SAPIENZA
UNIVERSITÀ DI ROMA

Università degli Studi di Roma "La Sapienza"

Riassunti di Interazione Uomo Macchina

Alessandro Dori

Docente:
Prof. Maurizio Mancini

Anno Accademico 2024/2025

Indice

1 Panoramica su HCI (Human-Computer Interaction) 6
- 1.1 Obiettivi del corso . . . 6
- 1.2 Che cos'è HCI? . . . 6
- 1.3 Aspetti Multidisciplinari di HCI . . . 6
- 1.4 Modelli di Interazione . . . 6
  - 1.4.1 Modello di Norman . . . 6
  - 1.4.2 Modello di Abowd e Beale . . . 7
  - 1.4.3 Errori Umani nell'Interazione . . . 7
- 1.5 Processi di Progettazione e Usabilità . . . 7
  - 1.5.1 Usabilità . . . 7
  - 1.5.2 Design Thinking . . . 7
  - 1.5.3 User-Centered Design (UCD) . . . 8

2 Introduzione alle Interfacce 8
- 2.1 Definizione di Interfaccia . . . 8
- 2.2 Interfaccia nell'Interazione Uomo-Macchina . . . 8

3 Introduzione al Needfinding 8
- 3.1 Obiettivi . . . 8
- 3.2 Domande Principali del Needfinding . . . 9
- 3.3 Metodi di Needfinding . . . 9
- 3.4 Osservazione . . . 9
  - 3.4.1 Obiettivi dell'Osservazione . . . 10
  - 3.4.2 Come Osservare . . . 10
  - 3.4.3 Cosa Osservare . . . 10
- 3.5 Tecniche di Osservazione . . . 10
  - 3.5.1 Osservare senza coinvolgere l'utente . . . 10
  - 3.5.2 Mettersi nei panni degli utenti . . . 10
  - 3.5.3 Ascoltare e Osservare . . . 11
- 3.6 Attenzione alla Percezione degli Utenti . . . 11
- 3.7 Differenza tra Needs e Soluzioni . . . 11
- 3.8 Sfide e Rischi dell'Osservazione . . . 11
  - 3.8.1 Pazienza . . . 11
  - 3.8.2 Effetto Hawthorne . . . 11
- 3.9 Interviste . . . 11
  - 3.9.1 Come trovare partecipanti . . . 12
  - 3.9.2 Linee guida per le interviste . . . 12
  - 3.9.3 Attenzione alle domande . . . 12
  - 3.9.4 Esecuzione dell'intervista . . . 12
- 3.10 Needfinding: Altre Tecniche di Query . . . 13
  - 3.10.1 Interviste Specifiche . . . 13
  - 3.10.2 Tipologie di Interviste Particolari . . . 13
- 3.11 Sondaggi (Questionari) . . . 13
  - 3.11.1 Vantaggi dei Questionari . . . 14
  - 3.11.2 Considerazioni sui Questionari . . . 14
  - 3.11.3 Strumenti per Questionari . . . 14
- 3.12 Diari . . . 14

3.13 Pager Studies . . . 14
3.14 Camera Studies . . . 14
3.15 Inchiesta Contestuale . . . 15
3.16 Scale di Misurazione . . . 15
3.17 Conclusione . . . 15

4 Task Analysis **15**
4.1 Definizione . . . 15
4.2 Needs e Goals . . . 15
4.3 Caratteristiche della Task Analysis . . . 16
4.4 Modello di un Task . . . 16
4.5 Esempi . . . 16
4.6 Importanza della Task Analysis . . . 17
4.7 Esempi di Needs e Tasks . . . 17

5 Personas **17**
5.1 Definizione . . . 17
5.2 Obiettivi delle Personas . . . 17
5.3 Elementi di una Persona . . . 18
5.4 Vantaggi delle Personas . . . 18

6 Storyboards **18**
6.1 Definizione . . . 18
6.2 Elementi di uno Storyboard . . . 19
6.3 Tipologie di Storyboards . . . 19
6.4 Perché Disegnati a Mano? . . . 19
6.5 Benefici degli Storyboards . . . 19
6.6 Esempi di Storyboards . . . 19
6.7 Storyboarding Avanzato . . . 20
6.7.1 Aspetti Principali . . . 20
6.7.2 Forma dello Storyboarding . . . 20
6.7.3 Fumetti nello Storyboarding . . . 20
6.7.4 Vantaggi dello Storyboarding . . . 20
6.7.5 Strumenti per Storyboarding . . . 20
6.7.6 Quanti Storyboard Creare? . . . 21

7 Prototyping **21**
7.1 Definizione . . . 21
7.2 Obiettivi del Prototyping . . . 21
7.3 Approcci al Prototyping . . . 21
7.4 Ciclo Iterativo di Prototipazione . . . 21
7.5 Potenziali Problemi del Prototyping . . . 22
7.5.1 Tempo . . . 22
7.5.2 Pianificazione . . . 22
7.5.3 Caratteristiche Non Funzionali . . . 22
7.5.4 Inerzia del Design . . . 22
7.6 Tipologie di Prototipi . . . 22
7.6.1 Low-Fidelity Prototypes . . . 22
7.6.2 Medium-Fidelity Prototypes . . . 22

7.6.3 High-Fidelity Prototypes . . . 23
7.7 Wizard of Oz Prototyping . . . 23
7.8 Benefici e Limiti del Prototyping . . . 23
7.8.1 Benefici . . . 23
7.8.2 Limiti . . . 23
7.9 Paper Prototyping . . . 23
7.9.1 Che cos'è? . . . 23
7.9.2 Cosa Disegnare? . . . 24
7.9.3 Strumenti e Materiali . . . 24
7.9.4 Testing con il Paper Prototyping . . . 24
7.9.5 Vantaggi del Low-Fidelity Prototyping . . . 24
7.9.6 Cosa Permette di Testare? . . . 24
7.9.7 Miti sul Paper Prototyping . . . 25
7.10 Tool e Mago di Oz . . . 25
7.10.1 Paper Prototyping con il supporto del computer . . . 25
7.10.2 POP (by Marvel) . . . 25
7.10.3 Prototipi Mago di Oz . . . 25
7.10.4 Vantaggi del Mago di Oz . . . 26
7.10.5 Svantaggi del Mago di Oz . . . 26

8 Observational Methods **26**
8.1 Valutazione . . . 26
8.2 Obiettivi della Valutazione . . . 26
8.3 Metodi di Osservazione . . . 27
8.3.1 Think Aloud . . . 27
8.3.2 Cooperative Evaluation . . . 27
8.3.3 Protocol Analysis . . . 27
8.3.4 Post-task Walkthroughs . . . 28
8.4 Stili di Valutazione . . . 28
8.5 Questioni Etiche . . . 28

9 Esperimenti **28**
9.1 Valutazione Sperimentale . . . 28
9.2 Fattori Sperimentali . . . 28
9.3 Tipi di Ipotesi . . . 29
9.4 Design Sperimentale . . . 29
9.5 Condizioni . . . 29
9.6 Metodi Sperimentali . . . 29
9.6.1 Between Subjects . . . 29
9.6.2 Within Subjects . . . 29
9.7 Analisi dei Dati . . . 30
9.8 Analisi Statistica . . . 30

10 Expert-based Evaluation **30**
10.1 Valutazioni senza utenti . . . 30
10.1.1 Vantaggi e svantaggi . . . 30
10.2 Tipi di valutazione Expert-based . . . 31
10.3 Esempio: Zoom Screen Sharing . . . 31
10.3.1 Cognitive Walkthrough . . . 31

10.3.2 Heuristic Evaluation 31

11 Progetto e sviluppo di sistemi interattivi II **32**
11.1 Metodologie di Sviluppo 32
11.2 Waterfall Model 32
11.3 Agile Methodology 32
11.4 Confronto tra Waterfall e Agile 33

12 Agile UCD (User-Centered Design) **33**
12.1 Definizione di Agile UCD 33
12.2 Caratteristiche principali 34
12.3 Ciclo di vita Agile UCD 34
12.4 Vantaggi di Agile UCD 34
12.5 Sfide di Agile UCD 34

13 Affordances e Signifiers **34**
13.1 Definizione di Affordances (Norman) 34
13.1.1 Esempi di Affordances 35
13.2 Definizione di Signifiers (Norman) 35
13.2.1 Esempi di Signifiers 35
13.3 Affordances vs Signifiers 35
13.4 Matrice di Affordances e Signifiers 35

14 Modi e Gestione dell'Interazione **36**
14.1 Terminologia 36
14.2 Modi 36
14.3 Minimizzare gli Errori di Modo 36
14.4 Tipi di Modi 36
14.5 Interazione Noun-Verb e Verb-Noun 37
14.6 Pro e Contro di Noun-Verb e Verb-Noun 37

