

Interazione Uomo Macchina



Sommario

1. Introduzione	3
1.1 Interazione Uomo Macchina	3
1.2 Le Interfacce Utente	3
1.3 Requisiti di Usabilità	4
1.4 Ingegnerizzazione dell’interfaccia utente.....	4
1.5 Modelli mentali.....	5
1.6 I Framework dell’interazione.....	6
1.7 Capacità Cognitive	7
2. Stili dell’Interazione	10
2.1 Stili dell’Interazione	10
2.2 Interfacce Wimp	10
3. Paradigmi di Interazione e Principi di Usabilità.....	12
3.1 Introduzione	12
3.2 Paradigmi per l’usabilità	12
3.3 Antropomorfismo	15
4. Le Personas	16
4.1 Regole di Design	16
4.2 Analisi degli stakeholder.....	16
4.3 I profili utente – “personas”	16
4.4 Metodi di raccolta dati	19
4.5 Studi pilota.....	19
4.6 Task Analysis (<i>Analisi dei compiti</i>)	20
4.7 Empowerment (<i>Potenziamento personale</i>).....	21
5. Regole di Design	22
5.1 Introduzione	22
5.2 Principi di Usabilità	22
5.3 Standard ed Euristiche.....	24
5.4 Linee Guida	25
5.5 Regole d’oro ed Euristiche.....	25
5.6 Design Pattern	27

6. Regole di Design	28
6.1 Prototipi	28
6.2 Tecniche di prototipazione	28
6.3 Fedeltà nei prototipi	30
6.4 Compromessi nella prototipazione	31
6.5 Design concettuale	31
7. La collaborazione	32
7.1 Cos'è la collaborazione	32
7.2 Classificazione per funzione	32
7.3 L'integrazione della comunicazione e del lavoro	33
8. I Pattern	35
8.1 Introduzione	35
8.2 Design Pattern Vs. Linee guida di design	35
8.3 Design Pattern HCI.....	36
9. Paradigmi e tecniche di valutazione dell'usabilità.....	38
9.1 Valutazione dell'usabilità.....	38
9.2 Valutazione del design.....	38
10. Accessibilità e Usabilità.....	43
10.1 Concetto di accessibilità.....	43
10.2 Tecnologia assistiva.....	44
10.3 La legge Stanca.....	44
10.4 Il W3C e la Web Accessibility Iniziative (WAI)	44
10.5 Relazione tra accessibilità e usabilità.....	45

Capitolo 1 – Introduzione

1.1 Interazione Uomo Macchina

L'interazione uomo macchina si occupa della **progettazione**, **valutazione** e **implementazione** di sistemi di calcolo interattivo per uso umano e dello studio dei principali fenomeni che li circondano.

È una materia interdisciplinare poiché unisce la:

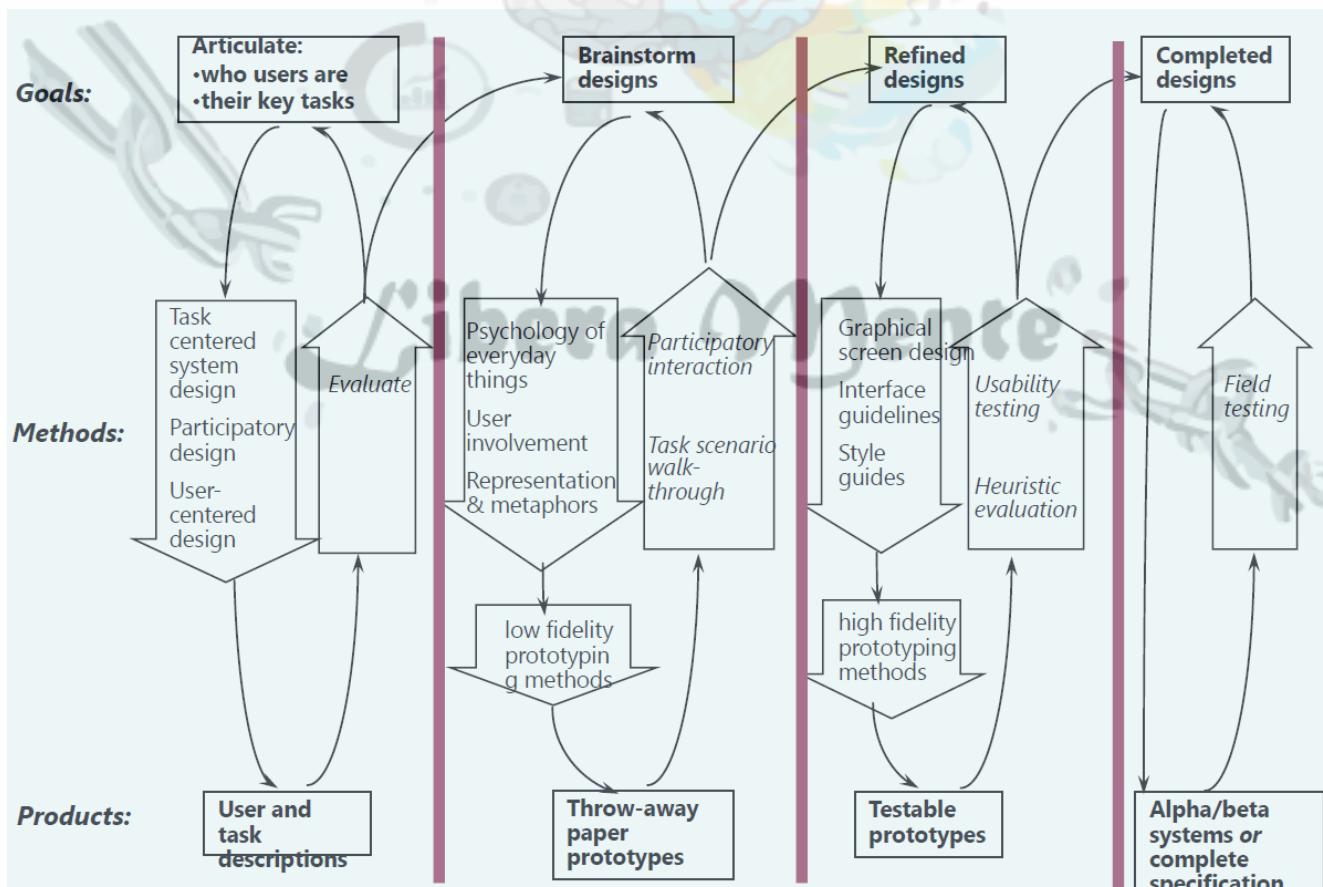
- Psicologia cognitiva;
- Informatica;
- Psicologia sociale organizzativa;
- Ergonomia e fattori umani.

Inoltre, l'*interazione uomo macchina* è una **disciplina rigorosa**. Innanzitutto, partiremo dal **design** che si basa:

- sulla comprensione dei vincoli;
- sullo studio dello spazio di progettazione;
- su una profonda conoscenza del materiale di progetto, cioè l'utente, il task e la macchina.

Quindi, parleremo da una parte di **user-centered design** e dall'altra di **task-centered design**.

L'obiettivo è di capire come avviene l'interazione e lo sviluppo di **prodotti usabili**. Informalmente, **"usabile"** significa facile da apprendere, efficace e pratico da usare, e il cui uso fornisce un'esperienza piacevole agli utenti.



1.2 Le Interfacce Utente

Le **interfacce utente** pongono la loro attenzione a livello del singolo utente. Naturalmente, in grandi aziende, dove ci sono diverse figure coinvolte nel design dell'interazione, diventa più difficile comunicare e progredire nel processo di design.

1.3 Requisiti di Usabilità

Il termine “**user-friendly**” sta ad indicare: “*facile da usare*”, “*accessibile*”, “*comprendibile*”, “*intellegibile*”, etc. Ma un “*friend*” cerca anche di aiutare ed essere prezioso. Pertanto, è necessario un processo sistematico per sviluppare sistemi **usabili** per un utente specifico in uno specifico contesto.

L’usabilità richiede il **project management** e un’attenta **cura dell’analisi dei requisiti**, con un testing che miri ad avere obiettivi ben definiti.

1.4 Ingegnerizzazione dell’interfaccia utente

Tra gli obiettivi dell’**analisi dei requisiti** abbiamo:

1. Stabilire le necessità dell’utente e analisi dei task per assicurare adeguate funzionalità.

Ciò vuol dire:

- definire quali task e sotto-task devono essere eseguiti;
- includere anche task che devono essere eseguiti occasionalmente. I task comuni sono facili da identificare;
- le funzionalità dovranno riflettere le necessità altrimenti gli utenti rifiuteranno o sottoutilizzeranno il prodotto.

2. Affidabilità, Disponibilità, Sicurezza e Integrità dei Dati:

- Le azioni devono funzionare come specificato;
- I dati visualizzati devono riflettere il reale contenuto del database sottostante;
- Mitigare il senso di sfiducia da parte dell’utente;
- Il sistema dovrebbe essere disponibile più frequentemente possibile;
- Il sistema non dovrebbe introdurre errori;
- Assicurare la privacy dell’utente e la sicurezza dei dati, proteggendolo contro accessi non certificati, distruzione di dati e interferenze dolose;

3. Promuovere la:

- **Standardizzazione**: usare standard industriali preesistenti laddove esistono, per aiutare nell’apprendimento ed evitare errori (es. gli standard W3C e ISO).
- **Integrazione**: il prodotto dovrebbe essere eseguito all’interno di diversi strumenti e pacchetti software.
- **Consistenza**: compatibilità tra diverse versioni del prodotto; compatibilità con sistemi su carta e non basati su computer correlati al sistema che si intende progettare; uso di comuni sequenze di azioni, termini, unità, colori ecc. all’interno del programma.
- **Portabilità**: consentire all’utente di convertire dati tra diversi ambienti software e hardware.

4. Pianificazione e Budget:

- Completare i progetti in tempo e rientrando nel budget. Infatti, i prodotti consegnati in ritardo o che hanno sforato il budget iniziale possono nascondere una serie di pressioni all’interno di un’azienda e possono significare un cliente non soddisfatto e una perdita dell’affare.

Tra gli obiettivi della **progettazione dell’Interfaccia Sistema-Utente** abbiamo la definizione della comunità utente ‘target’ associata all’interfaccia. Le comunità evolvono e cambiano e vengono valutate tramite 5 fattori umani centrali:

1. **Tempo di apprendimento**: quanto tempo impiega tipicamente un membro della comunità per imparare task rilevanti?
2. **Velocità di esecuzione**: qual è il tempo di esecuzione dei benchmark rilevanti?
3. **Quantità di errori da parte degli utenti**: quanti e che tipo di errori vengono comunemente commessi mentre si eseguono i task dei benchmark?
4. **Ritenzione nel tempo**: la frequenza d'uso e la facilità di apprendimento favoriscono una migliore ritenzione da parte dell'utente.
5. **Soddisfazione soggettiva**: consentire il feedback da parte dell'utente tramite interviste, commenti liberi, e scale di soddisfazione.

A volte ci devono essere dei **compromessi** nello sviluppo, usare strumenti come macro e abbreviazioni per facilitare alcuni carichi. Infatti, le modifiche all'interfaccia in una nuova versione possono creare problemi di inconsistenza ma i cambiamenti potrebbero introdurre nuove funzionalità necessarie.

Inoltre, bisogna testare tutte le alternative di design usando un ampio range di **modelli dimostrativi**. Le alternative possono essere valutate dai progettisti e dagli utenti tramite dei mockup o prototipi *high-fidelity*. Un compromesso fondamentale è ottenere feedback nel processo di sviluppo e in modo meno costoso piuttosto che avere una interfaccia autentica da valutare.

1.5 Modelli mentali

Le persone hanno un **modello mentale** di come funzionano le cose:

- *come si avvia un'auto?*
- *come funziona uno sportello bancomat?*
- *come parte un computer?*

Tale *modello* consente alle persone di fare previsioni su come funzioneranno le cose. I **modelli mentali** sono costruiti a partire da:

- **Affordances**: è la relazione fra l'oggetto, per come appare, e come esso possa essere utilizzato. Secondo Norman “*Le affordance percepite ci aiutano a indovinare quali azioni siano possibili, senza bisogno di cartelli o istruzioni*”.

Nell'**interactive design** l'affordance rappresenta la prima regola fondamentale: essere intuitivi, cioè far capire l'interfaccia al primo sguardo senza per forza esplicitare l'usabilità con informazioni maggiori (etichette, testi, CallToAction).

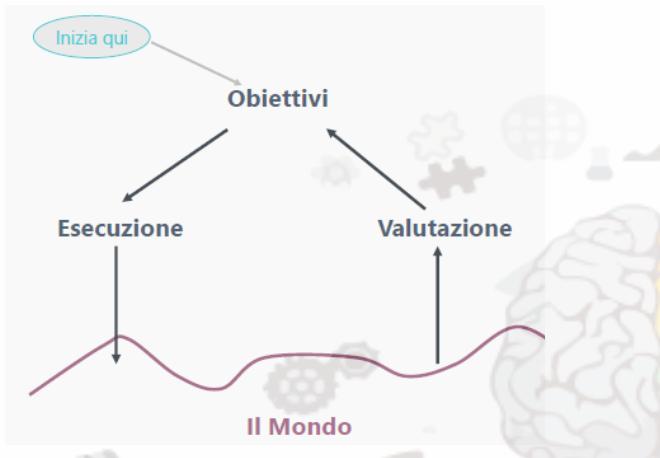
- **Vincoli**: possono essere **fisici** (come la dimensione dello schermo) o **logici** (come la trasparenza di un'icona).
- **Mappings**: si riferisce alla progettazione di controlli, in modo tale che riflettano il modo in cui si comportano, o i risultati ottenuti usandoli.
- **Positive transfer** (conoscenze acquisite in situazioni analoghe);
- **Associazioni / Standard culturali**;
- **Istruzioni**;
- **Interazioni**.

1.6 I Framework dell'interazione

L'**interazione** è la comunicazione tra l'utente e il sistema e grazie ai **framework** abbiamo la contestualizzazione, cioè una vista globale, di questo dialogo tra utente e sistema.

Il **ciclo di esecuzione e valutazione di Norman** è un modello basato su azioni. Secondo questo modello l'utente formula un piano d'azione che poi viene eseguito tramite l'interfaccia del sistema.

