkeiten
- Verhaltensmuster, Vorgehensweisen
- Werte, Ängste, Sehnsüchte, Vorlieben
- Computerkenntnisse
- Kenntnisse über verwandte Produkte, Vorgängersysteme und Konkurrenzprodukte
- Verbesserungspotential (Ideen) an der heuten Situation
- Erwartungen an die neue Lösung

Hilfreich bei der Entwicklung ist es ein konkretes Bild dieser Persona im Kopf zu haben und dabei helfen Eigenschaften wie Name, Alter, Geschlecht, Charakterzüge, Bild, Zitate,…

Es müssen folgende Personas unterschieden werden

- Primäre Personas: für deren Bedürfnisse wird das System entwickelt und das UI erstellet
- Sekundäre -„-: Bedürfnisse sind größtenteils durch primäre Personas abgedeckt. Kleine Erweiterungen nötig.
- Ergänzende -„- : Bedürfnisse sind vollständig durch primäre Personas gedeckt.
- Non- -„-: Bedürfnisse werden explizit nicht im System berücksichtigt.

Szenarien

Zentrales Element der benutzerorientierten Entwicklung. Repräsentiert ein realistisches Beispiel einer Benutzer–System Interaktion.

Szenarien werden basierend an den Anforderungen an das System erstellt und werden entweder iterativ entwickelt oder in Workshops mit den Benutzern. Sie sollen leicht verständlich sein und vom Auftraggeber/Benutzer/… überprüft, erweitert und korrigiert werden. Die Szenarien stellen eine konkrete Anwendung dar und sollen für bestimmte Benutzergruppen entworfen werden und zeigen wie diese das Produkt verwendet wird und somit deren Bedürfnisse erfüllen sollen. Sie beschreiben somit relevante Aspekte für den realen Einsatz, decken aber auch Sonder- und Fehlerfälle ab.

- Usability Testszenarien: für Evaluierung von Prototypen-/Systemtests
- Testszenarien: Grundlage für Softwaretesting
- Schulung: von Benutzern und Erstellung von Anleitungen

Storyboards

Ist teilweise eine Verbildlichung von Szenarios, soll der Kommunikation von ausgewählten Abläufen in einem System zwischen Auftraggeber, Benutzer und Entwickler dienen und zeigen wie die Interaktion von statten gehen wird.

Storyborads sollen durch die bildliche Darstellung Dialogabläufe der Benutzerschnittstellen, schwer verständliche Konzepte/Sachverhalte, den Anwenungskontext und die Umgebung in die das System eingebettet wird, darstellen. Außerdem stellen sie ein konkretes Fallbeispiel dar in dem kritische und wichtige Punkte detailliert dargestellt werden. Die handelnden Personen sollen charakterisiert sein (→Personas).

Folgende Details sollen enthalten sein:

- Berücksichtigte und nicht berücksichtigte Bedürfnisse
- Änderungen in Geschäftsprozessen
- Neuerungen der Arbeitsweise
- Enthaltene bzw. Nicht enthaltene Funktionalität
- Grundsätzlicher Aufbau der Benutzerschnittstellen
- Ausgewählte Details des UI

User Interface Prototyping

Frühe Darstellung der Umsetzung -> Fehler sollen früh gefunden werden. Konzipieren und Optimieren der Benutzerschnittstellen. Sie eignen sich um den Funktionsumfang zu illustrieren, die Funktionsweise zu verdeutlichen, die UI-Elemente zu spezifizieren, Navigation und Interaktion darzustellen, etc. Den Entwicklern dienen außerdem als Abschätzung des Realisierungsaufwandes.

Unterschieden wird zwischen Low-fi und High-fi Prototypen. Erstere werden Storyborads, Papier Mock-ups, Sketches, Vidos, Skizzen,….genannt um ein frühes Feedback von den Benutzern zu erhalten. High-fi Prototypen ist zum Beispiel Software, Flash/Director, Visual Basic Prototypen,…

Wichtig ist es die Dimension der Prototypen zu bestimmten, also welcher Funktionsumfang wird gezeigt bzw. wie detailliert soll dieser dargestellt werden, soll er interaktiv benutzbar sein (High-fi Prototypen), sollen realistische Daten zum Einsatz kommen. Wie viel UI-Technologie soll zum Einsatz kommen.