15 Progetto di Applicazioni Mobili **37**
15.1 Contesto d'uso 37
15.2 Attenzione 37
15.3 Differenze rispetto al Desktop 37
15.3.1 Schermo dello smartphone 37
15.3.2 Memoria limitata 37
15.3.3 Una schermata alla volta 38
15.3.4 Un'applicazione alla volta 38
15.3.5 Help minimale 38

16 Stili di Applicazioni Mobili **38**
16.1 Productivity Application 38
16.2 Utility Application 39
16.3 Immersive Application 39

17 Productivity App e Modi **39**
17.1 Caratteristiche delle App di Produttività . . . 39
17.2 Approccio tipico . . . 40
17.3 Esempio: App Mail (Gmail) . . . 40
17.4 Approccio Noun-Verb . . . 40
17.5 Vantaggi di Noun-Verb . . . 40
17.6 Animazione delle View . . . 40
17.7 Aggiunta di un Oggetto: Verb-Noun . . . 41
17.8 Vista di Default . . . 41

18 Design Patterns **41**
18.1 Introduzione ai Design Patterns . . . 41
18.2 Caratteristiche dei Design Patterns . . . 41
18.3 UI Design Patterns . . . 41
18.4 Design Patterns in App Mobile (Android) . . . 42
18.5 Dark Patterns . . . 42
18.6 Attention-Capture Damaging Patterns (ACDPs) . . . 42

19 Interfacce Android **42**
19.1 Principi di Base . . . 42
19.2 Leverage on Users' Knowledge . . . 43
19.3 Scopribilità . . . 43
19.4 Azione e Reazione . . . 43
19.5 Memoria . . . 43
19.6 Controllo dell'Utente . . . 43
19.7 Coerenza . . . 43
19.8 Material Design . . . 43
19.9 Lists e Cards . . . 44
19.9.1 Lists . . . 44
19.9.2 Cards . . . 44
19.10 Alerts e Bottom Sheets . . . 44
19.10.1 Alerts . . . 44
19.10.2 Bottom Sheets . . . 44
19.11 Componenti principali . . . 44
19.12 Sheets . . . 45
19.13 Navigazione e Barre . . . 45
19.14 Transizioni . . . 45
19.15 Sistema cromatico di Material Design 3 (M3) . . . 45
19.16 Componenti del Sistema Cromatico . . . 45
19.17 Principali Ruoli di Colore . . . 46
19.18 Generazione della Palette di Tonalità . . . 46
19.19 Creazione di Temi con Figma . . . 46
19.20 Uso con M3 Design Kit in Figma . . . 46
19.21 Riferimenti Utili . . . 46

1 Panoramica su HCI (Human-Computer Interaction)

1.1 Obiettivi del corso

- Comprendere come progettare esperienze utente interagendo con applicazioni, dispositivi e ambienti moderni.
- Acquisire conoscenze approfondite su un processo centrato sull'uomo per creare sistemi interattivi e applicarlo nella pratica.
- Familiarizzare con metodi per raccogliere e ascoltare i bisogni degli utenti.
- Imparare a valutare sistemi interattivi con i loro utenti.

1.2 Che cos'è HCI?

- Un campo multidisciplinare che si occupa di progettazione, valutazione e implementazione di sistemi informatici interattivi per l'uso umano.
- Coinvolge l'interazione tra due entità:
  - L'essere umano, con i suoi obiettivi e compiti.
  - Il computer, con il suo stato e i suoi linguaggi.
- Studia il comportamento reciproco nel tempo di questi due attori.

1.3 Aspetti Multidisciplinari di HCI

- **Psicologia e scienze cognitive**: capacità percettive e cognitive degli utenti.
- **Ergonomia**: capacità fisiche degli utenti.
- **Sociologia**: contesto sociale dell'interazione.
- **Informatica e ingegneria**: costruzione di artefatti necessari (HW, SW).
- **Design grafico**: presentazione efficace dell'interfaccia.
- **Scrittura tecnica**: documentazione e contenuti su schermo.

1.4 Modelli di Interazione

1.4.1 Modello di Norman

- Descrive l'interazione come un ciclo di esecuzione e valutazione:
  1. Stabilire il goal ($WHAT$).
  2. Formare l'intenzione (HOW).
  3. Specificare la sequenza di azioni.
  4. Eseguire l'azione.

5. Percepire lo stato del sistema.
6. Interpretare lo stato del sistema.
7. Valutare lo stato del sistema rispetto al goal.

- Evidenzia due **gap** principali:
  - **Gulf of Execution**: difficoltà di tradurre l'intenzione in azione.
  - **Gulf of Evaluation**: difficoltà di interpretare lo stato del sistema.

1.4.2 Modello di Abowd e Beale

- Introduce l'importanza dell'interfaccia utente (linguaggio UI) come mediatore tra linguaggio utente (task) e linguaggio di sistema (core).
- Evidenzia l'importanza della presentazione, performance, osservazione e articolazione nell'interazione.

1.4.3 Errori Umani nell'Interazione

- **Slip**: azione errata nonostante l'intenzione corretta.
- **Mistake**: goal o intenzione errati a causa di una comprensione incompleta del sistema.
- Gli errori umani non devono essere considerati colpa dell'utente, ma spesso riflettono un design inadeguato.

1.5 Processi di Progettazione e Usabilità

1.5.1 Usabilità

- Definita come la capacità di un sistema di essere utilizzato efficacemente, efficientemente e con soddisfazione in un contesto specifico.
- Dimensioni principali:
  - Utilità, apprendibilità, memorizzabilità, efficacia, efficienza, visibilità, gestione degli errori, soddisfazione.

1.5.2 Design Thinking

- Processo iterativo in 5 fasi:
  1. Empatizzare.
  2. Definire.
  3. Ideare.
  4. Prototipare.
  5. Testare.

1.5.3 User-Centered Design (UCD)

- Si focalizza sui bisogni, desideri e limitazioni degli utenti finali in tutte le fasi del processo di progettazione.
- Coinvolge utenti reali in modo collaborativo per creare prototipi e scenari.

2 Introduzione alle Interfacce

2.1 Definizione di Interfaccia

Secondo il *Merriam-Webster Dictionary*, un'interfaccia è:

- Una superficie che costituisce il confine comune tra due corpi, spazi o fasi, ad esempio un'interfaccia olio-acqua.
- Un punto d'incontro dove sistemi indipendenti e spesso non correlati interagiscono o comunicano, come l'interfaccia uomo-macchina.
- I mezzi attraverso i quali si realizza l'interazione o la comunicazione in un'interfaccia.

2.2 Interfaccia nell'Interazione Uomo-Macchina

L'interfaccia utente è lo spazio dove avviene l'interazione tra esseri umani e macchine. L'obiettivo è consentire un'operazione efficace e il controllo della macchina da parte dell'utente, mentre la macchina fornisce un feedback che supporta il processo decisionale.

Elementi dell'interfaccia utente:

- **Hardware**: componenti fisici.
- **Software**: componenti logici.

Le interfacce utente offrono:

- **Input**: per manipolare un sistema.
- **Output**: per indicare gli effetti delle manipolazioni degli utenti.

3 Introduzione al Needfinding

3.1 Obiettivi

Il **Needfinding** si focalizza sulla comprensione dei requisiti di sistema e dei bisogni degli utenti. Gli obiettivi principali includono:

- Identificare i bisogni latenti nel sistema (*gaps*).
- Scoprire opportunità riconoscendo queste lacune.
- Raccontare una nuova "storia" sul come e sul perché migliorare un sistema.

3.2 Domande Principali del Needfinding

Per identificare i bisogni degli utenti, bisogna rispondere a queste domande:

- Chi sono gli utenti del sistema?
  - Gruppo omogeneo o diverse categorie (es. giovani/anziani, esperti/principianti)?
- Cosa fanno attualmente per soddisfare questi bisogni?
- Qual è il contesto d'uso?

Nota Importante:

- **Tu non sei un utente rappresentativo**: le competenze di designer e sviluppatori sono diverse da quelle degli utenti finali.
- **Il cliente non è un utente rappresentativo**: anche i manager potrebbero non comprendere appieno le esigenze degli utenti finali.

3.3 Metodi di Needfinding

Tra i metodi più utilizzati troviamo:

- **Osservazione** (naturale o controllata).
- **Diari** compilati dagli utenti.
- **Interviste** e **Focus Group**.
- **Sondaggi** e **Inchieste contestuali**.

3.4 Osservazione

Osservazione Etnografica

- Immergersi nell'ambiente degli utenti per comprendere lingua, cultura e comportamenti.
- Registrare audio-video o prendere appunti dettagliati.
- Rischi: interpretazioni errate, interruzione delle attività normali, trascurare dettagli importanti.

Osservazione Naturale

- Studiare gli utenti "sul campo" per scoprire problemi reali d'uso.
- Genera idee per miglioramenti, ma è difficile da replicare e controllare.