Quando il piano è stato realizzato, l'utente osserva l'interfaccia per valutare il nuovo stato del sistema e stabilisce ulteriori azioni. Il *modello di Norman* si concentra solo sulla vista della interfaccia da parte dell'utente.



L'**esecuzione** ha tre stadi:

1. Inizia con un obiettivo;
2. Traduce l'obiettivo in un'intenzione;
3. Traduce l'intenzione in una sequenza di azioni.

La **valutazione** ha tre stadi:

1. Percepisce il mondo;
2. Interpreta ciò che ha percepito;
3. Confronta le interpretazioni con le intenzioni di partenza.

Questo ciclo interattivo può essere diviso, quindi, in due fasi principali: **esecuzione** e **valutazione**, che, a loro volta, possono essere suddivise in altre 7 fasi:

1. l'utente stabilisce un obiettivo;
2. formula l'intenzione (come fare);
3. specifica la sequenza di azioni sull'interfaccia;
4. esegue l'azione;
5. percepisce il nuovo stato del Sistema;
6. interpreta lo stato del sistema;
7. valuta lo stato del sistema rispetto all'obiettivo.

Golfo dell'esecuzione

Il **golfo dell'esecuzione** è la formulazione delle azioni (le intenzioni) da parte dell'utente che può essere diversa dalle azioni messe a disposizione dal Sistema. In altri termini, il *golfo dell'esecuzione* è la differenza fra la formulazione delle azioni da parte dell'utente e le azioni consentite dal sistema. Solo se le azioni previste dal sistema corrispondono a quelle previste dall'utente, l'interazione sarà efficace.

Golfo della valutazione

Il **golfo della valutazione** riguarda le aspettative dell'utente relative allo stato del sistema modificato che possono essere diverse dalla reale presentazione di questo stato.

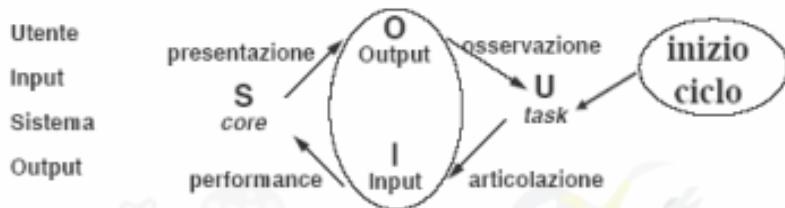
In altri termini, il *golfo della valutazione* calcola lo sforzo richiesto da parte dell'utente per interpretare lo stato fisico del sistema e fino a che punto le aspettative e le intenzioni sono state soddisfatte.

Abowd e Beale

Tentano una descrizione più realistica dell'interazione uomo-macchina, includendo esplicitamente il sistema.

I **nodi** rappresentano i 4 componenti principali: **Utente**, **Input**, **Sistema** ed **Output**.

Ogni componente ha il proprio linguaggio e l'**interazione** consiste in un processo di traduzione da un linguaggio ad un altro. Si avranno **problemi d'interazione** quando la traduzione da un linguaggio all'altro è difficile o impossibile.



Questi sono *framework generali* per capire l'**interazione**:

- non sono ristretti ai computer system;
- identificano tutte le principali componenti dell'interazione;
- permettono un assestamento comparativo dei sistemi;
- sono un'astrazione.

1.7 Capacità Cognitive

L'**Ingegneria cognitiva** o **Cognetica** è lo studio delle capacità mentali umane.

La **cognetica** è strettamente legata all'**ergonomia** che riguarda lo studio delle caratteristiche fisiche dell'interazione.

Nell'ambito psicologico un *processo cognitivo* è una sequenza di eventi necessari alla formazione di un contenuto di conoscenze attraverso l'attività mentale.

Un'**interfaccia** è il modo in cui si fa qualcosa con uno strumento, le azioni che si devono eseguire e il modo in cui lo strumento risponde all'utente. È essenziale conoscere la **cognetica** per la progettazione di interfacce a misura d'uomo.

Ergonomia

L'**ergonomia** considera cose quali:

- **Disposizione dei controlli e dei display**: raggruppati secondo la funzione, frequenza d'uso, sequenzialità.
- **Ambiente circostante**: la seduta adattabile ad ogni misura di utente.
- **Salute**: la posizione fisica, la temperatura, l'illuminazione ed il rumore.
- **Uso dei colori**: non bisogna usare i colori come un unico tipo di segnalatore, ma unito ad altre informazioni. I colori devono corrispondere a convenzioni comuni (rosso per gli avvertimenti, verde per ok, ecc.).

L'**ergonomia** è utile per definire degli standard e delle linee guida per la progettazione di alcuni aspetti del sistema.

Cognetica

La **cognetica** considera la variabilità cognitiva degli esseri umani dal punto di vista statistico (come l'ergonomia per la variabilità fisica). È una visione pragmatica ed empirica:

- di ciò che la mente può o non può fare;
- di quanto tempo occorre per compiere una data operazione;
- delle circostanze che aumentano la probabilità di commettere errori.

Raskin sulla cognizione

Raskin è uno studioso della *cognetica* e ha teorizzato l'importanza del **conscio/inconscio cognitivo**. Inoltre, ha evidenziato il concetto di **Locus Attention** (*Fuoco dell'attenzione*).

Inconscio cognitivo

L'**inconscio cognitivo** comprende tutti i processi mentali inconsci che si compiono senza che noi ne siamo consapevoli. Quindi, l'*inconscio cognitivo* è definito il “**luogo della mente**” da cui possiamo recuperare all'occorrenza informazioni che vi giacevano inutilizzate.

Esempio:

Qual è la terza lettera del vostro nome?

È un'informazione conosciuta, ma ci dobbiamo ragionare e quindi richiamare l'inconscio per arrivare alla risposta.

Conscio cognitivo

Il **conscio cognitivo** è uno stimolo (la lettura di una frase, la necessità di rispondere a una domanda ecc.) che può far sì che un'informazione o un qualsiasi altro aspetto della nostra memoria passi dall'inconscio al conscio cognitivo. Non posso passare liberamente da conscio a inconscio.

PROPRIETÀ	CONSCIO	INCONSCIO
attivato da	novità – emergenza – pericolo	ripetizione – eventi attesi – sicurezza
usato in	circostanze nuove	routine
gestisce	decisioni	operazioni senza alternative
accetta	proposizioni logiche	logica o inconsistenze
opera	sequenzialmente	simultaneamente
controlla	volontà	abitudini
capacità	minima	enorme
durata	decine di secondi	anni

Fuoco dell'attenzione

Il **fuoco dell'attenzione** è ciò a cui stiamo pensando attivamente e consciamente. Può essere fuori dal nostro controllo (es. un rumore improvviso ci fa perdere il *fuoco dell'attenzione*, cioè l'attenzione, su una determinata attività) e possiamo concentrarci su **UN SOLO fuoco dell'attenzione**, anche se percepiamo molto di più. Le percezioni dirette hanno solitamente un breve periodo di persistenza.

Quali sono le conseguenze, di quanto detto, sull'Interfaccia?

Le percezioni non diventano automaticamente ricordi: se vogliamo dare un'informazione all'utente, questa deve rimanere visibile per tutto il tempo necessario (es. messaggi d'errore con informazioni specifiche).

Se l'attenzione dell'utente è fissata su qualcosa, possiamo apportare cambiamenti ad altre parti del sistema sapendo che ciò non arrecherà disturbo (es. immagine di "loading" che maschera i tempi di attesa).

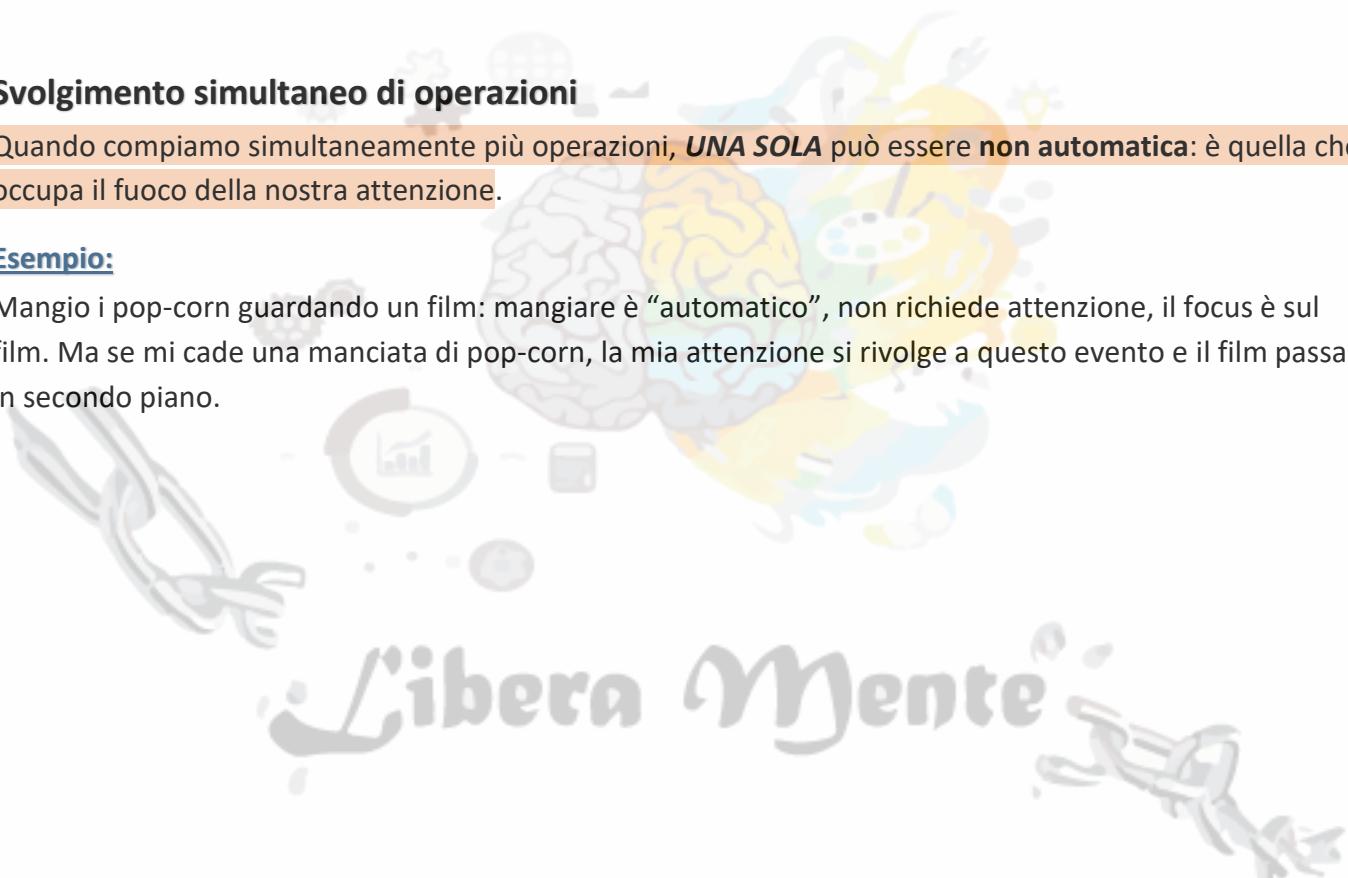
In fase "*critica*" (il programma reagisce in modo inatteso) i messaggi d'errore e di aiuto non vengono recepiti. Maggiore è il coinvolgimento dell'utente in una data operazione, maggiore è la difficoltà a cambiare il **fuoco dell'attenzione**.

Svolgimento simultaneo di operazioni

Quando compiamo simultaneamente più operazioni, **UNA SOLA** può essere **non automatica**: è quella che occupa il fuoco della nostra attenzione.

Esempio:

Mangio i pop-corn guardando un film: mangiare è "automatico", non richiede attenzione, il focus è sul film. Ma se mi cade una manciata di pop-corn, la mia attenzione si rivolge a questo evento e il film passa in secondo piano.



Capitolo 2 – Stili dell’Interazione

2.1 Stili dell’Interazione

L’**interazione** è un dialogo tra il calcolatore e l’utente. Ci sono diversi tipi di dialogo.

- **Interfacce a riga di comando:** si esprimono istruzioni direttamente al computer per mezzo di una combinazione di abbreviazioni e parole intere. Meglio per utenti esperti perché si basa sulla memoria e per aiutare l’utente si usano nomi significativi.
- **Menù:** insieme di opzioni disposte sullo schermo. Le opzioni sono visibili e richiedono meno memoria e si possono selezionare con dispositivi di puntamento o con la tastiera. Spesso le opzioni sono raggruppate gerarchicamente:
 - **Pull-Down:** trascinati giù da un titolo al top dello schermo.
 - **Pop-Up:** compaiono quando si clicca su una particolare regione dello schermo.
 - **Pin-Up:** rimangono sullo schermo finché non gli viene chiesto esplicitamente di scomparire.
 - **Fall-Down:** simile a *pull-down*, ma la barra non è selezionata esplicitamente.
- **Linguaggio naturale:** è familiare si può usare il riconoscimento parlato o scritto. I problemi sono che può essere vago ed ambiguo. Inoltre, è ristretto ad un sottoinsieme del *linguaggio naturale*.
- **Question/Answer e Query:** l’interazione è condotta mediante una serie di domande. Adatto per utenti inesperti, ma offrono funzionalità ridotte. I *linguaggi d’interrogazione* (es. SQL) sono usati per ritrovare informazioni nei database, ma è richiesta la comprensione della struttura del database e della sintassi del linguaggio, quindi, richiede qualche livello di esperienza.

Form-Fill e Spreadsheets

I **Form-Fill** riproducono lo schema di un modulo cartaceo e per questo richiedono una buona progettazione per porre i dati importanti in posizioni rilevanti.

Gli **spreadsheet** sono una griglia di celle ciascuna delle quali può contenere un valore o una formula. Le formule possono comprendere valori di altre celle (es. somma tutte le celle di questa colonna) e l’utente può immettere ed alterare i valori delle celle ma lo *spreadsheet* manterrà la formula sempre corretta.

2.2 Interfacce WIMP

Windows Icons Menus Pointers, raggruppa gli stili d’interazione precedenti in un unico stile.