Wichtige Aspekte sind

- Finden sich die Benutzer in den Strukturen und Menüs zurecht
- Informationsdarstellung intuitiv
- Kommen spezielle Technologien zum Einsatz
- Grundsätzlicher Aufbau und Design
- Aufteilung und Struktur von Informationen
- Verwendung und Verhalten von Fenstern
- Wichtige Bedienelemente
- Navigation
- Prüfung von Eingaben und Anzeigen von Fehlermeldungen
- Speicherfunktionen
- …

UI Prototypen soll das Optimieren des UI ermöglichen und Punkte wie

- Flüssiges Arbeiten möglich
- Intuitive Benutzung oder gibt es Hürden
- Effiziente Navigation
- Werden gewünschte Informationen gefunden
- Warm- / Rückmeldungen bemerkbar und verständlich
- Passt die Schnittstelle zu den Details des Arbeitsablaufes

Graphische Darstellung bereits bei Prototypen relevant, da ein gut funktionierender aber schlecht designter Prototyp von den Benutzern eher abgelehnt wird.

Methoden

- **Papier/Stift:** Interviews, Workshops
- **Powerpoint, Keynote:** wenig Aufwand für erste Interaktionen
- **Photoshop:** schnell gut aussehende Mock-ups
- **Director / Flash:** Interaktive Prototypen mit hohen Designanforderungen oder Animationen
- **Programmierwerkzeuge:** Interaktive Prototypen für größere Datenmengen und komplexerem Verhalten

Entwicklerseitig sollte darauf geachtet werden, dass Prototypen nicht nach dem Motto „jetzt hab ich das aber eh schon implementiert“ weiterverwendet werden. Die ersten Prototypen werden fast immer verworfen. Anfängliche Prototypen sollten daher nicht programmiert werden.

Use Cases

Use Cases dienen zur Spezifikation funktionaler Anforderungen für die Entwicklung. Sie zeigen das Verhalten des Systems aus Benutzersicht. Die Funktionsvielfalt (Use Cases stellen immer für den Benutzer abschließende Handlungen dar) wird aus Sicht der Benutzer, hier Akteure, in zusammengehörige Einheiten aufgeteilt und erst später nach und nach spezifiziert. Die Akteure werden anhand von Workshops und Interviews erstellt, sollen aber keine realen Benutzer repräsentieren, aber trotzdem existierende Rollen darstellen.

Die Use Case Spezifikation bieten eine detaillierte Beschreibung der Anwendungsfälle und beschreibt alle Schritte zwischen Akteur und System, auch Sonder- und Fehlerfälle. Sowohl für Auftraggeber- sowie auch Entwicklerseite relevant.

Essential Use Cases sind abstrakte Use Cases die möglichst technikfrei formuliert sind. Ziel ist es sich nicht auf eine technische Lösung festzulegen.

Use Case vs Szenarien

Use Cases halten einen bestimmten Teil des Systemverhaltens für die Entwicklung fest während Szenarien konkrete Beispiele für die Systembenutzung darstellen.

Akteure vs Personas

Akteure definieren Rollen die mit dem System interagieren, während Personas prototypische Benutzer mit ihren Eigenschaften beschreiben.

Guidelines und Styleguides

Definieren von Regeln für die Gestaltung, sowohl globale Richtlinien sowie auch detaillierte Vorgaben.

Guidelines sind generelle Richtlinien für die Verwendung und Verhalten des UI wie zum Beispiel gesetzliche Vorschriften oder Normen aber auch Regelsammlungen für die Entwicklung von UI, User Interface Pattern, wo hingegen Styleguides konkrete Vorgaben für die visuelle Gestaltung und das Layout der Oberfläche sind. Letztere beschreiben das Look & Feel eines Systems und definieren Dinge wie technologische Rahmenbedingungen, Zielgruppe, allgemeine Regeln wie zum Beispiel Anzahl der Menüeinträge, Bedienung über Tastatur oder Punkte wie welche UI-Elemente werden in welchen Situationen verwendet oder wie haben sich GUI Elemente zu verhalten. Weiters sollten Styleguides Dinge definieren wie

- Navigation
- Visuelles Design
- Technische Umsetzbarkeit
- Terminologie (Begriffe, Bezeichnungen, Fachbegriffe,…)
- Tastaturbelegung (Shortcuts, Tabulatorreihenfolge,…)

Bei der Planung müssen Dinge wie

- Müssen Normen oder Gesetze eingehalten werden
- Gibt es bestehende Regelwerke (zum Beispiel Designvorgaben vom Auftraggeber)
- Lohnt sich die Entwicklung eigener Styleguides
- Sind vorhandene Styleguides überhaupt brauchbar bzw. detailliert genug
- Kann durch geeignete Styleguides die Kommunikation im Unternehmen erhöht werden

Usability Testing

Bewerten des neuen Systems durch Benutzer

Fragebögen

Feedback zum neuen System in großer Zahl um aussagekräftige Zahlen zur Analyse zu Benutzer und Kontext, Prototypen oder dem neuen System ableiten zu können.