3.4.1 Obiettivi dell'Osservazione

- Individuare le necessità.
- Comprendere gli obiettivi degli utenti.
- Identificare i problemi esistenti.

Citazione: "Innamòrati del problema e non della soluzione." – Uri Levine

3.4.2 Come Osservare

- Senza preconcetti su cosa cercare.
- Essere aperti a scoperte inaspettate.
- Usare l'osservazione per definire il problema.

3.4.3 Cosa Osservare

- Cosa fanno gli utenti ora.
- I loro obiettivi.
- Gli strumenti utilizzati.
- Le attività più ampie a cui appartiene quella osservata.
- Il contesto e i momenti della giornata.
- Somiglianze e differenze tra gli utenti.

3.5 Tecniche di Osservazione

3.5.1 Osservare senza coinvolgere l'utente

- Stare vicino agli utenti senza disturbarli.
- Osservare in spazi pubblici (strada, locali, trasporti pubblici).
- Analizzare registrazioni video.

3.5.2 Mettersi nei panni degli utenti

- Lavorare insieme agli utenti.
- Farsi insegnare i passi del loro processo.
- Notare:
  - Trucchi, scorciatoie, modifiche.
  - Errori.

3.5.3 Ascoltare e Osservare

- Evitare di influenzare gli utenti.
- Tralasciare i propri obiettivi.
- Lasciar parlare gli utenti, concentrandosi su:
  - Emozioni.
  - Paure.
  - Frustrazioni.

3.6 Attenzione alla Percezione degli Utenti

- La percezione dell'utente può essere inconsapevolmente errata.
- Osservare ciò che fanno realmente, non solo ciò che dicono di fare.
- Questo offre opportunità per innovazioni.

3.7 Differenza tra Needs e Soluzioni

- L'obiettivo è individuare i **needs** (necessità).
- I needs aprono nuove possibilità.
- Le soluzioni limitano l'innovazione.

3.8 Sfide e Rischi dell'Osservazione

3.8.1 Pazienza

- Le osservazioni possono sembrare di routine.
- Le intuizioni emergono spesso dai dettagli e dalle sfumature.

3.8.2 Effetto Hawthorne

- Il comportamento degli utenti può cambiare a causa dell'osservazione stessa.

3.9 Interviste

Le interviste sono uno strumento fondamentale per il Needfinding:

- **Vantaggi**:
  - Informali e a basso costo.
  - Permettono approfondimenti e scoperte inaspettate.
- **Svantaggi**:
  - Soggettive e richiedono tempo.

- Possono essere in persona, strutturate o non strutturate, e di tipo individuale o di gruppo.
- Esempi di domande aperte:
  - "Parlami della tua giornata tipica."
  - "Quali sono i tre aspetti migliori e peggiori di questo sistema?"
- Evitare domande chiuse, suggestive o ipotetiche.

3.9.1 Come trovare partecipanti

- Utilizzare i social network o amici di amici.
- Offrire incentivi se possibile.
- Anche partecipanti non ideali possono offrire spunti interessanti.

3.9.2 Linee guida per le interviste

- Preparare le domande in anticipo.
- Registrare l'intervista o prendere appunti dettagliati (previo consenso).
- Rimanere neutrali rispetto alle opinioni dell'intervistato.
- Incoraggiare l'approfondimento delle risposte.

3.9.3 Attenzione alle domande

- **Evita domande**:
  - Con risposta scontata o troppo generiche.
  - Basate su scenari ipotetici.
  - Che chiedono quanto spesso fai qualcosa.
  - Con risposta sì/no o che iniziano con "Ti piacerebbe...?".
- **Preferisci domande**:
  - Aperte, specifiche e concrete.
  - Focalizzate su situazioni note all'intervistato.
  - Che chiedano "quando" (ma non lontano nel tempo).

3.9.4 Esecuzione dell'intervista

Fasi principali (Robson e McCartan, 2016):

1. **Introduzione**: presentare se stessi, spiegare lo scopo e rassicurare sugli aspetti etici.
2. **Warm-up**: domande semplici e non minacciose (es. informazioni demografiche).

3. **Sessione principale**: domande in ordine logico, con le più complesse alla fine.
4. **Cooling-off**: domande facili per rilassare la tensione.
5. **Chiusura**: ringraziare l'intervistato e segnalare la fine dell'intervista.

3.10 Needfinding: Altre Tecniche di Query

3.10.1 Interviste Specifiche

- **Lead users**:
  - Utenti con necessità innovative, prima di altri.
  - Altamente competenti e sofisticati.
  - Spesso trovano da soli soluzioni ai loro bisogni.
  - I loro bisogni anticipano le esigenze future di molti.
- **Extreme users**:
  - Spingono un sistema esistente al limite.
  - Evidenziano problemi difficilmente osservabili.
  - Hanno bisogni amplificati, con soluzioni e workaround più evidenti.
- **Esperti**:
  - Forniscono conoscenze aggregate sul comportamento degli utenti.
  - Possono discutere problemi complessi o astratti.

3.10.2 Tipologie di Interviste Particolari

- **History interviews**: analizzano sequenze di eventi per comprendere comportamenti.
- **Process mapping**: richiedono all'intervistato di descrivere un processo completo.
- **Laddering**: tecniche basate su domande iterative del tipo "Perché?".
- **Cultural context**: non strutturate, per esplorare il contesto culturale.
- **Intercepts**: una singola domanda per raccogliere insight rapidi.

3.11 Sondaggi (Questionari)

- Utili per ottenere una visione generale, ma non adatti a un'analisi profonda.
- Struttura consigliata:
  - Dichiarare lo scopo del sondaggio.
  - Domande su background demografico, esperienza, responsabilità lavorative, ecc.
- Rischi: dati parziali o non rappresentativi.

3.11.1 Vantaggi dei Questionari

- Uniformità: stesse domande per tutti gli utenti.
- Velocità nella raccolta di risposte.
- Analisi rigorosa e dati strutturati.

3.11.2 Considerazioni sui Questionari

- Meno flessibili delle interviste.
- Richiedono una progettazione accurata e test pilota.
- Diversi tipi di domande: aperte, chiuse, a scelta singola, multipla, scale di valori, ranking, ecc.

3.11.3 Strumenti per Questionari

- Tool online: Survey Monkey, TypeForm, Google Forms, ecc.
- Caratteristiche:
  - Creazione e organizzazione delle domande.
  - Invio via email o pubblicazione online.
  - Elaborazione automatica delle statistiche e raccolta dati.

3.12 Diari

- Utili per raccogliere informazioni su attività sporadiche o per studi longitudinali.
- Esistono app dedicate alla gestione di diari digitali.

3.13 Pager Studies

- L'intervistato annota in tempo reale:
  - In momenti specifici della giornata.
  - In un luogo determinato.
- Note libere o compilazione di moduli.
- Ricerca di pattern o ripetizioni nei dati raccolti.

3.14 Camera Studies

- Gli utenti utilizzano videocamere per registrare le loro attività.
- Permette di analizzare l'attività "con gli occhi dell'utente".

3.15 Inchiesta Contestuale

- Combina osservazione e intervista: osservare gli utenti nel loro ambiente naturale mentre svolgono compiti.
- Permette di ottenere una comprensione profonda dei loro pensieri e processi di lavoro.

3.16 Scale di Misurazione

- **Scala Nominale**: classi distinte senza ordine (es. città, colore preferito).
- **Scala Ordinale**: classi ordinate senza distanza definita (es. preferenze, punteggi Likert).
- **Scala di Rapporto**: valori numerici con uno zero fisso (es. durata di un compito).

3.17 Conclusione

Il Needfinding è un processo essenziale per progettare sistemi centrati sull'utente, combinando osservazioni, interviste e strumenti di analisi per comprendere a fondo le esigenze degli utenti.

4 Task Analysis

4.1 Definizione

La **Task Analysis** è lo studio del modo in cui le persone svolgono le loro attività, con l'obiettivo di determinare:

- **Cosa fanno**: i passi e le sequenze delle azioni.
- **Cosa usano**: artefatti e strumenti coinvolti.
- **Quanto bene raggiungono gli obiettivi**: analisi dei risultati.

4.2 Needs e Goals

- I **needs** (bisogni) guidano i **goals** (obiettivi).
- Un need può generare molteplici goals:
  - Es. migliorare il benessere (need) → fare esercizio, monitorare il sonno (goals).
- I goals traducono i bisogni in azioni concrete:
  - Es. bisogno di sicurezza → installare un sistema di sicurezza (goal).

4.3 Caratteristiche della Task Analysis

- Permette di apprendere sui normali utenti osservandoli in azione.
- Contribuisce a:
  - Identificare i task che un'applicazione deve supportare.
  - Raffinare la navigazione o la struttura di un'applicazione.
  - Definire i requisiti applicativi e strategia di contenuto.
  - Valutare usabilità e creare prototipi iniziali.
- Sfide:
  - Non si progettano task, ma interfacce.
  - Task e oggetti non mappano 1:1.