- **Windows:** sono aree dello schermo che si comportano come terminali indipendenti e possono contenere testo o grafica. Inoltre, possono sovrapporsi, ridimensionarsi, oscurarsi e disporsi l’una a fianco dell’altra. Sono dotate di barre di scorrimento che consentono di scorrere il contenuto della finestra sia orizzontalmente che verticalmente. Le barre dei titoli descrivono il nome della finestra.
- **Icons:** piccole figure o immagini che rappresentano un oggetto dell’interfaccia. Le **finestre** possono essere ridotte a questa piccola rappresentazione.
- **Menus:** scelta offerta sullo schermo di operazioni o servizi che possono essere eseguiti. Il **problema** dei menu è che possono occupare molto spazio sullo schermo. La **soluzione** è di usare alcuni *menu* particolari:

- i **menu pull-down** sono trascinati giù da un singolo titolo al top dello schermo;
- i **menu pop-up** compaiono quando si clicca su una particolare regione dello schermo (eventualmente designata da un'icona);
- i **menu pin-up** rimangono sullo schermo finché non gli viene esplicitamente richiesto di scomparire.
- il **menu fall-down** è simile al *pull-down*, ma la barra non deve essere selezionata esplicitamente.
- I **menu a cascata** sono una selezione di menu che ne apre un altro adiacente e così via
- **menu pie**: le opzioni sono disposte in un cerchio. Più facile selezionare le voci (più ampia area target) e anche più veloce (stessa distanza da ogni opzione). Comunque, occupano spazio sullo schermo e sono meno comuni.

Qualche volta sono forniti degli acceleratori da tastiera: combinazioni di tasti della selezione di una voce da un menu. Il problema generale è cosa includere nei menu e come raggruppare le voci.

- **Pointers**: componente su cui si basa lo stile *WIMP*. Si basa sul puntamento e la selezione di oggetti quali icone e voci di menu. Ottenuto con il mouse o altri sistemi di puntamento, che permettono di far muovere un'ampia varietà di cursori (come trackball, tasti cursore o abbreviazioni da tastiera).

Altre funzionalità di WIMP

Ci sono cose aggiuntive associate ai **sistemi WIMP**, i **widgets**:

- **Pulsanti**: regioni individuali ed isolate all'interno di un display che possono essere selezionate per invocare un'azione.
 - **pulsanti radio**: insieme di scelte mutuamente esclusive;
 - **check boxes**: insieme di scelte non esclusive.
- **Palette**: indicano un insieme di modi possibili disponibili, oltre a quello corrente. In genere una collezione di icone disposte a mattonella. (es. un applicativo per disegno può avere una paletta che indica se si sta disegnando quadrati, cerchi, linee o testo, un'altra che indica l'insieme di pattern di riempimento disponibili e un'altra che indica i colori disponibili).
- **Box di dialogo**: finestre di informazioni che compaiono per informare di qualche evento importante o richiedere certe informazioni.

Capitolo 3 – Paradigmi di Interazione e Principi di Usabilità

3.1 Introduzione

L'obiettivo della progettazione è di progettare al fine della **massima usabilità**. I **principi di usabilità** sono un mezzo più generale per comprendere l'**usabilità**. La storia del progetto di sistemi interattivi fornisce paradigmi per progetti usabili.

3.2 Paradigmi per l'usabilità

Le metafore

I **paradigmi per l'usabilità** sono tecniche interattive. La **metafora** mette la computazione in relazione con altre attività del mondo reale e può essere considerata una tecnica di apprendimento (es. la gestione dei file su una scrivania d'ufficio, l'elaborazione di testi come una macchina da scrivere e la realtà virtuale in cui l'utente è dentro la metafora).

L'**interfaccia** è progettata in modo da essere simile all'entità fisica, ma ha anche delle caratteristiche proprie. Infatti, sfrutta la familiarità degli utenti per aiutarli a comprendere ciò che a loro non è familiare.

L'uso delle metafore porta i seguenti **vantaggi**:

- apprendimento più facile;
- possono essere innovative in modo da rendere il mondo dei computer più accessibile;
- aiutano gli utenti a comprendere il modello concettuale sottostante.

Tra gli **svantaggi** abbiamo:

- possono violare le regole convenzionali e culturali (cestino sul desktop = cestino delle carte collocato sulla scrivania);
- conflitti con i principi di progettazione (forzare troppo la metafora rende il sistema poco chiaro);
- si possono utilizzare dei cattivi design esistenti (utilizzo di cattivi esempi nella metafora);
- limitano l'immaginazione del progettista (limitarsi alla metafora esclude soluzioni innovative);
- gli utenti comprendono il sistema esclusivamente in termini della metafora.

Manipolazione Diretta

La **manipolazione diretta** è il paradigma che ha avuto maggior successo e che ancora oggi utilizziamo. Questo è un paradigma che fa uso della **metafora del mondo ideale**, ovvero l'interfaccia non è un mediatore tra utente e sistema, ma dal punto di vista dell'utente l'interfaccia è il sistema. Infatti, le operazioni eseguite sull'interfaccia evitano la necessità di comprenderne il significato ad un livello più profondo del sistema.

Non c'è più la distinzione fra *linguaggio di Input* e *linguaggio di Output*. Le **espressioni di output** vengono, infatti, utilizzate per articolare espressioni di input (es. l'icona del documento è l'output, ma viene utilizzata come input per l'operazione di spostamento). I **widgets** sono così **oggetti di output e di input**.

Correlato alla **Manipolazione Diretta** è il paradigma “*What You See Is What You Get*” (**WYSIWYG**), dove ciò che conta nelle interfacce è la semplicità e l'immediatezza tra la rappresentazione e il prodotto finale.

I suoi principi base sono:

- la continua rappresentazione degli oggetti e delle azioni d'interesse;
- usare azioni fisiche e pressione di pulsanti anziché impartire comandi con una sintassi complessa;
- rapide azioni reversibili con feedback immediato.

Shneiderman definisce la *Manipolazione Diretta* attribuendogli le seguenti caratteristiche:

- visibilità degli oggetti che interessano;
- azione incrementale e feedback rapido su tutte le azioni;
- reversibilità delle azioni in modo da incoraggiare gli utenti nell'esplorazione;
- correttezza sintattica di tutte le azioni in modo che ogni operazione sia ammessa;
- sostituire il linguaggio dei comandi con le azioni per manipolare gli oggetti visibili.

I vantaggi di queste caratteristiche sono:

- gli utenti inesperti imparano rapidamente le funzionalità di base ed acquistano sicurezza;
- gli utenti esperti portano avanti in modo molto veloce più task contemporaneamente;
- messaggi di errore raramente necessari.
- gli utenti possono vedere immediatamente se le loro azioni stanno perseguitando i loro obiettivi e in caso contrario agire diversamente;
- gli utenti provano minor ansia.

Gli svantaggi, invece, sono:

- alcune persone prendono troppo alla lettera la *metafora* della *manipolazione diretta*;
- non tutti i task possono essere descritti dagli oggetti e non tutte le azioni possono essere eseguite direttamente;
- alcuni task sono svolti in maniera più efficace se si delega il sistema;
- muovere il mouse sullo schermo può richiedere più tempo rispetto alla pressione di tasti per fare le stesse azioni.

Le *interfacce a manipolazione diretta* implicano il **paradigma delle azioni**, nel quale le azioni fatte sull'interfaccia fanno superare la necessità di comprendere il significato a qualsiasi livello più profondo. Tuttavia, alcune attività sono più difficili da esprimere tramite azioni, se non impossibili.

Nel **paradigma del linguaggio**, l'interfaccia è un mediatore tra l'utente ed il sistema: l'utente dà all'interfaccia delle istruzioni ed è poi responsabilità dell'interfaccia controllare che quelle istruzioni vengano eseguite. A questo **paradigma linguistico** si possono associare due interpretazioni importanti:

1. la prima richiede che l'utente capisca il funzionamento del sistema di base e l'interfaccia esegue poca traduzione. Questa interpretazione di *paradigma linguistico* è simile al tipo d'interazione che esisteva prima che fossero introdotte le interfacce di *manipolazione diretta*.
2. la seconda non esige che l'utente comprenda la struttura del sistema di base e prevede che l'interfaccia rivesta un ruolo più attivo, fungendo da interprete tra l'operazione progettata dall'utente e le operazioni possibili del sistema che devono essere attivate per eseguire quel progetto.

Esempio:

La “**programmazione**” comprende sia il **linguaggio** che le **azioni**: l'utente esegue dei task di routine nel **paradigma delle azioni** ed il sistema registra ciò come procedura generale. Il sistema interpreta le operazioni dell'utente come uno script linguistico che poi si può eseguire.

World Wide Web

Il web è costruito sulla rete Internet ed offre un'interfaccia prevalentemente grafica di facile uso per le informazioni, nascondendo la complessità di base dei protocolli di trasmissione, degli indirizzi e dell'accesso remoto ai dati.

Nonostante Internet esistesse dal 1969 non diventò un **paradigma di interazione** fin quando non divennero disponibili delle interfacce grafiche ben progettate (**browser**) che consentissero l'accesso a informazioni multimediali.

Computer Supported Cooperative Work

Con l'arrivo delle reti di computer fu possibile la comunicazione tra macchine eterogenee. In questo modo presero piede delle tecniche di collaborazione tra singoli via computer, il cosiddetto **Computer Support Cooperative Work (CSCW)**.

La principale differenza fra i sistemi CSCW e i *sistemi interattivi* progettati per un singolo utente è che i progettisti non possono più trascurare l'ambiente sociale all'interno del quale ogni individuo opera.

I **sistemi CSCW** vengono costruiti per consentire l'interazione tra molte persone attraverso il computer e un solo prodotto deve soddisfare le necessità di molti.

Un ottimo esempio di CSCW è l'*e-mail* (posta elettronica), una metafora per mezzo della quale singoli utenti posti in posizioni fisicamente separate possono comunicare con messaggi elettronici che funzionano in un modo simile ai sistemi postali convenzionali. Gli scambi di comunicazione tra siti di tutto il mondo avvengono nel giro di minuti e non più di settimane.

Multimodalità

La **multimodalità** combina varie modalità di input simultanee che possono presentare le informazioni in molte modalità di output diverse. In altri termini, vengono usati più canali di comunicazione tra utente e sistema.

La *multimodalità* fa uso di un **canale di comunicazione umano**. Particolare enfasi va posta sull'uso simultaneo di più *canali di comunicazione* per l'input e per l'output.

La *multimodalità* fa uso di due paradigmi:

1. **il computer come strumento**: la macchina è uno strumento passivo e cerca di comprendere l'utente attraverso tutte le diverse modalità di input che riconosce. L'utente è sempre responsabile dell'iniziativa nelle operazioni. Inoltre, questo paradigma segue i principi della **manipolazione diretta**.
2. **il computer come partner di dialogo**: le modalità multiple sono usate per aumentare l'**antropomorfismo** nell'interfaccia utente (riconoscimento parlato). L'output multimodale è espresso come teste parlanti e altre modalità "Human like".

Messaggi di errore

Shneiderman definisce delle linee guida per i **messaggi di errore**:

- Evitare i termini come *FATAL, INVALID, BAD*.
- Avvertimenti audio.
- Evitare *MAIUSCOLE* e codici con lunghi numeri.
- Messaggi precisi.
- Fornire aiuto relativo al contesto.

3.3 Antropomorfismo

Abbiamo due tipi di aspetti:

1. **l'aspetto cognitivo** (la componente mentale): situazioni che danno luogo ad emozioni.
2. **l'aspetto fisico** (la componente fisica): risposta fisiologica che avviene con un'emozione o la segue immediatamente.

L'**antropomorfismo** riguarda nell'attribuire qualità umane ad oggetti inanimati. Noto nella pubblicità e molto sfruttato nell'interazione uomo-macchina per rendere più divertente l'esperienza dell'utente e ridurre la sua ansietà.

I **vantaggi** dell'antropomorfismo sono:

- Incoraggia l'utente a continuare nell'interazione in caso di feedback positivi.
- Riduce l'ansia degli utenti inesperti.
- Può coinvolgere l'utente.

Tra gli **svantaggi** abbiamo:

- Può far sentire inferiori gli utenti in caso di feedback negativi e provocare ansia.
- Feedback personalizzati possono rendere gli utenti meno responsabili delle proprie azioni.
- Portano la gente a un falso senso di confidenza, a parlare di fatti personali con personaggi virtuali

Legato al paradigma dell'**antropomorfismo** è quello delle “**interfacce basate su agenti**” che operano per conto degli utenti all'interno del mondo elettronico in modo da metterli più a loro agio e non farli sentire in difficoltà. A volte, tali interfacce sono frustranti.

Gli **agenti di interfaccia** si possono classificare sulla base del grado di *antropomorfismo* che esibiscono:

- **Personaggi sintetici**: sono autonomi, con stati interni e in grado di rispondere ad eventi esterni;
- **Agenti animati**: giocano un ruolo collaborativo sull'interfaccia e spesso sono del tipo cartone animato;
- **Agenti emotivi**: hanno una personalità predefinita e un insieme di emozioni che l'utente può cambiare;
- **Agenti conversativi emotivi**: hanno una personalità predefinita e un insieme di emozioni che l'utente può cambiare.

Capitolo 4 – Le Personas

4.1 Regole di Design

Le **regole di design** sono regole che un progettista segue per aumentare l'**usabilità** del prodotto software finale. Bisogna tener conto di:

- Chi sono gli utenti;
- Quali attività si stanno svolgendo;
- Dove ha luogo l’interazione.

Bisogna ottimizzare le interazioni che gli utenti hanno col prodotto in modo che corrispondano alle attività e alle esigenze degli utenti. Abbiamo 5 questioni chiave:

1. **Fissare gli obiettivi**: decidere cosa osservare/studiare e decidere come analizzare i dati una volta raccolti;
2. **Identificare i partecipanti**: decidere da chi ottenere dati;
3. **Relazione con i partecipanti**: chiara e professionale e consenso informato quando opportuno;
4. **Triangolazione**: guardare ai dati da più prospettive raccogliendo più tipi di dati;
5. **Studi piloti**: piccolo trial dello studio principale.