4.4 Modello di un Task

Definizione di Benyon:

- Un **task** è un obiettivo accompagnato da un insieme di azioni ordinate.
- Strutturato in livelli di descrizione fino ad arrivare ad azioni semplici.
- Le **azioni** sono task semplici, senza problem-solving.

4.5 Esempi

- Es. "Pulire la casa":
  1. Prendere l'aspirapolvere.
  2. Sistemare gli accessori necessari.
  3. Pulire le stanze.
  4. Svuotare il sacco della polvere.
  5. Riporre l'aspirapolvere.
- Es. "Usare un proiettore":
  1. Collegare il proiettore alla corrente e accendere.
  2. Trovare e premere il pulsante di accensione.
  3. Inserire la diapositiva e regolarla.
  4. Allineare il proiettore allo schermo.
  5. Mettere a fuoco.

4.6 Importanza della Task Analysis

- Scoprire i **goals** degli utenti e come li raggiungono.
- Comprendere:
  - Le esperienze (personali, sociali, culturali) degli utenti.
  - Come l'ambiente fisico influenza i loro comportamenti.
  - Come conoscenze pregresse influenzano il workflow e i punti critici.

4.7 Esempi di Needs e Tasks

- **Need**: trasporto conveniente → **Task**: prenotare un viaggio tramite app.
- **Need**: sicurezza → **Task**: monitorare i dettagli del conducente e seguire la corsa.
- **Need**: risparmio → **Task**: confrontare opzioni di viaggio e prezzi.

5 Personas

5.1 Definizione

Le **Personas** sono uno strumento utilizzato nel design di interfacce. Rappresentano modelli astratti di utenti reali e servono per comprendere meglio:

- Le esigenze (*needs*).
- I comportamenti (*actions*).

Definizione Formale:

- Rappresentazioni semi-fittizie di utenti reali basate su dati demografici, comportamentali e psicografici.
- Identificano categorie di utenti con caratteristiche, obiettivi e necessità specifiche.

5.2 Obiettivi delle Personas

- Creare **empatia** verso gli utenti, permettendo ai designer di comprendere meglio i loro bisogni.
- Guidare le decisioni di progettazione per ottimizzare l'esperienza utente.
- Fornire un quadro di riferimento per identificare esigenze e sfide degli utenti.

5.3 Elementi di una Persona

Una **Persona** include:

- **Nome**: conferisce un'identità.
- **Foto**: aiuta a creare una connessione umana.
- **Caratteristiche demografiche**: età, sesso, professione, istruzione, ecc.
- **Obiettivi**: cosa vuole ottenere utilizzando l'interfaccia?
- **Frustrazioni**: quali problemi potrebbe incontrare?
- **Comportamenti**: come utilizza il sistema? Quali sono le sue abitudini?

5.4 Vantaggi delle Personas

- **Creare empatia**: i designer possono interiorizzare obiettivi, bisogni e desideri degli utenti.
- **Sviluppare il focus**: aiuta a focalizzarsi su specifiche categorie di utenti evitando di progettare per tutti.
- **Comunicare e raggiungere consenso**: sintetizzano le conclusioni della ricerca per chi non ha partecipato direttamente.
- **Prendere decisioni**: vedere il mondo dalla prospettiva dell'utente facilita le scelte progettuali.
- **Misurare l'efficacia**: fungono da sostituti degli utenti reali per valutare il design.

6 Storyboards

6.1 Definizione

Uno **Storyboard** è una rappresentazione grafica dell'aspetto esteriore di un sistema previsto, senza funzionalità associate. Può essere descritto come:

- Una rappresentazione simile a un fumetto che illustra l'esecuzione di un task.
- Una sequenza di pannelli o schizzi che mostra cosa può fare una persona in un determinato scenario.
- Uno strumento per comunicare il flusso di eventi in punti chiave.
- Non richiede competenze artistiche: è focalizzato sulla comunicazione di idee, non sulla qualità delle immagini.

6.2 Elementi di uno Storyboard

- **Goal**: illustra l'obiettivo del task.
- **Sequence**: mostra come si sviluppa il task, incluso:
  - Passaggi principali coinvolti.
  - Ruolo dell'interfaccia nel supportare gli utenti.
  - Trigger che porta all'utilizzo del sistema.
- **Satisfaction**: rappresenta la motivazione dell'utente e il risultato finale raggiunto.

6.3 Tipologie di Storyboards

- **Tradizionali**: simili ai fumetti, includono:
  - Attori, fumetti per i dialoghi, sfondi.
  - Annotazioni descrittive per spiegare ogni scena.
- **Scored storyboards**: si concentrano su dinamiche specifiche (movimenti, colori, suoni) con annotazioni dettagliate.
- **Solo Testo**: utilizzati quando il comportamento dell'interazione è troppo complesso per essere illustrato.

6.4 Perché Disegnati a Mano?

- **Rapidità**: non richiede strumenti grafici avanzati, facilitando sperimentazioni rapide.
- **Imprecisione**: incoraggia gli utenti a fornire feedback e suggerimenti senza essere influenzati da dettagli superflui.

6.5 Benefici degli Storyboards

- Sottolineano come un'interfaccia supporta il completamento di un task.
- Focalizzano la discussione e il feedback sui task degli utenti.
- Favoriscono la coesione tra i membri del team sugli obiettivi dell'applicazione.
- Evitano distrazioni su dettagli non rilevanti (es. pulsanti o stili grafici).

6.6 Esempi di Storyboards

Scenario: Applicazione per Ricette Un esempio illustra come un'app scarica automaticamente la ricetta del giorno sullo smartphone dell'utente, permettendogli di:

- Consultare la lista della spesa mentre è in metro.
- Acquistare gli ingredienti necessari.
- Preparare un pasto sano.

6.7 Storyboarding Avanzato

6.7.1 Aspetti Principali

A questo livello, lo storyboarding si concentra su:

- **Ruolo dell'interfaccia utente (UI)**: comprendere come supporta i task.
- **Attori coinvolti**: identificare gli utenti e il loro contesto.
- **Contesto e ambiente**: descrivere dove avvengono i task.
- **Task principali**: rappresentare i task senza progettare l'interfaccia.

6.7.2 Forma dello Storyboarding

- Disegnare i task principali in modo semplice e leggibile.
- Considerare vincoli come:
  - Efficienza: evitare di perdere tempo.
  - Comprensibilità: descrizioni facilmente interpretabili.
  - Comunicazione: condividere i task con altri designer.

6.7.3 Fumetti nello Storyboarding

- Disegni a mano libera, semplici e chiari (2-4 vignette per ogni task).
- Poco testo, focalizzato su azioni e flusso.
- Non includere le schermate dell'interfaccia, poiché non sono ancora progettate.

6.7.4 Vantaggi dello Storyboarding

1. Mostra il sistema e il contesto d'uso.
2. Facilita la condivisione con altri membri del team.
3. Non vincola a un'interfaccia specifica.
4. Disegni rapidi e facili da correggere.
5. Permette discussioni e miglioramenti.

6.7.5 Strumenti per Storyboarding

- Esistono strumenti software per creare storyboard in stile fumetto.
- Per il corso, non sono necessari strumenti avanzati.

6.7.6 Quanti Storyboard Creare?

- Di solito, 1 storyboard per 1-2 task principali.
- Ridurre il numero di storyboard e focalizzarsi su task essenziali.
- Tipicamente, 2-4 storyboard per 3-5 task sono sufficienti per il corso.

7 Prototyping

7.1 Definizione

Il **prototyping** consiste nella creazione di un modello di un sistema interattivo. Questo modello:

- Simula o anima alcuni aspetti/caratteristiche/funzioni del sistema.
- Non include necessariamente tutte le funzionalità.

7.2 Obiettivi del Prototyping

- Valutare l'impatto sugli utenti.
- Provare e validare un'idea o concetto.
- Imparare dai feedback.
- Testare versioni iniziali, intermedie o finali.

7.3 Approcci al Prototyping

- **Throw-away**: il prototipo viene scartato dopo aver acquisito le conoscenze necessarie.
- **Incrementale**: le funzioni vengono aggiunte gradualmente al prototipo.
- **Evolutivo**: il prototipo funge da base per versioni successive che lo migliorano.

7.4 Ciclo Iterativo di Prototipazione

- Il primo progetto non sarà perfetto: i progettisti devono aspettarsi errori.
- Processo iterativo:
  1. Progettazione.
  2. Creazione del prototipo.
  3. Validazione (*OK/NOK*).
  4. Revisione o riprogettazione, se necessario.
- Miglioramento continuo (*hill climbing*).

7.5 Potenziali Problemi del Prototyping

7.5.1 Tempo

- Realizzare prototipi richiede tempo.
- Prototipi scartati possono sembrare una perdita di tempo.
- Soluzione: **rapid prototyping** (ma evitare valutazioni affrettate).

7.5.2 Pianificazione

- Difficile pianificare un processo di design con prototipi.
- Complessità nella stima dei costi.
- Contrattazioni tra committente e progettista.

7.5.3 Caratteristiche Non Funzionali

- Aspetti come sicurezza, affidabilità e tempo di risposta sono spesso sacrificati nello sviluppo del prototipo.

7.5.4 Inerzia del Design

- Decisioni iniziali sbagliate possono non essere riviste (vedi *hill climbing*).
- Importanza di comprendere le cause dei problemi, non solo i sintomi.

7.6 Tipologie di Prototipi

7.6.1 Low-Fidelity Prototypes

- Permettono di visualizzare rapidamente idee.
- Materiali: carta, penne, post-it, stencil, ecc.
- Simulano l'interazione attraverso schemi disegnati.
- Il computer è simulato da un attore umano.

7.6.2 Medium-Fidelity Prototypes

- Wireframes e mockup statici.
- Utilizzano software come Figma, Balsamiq o Moqups.
- Interattività limitata e percorso predefinito.
- Ideali per testare il flusso e il layout dell'interfaccia.

7.6.3 High-Fidelity Prototypes

- Prototipi interattivi che sembrano prodotti finiti.
- Costosi e richiedono molto tempo.
- Feedback focalizzato su dettagli grafici (colori, font).
- Utili per testare comportamento realistico e problemi di usabilità.

7.7 Wizard of Oz Prototyping

- Simulazione di tecnologie future non ancora implementabili.
- Un essere umano (wizard) finge di essere il sistema.
- Spesso utilizzato per testare interfacce vocali o sistemi basati sull'apprendimento.
- Benefici:
  - Rapidità ed economicità.
  - Sperimentazione di idee avanzate.
- Rischi:
  - Sovrastima delle capacità tecnologiche.
  - Complessità nel simulare risposte realistiche.

7.8 Benefici e Limiti del Prototyping

7.8.1 Benefici

- Permette feedback iterativi e miglioramenti incrementali.
- Supporta la visualizzazione e la comunicazione di idee.
- Identifica problemi e limiti nella fase iniziale.

7.8.2 Limiti

- Richiede risorse (tempo e materiali).
- Non rappresenta completamente le performance reali del sistema.
- Rischio di concentrarsi troppo su dettagli estetici rispetto a quelli funzionali.

7.9 Paper Prototyping

7.9.1 Che cos'è?

- Disegnare a mano su carta per rappresentare l'interfaccia.
- Processo rapido, pronto per essere modificato e rifatto.
- Concentrarsi sull'idea, non sull'estetica.

7.9.2 Cosa Disegnare?

- Elementi base dell'interfaccia: etichette, bottoni, widget.
- Mantenere proporzioni simili a quelle dell'interfaccia reale.
- Evitare di curare troppo l'estetica per risparmiare tempo.

7.9.3 Strumenti e Materiali

- Disegni a mano libera.
- Uso di post-it, trasparenze, fotocopie, ritagli.
- Riutilizzo di parti già create.

7.9.4 Testing con il Paper Prototyping

- Mostrare il prototipo a un potenziale utente.
- Ruoli:
  - **Administrator**: guida il test.
  - **Observer**: annota osservazioni.
  - **Human Computer**: simula il comportamento del sistema.
- Cambiare i foglietti quando l'utente interagisce (clicca/tocca).

7.9.5 Vantaggi del Low-Fidelity Prototyping

- Rapidi ed economici da creare e modificare.
- Feedback immediato dagli utenti.
- Gli utenti si concentrano sul contenuto, non sulla grafica.
- È poco costoso rifare prototipi non validi.

7.9.6 Cosa Permette di Testare?

- Interazione.
- Navigazione.
- Workflow.
- Terminologia.
- Funzionalità.

7.9.7 Miti sul Paper Prototyping

- "Non so disegnare" — Non è necessario; si valuta l'idea, non l'estetica.
- "Non è professionale" — La professionalità consiste nel coinvolgere gli utenti sin dalle prime fasi.
- "Non si può prototipare l'interattività" — La maggior parte delle UI è basata su point&click.

7.10 Tool e Mago di Oz

7.10.1 Paper Prototyping con il supporto del computer

- Fotografare i disegni su carta e importarli in un tool.
- Marcare le zone cliccabili sull'interfaccia.
- Collegare le zone cliccabili ad altri disegni o schermate.
- Questo metodo sostituisce l'**Human Computer**.
- Può registrare automaticamente i test.

7.10.2 POP (by Marvel)

- Uno strumento per prototipazione rapida tramite fotografie.
- Permette di creare prototipi interattivi con immagini dei disegni cartacei.
- Maggiori informazioni e video illustrativo:
  - `https://marvelapp.com/pop`
  - `https://www.youtube.com/watch?v=wSIvYdwxa9c`

7.10.3 Prototipi Mago di Oz

Definizione Un'interfaccia che simula il funzionamento reale mediante l'intervento umano (Wizard).

- Il Wizard è nascosto o remoto.
- Traduce l'input dell'utente in azioni:
  - Inserire testo.
  - Passare alla schermata successiva.
  - Selezionare un punto su una mappa.
- Usata per UI di dialogo naturale, computer vision, interfacce vocali, location-based UI, ecc.

7.10.4 Vantaggi del Mago di Oz

- Applicazione interattiva con poco codice da sviluppare.
- Flessibile e adatta a iterazioni rapide.
- Più realistica del paper prototyping.
- Coinvolge gli utenti nel processo.
- Permette di immaginare applicazioni innovative.
- I designer imparano facendo da Wizard.

7.10.5 Svantaggi del Mago di Oz

- Simulazione non sempre fedele della tecnologia reale.
- La tecnologia simulata potrebbe non essere realizzabile.
- I Wizard richiedono allenamento e possono essere incoerenti.
- Fare il Wizard è stancante.
- Alcune caratteristiche e limitazioni del sistema non si possono simulare.
- Non sempre appropriata in certi contesti o luoghi.

8 Observational Methods

8.1 Valutazione

- Richiede un artefatto: simulazione, prototipo o implementazione completa.
- Testa l'usabilità e la funzionalità del sistema.
- Può avvenire in laboratorio, sul campo o in collaborazione con gli utenti.
- Valuta design e implementazione.
- Deve essere considerata in tutte le fasi del ciclo di vita del design.

8.2 Obiettivi della Valutazione

- Valutare l'estensione della funzionalità del sistema.
- Valutare l'effetto dell'interfaccia sull'utente.
- Identificare problemi specifici.

8.3 Metodi di Osservazione

- Think Aloud.
- Cooperative Evaluation.
- Protocol Analysis.
- Post-task Walkthroughs.

8.3.1 Think Aloud

- L'utente esegue un task e descrive le azioni e i pensieri.
- **Vantaggi**:
  - Semplice e non richiede competenze avanzate.
  - Fornisce insight utili.
  - Mostra come il sistema viene effettivamente utilizzato.
- **Svantaggi**:
  - Soggettivo e selettivo.
  - Descrivere le azioni può alterare la performance.

8.3.2 Cooperative Evaluation

- Variante del Think Aloud.
- L'utente collabora nella valutazione.
- Utente e valutatore possono porre domande reciproche.
- **Vantaggi**:
  - Meno rigida e più facile da utilizzare.
  - L'utente è incoraggiato a criticare il sistema.
  - Possibilità di chiarimenti in tempo reale.

8.3.3 Protocol Analysis

- Protocollo = registrazione della sessione di valutazione.
- Strumenti:
  - Carta e penna: economico ma limitato.
  - Audio: utile per Think Aloud, difficile da sincronizzare.
  - Video: accurato ma intrusivo.
  - Log computerizzati: automatici, ma dati difficili da analizzare.
- La trascrizione di audio/video richiede abilità specifiche.

8.3.4 Post-task Walkthroughs

- La trascrizione della sessione viene rivista con il partecipante.
- Identifica motivazioni e alternative considerate.
- Necessario quando il Think Aloud non è possibile.

8.4 Stili di Valutazione

- **In laboratorio**:
  - Ambiente controllato e attrezzature specialistiche.
  - Manca il contesto reale.
- **Sul campo**:
  - Ambiente naturale e contesto reale.
  - Possibilità di distrazioni e rumori.

8.5 Questioni Etiche

- Test anonimi e aggregazione delle informazioni.
- Ricompense per i partecipanti.
- Uso di comitati etici per approvare i test.

9 Esperimenti

9.1 Valutazione Sperimentale

- Valutazione controllata di specifici aspetti del comportamento interattivo.
- Il valutatore sceglie un'ipotesi da testare misurando il comportamento dei partecipanti.
- Diverse condizioni sperimentali sono considerate, variando solo una variabile controllata.
- Le variazioni nel comportamento sono attribuite alle diverse condizioni.