Inoltre, bisogna anche **comprendere le esigenze degli utenti**, ovvero:

- considerare le persone cosa sanno e cosa non sanno fare;
- cosa potrebbe aiutarli a svolgere le attività che già svolgono;
- ascoltare cosa vogliono e coinvolgerli nelle scelte;
- utilizzare metodi *user-based* già utilizzati e testati.

4.2 Analisi degli stakeholder

L'**analisi degli stakeholder** è uno strumento semplice per incoraggiare il team di progetto a considerare le necessità di tutte le persone che potrebbero essere influenzate dalle decisioni di progetto.

Gli **utenti finali** sono gli **stakeholder esterni**, il gruppo critico su cui ci si concentra per individuare il target dell’applicazione:

- i clienti attuali;
- i clienti potenziali;
- i clienti perduti;
- i clienti della concorrenza.

4.3 I profili utente – “*personas*”

Si deve identificare un **insieme di personaggi** rispetto ai quali testare idee e concetti di design. Un **personaggio** contiene:

- una mini-biografia di un utente fittizio per il prodotto proposto;
- informazioni precise sul personaggio e descrizione di obiettivi e motivazioni.

Inoltre, incoraggia il team di progettisti a mettersi nei panni dell’utente/cliente e a comprendere le sue motivazioni come persona.

Possono essere identificati diversi tipi di personaggi: l’ideale è identificare un utente medio, chiamato **utente-stereotipo**, oppure gli **utenti estremi** (frequenza d’uso, complessità richiesta, ecc.). Inoltre, non ci si preoccupa di essere ‘**politically correct**’.

La ‘**persona**’ (descrizione del personaggio) dovrebbe rappresentare:

- il **comportamento**;
- le **motivazioni**.

Processo di definizione dei personaggi

Questo processo è veloce e può essere uno strumento molto efficace per comprendere l’utente. Inoltre, richiede materiale semplice (fogli A2, riviste e un word processor) e deve essere mantenuto semplice e divertente (non si deve riprodurre una persona perfetta).

Si identifica il **range dei possibili utenti** assicurandosi di interagire con tutta le diverse tipologie di potenziali utenti.

Si identificano gli **obiettivi** di ciascun personaggio:

- Obiettivi di esperienza
- Obiettivi finali
- Obiettivi collegiali
- Obiettivi pratici
- Falsi obiettivi

Sorgenti per la raccolta dati

- | | | |
|--|--|--|
| <p>➤ Interviste</p> <p>➤ User survey</p> <p>➤ Questionari</p> | <p>➤ Indagini per telefono</p> <p>➤ Indagini via Web</p> | <p>➤ Analisi documentazione esistente</p> |
|--|--|--|

Visite sul posto – Pianificazione

- **Obiettivi ed argomenti**: cosa si intende apprendere attraverso la visita sul luogo.
- **Partecipanti**: chi si vuole osservare o intervistare. Quanti partecipanti includere. Quale tipo di utente vorrebbe partecipare.
- **Sedi**: dove svolgere l’intervista o l’osservazione.
- **Programmazione delle visite**: quando si potrebbe andare sul posto. Quanto tempo si prevede che si possa dedicare a ciascuna visita.
- **Selezione**: come trovare le persone che si ha bisogno di incontrare? Come convincerle a partecipare.
- **Tecniche di raccolta dei dati**: quali tecniche utilizzare per raccogliere i dati. Si parteciperà attivamente alla sessione o si osserverà solamente?
- **Media**: useremo videoregistratori, registratori audio, carta, penna, moduli prestampati?

Visite sul posto – Attuazione

Chiedere all’utente di pensare ad alta voce; osservare, parlare, ascoltare l’utente ed annotare le caratteristiche interessanti per il progetto.

- Prender nota dell’ambiente di lavoro di ciascun utente;
- Capire gli obiettivi degli utenti;
- Capire i compiti degli utenti (valutare se la tecnologia richiesta è necessaria o opzionale);
- Annotare quali fattori determinano l’esecuzione dei compiti, la situazione all’inizio dei compiti;
- Separare le osservazioni dalle inferenze durante l’osservazione;
- Fare attenzione alle interazioni con altre persone, altri programmi, documenti;
- Osservare cosa succede quando l’utente ha terminato il compito;

Informazioni da reperire

Il tipo di informazioni da raccogliere è un fattore rilevante per determinare come strutturare l'intervista o come formulare le domande. Bisogna identificare i bisogni degli utenti, cioè capire quale obiettivo vogliono perseguire attraverso l'applicazione che si vuole sviluppare.

Si deve identificare il loro **modello di comportamento**, quindi, capire quali azioni eseguono per realizzare i loro goal, quali sono le loro priorità e quali difficoltà incontrano.

Chi intervistare?

Utenti finali del prodotto, se lo scopo dell'intervista è determinare il profilo o il modello di comportamento dell'utente.

Interviste singole:

- consentono di ottenere risposte più dettagliate;
- consentono di percepire differenze e analogie tra utenti della stessa categoria;
- si protraggono più a lungo.

Interviste di gruppo:

- consentono dialoghi tra gli utenti;
- consentono di parlare con più persone impiegando minor tempo;
- sono meno prevedibili e quindi difficili da pianificare.

Luogo dell'intervista?

L'ambiente di lavoro dell'utente: il luogo più adatto permette all'intervistatore di osservare direttamente il contesto d'uso e consente agli utenti di **mostrare piuttosto che parlare**.

Se l'ambiente di lavoro non è disponibile bisogna mettere gli utenti a proprio agio in un luogo poco rumoroso e prevedere delle domande per individuare il contesto lavorativo.

Raccolta di informazioni

I metodi appropriati per reperire informazioni utili in diverse fasi di sviluppo di un'applicazione sono:

- identificare i modelli di comportamento degli utenti;
- individuare le opinioni degli utenti su prodotti esistenti in mercato;
- identificare le aspettative degli utenti sul prodotto che deve essere sviluppato;
- verificare il grado di soddisfazione degli utenti riguardo le varie versioni (prototipali e non) del sistema.

Questo è il punto di partenza per il processo di **analisi dei requisiti** prima ancora dell'osservazione diretta dell'utente, per:

- capire il contesto generale del lavoro;
- individuare il vocabolario dell'utente;
- pianificare in modo adeguato i successivi contatti con gli utenti.

Non è raccomandabile portare l'utente ad immaginare situazioni ipotetiche. Spesso gli utenti non hanno la capacità di predire ciò che può essere di loro gradimento ma più semplicemente, spesso non sono in grado di specificare le loro preferenze.

Caratteristiche psicologiche

Gli individui apprendono in maniera diversa l'uno dall'altro secondo le modalità e le strategie con cui ciascuno elabora le informazioni. Gli **stili cognitivi** si riferiscono alla scelta delle *strategie cognitive* utilizzate per risolvere un compito e vanno considerati come delle preferenze nell'uso delle proprie abilità:

- **sistematico-intuitivo**: riguarda il modo di classificare e formulare ipotesi da parte di un individuo;
- **globale-analitico**: concerne, nella percezione, la preferenza per la considerazione dell'insieme o del dettaglio;
- **impulsivo-riflessivo**: riguarda i processi decisionali, è il modo in cui si affronta un problema.
- **verbale-spaziale**: è trasversale ai vari stili cognitivi, riguarda la percezione (i tipi di informazione su cui si focalizza l'attenzione dell'individuo), la memoria (le informazioni che meglio vengono registrare in memoria e come sono immagazzinate) e le preferenze di risposta.

4.4 Metodi di raccolta dati

Metodi basati su interviste

Uno dei metodi di raccolta dati è basato sulle **interviste**:

- **Interviste e focus groups**: adeguate per esplorare questioni in profondità con un basso numero di soggetti. L'intervistatore può annotare nuove questioni. Inoltre, può registrare o prendere nota.
- **Questionari e survey**: adeguati per raccogliere dati da un elevato numero di soggetti che rispondono. Usate per determinare la frequenza e la distribuzione.
- **La formulazione delle domande** deve essere fatta con un linguaggio semplice e chiaro e bisogna evitare influenze.

Metodi basati su osservazioni

Un altro metodo di raccolta dati è basato sulle **osservazioni**:

- **Log di sistema**: è un'attività di registrazione dettagliata, ma spesso difficile da interpretare.
- **Registrazione video/osservazione da parte dell'analista**: interazione uomo-sistema nel contesto di lavoro: un ricco insieme di dati dettagliati. Registrazione video per analisi di dettaglio, spesso da parte di più osservatori.

Un protocollo verbale concorrente o successivo, con un piccolo numero di utenti, può aiutare l'interpretazione delle osservazioni e dei log di sistema.

4.5 Studi pilota

Lo **studio pilota** consiste nel testare ogni approccio di raccolta dati con un piccolo numero di utenti potenziali. Revisionare:

- i **dati raccolti** (forniscono le informazioni richieste?)
- le **risposte degli utenti** (si stanno ponendo domande rilevanti/importanti?)
- le **risorse/competenze** richieste per raccogliere/analizzare i dati.

4.6 Task Analysis (Analisi dei compiti)

Obiettivo: uno stato target del mondo che l'utente vuole ottenere.

Compito (task): un'attività (sempre espressa con un verbo) intrapresa da una persona per perseguire un obiettivo usando un particolare dispositivo, o un particolare strumento o artefatto:

- potrebbe esserci esattamente un task oppure *task* alternativi per raggiungere l'obiettivo;
- i task possono essere decomposti in *sotto-task*.

Dominio: contesto in cui viene eseguito un *task*.

La **decomposizione dei task** è generalmente gerarchica. La base per il design è:

- Allocazione dei task (uomo-sistema);
- Requisiti di performance;
- Ordinamento e raggruppamento delle funzioni;
- Quali oggetti rappresentare;
- Quali dispositivi di input/output e stili di dialogo;
- Di quali conoscenze/competenze necessitano gli utenti per completare i task.

Tipi di Task

Task fisici

Conoscenza o rappresentazione di un insieme appropriato di azioni o **comportamenti fisici** (es. Inserire una scheda, premere pulsanti).

Task mentali

Conoscenza o rappresentazione di un insieme appropriato di azioni o **comportamenti mentali** (es. recuperare il numero di PIN).

Analisi dei Task basati sulla conoscenza

- Elencare tutti gli oggetti e le azioni coinvolte nel task;
- Elencare tutte le conoscenze di cui necessitano gli utenti per compiere ogni azione correttamente;
- Si raggruppino in *tassonomie*.

Lo **scopo** è di:

- identificare la conoscenza di cui ha bisogno l'utente per eseguire il task;
- definire le competenze, preparare il training, la documentazione.

Impatto sul design (quando la performance non è come desiderata)

1. Incompatibilità tra modello mentale dell'utente e le rappresentazioni del computer del:

- **Dominio:** gli oggetti e le funzioni che l'utente si aspetta non sono presenti o sono difficili da trovare;
- **Task:** l'utente non è in grado di eseguire il task oppure è costretto a eseguirlo in maniera sgradevole.

2. Scorretta allocazione del task tra l'utente e il sistema e troppo carico (mentale o fisico) per l'utente.

L'obiettivo è avere un design guidato dagli obiettivi, ciò vuol dire che:

- non dimenticare mai l'obiettivo (cosa si deve ottenere);
- documentare l'obiettivo nell'analisi dei task;
- gli obiettivi sono più duraturi, mentre i task sono transitori;
- *"Progettare a partire dai task anziché dai goal è una delle cause di un'interazione frustrante e inefficace"* [Alan Cooper].

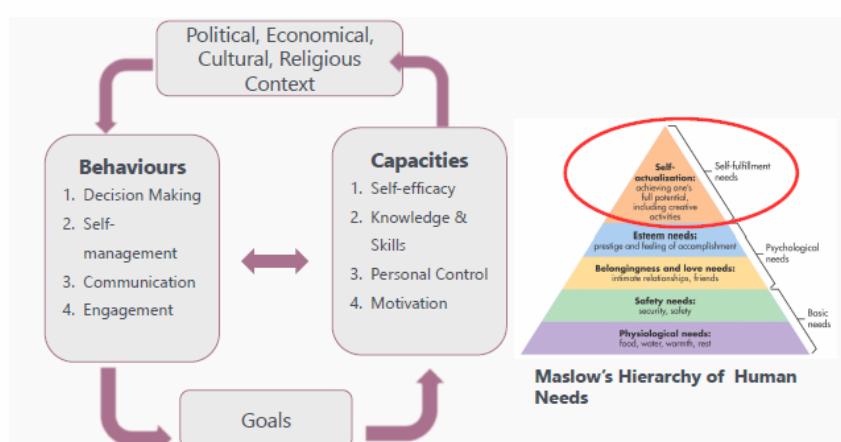
Gli obiettivi degli utenti sono spesso diversi da quelli che ci aspetteremmo. Quando si soddisfano le **esigenze personali** e gli **obiettivi degli utenti**, i business goal si perseguono in modo più efficace.

4.7 Empowerment (Potenziamento personale)

Introdurremo una metodologia per identificare i requisiti di **User eXperience (UX)** per l'**empowerment** dei singoli utenti sui task che questi svolgono nel contesto che stiamo osservando.

Il processo di **identificazione degli obiettivi di empowerment** si articola in diversi step:

1. associare ciascun task a una o più categorie comportamentali;
 2. per ogni categoria individuata determinare quale/i abilità ne subiscono l'influenza;
 3. costruire un **questionario** che permetta di valutare gli indicatori di abilità associati a ciascun task con una scala a **intervalli da 1 a 5** (corrispondenti a Scarso, Sufficiente, Buono, Molto Buono, Eccellente), ma è sempre bene includere anche una domanda a risposta aperta per trattare necessità eventualmente trascurate.
- Le abilità acquisite finora da ciascun partecipante allo studio sono valutate come misura del livello di **empowerment** percepito dal partecipante stesso, questo avviene per derivare i **requisiti di UX** che porterebbero al raggiungimento degli obiettivi individuali di empowerment.
4. fare la media dei valori ottenuti da tutti i partecipanti dello studio: verrà costruito un **foglio Excel** per ogni partecipante e sarà riempito con i valori ottenuti dal questionario. Si fa la media dei valori ottenuti da tutti i partecipanti allo studio e saranno riassunti i valori ottenuti in una tabella.
 5. alla fine, si evidenziano i valori che sono ≤ 3 e verranno elencati gli **obiettivi di empowerment**.



Definizione UX

La **User eXperience** comprende tutte le emozioni, le convinzioni, le preferenze, le percezioni, le risposte fisiche e psicologiche, i comportamenti e i risultati dell'utente, che occorrono prima, durante e dopo l'uso. Essa rappresenta lo stato interiore e fisico dell'utente che risulta da precedenti esperienze, atteggiamenti, abilità e personalità, e dal contesto d'uso.

Capitolo 5 – Regole di Design

5.1 Introduzione

Le **regole di design** sono regole che un progettista segue per aumentare l'**usabilità** del prodotto software finale.

Le si può classificare sulla base di due elementi :

- **autorità** : indica se la regola deve essere eseguita obbligatoriamente o sia solo un suggerimento;
- **generalità**: indica se la regola può essere applicata in molte situazioni di progetto o se è valida solo in una situazione più specifica.

Le regole di design possono essere suddivise in 3 tipi di **regole principali**:

1. **Principi di usabilità**: regole di design astratte con alta generalità e bassa autorità.
2. **Standard ed Euristiche**: regole di design specifiche con alta autorità e limitata applicabilità.
3. **Linee guida**: tendono ad avere minore autorità ed essere più generali nell'applicabilità.

5.2 Principi di Usabilità

I **principi di usabilità** sono un mezzo più generale per comprendere e promuovere l'**usabilità**. Sono guidati teoricamente da conoscenza psicologica, computazionale e sociologica.

L'obiettivo dei *principi di usabilità* è di massimizzare i benefici di un buon progetto astraendo le proprietà generali e rendendole ripetibili così da poter guidare il progetto di nuovi sistemi interattivi.

Vengono raggruppati in 3 categorie principali:

1. **Capacità di apprendimento o learnability**: facilità con cui i nuovi utenti possono iniziare un'interazione efficace e raggiungere le massime prestazioni. Si suddivide in:
 - **Predicibilità**: per determinare l'effetto di azioni future è sufficiente ragionare sulla base della storia delle interazioni precedenti.
 - **Sinteticità**: l'utente deve poter assestarsi e valutare l'effetto delle operazioni precedenti sullo stato corrente. Quando un'operazione cambia lo stato del sistema è importante che l'utente si accorga della modifica. In alcuni casi questa notifica può arrivare subito non richiedendo alcuna ulteriore interazione dell'utente (**Onestà immediata**) . In altri casi può arrivare alla fine dopo esplicite direttive dell'utente (**Onestà ritardata**).
(es. confronto del copia incolla in sistemi a *manipolazione diretta* e in sistemi a *riga di comando*).
- **Familiarità**: gli utenti portano con sé molta esperienza ottenuta attraverso l'interazione con altri sistemi. Si è interessati al come la conoscenza dei sistemi precedenti si applica al nuovo sistema. Il sistema dovrebbe risultare intuitivo.
- **Generalizzabilità**: gli utenti cercano di estendere la conoscenza specifica di un'interazione a nuove situazioni che sono simili ma che non hanno mai incontrato prima. Un buon esempio di generalizzazione fra un'applicazione ed un'altra si nota negli editor di testi che tentano di fornire le operazioni di copia e incolla nello stesso modo.
- **Coerenza**: similitudini nel comportamento input/output che nascono da situazioni o obiettivi simili. Si può avere consistenza nel nominare i comandi o nell'invocazione dei comandi.

2. **Flessibilità**: si riferisce alla molteplicità di modi in cui l'utente e il sistema si scambiano informazioni. Si suddivide in:

- **Iniziativa nel dialogo**: il sistema può iniziare il dialogo e l'utente risponde semplicemente alle richieste di informazioni. Il sistema non può essere interrotto (dialogo **System pre-emptive**). Una finestra modale fa sì che l'utente possa comunicare con il sistema solo tramite questa finestra, in alternativa l'utente può essere libero di interrompere il sistema e di intraprendere qualsiasi azione verso di esso (dialogo **User pre-emptive**). In generale si tenta di aumentare la capacità dell'utente di interrompere il sistema e ridurre la possibilità che il sistema interrompa l'utente. Alcune situazioni possono però richiedere un dialogo *system pre-emptive* (es. cancellazione di testo in un editor cooperativo).
- **Multithreading**: capacità del sistema di supportare l'interazione con l'utente su più task contemporaneamente. Consente quindi al sistema di supportare più di un compito alla volta. Il multithreading può essere:
 - **concorrente** quando consente la comunicazione separata di informazioni che riguardano compiti separati;
 - **interfogliato** permette una sovrapposizione temporale tra compiti separati ma in ogni dato momento il dialogo si limita ad un solo compito.

Un *sistema a finestre* supporta un *dialogo interfogliato*, mentre un **sistema multimodale** potrebbe supportare un **dialogo concorrente**.

- **Migrabilità di un task**: riguarda il passaggio di responsabilità tra utente e computer per l'esecuzione di un task. Dovrebbe essere possibile per un utente o un sistema passare il controllo all'altro in modo da ottimizzare alcuni compiti. Un ottimo esempio di **migrabilità** dei task avviene nel controllo dell'ortografia.
- **Personalizzazione**: viene modificata l'interfaccia utente da parte dell'utente o da parte del sistema. Si fa una distinzione tra modifica iniziata dall'utente (**adattabilità**) e modifica iniziata dal sistema (**adattività**).

L'**adattabilità** avviene quando, ad esempio, l'utente personalizza l'interfaccia sulla base di sue preferenze spostando dei pulsanti.

L'**adattività** invece è la personalizzazione automatica dell'interfaccia sulla base dell'osservazione dei compiti utente.

- **Sostituibilità**: concedere di sostituire fra loro valori di input e di output equivalenti. Ciò contribuisce alla flessibilità permettendo all'utente di scegliere la forma che si adatta meglio.

Possiamo avere sostituibilità per:

- l'**input** in modo da immettere valori anche in diverse unità di misura;
- per l'**output** avendo molteplicità nella rappresentazione degli stati del sistema;
- per **opportunità** dove non riscontriamo differenza fra input e output (es. in un pacchetto da disegno tramite manipolazione diretta possiamo disegnare una linea e il sistema calcola la lunghezza, oppure possiamo inserire le coordinate e il sistema disegna la linea).

3. **Robustezza:** il livello di supporto fornito all'utente nel determinare un comportamento di successo rispetto ai suoi goal. I principi di robustezza sono:

- **Osservabilità:** capacità dell'utente di valutare lo stato interno del sistema tramite la sua rappresentazione percepibile attraverso l'interfaccia. Essa può essere ottenuta tramite:
 - **Navigabilità:** consente all'utente di esaminare lo stato interno corrente del sistema tramite la visualizzazione fornita sull'interfaccia.
 - **Default:** i valori predefiniti assistono l'utente tramite un ricordo passivo e riducono il numero di azioni fisiche per immettere un valore.
 - **Raggiungibilità:** si riferisce alla possibilità di spostarsi tra gli stati osservabili del sistema.
 - **Persistenza:** tratta la durata dell'effetto di un atto comunicativo e la capacità dell'utente di usare quell'effetto.
- **Recuperabilità:** capacità dell'utente di intraprendere azioni correttive una volta rilevato un errore. Gli utenti spesso commettono errori e vogliono ripristinare lo stato precedente del sistema. La recuperabilità può avvenire in due direzioni:
 - **forward (in avanti)** che implica l'accettazione dello stato corrente e la negoziazione del percorso per raggiungere lo stato desiderato;
 - **backward (in indietro)** tenta di annullare gli effetti di un'interazione appena eseguita per ritornare a uno stato precedente prima di procedere;
 - **principio di sforzo commisurato:** afferma che se è difficile annullare un effetto dato sullo stato dovrebbe essere altrettanto difficile eseguirlo.
- **Risposta:** come l'utente percepisce la comunicazione con il sistema in termini del tempo che il sistema impiega ad esprimere cambiamenti di stato. Misura la rapidità di comunicazione tra sistema e utente.
- **Conformità dei task:** il grado con cui i servizi offerti dal sistema supportano tutti i task degli utenti. Poiché lo scopo di un sistema interattivo è consentire a un utente di eseguire vari compiti per raggiungere i suoi obiettivi all'interno di uno specifico dominio, potremmo chiederci se il sistema supporta tutti i compiti di interesse (**completezza dei compiti**) e se lo fa come vuole l'utente. L'**adeguatezza dei compiti** affronta il grado di comprensione che l'utente ha dei compiti.

5.3 Standard ed Euristiche

Gli **standard** sono stabiliti da organismi nazionali o internazionali per assicurare la conformità da parte di una vasta comunità di progettisti, inoltre, richiedono una **teoria di base corretta** e una **tecnologia che cambia lentamente**. Gli **standard hardware** più diffusi di quelli **software**: autorità elevata e basso livello di dettaglio.

ISO 9241

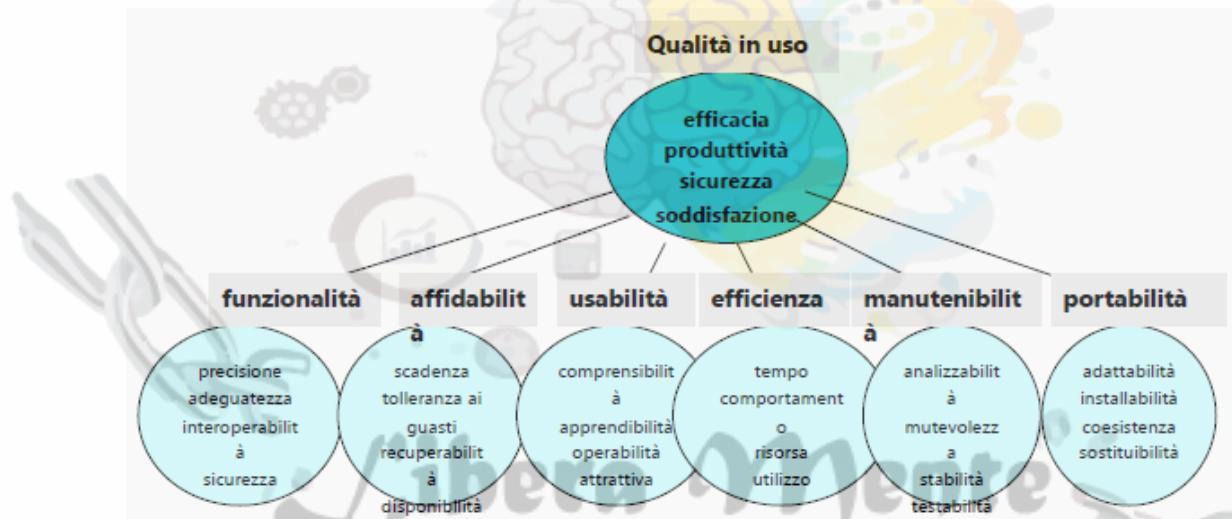
ISO 9241 definisce l'**usabilità** come **efficacia, efficienza e soddisfazione** con cui determinati utenti eseguono i compiti in un determinato contesto d'uso.

Quindi, l'**usabilità** di un *prodotto software* è uno dei fattori che contribuiscono alla **qualità d'uso**.

ISO 9241 stabilisce **7 principi di dialogo** generali:

1. **idoneità al compito**: il dialogo dovrebbe essere adatto al compito e al livello di abilità dell'utente;
2. **auto-descrittività**: il dialogo dovrebbe chiarire che cosa l'utente dovrebbe fare dopo;
3. **controllabilità**: l'utente dovrebbe essere in grado di controllare il ritmo e la sequenza dell'interazione;
4. **conformità alle aspettative dell'utente**: dovrebbe essere coerente;
5. **tolleranza agli errori**: il dialogo dovrebbe essere tollerante;
6. **idoneità all'individualizzazione**: il dialogo dovrebbe essere personalizzato per adattarsi all'utente;
7. **idoneità all'apprendimento**: il dialogo dovrebbe supportare l'apprendimento.

Parte I – Modello di Qualità



L'**usabilità** è la capacità di un prodotto software di essere **compresso, appreso, utilizzato** e di essere **attraente** per l'utente se utilizzato in un determinato contesto.

Qui si sottolinea il fatto che l'**usabilità** di un prodotto software dipende dal contesto in cui il prodotto è usato.

5.4 Linee Guida

Le **linee guida** sono più **suggerite, generali ed astratte**, inoltre, comprendere le motivazioni delle *linee guida* aiuta a risolvere i conflitti:

- le **linee guida** maggiormente **astratte** sono applicabili durante le prime attività del ciclo di vita;
- le **linee guida** maggiormente **dettagliate** sono applicabili durante le attività successive del ciclo di vita.

5.5 Regole d'oro ed Euristiche

Le *regole di design* descritte in precedenza richiedono per il progettista un certo impegno per tener traccia delle *linee guida* oppure per interpretare i principi. Quindi esistono diversi tipi di **euristiche** che bisogna seguire per creare un buon sistema interattivo.

Le **euristiche** più famose sono:

- le **10 euristiche di Nielsen**;
- le **8 regole d'oro di Shneiderman**;
- i **7 principi di Norman**.

Le 8 regole d'oro di Shneiderman

Queste regole forniscono un riassunto breve ed utile dei principi chiave di design.