9.2 Fattori Sperimentali

- **Partecipanti**: campione rappresentativo e sufficiente.
- **Variabili**:
  - **Indipendenti (IV)**: caratteristiche modificate per produrre condizioni differenti (es. stile dell'interfaccia, numero di elementi di menu).
  - **Dipendenti (DV)**: caratteristiche misurate nell'esperimento (es. tempo impiegato, numero di errori).

- **Ipotesi**: predizione dell'effetto della IV sulla DV (es. "il tasso di errore aumenta al diminuire della dimensione del font").
- **Design sperimentale**: come condurre l'esperimento.

9.3 Tipi di Ipotesi

- **Ipotesi alternativa**: predice un effetto della IV sulla DV.
- **Ipotesi nulla**: predice nessuna differenza nella DV; l'obiettivo è confutarla.

9.4 Design Sperimentale

- Scegliere l'ipotesi.
- Definire le variabili indipendenti e dipendenti.
- Selezionare i partecipanti.
- Decidere il metodo sperimentale:
  - **Between subjects**: ogni partecipante esegue una sola condizione.
  - **Within subjects**: ogni partecipante esegue tutte le condizioni.

9.5 Condizioni

- **Condizione di controllo**: situazione base senza modifiche.
- **Condizione sperimentale**: una IV è modificata rispetto alla condizione di controllo.

9.6 Metodi Sperimentali

9.6.1 Between Subjects

- Ogni partecipante esegue una sola condizione.
- **Pro**: non risente del trasferimento dell'apprendimento.
- **Contro**: richiede più partecipanti; differenze individuali possono alterare i risultati.

9.6.2 Within Subjects

- Ogni partecipante esegue tutte le condizioni.
- **Pro**: meno partecipanti richiesti; differenze individuali hanno minore impatto.
- **Contro**: rischio di trasferimento dell'apprendimento.

9.7 Analisi dei Dati

- Osservare e graficare i dati raccolti.
- Identificare outliers.
- Conservare i dati originali per ulteriori analisi.
- Classificazione delle variabili:
  - Variabili discrete o continue.
  - Variabili continue positive (es. tempo impiegato).

9.8 Analisi Statistica

- **Test parametrici**: usati se la distribuzione è normale.
- **Test non parametrici**: usati in caso contrario.
- Verifica delle ipotesi: accettazione o rifiuto dell'ipotesi nulla.

10 Expert-based Evaluation

10.1 Valutazioni senza utenti

- Gli esperti valutano il design rispetto all'impatto sull'utente tipico.
- Obiettivo: identificare problemi che:
  - Violano principi cognitivi noti.
  - Ignorano risultati empirici consolidati.

10.1.1 Vantaggi e svantaggi

- **Vantaggi**:
  - Economiche.
  - Applicabili in qualsiasi fase del progetto.
- **Svantaggi**:
  - Non valutano l'uso reale del sistema.
  - Misurano l'aderenza a principi noti.

10.2 Tipi di valutazione Expert-based

- **Cognitive Walkthrough**:
  - Valuta quanto il design supporta l'apprendimento dei task.
  - Esperti seguono il design identificando problemi usando principi psicologici.
- **Heuristic Evaluation**:
  - Usa linee guida o principi generali per identificare problemi di usabilità.
  - Dieci euristiche di Nielsen per analizzare violazioni e severità.
- **Model-based Evaluation**:
  - Usa modelli teorici come:
    - **GOMS**: prevede performance degli utenti.
    - **KLM**: stima i tempi necessari per compiti fisici (tastiera e mouse).
    - **Dialog Models**: valuta sequenze di dialoghi, stati irraggiungibili, dialoghi circolari.
- **Review-based Evaluation**:
  - Usa risultati di studi precedenti per confermare o rifiutare parti di design.
  - Richiede esperti per assicurare corrette ipotesi.

10.3 Esempio: Zoom Screen Sharing

10.3.1 Cognitive Walkthrough

- Obiettivo: abilitare la condivisione dello schermo per i partecipanti.
- Problemi identificati:
  - Icona poco visibile.
  - Opzioni avanzate non evidenti.
  - Linguaggio poco intuitivo.
- Soluzioni:
  - Rendere visibili le opzioni avanzate.
  - Rinominare "Advanced Sharing Options" in "Manage Screen Sharing".
  - Aggiungere notifiche sull'impostazione predefinita.

10.3.2 Heuristic Evaluation

- Dieci euristiche applicate per valutare problemi.
- Soluzioni:
  - Migliorare visibilità del sistema.
  - Prevenire errori tramite notifiche.
  - Fornire documentazione e messaggi chiari.

11 Progetto e sviluppo di sistemi interattivi II

11.1 Metodologie di Sviluppo

Il progetto e sviluppo di sistemi interattivi utilizza due metodologie principali:

- **Waterfall**: modello a cascata.
- **Agile**: approccio iterativo e incrementale.

11.2 Waterfall Model

- Modello lineare e sequenziale.
- Ogni fase deve essere completata prima di passare alla successiva.
- **Fasi**:
  1. Requisiti.
  2. Design.
  3. Implementazione.
  4. Test.
  5. Deployment.
  6. Manutenzione.
- **Vantaggi**:
  - Struttura chiara e ben definita.
  - Facilità di gestione.
  - Adatto per progetti con requisiti ben definiti.
- **Svantaggi**:
  - Difficoltà a gestire modifiche.
  - Problemi scoperti tardi nel ciclo.
  - Poco flessibile.

11.3 Agile Methodology

- Modello iterativo e incrementale.
- Basato su brevi cicli chiamati **sprint**.
- **Principi chiave**:
  - Collaborazione continua con il cliente.
  - Rispondere al cambiamento piuttosto che seguire un piano.
  - Consegna frequente di software funzionante.
  - Focus sugli individui e le interazioni.

- **Framework Agile comuni**:
  - Scrum.
  - Kanban.
  - Extreme Programming (XP).
- **Vantaggi**:
  - Maggiore flessibilità e adattabilità.
  - Coinvolgimento continuo del cliente.
  - Riduzione dei rischi.
- **Svantaggi**:
  - Richiede team esperti e autonomi.
  - Può essere difficile da pianificare.

11.4 Confronto tra Waterfall e Agile

- **Flessibilità**:
  - Waterfall: poco flessibile.
  - Agile: altamente flessibile.
- **Adattabilità**:
  - Waterfall: adatto per requisiti stabili.
  - Agile: adatto per requisiti in evoluzione.
- **Consegna**:
  - Waterfall: consegna finale.
  - Agile: consegna incrementale.

12 Agile UCD (User-Centered Design)

12.1 Definizione di Agile UCD

Agile UCD integra i principi dell'Agile con il Design Centrado sull'Utente (*User-Centered Design*), focalizzandosi su:

- **Coinvolgimento continuo dell'utente**: utenti finali partecipano attivamente nel ciclo di sviluppo.
- **Iterazioni frequenti**: validazione continua attraverso prototipi e feedback.
- **Team collaborativi**: progettisti, sviluppatori e utenti lavorano insieme.

12.2 Caratteristiche principali

- **Iterazioni rapide**: ogni sprint produce un incremento utilizzabile.
- **Prototipi usabili**: evoluzione continua basata sui test utente.
- **User Stories**: descrivono funzionalità dal punto di vista dell'utente.
- **Definizione del Done (DoD)**:
  - Include test di usabilità.
  - Garantisce che le funzionalità siano pronte per l'utente finale.

12.3 Ciclo di vita Agile UCD

1. **Identificazione dei bisogni utente**: interviste, osservazioni e brainstorming.
2. **Creazione di Personas**: modelli rappresentativi degli utenti target.
3. **Prototipazione**: prototipi low-fidelity per iniziare, evolvendo verso quelli ad alta fedeltà.
4. **Test utente**: iterazioni basate sul feedback ricevuto.
5. **Sviluppo incrementale**: realizzazione di funzionalità complete per ogni sprint.

12.4 Vantaggi di Agile UCD

- **Adattabilità**: risponde rapidamente a cambiamenti nei requisiti.
- **Riduzione dei rischi**: identifica problemi di usabilità durante lo sviluppo.
- **Coinvolgimento utente**: miglior allineamento con i bisogni reali.

12.5 Sfide di Agile UCD

- **Tempo limitato per i test**: gli sprint brevi possono comprimere le attività di valutazione.
- **Conflitti di priorità**: bilanciare esigenze di sviluppo e di design.
- **Competenze del team**: richiede conoscenze sia in Agile che in UCD.