1. **Coerenza - preservare la coerenza**: sequenze di operazioni simili dovrebbero essere effettuate sempre con lo stesso tipo di azioni. Bisogna usare sempre stesse convenzioni e terminologia per i prompt, menù, colori e si devono limitare il numero delle eccezioni.
2. **Snellimenti - consentire agli utenti abituali di usare comandi rapidi**: consentire l'uso di macro ed abbreviazioni per ridurre il numero di interazioni ed abbreviare i tempi di risposta del sistema.
3. **Offrire feedback informativo**: ad ogni azione del sistema dovrebbe corrispondere un feedback del sistema. Il responso di azioni frequenti dovrebbe essere modesto, mentre azioni occasionali dovrebbero avere responsi più dettagliati. Mostrare i cambiamenti presentando gli oggetti di interesse.
4. **Chiusura - progettare dialoghi provvisti di chiusura**: organizzare le sequenze di azioni in gruppi, prevedendo feedback informativo alla fine di ciascun gruppo di azioni. Dare all'utente la sensazione di poter scaricare la mente alla fine di una sequenza per dedicarsi interamente al task successivo.
5. **Errori - Offrire una prevenzione e una gestione semplice degli errori**: progettare l'interfaccia in modo che sia quanto più difficile commettere errori. In presenza di un errore il sistema dovrebbe offrire istruzioni semplici e specifiche per risolverlo. Migliorare i messaggi di errore aumenta la possibilità di usare il sistema con successo. Si possono aiutare gli utenti ad evitare gli errori seguendo tre possibili tecniche:
 - **Copie corrispondenti corrette**: la mancata chiusura di una parentesi graffa può essere prevenuta usando un editore "smart". Evita gli errori e la necessità di gestirli ma potrebbe essere preferito un approccio meno rigido.
 - **Sequenze complete**: una fonte di errori è il mancato completamento di una fissata sequenza di passi. Si cerca di evitare questo problema raggruppando sequenze di passi in singole azioni. A volte l'utente ha bisogno di accedere alle operazioni atomiche piuttosto che alla sequenza di passi raggruppati: può essere preferibile lasciare all'utente stesso la possibilità di definire delle "macro". La scelta delle sequenze da raggruppare può essere fatta in base ad uno studio sull'uso effettivo del sistema e sul tipo di errori ricorrenti.
 - **Comandi corretti**: altra causa di errori è l'uso di un linguaggio di comandi che spesso porta errori di battitura, di nomi, di comandi etc... Il completamento automatico può evitare questo tipo di errori riducendo l'uso della tastiera. Non sempre la correzione automatica completa il comando nella maniera voluta. Lo stesso principio si applica alla manipolazione diretta: il sistema può presentare all'utente solo le azioni ammissibili, e l'utente seleziona quelle che desidera, per esempio per mezzo del mouse.

6. **Reversibilità - permettere un'inversione semplice delle azioni:** rendere le azioni reversibili il più possibile in modo da incoraggiare l'utente a esplorare opzioni sconosciute.
7. **Controllo - supportare il controllo interno:** dare all'utente la sensazione di essere il responsabile del sistema, che non fa altro che rispondere alle sue azioni. Evitare reazioni sorprendenti.
8. **Memoria - ridurre il carico della memoria a breve termine:** la memoria umana a breve termine può elaborare un numero limitato di informazioni. (7 ± 2 elementi di informazione). Mantenere il display semplice e ridurre la frequenza di spostamenti di finestre. Fornire, se necessario, accesso on-line alle forme sintattiche, abbreviazioni di comandi, codici.

I 7 principi di Norman

Sono un riassunto della filosofia di design centrata sull'utente di Norman.

1. **Bisogna usare sia la conoscenza presente nel mondo sia la conoscenza mentale:** le persone lavorano meglio quando le conoscenze di cui hanno bisogno per eseguire un compito sono disponibili all'esterno (concetto di Affordance).
2. **Si deve semplificare la struttura dei compiti:** i compiti devono essere semplici per evitare all'utente una risoluzione complicata dei problemi e un eccessivo carico di memoria.
3. **Si rendano le cose visibili:** bisogna gettare un ponte sul golfo dell'esecuzione e sul golfo della valutazione. L'interfaccia deve mostrare ciò che il sistema può fare e come farlo, permettendo di vedere chiaro l'effetto delle sue azioni.
4. **Le corrispondenze vanno chiarite:** le intenzioni dell'utente dovrebbero corrispondere chiaramente ai controlli del sistema e le sue azioni dovrebbero corrispondere chiaramente agli eventi del sistema. Dovrebbe essere chiaro, quindi, quale controllo fa cosa e in che misura lo fa.
5. **Si sfrutti il potere dei vincoli, sia naturali sia artificiali:** i vincoli sono elementi nel mondo che consentono di eseguire solo l'azione corretta nel modo corretto. Un esempio semplice è il puzzle i cui pezzi si incastrano in un solo modo. I vincoli fisici del design guidano l'utente nel completamento del compito.
6. **Bisogna progettare gli errori:** bisogna anticipare gli errori dell'utente e progettare il recupero del sistema.
7. **Quando tutto il resto non ha successo, si creano degli standard:** se non esistono corrispondenze naturali quelle arbitrarie dovrebbero andare standardizzate. È questo principio di standardizzazione che permette ad una persona di guidare qualsiasi macchina con pochissime difficoltà.

5.6 Design Pattern

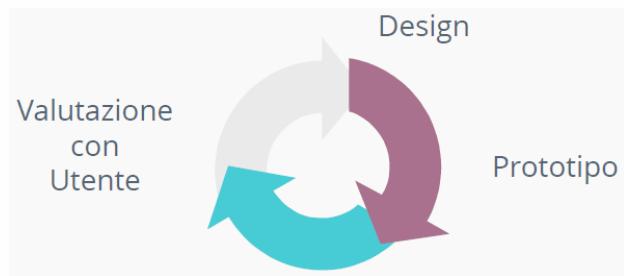
Sono un approccio per riutilizzare le conoscenze sulle soluzioni di progettazione di successo. Un **modello** è una soluzione parzialmente specificata ad un problema ricorrente all'interno di un contesto specifico. Si concentrano sulla prassi della progettazione non sulla teoria, raccogliendo le proprietà essenziali comuni ad un buon progetto. Per essere descritti si usa un linguaggio appropriato. Essi sono ripetibili perché astraggono le proprietà generali dei tipi di design.

Capitolo 6 – Prototipazione e Design

6.1 Prototipi

I prototipi si usano per:

- avere un rapido feedback sul design;
- sperimentare design alternativi;
- tenere il design centrato sull'utente;
- eliminare i problemi legati all'implementazione;
- simulano le caratteristiche del sistema.



Abbiamo tre tipi di prototipi: **Throw-away**, **Incremental** ed **Evolutionary**.

Gli obiettivi, invece, possono essere:

- **esplorativo**: il prototipo aiuta a sviluppare una visione comune e a dedurre i requisiti dell'utente;
- **sperimentale**: il prototipo consente studi di fattibilità.

Lavorando con i *prototipi* avremo diversi livelli di dettaglio:

- **mock-up** è un prospetto dell'interfaccia uomo-macchina senza alcuna funzionalità;
- **prototipo funzionale** è un'implementazione di un sistema parziale;
- **sistema pilota** è un sistema con funzionalità limitate (i casi d'uso).

Tali *prototipi* sono costruiti con diversi strumenti (es. tool di presentazione, costruttori di interfaccia, ...).

Lavorare con un prototipo

Esaminiamo gli steps per creare dei *prototipi*:

1. creare un modello che catturi l'uso del sistema dal punto di vista dell'utente (es. diagrammi UML dei casi d'uso);
2. creare viste interattive prototipali con Powerpoint (o Balsamiq, HTML, Visual Studio, ...) che possano animare i principali scenari d'uso;
3. valutare e migliorare il prototipo con diversi gruppi di utenti;
4. far evolvere il modello per fornire specifiche di design, documentazione, casi di testing del sistema e materiale illustrativo.

6.2 Tecniche di prototipazione

Paper Sketches di interfacce Utente

Vantaggi

- ✓ supportano il *brainstorming*;
- ✓ non richiedono la definizione di dettagli;
- ✓ non richiedono particolari abilità tecniche.

Svantaggi

- ❖ non si modificano facilmente;
- ❖ bisogna tradurli manualmente in forma elettronica;
- ❖ non interagiscono con l'utente.

Brainstorming: rende visibile il design e permette di confrontare rapidamente idee diverse.

Design "in bozza": mostra gli esempi importanti e non serve usarlo per coprire tutti i casi.

Paper Mock-Ups

I **mock-ups** hanno le schermate principali che possono essere disegnate a mano e i dialoghi sono inclusi (su fogli A4 separati). I **mock-ups** sono un modo semplice per verificare idee con gli utenti poiché sono limitate sull'interazione ma mostrano sequenze di schermate se animate dal presentatore.

Se sono disegnati su computer possono dare più feedback poiché danno all'utente un'idea più chiara (più precisa) di quello che si vuole realizzare.

Storyboarding

Lo **storyboarding** è un'altra semplice tecnica basata su carta, ma un po' più dettagliata. Implica l'uso di figure e testo per illustrare grossolanamente l'aspetto grafico di un insieme di schermate, inoltre, non deve essere elaborato o totalmente accurato, poiché aspira a illustrare l'idea di base (è limitato nell'interazione ma mostra sequenze di schermate).

Gli **storyboards** non sono realizzati necessariamente su supporti digitali, ma possono essere dei "paper-sketches" di interfacce utente e possono essere animati. Inoltre, non interagiscono con l'utente, ma si realizzano in poco tempo e con minime capacità tecniche.

Lo **storyboarding** mostra schermate chiave in sequenza: ogni schermata è accompagnata da testo che descrive la scena, l'interazione dell'utente e qualsiasi oggetto mediale (es. suono).

Scenari

Gli **scenari** consentono ai progettisti di esplorare il proprio pensiero. Prima di tutto, è necessario identificare i profili utente e le attività di un task, successivamente, il progettista costruisce un chiaro scenario per le attività di quel task (es. prenotare un viaggio tramite una 'agenzia di viaggi' interattiva).

Per costruire gli **scenari** bisogna assestarsi le idee sui risultati, anticipando possibili problemi (es. l'utente A farà ..., l'utente B farà ...) e riflettere sullo *scenario* per evidenziare problemi.

Flowchart

- Indicano sequenza e struttura.
- Mostrano percorsi di interazione all'utente.
- Come le schermate sono collegate le une alle altre.
- Presi da soli non sono molto utili per il design ma in combinazione con *paper mock-up* e *storyboard* iniziano a dare all'utente un quadro completo di come è strutturata un'applicazione.

Simulazione di funzionalità limitate

Gli approcci basati su carta ricadono nelle attività di testing dell'interazione: per testare questo aspetto del design l'utente deve essere in grado di interagire col design in un approccio realistico che fornisca feedback.

La **simulazione di funzionalità limitate** non è altro che un approccio semplice che consiste nel prendere un *mock-up* su carta e automatizzarlo su computer: si disegna una figura e la si rende interattiva, consentendo il testing di una sequenza di interazioni.

Questo metodo è veloce da costruire se si usano tool come *PowerPoint* o *Balsamiq*.

6.3 Fedeltà nei prototipi

La “**fedeltà**” si riferisce al livello di dettaglio:

- **Alta fedeltà (HI-FI)**: i prototipi assomigliano al prodotto finale;
- **Bassa fedeltà (LO-FI)**: sketch approssimato con molti dettagli mancanti.

Tipo	Vantaggi	Svantaggi
Low-fidelity prototype	<ul style="list-style-type: none">• Revisione rapida possibile;• È possibile dedicare più tempo al miglioramento del design prima di iniziare lo sviluppo;• Valuta più concetti di progettazione;• Utile dispositivo di comunicazione;• Proof of concept.	<ul style="list-style-type: none">• Controllo errori limitato;• Scarse specifiche dettagliate per lo sviluppo;• Facilitator-driven;• Utilità limitata per i test di usabilità;• Limitazioni di navigazione e flusso.
High-fidelity prototype	<ul style="list-style-type: none">• Funzionalità (quasi) completa;• Totalmente interattivo;• User-driven;• Definisce chiaramente lo schema di navigazione;• Utilizzare per esplorazione e test;• Look and feel previsto;• Serve come una specifica "vivente" o in evoluzione;• Strumento di marketing e vendita.	<ul style="list-style-type: none">• Più dispendioso in termini di risorse da sviluppare;• Richiedere tempo per modificare;• Inefficiente per i progetti di prova di concetto;• Potenziale di essere scambiato per il prodotto finale;• Potenziale di stabilire aspettative inadeguate

Perché usare prototipi LO-FI?

I metodi tradizionali richiedono troppo tempo:

- sketches --> **prototype** --> evaluate --> iterate
- sketches --> evaluate --> iterate

Gli **sketch** fungono da *prototipi*: il progettista interpreta il computer, mentre, gli altri membri del team osservano e registrano. Non sono richieste competenze tecniche possono partecipare non programmatori.

Tecnica del Mago di OZ

Uno dei designer riceve i comandi in input dall'utente che sta testando il prototipo e li traduce in comandi che il prototipo potrà riconoscere. Ciò consente di intervenire tra l'utente ed il sistema in modo da aumentare le funzionalità e di migliorare il prototipo nelle sue versioni successive.

6.4 Compromessi nella prototipazione

Tutti i prototipi comportano **compromessi**, in particolare, abbiamo 2 tipi comuni di compromesso:

1. **orizzontale**: fornisce una vasta gamma di funzioni, ma con pochi dettagli;
2. **verticale**: fornisce molti dettagli solo per alcune funzioni.

I compromessi nei prototipi non devono essere ignorati, bensì, il prodotto ha bisogno di essere ingegnerizzato.

6.5 Design concettuale

Il **design concettuale** si occupa è di trasformare requisiti/esigenze degli utenti in un **modello concettuale**.

Un **modello concettuale** è uno schema di ciò che le persone possono fare con un prodotto e quali concetti sono necessari per comprendere e interagire con esso.

- **Metafore e analogie** che comunicano alle persone come capire a cosa serve un prodotto e come usarlo per un'attività. Le metafore dell'interfaccia combinano **conoscenza familiare** e **conoscenza nuova** in un modo da aiutare l'utente a comprendere il prodotto.

Abbiamo 3 passaggi: comprendere la funzionalità, identificare potenziali aree problematiche, generare metafore.

Per valutare una metafora dobbiamo porci le seguenti domande:

- *Quanta struttura fornisce?*
- *Quanto è rilevante per il problema?*
- *È facile da rappresentare?*
- *Il pubblico lo capirà?*
- *Quanto è estensibile?*

- I **concetti** a cui le persone sono esposte attraverso il prodotto, inclusi gli oggetti del dominio delle attività che creano e manipolano, i loro attributi e le operazioni che possono essere eseguite su di essi (come il salvataggio, la revisione e l'organizzazione).
- Le **relazioni** tra questi concetti (ad esempio, se un oggetto ne contiene un altro).
- Le **mappature** tra i concetti e l'esperienza utente che il prodotto è progettato per supportare o richiamare (ad esempio, si può rivisitare una pagina guardando un elenco di siti visitati, visitati più di frequente o siti web salvati).

Mood Board

Nell'ambito della *User eXperience Design* la **mood board** è una raccolta di risorse e materiali destinati a comunicare lo stile, la voce, la direzione e il linguaggio di un particolare design, marchio o progetto.

Oltre questo, serve anche ad immaginare come dovrà sentirsi l'utente quando userà il nostro prodotto. Quindi, la **mood board** (tavola dell'umore) può essere usata per catturare la sensazione dell'utente.

Esplorare l'esperienza dell'utente

Si utilizzano personas, prototipi card-based o post-it per modellare la user experience. Le rappresentazioni visuali sono chiamate:

- **Mappa di design;**
- **Mappa di esperienza.**

Inoltre, abbiamo 2 rappresentazioni comuni:

1. **Ruota**;
2. **Timeline**.

Riassumendo

- Diversi tipi di prototipi vengono utilizzati per scopi diversi e in diverse fasi.
- I **prototipi** rispondono a delle domande.
- Il **prodotto finale** deve essere appropriatamente ingegnerizzato.
- Due aspetti del design: **concettuale** e **concreto**.
- Per generare un **design concettuale**, si prendono in considerazione metafore di interfaccia, tipi di interazione e tipi di interfaccia.
- Gli *storyboard* possono essere generati dagli scenari.
- I **prototipi card-based** possono essere generati da casi d'uso.
- I *kit di programmazione* e gli *SDK* facilitano il passaggio dalla progettazione alla costruzione.

Capitolo 7 – La collaborazione

7.1 Cos'è la collaborazione

La **collaborazione** è un **software** progettato specificamente per supportare il lavoro di gruppo avendo in mente dei **requisiti di cooperazione**. Non sono solo strumenti di comunicazione.

La **collaborazione** può essere classificata in base a quando e dove i partecipanti stanno lavorando o la funzione che svolge la **collaborazione** per il lavoro cooperativo.

Matrice Spazio - Tempo

Per analizzare la collaborazione, facciamo uso di una **matrice Spazio/Tempo**:

	Stesso Tempo	Tempi Diversi
Stesso luogo	face to face (classrooms, meeting rooms)	interazione asincrona (scheduling di progetto, strumenti di coordinamento)
Luoghi Diversi	sincrona distribuita (editori, video, finestre condivisi)	asincrona distribuita (email, listservs, conferenze)

Prendiamo, in esempio, un sistema di collaborazione in luoghi differenti e tempo differente.

Un esempio è la posta elettronica: esso viene detto **sistema asincrono distribuito**.

Ora consideriamo un sistema di collaborazione in luogo differente ma stesso tempo.

Un esempio è il sistema Doodle per lo scheduling di eventi: esso viene detto **sistema sincrono distribuito**.

Esse sono composte da editing di gruppo, schermate condivise per assistenza cliente, simultaneità su più siti. Un altro esempio può essere una chat, un videogioco e molto altro.

7.2 Classificazione per funzione

Un altro tipo di classificazione della **collaborazione** è per **funzione**: il lavoro cooperativo coinvolge sia i **partecipanti** che lavorano, sia gli **artefatti** sui quali lavorano.



- **Supporto a meeting e decisioni:** nel design, nella gestione e nella ricerca vogliamo:

- generare idee;
- sviluppare idee;
- registrare idee.

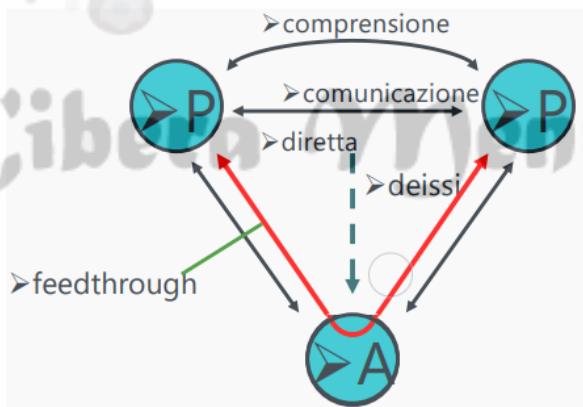
Vi sono tre tipi di sistemi:

1. **Strumenti di argomentazione: asincroni co-situati.** Hanno due scopi: ricordare ai progettisti le ragioni delle loro decisioni e comunicare il principio di base tra squadre di design. Vi sono dei problemi in questo tipo di sistema, ad esempio, cosa succede se due persone accedono allo stesso nodo? Bisogna anche tener traccia di quali cambiamenti sono stati apportati e da chi.
2. **Sale riunioni: sincroni co-situati.** Supporto elettronico per riunioni face-to-face. Possiamo fare un esempio banale, nel quale ci troviamo in uno studio dove tutti i componenti del gruppo condividono una lavagna. Anche in questo vi sono alcune complicanze del tipo, chi può scrivere e quando sulla lavagna?
3. **Superfici di disegno condivise: sincrone remote.** Lavagna di disegno condivisa a distanza. Anche in questo vi sono altri tipi di problemi, come ad esempio il problema di ritardo di connessione.

7.3 L'integrazione della comunicazione e del lavoro

Aggiunto:

- **Deissi** → riferimenti agli oggetti di lavoro;
- **feedthrough** → per la comunicazione tramite artefatti.



Capitolo 8 – I Pattern

8.1 Introduzione

Un **design efficace** e **flessibile** è difficile da ottenere al primo colpo, eppure i progettisti esperti realizzano buoni progetti, mentre, i progettisti principianti di solito sono sopraffatti da tutte le opzioni di design disponibili.

I progettisti esperti di solito non risolvono tutti i problemi partendo da zero, ma riusano soluzioni che hanno funzionato per loro in passato: quando trovano una buona soluzione, la usano ancora e ancora. Tale esperienza fa parte di ciò che li rende esperti. Questi tipi di esperienze possono essere chiamate **design pattern**.

8.2 Design Pattern Vs. Linee guida di design

Le **linee guida di design** possono essere astratte o concrete:

- le linee guida astratte di solito non suggeriscono come risolvere un problema;
- le linee guida concrete sono di solito troppo specifiche per una specifica interfaccia.

Le **linee guida** di solito assumono una **validità assoluta** mentre i **design pattern** enfatizzano l'**efficacia** in un particolare contesto.

Le **linee guida** sono più utili per **descrivere i requisiti** mentre i **pattern** sono strumenti utili per coloro che devono tradurre i **requisiti in soluzioni software specifiche**.

La struttura dei pattern di Alexander (Design Pattern Architetturali)

- **Nome** descrive l'effetto dell'uso del pattern;
- **Fotografia** mostra un esempio archetipico del pattern in uso;
- Un **paragrafo introduttivo** colloca il pattern nel contesto di altri pattern di più ampia scala;
- **Titolo** incapsulamento del problema;
- **Corpo** del problema;
- La **soluzione** indicata sotto forma di un'istruzione;
- Un **digramma** mostra la soluzione sotto forma di diagramma;
- Un **paragrafo** di chiusura mostra come questo pattern è consistente con altri pattern più piccoli.

Un **design pattern** non deve essere né troppo generale e né troppo specifico ma deve permettere di essere usato come una soluzione “*un milione di volte, senza mai farlo allo stesso modo due volte*”.

Ciascun pattern di *Alexander* possiede tale formato:

- **Nome pattern;**
- **Contesto;**
- **Forze**, cioè la descrizione del problema;
- **Dichiarazione problema;**
- **Soluzione;**
- **Altri schemi** da considerare.

Linguaggio dei Pattern

Alexander ha sottolineato l'importanza dei **linguaggi dei pattern**. Un *linguaggio* è un insieme di pattern che riempiono uno spazio di progettazione e sono scelte per completarsi a vicenda.

I pattern in un determinato dominio dovrebbero essere sempre organizzati in una struttura logica e intuitiva, dove ogni pattern dovrebbe indicare la sua relazione con altri pattern e con l'intero linguaggio.

8.3 Design Pattern HCI

Negli anni i *pattern* hanno raggiunto la comunità **HCI** (human community interface) mantenendo una forte somiglianza con i *pattern architetturali*.

A supporto di questo vi sono le **12 tesi di Borchers**:

1. L'architettura è più vicina all'HCI che all'ingegneria del software.
2. Il libro della banda dei quattro non contiene pattern.
3. *HCI* deve derivare la sua idea di pattern dalla fonte originale.
4. I linguaggi di pattern *HCI* non sono guide di stile, né regole d'oro, né standard.
5. I linguaggi dei pattern dell'architettura devono essere estesi con la nozione di struttura temporale per gestire l'*HCI*.
6. La struttura e i componenti dei singoli pattern devono tenere conto della dimensione temporale.
7. I pattern dell'*HCI* devono fornire prove empiriche della loro validità.
8. I pattern dell'*HCI* devono essere leggibili dagli utenti.
9. I pattern dell'*HCI* toglieranno il potere ai progettisti *HCI* e lo metteranno nelle mani degli utenti.
10. I pattern possono modellare molti domini di applicazioni.
11. L'uso di pattern nell'architettura software, nella progettazione dell'interazione e nel dominio dell'applicazione di un progetto può migliorare la comunicazione nei team di progettazione interdisciplinare.
12. L'uso di pattern nei vari domini può essere mappato verso la maggior parte delle fasi del ciclo di vita dell'ingegneria dell'usabilità.

Un **design pattern HCI** cattura l'essenza di una soluzione di successo per un problema di usabilità ricorrente nei sistemi interattivi.

Esso consiste di:

- **Un nome;**
- **Un grado di validità** (ranking);
- **Un esempio tangibile** (image);
- **Un contesto** (context);
- **Una descrizione del problema** (problem statement);
- **Esempi** (examples);
- **La soluzione** (solution);
- **Uno schizzo** (sketch o diagramma);
- **Riferimenti** ad altri patterns.

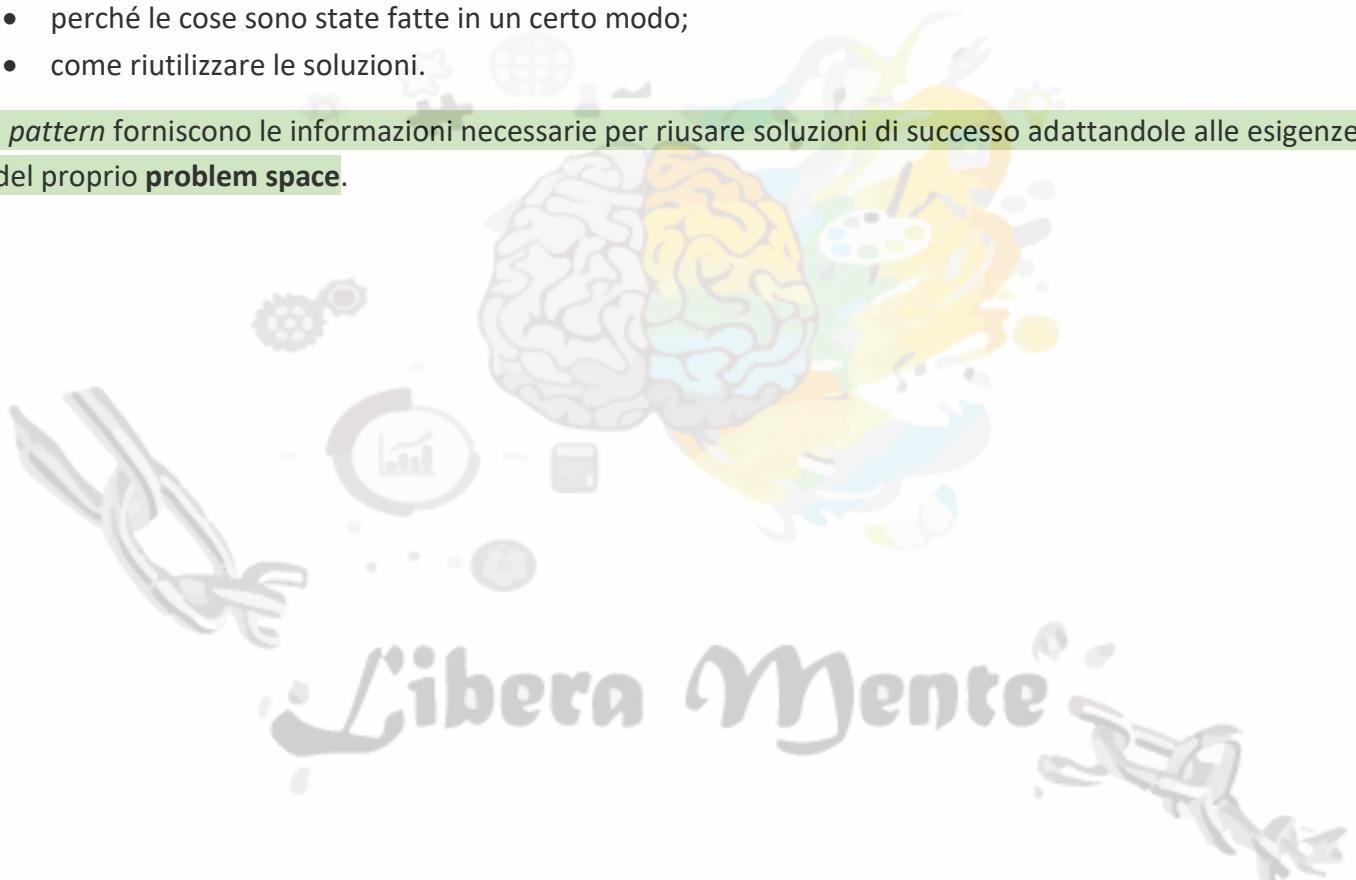
I design pattern HCI:

- colgono la prassi della progettazione non la teoria;
- colgono le proprietà essenziali comuni di un buon progetto;
- rappresentano la conoscenza della progettazione a vari livelli: sociale, organizzativo, concettuale e di dettaglio;
- incorporano valori e possono esprimere ciò che è umano nel progetto dell'interfaccia;
- sono intuitivi e leggibili e quindi possono essere usati per la comunicazione tra tutti gli stakeholder;
- un linguaggio di pattern dovrebbe essere generativo e perciò utile nello sviluppo di design completi.