13 Affordances e Signifiers

13.1 Definizione di Affordances (Norman)

- **Affordance**: relazione tra le proprietà di un oggetto e le capacità di un agente che determinano come l'oggetto può essere usato.
- La forma, dimensione e aspetto dell'oggetto suggeriscono cosa un agente può fare con esso.
- L'esistenza di un'affordance dipende dall'oggetto e dall'agente (se l'agente può percepire come usare l'oggetto).

13.1.1 Esempi di Affordances

- Porte: capire se possono essere spinte o tirate.
- Oggetti quotidiani come maniglie, lavatrici, telefoni.

13.2 Definizione di Signifiers (Norman)

- **Signifier**: qualsiasi segno, suono o indicatore percepibile che comunica un comportamento appropriato a una persona.
- Guida le azioni dell'utente fornendo indicazioni chiare su cosa fare con l'oggetto (esempi: etichette, frecce, simboli, o altri segnali visivi).
- Rende più esplicite le azioni previste, specialmente quando le affordances non sono immediatamente evidenti.

13.2.1 Esempi di Signifiers

- Frecce su una porta che indicano se spingere o tirare.
- Indicatori visivi o auditivi che spiegano come interagire con un oggetto.

13.3 Affordances vs Signifiers

- **Affordances**:
  - Presenza di affordances: l'oggetto suggerisce direttamente come può essere utilizzato.
  - Mancanza di affordances: il significato non è percepibile senza istruzioni.
- **Signifiers**:
  - Presentano indicazioni esplicite su come utilizzare un oggetto.
  - Più importanti delle affordances nell'interazione con un sistema.

13.4 Matrice di Affordances e Signifiers

Affordance	**Signifier Presente**	**Signifier Assente**
Presente	Perceived Affordability	Hidden Affordability
Assente	False Affordability	Affordability Correctly Rejected

Tabella 1: Relazione tra Affordances e Signifiers

14 Modi e Gestione dell'Interazione

14.1 Terminologia

- **Content**: insieme di informazioni significative e utili per l'utente.
- **GID (Graphical Input Device)**: meccanismo per comunicare al sistema una posizione o una scelta.
- **GID Button**: bottone principale del GID.
- **Tap**: azione di premere e rilasciare un tasto.
- **Click**: posizionare il GID e fare un tap sul GID Button.
- **Double-click**: due click consecutivi sul GID senza spostamento.
- **Drag/Swipe**: premere il GID Button, muoverlo e rilasciarlo in un'altra posizione.
- **Gesture**: sequenza di azioni completata automaticamente (es. premere "Invio").

14.2 Modi

- Un'interfaccia è **modale** se l'interpretazione di un gesto dipende dallo stato del sistema.
- Problemi dei modi:
  - Rendono difficile prevedere il comportamento dell'interfaccia.
  - Necessitano di una verifica dello stato del sistema.
- Esempi:
  - **Torcia**: accendere o spegnere dipende dallo stato non visibile.
  - **Caps Lock**: crea un modo spesso fonte di errori.

14.3 Minimizzare gli Errori di Modo

- Non utilizzare modi, quando possibile.
- Marcare chiaramente i modi attivi.
- Differenziare i comandi in base ai modi per evitare effetti indesiderati.

14.4 Tipi di Modi

- **Modo Temporaneo**: svanisce dopo l'uso (es. pennello di Word).
- **Quasi-modo**: richiede il mantenimento di un controllo fisico (es. tasto CTRL).

14.5 Interazione Noun-Verb e Verb-Noun

- **Noun-Verb (Oggetto-Azione)**: si seleziona prima l'oggetto, poi l'azione.
- **Verb-Noun (Azione-Oggetto)**: si seleziona prima l'azione, poi l'oggetto.

14.6 Pro e Contro di Noun-Verb e Verb-Noun

- **Verb-Noun**:
  - Focus sull'azione iniziale.
  - Rischio di errori di modo in caso di distrazione.
- **Noun-Verb**:
  - Focus iniziale sull'oggetto.
  - Minore rischio di errori, poiché l'azione è il passo finale.

15 Progetto di Applicazioni Mobili

15.1 Contesto d'uso

- Gli utenti di dispositivi mobili spesso sono in movimento e non possono concentrarsi a lungo.
- Eseguono compiti nei ritagli di tempo, con fretta e poco tempo a disposizione.
- Sono frequentemente interrotti da eventi esterni, anche asincroni.
- A volte necessitano di non disturbare altri (es. in cinema o teatro).

15.2 Attenzione

- Gli utenti potrebbero dover dividere l'attenzione con altre attività (es. guida).
- Vista, udito e mani possono essere impegnati.
- Condizioni ambientali variabili (scarsa illuminazione, rumori).

15.3 Differenze rispetto al Desktop

15.3.1 Schermo dello smartphone

- Piccole dimensioni (5"-7") ma alta risoluzione (¿1Mpixel).
- Leggibilità è cruciale; evitare elementi non necessari.

15.3.2 Memoria limitata

- Eliminare memory leaks.
- Ridurre le dimensioni dei file al minimo indispensabile.

15.3.3 Una schermata alla volta

- Non ci sono finestre multiple.
- Accesso sequenziale alle schermate.
- Porting di applicazioni desktop richiede:
  - Riorganizzazione su schermate sequenziali.
  - Porting di subtask selezionati.

15.3.4 Un'applicazione alla volta

- No multitasking per app di terze parti.
- Telefonate interrompono le applicazioni correnti.
- Le app devono:
  - Conservare dati e stato.
  - Ripartire rapidamente dallo stato precedente.
- Le impostazioni esterne all'app interrompono l'uso.

15.3.5 Help minimale

- Gli utenti mobili non leggono l'help.
- L'help non deve occupare spazio sullo schermo.
- Usare controlli standard e sequenze logiche delle schermate.
- Includere bottoni "Back" per tornare indietro.

16 Stili di Applicazioni Mobili

16.1 Productivity Application

- Permette di svolgere compiti basati sull'organizzazione e manipolazione di informazioni dettagliate.
- Esempi di task:
  - Gestire mail.
  - Organizzare liste.
  - Modificare elementi su livelli di dettaglio.
- **Caratteristiche**:
  - Informazioni strutturate gerarchicamente.
  - Interfaccia semplice e pulita.

  - Uso di controlli standard.
  - Preferenze per ridurre scelte ripetitive.
- Esempio: Google Photo.

16.2 Utility Application

- Task semplice con poco input dall'utente.
- Esempi di task:
  - Leggere previsioni meteo.
  - Verificare lo stato di un dispositivo.
- **Caratteristiche**:
  - Zero struttura gerarchica.
  - Molta informazione, poca interazione.
  - Liste non collegate gerarchicamente.
  - Cambi frequenti di preferenze dall'interno dell'app.
- Esempio: app per previsioni meteo.

16.3 Immersive Application

- Per compiti che presentano un ambiente particolare e richiedono attenzione continua.
- Esempi di task:
  - Esperienze interattive come videogiochi.
  - Ambienti immersivi come IKEA Place.
- **Caratteristiche**:
  - Ambienti ricchi a tutto schermo.
  - Focalizzate sul contenuto e sull'esperienza.
  - Non utilizzano controlli standard ma ne creano di propri.
  - Scoprire il funzionamento dei controlli è parte dell'esperienza.

17 Productivity App e Modi

17.1 Caratteristiche delle App di Produttività

- Focus sull'esecuzione di task specifici.
- Esempi:
  - Mail.

  - Social.
  - Home banking.
- Task tipici:
  - Creare e manipolare oggetti come post, messaggi, contatti, chiamate telefoniche, movimenti bancari.

17.2 Approccio tipico

- Uso di un elenco di oggetti organizzati in livelli gerarchici (albero n-ario).
- Navigazione gerarchica attraverso le schermate.

17.3 Esempio: App Mail (Gmail)

- Lista conversazioni.
- Tap su una conversazione per vedere le mail in una nuova vista (navigazione nell'albero).
- Swipe per azioni rapide (archivia, segna come letto/non letto).
- Tap sul bottone "penna" per creare una nuova conversazione.

17.4 Approccio Noun-Verb

- L'utente seleziona prima un oggetto e poi esegue l'azione:
  - Swipe per archiviare una conversazione.
  - Tap su un pulsante per eliminare una conversazione.
  - Tap su una conversazione per aprirla.
- L'animazione dell'oggetto conferma il successo dell'azione.

17.5 Vantaggi di Noun-Verb

- Riduce errori di modo.
- Adatto per utenti con attenzione ridotta o frequenti interruzioni.
- Evita di richiedere all'utente di conoscere o ricordare lo stato del sistema.

17.6 Animazione delle View

- La vista figlia entra da destra e si sovrappone alla vista padre.
- Bottone "Back" in alto a sinistra per tornare alla vista precedente.
- Le viste si comportano come una pila (LIFO).

17.7 Aggiunta di un Oggetto: Verb-Noun

- L'utente sceglie prima un'azione, poi agisce sull'oggetto.
- Esempio: Premere "scrivi" (icona penna) per iniziare una nuova email.
- Apertura di una vista modale per la scrittura.
- Al termine, la vista modale si chiude, mostrando l'aggiornamento nella vista precedente.
- L'animazione della lista rappresenta un feedback implicito.

17.8 Vista di Default

- La vista iniziale corrisponde alla radice dell'albero.
- Benefici:
  - Rapidità.
  - Semplicità d'uso.
  - Minor numero di errori.
- Esempi: Gmail, WhatsApp, Telegram, Google Drive, BancoPosta.

18 Design Patterns

18.1 Introduzione ai Design Patterns

- Originariamente usati in architettura da Christopher Alexander.
- Definizione: "Ogni pattern descrive un problema ricorrente e una soluzione che può essere riutilizzata in contesti simili, senza essere mai identica."
- I design patterns sono soluzioni ricorrenti a problemi comuni nel design.

18.2 Caratteristiche dei Design Patterns

- Non sono troppo generali né troppo specifici.
- Sono un linguaggio condiviso per designer e sviluppatori.
- Permettono discussioni e confronto su alternative.

18.3 UI Design Patterns

- Consentono di accelerare la comprensione dell'interfaccia da parte degli utenti.
- Alcuni esempi comuni:
  - **Accordion Menu**
  - **Dropdown Menu**

  - **Cards**
  - **Breadcrumbs**
  - **Hamburger Menu**

18.4 Design Patterns in App Mobile (Android)

- **Toolbar/App Bar**: gestisce azioni principali.
- **Tabs**: organizza contenuti in categorie.
- **Navigation Drawer**: menu laterale per navigazione.
- **Scrolling e Paging**: migliora la navigazione tra contenuti.

18.5 Dark Patterns

- Design ingannevoli contro l'interesse degli utenti.
- Coniati nel 2010 da Harry Brignull.
- Obiettivo: influenzare scelte non ottimali per l'utente.
- Esempi: infinite scrolling, autoplay, refresh compulsivi.

18.6 Attention-Capture Damaging Patterns (ACDPs)

- Sfruttano vulnerabilità psicologiche per catturare l'attenzione.
- Tecniche comuni:
  - Ricompensa variabile.
  - Gratificazione immediata.
- Esempi:
  - **Neverending Autoplay**: riproduzione automatica senza fine.
  - **Casino Pull-to-Refresh**: comportamento simile a una slot machine.
  - **Infinite Scrolling**: caricamento continuo di contenuti.
- Effetti negativi:
  - Promuovono la dipendenza digitale.
  - Riduzione dell'autonomia e produttività.

19 Interfacce Android

19.1 Principi di Base

- Le interfacce Android si basano su principi di usabilità consolidati, come quelli definiti da Material Design.
- Puntano a migliorare la **scopribilità** e garantire un'esperienza utente coerente e intuitiva.

19.2 Leverage on Users' Knowledge

- Utilizzo di **standard widgets** che gli utenti conoscono già.
- Integrazione di **standard gestures** per azioni comuni (es. swipe, tap, drag).
- Uso di **metafore** per rendere intuitive interazioni complesse.

19.3 Scopribilità

- Non nascondere controlli essenziali.
- Fornire segnali visibili (signifiers) che guidino l'utente verso le azioni disponibili.

19.4 Azione e Reazione

- **Direct Manipulation**: permettere agli utenti di manipolare direttamente gli elementi.
- **Feedback**: fornire una risposta visiva o tattile immediata per ogni azione.

19.5 Memoria

- Preferire la selezione rispetto al ricordo: presentare opzioni visibili invece di richiedere input da memoria.

19.6 Controllo dell'Utente

- Informare gli utenti su "Dove sono?" (navigazione contestuale).
- Garantire navigazione inversa (*Back*) e progressi visibili (es. **progress bar**).
- Permettere di annullare azioni (**Undo**) o di personalizzare l'esperienza (**Customize**).

19.7 Coerenza

- Stesso gesto, stessa azione (**Same Gesture, Same Action**).
- Evitare modalità multiple e confusione nei comandi (**Avoid Modes**).
- Stessa funzione, stesso widget (**Same Function, Same Widget**).
- Utilizzo di temi coerenti (**Themes**) per tutte le schermate.

19.8 Material Design

- Standard di progettazione per interfacce Android sviluppato da Google.
- Obiettivo: fornire un linguaggio visivo unificato e coerente tra dispositivi e piattaforme.
- Principi fondamentali:

- **Materiale come metafora**: utilizzo di ombre, profondità e movimenti.
- **Design Bold e Grafico**: enfasi su colori vivaci, tipografia chiara e layout semplice.
- **Movimenti significativi**: animazioni utili per guidare l'utente.

19.9 Lists e Cards

19.9.1 Lists

- Organizzano elementi multipli in un'unica struttura verticale.
- Usate per rappresentare contenuti correlati.

19.9.2 Cards

- Elementi visivi che racchiudono dati correlati.
- Simili a fogli di carta: contenuti unici e pertinenti.

19.10 Alerts e Bottom Sheets

19.10.1 Alerts

- Informano l'utente su situazioni critiche.
- Richiedono all'utente di prendere decisioni.
- Generati dal sistema.

19.10.2 Bottom Sheets

- Presentano un set di azioni o opzioni.
- Sollecitati dall'utente.

19.11 Componenti principali

- **Cards**: Mostrano contenuti e azioni su un argomento specifico.
  - Utilizzabili come punti di ingresso per navigazioni più profonde.
  - Possono contenere titoli, testo, immagini, icone e bottoni.
  - Organizzabili in griglie, liste verticali o caroselli.
- **Lists**: Indici verticali continui di testo e immagini.
  - Tre formati: una riga, due righe, tre righe.
  - Ordinabili logicamente (alfabetico o numerico).
- **Buttons**: Comunicazione di azioni possibili.
  - Forme arrotondate, abbastanza ampie da contenere il testo.

  - Icone opzionali posizionate prima del testo.

- **Floating Action Buttons (FAB)**:
  - Rappresentano l'azione più comune o importante.
  - Posizionati sopra la barra di navigazione o all'interno.

19.12 Sheets

- **Bottom Sheets**: Contenuti supplementari e azioni.
  - Tipi: standard e modal.
- **Side Sheets**: Mostrano contenuti secondari ancorati al lato dello schermo.

19.13 Navigazione e Barre

- **Tabs**: Organizzano contenuti su schermate e viste diverse.
- **Bottom App Bars**: Navigazione e azioni principali.
- **Top App Bar**: Titoli e azioni relative alla schermata corrente.
- **Navigation Bars**: 3-5 destinazioni principali accessibili ovunque nell'app.

19.14 Transizioni

- **Forward e Backward**: Navigazione gerarchica.
- **Lateral**: Navigazione tra contenuti correlati, come i tab di una libreria media.

19.15 Sistema cromatico di Material Design 3 (M3)

- M3 definisce, assegna e personalizza i colori per interfacce coerenti e accessibili.
- Sistema cromatico basato su algoritmi per generare palette coerenti da un **seed color**.
- Temi di colore applicabili anche su iOS.

19.16 Componenti del Sistema Cromatico

- **Color Roles**: etichette che descrivono l'uso dei colori (es. pulsanti, sfondi).
- **Assegnazione Dinamica**: ogni elemento è associato a un ruolo che determina il colore assegnato.
- **Algoritmo della Palette**: genera tonalità e livelli di luminosità per ciascun key color.

19.17 Principali Ruoli di Colore

- **Primary**: elementi principali interattivi.
- **Secondary**: accenti visivi e complementi.
- **Tertiary**: variazioni stilistiche opzionali.
- **Background**: sfondo dell'interfaccia.
- **Surface**: superfici come card o finestre.
- **Error**: stati di errore o avvisi.
- **Outline**: bordature o contorni di elementi.
- **On-roles**: testi o icone sovrapposti a sfondi specifici (es. on-primary, on-background).

19.18 Generazione della Palette di Tonalità

- **Seed Color**: colore di riferimento da cui generare le tonalità.
- **5 Key Color Complementari**: derivati dal seed color.
- Ogni key color ha una palette con 13 livelli di luminosità.
- Tonalità chiare per sfondi (modalità chiara), scure per testi (modalità scura).

19.19 Creazione di Temi con Figma

- Usare il **Material Theme Builder Plugin**.
- Inserire il seed color per generare automaticamente la palette e assegnare i color roles.
- Supporta temi chiari, scuri e ad alto contrasto.

19.20 Uso con M3 Design Kit in Figma

- Utilizzare componenti predefiniti con color roles già assegnati.
- Scegliere un seed color rappresentativo del brand.
- Applicare e aggiornare automaticamente il tema selezionato.

19.21 Riferimenti Utili

- `https://m3.material.io/`
- `https://m3.material.io/styles/color/system/overview`