Il design è trovare soluzioni ma sfortunatamente, i progettisti si trovano a reinventare poiché:

- è difficile sapere come sono state fatte le cose prima;
- perché le cose sono state fatte in un certo modo;
- come riutilizzare le soluzioni.

I pattern forniscono le informazioni necessarie per riusare soluzioni di successo adattandole alle esigenze del proprio **problem space**.



Capitolo 9 – Paradigmi e tecniche di valutazione dell’usabilità

9.1 Valutazione dell’usabilità

La **valutazione dell’usabilità** ha come obiettivo di stimare le funzionalità offerte dal sistema, stimare gli effetti dell’interfaccia sull’utente e identificare specifici problemi.

Esistono due stili di valutazione principale.

1. Indagini di Laboratorio

Il sistema viene studiato all’interno di un **laboratorio**. Solitamente si coinvolgono soltanto i designer e i valutatori, anche se gli utenti possono essere portati all’interno del laboratorio per prendere parte agli studi di valutazione.

I **vantaggi** che derivano dall’indagini di laboratorio sono:

- disporre di attrezzatura specifica (telecamere, registratori audio, meccanismi di logging etc.);
- ambiente libero da interruzioni.

Gli **svantaggi** sono:

- perdita del contesto;
- difficoltà nell’osservare la cooperazione tra diversi utenti.

L’indagine di laboratorio diventa **appropriata** se:

- la posizione del sistema è pericolosa o impraticabile;
- per task per singolo utente che presentano molti vincoli;
- per manipolare di proposito il contesto per scoprire problemi e osservare procedure meno usate;
- per confrontare design alternativi in un contesto controllato.

2. Indagini sul campo

La **valutazione** viene effettuata nell’**ambiente di lavoro** dell’utente per osservare il sistema in azione.

I **vantaggi** che derivano dall’indagine sul campo sono:

- essere in un ambiente naturale;
- mantenimento del contesto (anche se le osservazioni possono alterarlo);
- possibilità di studi che durano nel tempo.

Gli **svantaggi** sono:

- distrazioni (alto livello del rumore, suono del telefono, etc.).

L’indagine sul campo è **appropriata**:

- quando il contesto è cruciale;
- per osservazioni a lungo termine.

9.2 Valutazione del design

La **valutazione del design** viene effettuata in fase di design oppure durante l’implementazione.

Per quanto riguarda la *fase di design*, la valutazione viene fatta prima che sia iniziata la fase d’implementazione in modo tale che possono essere evitati grandi sprechi di risorse.

Soltamente i metodi di valutazione del design non coinvolgono l'utente ma vengono effettuati dal designer e da esperti valutatori.

Tali metodi non vengono applicati solamente in fase di design ma possono essere applicati anche nel processo di sviluppo o su una versione completa del sistema.

Quando si procede nella valutazione del design c'è la possibilità di chiedere agli **esperti** oppure ricorrere a un **feedback** da parte degli **utenti** tramite interviste e questionari.

Gli **esperti** usano la loro conoscenza degli utenti e della tecnologia per rivedere l'usabilità del software restituendo critiche che possono essere rapporti formali o informali.

Esistono 4 possibili approcci alla valutazione del design:

1. **Cognitive Walkthrough**: richiede di attraversare uno scenario pre-pianificato notando potenziali problemi;
2. **Valutazione euristica**: è una revisione guidata da un insieme di euristiche;
3. **Valutazione basata su revisioni**;
4. **Valutazione basata sull'uso di modelli**.

Cognitive Walkthrough (Sondaggio Cognitivo)

È basato sulla teoria dell'apprendimento esplorativo di Polson. Valuta quando il design supporta l'utente nell'apprendimento dei task e di solito è effettuato da esperti in psicologia cognitiva che usano principi psicologici analizzando il design per identificare potenziali problemi.

Tale approccio deriva dal “**code walkthrough**”, tecnica usata in ingegneria del software, riferito a una sequenza di passi che rappresentano un segmento di codice di programma.

Per effettuare un **walkthrough cognitivo** bisogna avere:

1. descrizione del prototipo del sistema;
2. descrizione di un task rappresentativo che l'utente effettua sul sistema;
3. lista completa delle azioni necessarie per completare il task utilizzando il prototipo;
4. indicazioni di chi sono gli utenti del prodotto ed il loro tipo di esperienza e di conoscenza.

I valutatori seguono passo per passo la sequenza di azioni descritte al punto 3, dove per ogni azione dovranno rispondere alle seguenti domande:

- l'azione corretta sarà sufficientemente evidente per l'utente? L'utente saprà cosa fare per realizzare il task?
- l'utente noterà che è disponibile l'azione corretta? Gli utenti possono vedere il pulsante o la voce di menu che dovrebbero usare per l'azione successiva? E evidente quando è necessario?
- l'utente assocerà e interpreterà correttamente la risposta dell'azione? Gli utenti sapranno dal feedback che hanno fatto una scelta di azioni corretta o errata?

Durante la fase di valutazione gli esperti terranno un foglio a parte dove annotare i problemi che via via vengono osservati. Successivamente tale elenco di problemi verrà alla fine consegnato ai progettisti che ne esamineranno l'importanza e decideranno se e come intervenire per migliorare il design.

Walkthrough Pluralistico

È una variazione del walkthrough cognitivo e ha lo scopo di verificare se l'interfaccia del prodotto rispetta i principi fondamentali dell'usabilità.

Valutazione Euristica

Si tratta di un **metodo ispettivo** che non prevede il coinvolgimento degli utenti finali del prodotto, ma si basa sul giudizio di più esperti di usabilità, che fanno una critica dell'interfaccia, per individuarne eventuali problemi.

È utile in particolare per la valutazione iniziale, ma può anche essere usata su prototipi, storyboard e sistemi funzionanti.

L'idea che sta alla base è che più lavoratori esaminano autonomamente e criticano un sistema per individuare potenziali problemi d'usabilità.

Per effettuare una **valutazione euristica**:

- vengono identificati i criteri di usabilità, basati su linee guida e principi di progettazione dell'interfaccia;
- il design è esaminato da esperti per vedere se i criteri sono violati.

Inoltre, per aiutare i valutatori a scoprire i problemi d'usabilità vengono fornite le **10 euristiche di Nielsen** sono:

1. Visibilità dello stato del sistema
2. Corrispondenza tra sistemi e mondo reale
3. Controllo e libertà dell'utente
4. Consistenza e standard
5. Prevenzione degli errori
6. Riconoscimento piuttosto che ricordo
7. Flessibilità ed efficienza di utilizzo
8. Design estetico e minimalista
9. Aiutare gli utenti a riconoscere, diagnosticare e recuperare dagli errori
10. Aiuto e documentazione

La **valutazione euristica** comprende 3 fasi:

1. sessione informativa per spiegare agli esperti cosa fare;
2. c'è un periodo di valutazione di 1 – 2 ore in cui ogni esperto lavora separatamente e comprende due passaggi uno per avere un'idea del prodotto e un secondo per concentrarsi su aspetti specifici;
3. avviene una sessione in cui gli esperti lavorano insieme per dare priorità ai problemi

Valutazione basata su revisioni

Consiste nel basare la valutazione sul risultato di esperimenti analoghi già documentati e presenti in letteratura, tenendo presente che i risultati sperimentali non valgono in qualsiasi contesto.

La revisione deve tener conto delle similitudini e delle differenze tra il contesto dell'esperimento e il design che si sta considerando per la valutazione.

Valutazione basata su modelli

Si usano alcuni modelli cognitivi e di design per combinare specifiche di progetto e valutazione nello stesso framework.

Il **modello GOMS** (Goals Operators Methods and Selection) predice le prestazioni dell'utente con un particolare interfaccia e può essere usato per filtrare delle particolari scelte di progetto.

Il **modello “keystroke-level”** fornisce previsioni del tempo che l'utente impiegherà a eseguire task fisici di basso livello.

Valutazione dell'implementazione

Si distingue da quella del *design* perché richiede un prototipo del sistema o una completa implementazione e coinvolge gli utenti del sistema.

Un framework per guidare alla valutazione è **D E C I D E**:

- Determinare gli obiettivi che la valutazione intende perseguire.
- Esplorare le specifiche domande cui si deve rispondere.
- Scegliere (Choose) il paradigma e le tecniche di valutazione per rispondere alle domande.
- Identificare le questioni pratiche.
- Decidere come trattare le questioni etiche.
- Valutare (Evaluate), interpretare e presentare i dati.

Si possono attuare dei **metodi empirici**, **metodi sperimentali** e **metodi osservazionali** che richiedono l'interrogazione degli utenti.

- **Metodi empirici:** si valutano specifici aspetti del comportamento interattivo, mediante la scelta di:
 - ipotesi: predizioni dei risultati dell'esperimento. È formulata in termini di variabili indipendenti e dipendenti, affermando che una variazione nella variabile indipendente causerà una variazione nella variabile dipendente. Lo scopo dell'esperimento è mostrare che la predizione è corretta. Ciò è fatto **invalidando l'ipotesi nulla**, la quale stabilisce che le variabili dipendenti non variano in funzione delle variabili indipendenti.
 - variabili indipendenti: valori che sono cambiati per produrre condizioni differenti;
 - variabili dipendenti: caratteristiche misurate dell'esperimento.

La scelta dei soggetti (non meno di 10) è effettuata tenendo presente l'utente finale del sistema e di risultati vengono trattati mediante processi statistici.

- **Metodi sperimentali**

- **Within groups design:** ogni soggetto esegue gli esperimenti sotto tutte le condizioni. Ci possono essere problemi dovuti al trasferimento di conoscenze, ma possono essere risolti variando l'ordine delle condizioni.
- **Between groups design:** ogni soggetto esegue gli esperimenti soltanto sotto una condizione. È richiesto un gran numero di utenti.

- **Metodi osservazionali**

Think aloud: l'utente è osservato durante il task e gli è chiesto di descrivere cosa sta facendo ad alta voce.

I **vantaggi** sono la semplicità nell'applicare questo metodo e fornisce informazioni utili su come il sistema è realmente usato.

Gli **svantaggi** sono i risultati sull'usabilità del sistema sono molto soggettivi.

- **Valutazione cooperativa:** variazione del *Think aloud* dove il valutatore e l'utente interagiscono durante la sessione portando l'utente a criticare il prodotto e non solo usarlo. Il valutatore può intervenire nei momenti più critici dell'interazione per indagare sulle difficoltà e verificare proposte alternative.

Le **tecniche di interrogazione** chiedono all'utente stesso come sia il sistema. Esse sono soggettive e poco costose. Abbiamo due tipo di tecniche di interrogazione:

1. **Interviste:** seguono un approccio *top-down*. I **vantaggi** sono che le domande vengono adattate al contesto in modo da esaminare i problemi a fondo. Gli **svantaggi** è che sono molto soggettive ed impiegano molto tempo.

Esse possono essere:

- **strutturate:** rigorosamente scritte come un questionario;
- **semistrutturate:** guidate da un documento scritto, ma possono essere variate al bisogno;
- **non strutturate:** non sono guidate da un documento scritto.

Abbiamo delle **fasi fondamentali** per intervistare:

- **introduzione:** presentatevi, spiegate gli obiettivi dell'intervista, rassicurate l'intervistato sulle questioni etiche, chiedete di registrare, presentate un modulo di consenso informato.
 - **riscaldamento:** formulate le prime domande semplici e non minatorie.
 - **corpo principale:** presentate le domande in un ordine logico.
 - **raffreddamento:** includete poche facili domande per allentare la tensione alla fine.
 - **chiusura:** ringraziate l'intervistato, segnalate la fine, interrompete la registrazione.
2. **Questionari:** le domande possono essere chiuse o aperte, le prime possono essere più facili da analizzare anche dal calcolatore. Possono essere somministrate a popolazioni numerose. Le risposte possono essere basate su scale di valutazione ed è importante definire i punti delle scale stesse al fine di avere una corretta valutazione.

Il più famoso questionario è il **SUMI** che ha un valore internazionale e richiede circa 5 – 10 minuti per essere compilato, le sue misure sono basate su efficienza e conoscenza.

Capitolo 10 – Accessibilità e Usabilità

10.1 Concetto di accessibilità

Si stima che in Italia vi siano circa 3,1 milioni di persone diversamente abili.

Abbiamo due principali difficoltà:

1. **difficoltà tecnologica**: accesso da parte di vari utenti o “interoperabilità”;
2. **difficoltà sociali**: accesso usabile per persone diversamente abili e accesso per persone tecnologicamente/geograficamente/socialmente svantaggiate.

Principio della progettazione universale

Un ambiente multimediale, se non realizzato con criteri di accessibilità, può presentare barriere insormontabili, o comunque difficoltà di fruizione, per coloro che non possiedono tutte le funzionalità fisiche e sensoriali.

In base a tale **principio**, va studiata la possibilità di sviluppare strumenti informatici tali da consentirci di rivolgersi a un’utenza allargata, piuttosto che al classico “utente medio”.

Favorisce l’adozione di soluzioni, per i prodotti informatici, che mettano a disposizione degli utenti con problemi particolari modalità di uso alternativo, ma non per questo “speciale”, perché facenti parte delle scelte di riconfigurazione, disponibili sul prodotto stesso.

Gli utenti possono avere: disabilità visive, motorie, uditive, cognitive e in genere disabilità che aumentano con l’età. Le varie barriere incontrate sono:

- per i **non vedenti**: immagini che non hanno un testo alternativo, complesse, video che non possiede una descrizione con testo o con audio, tabelle che non hanno senso se lette in modo seriale, form che non possono essere compilati passando da un campo a un altro col tasto di tabulazione seguendo una sequenza di logica, formati di documenti che potrebbero essere difficili da interpretare da parte dei lettori di schermo;
- per **persone con vista scarsa**: pagine web con dimensioni di font fisse, che non cambiano, layout inconsistente ovvero difficile da navigare quando ingrandito, contrasto povero;
- per **persone con cecità ai colori**: colori usati come unico marcatore nell’enfatizzare il testo;
- per **persone con disabilità motorie**: limiti di tempo nell’accettazione di una risposta da parte dell’utente sulle pagine web, moduli online che non possono essere compilati passando da un campo all’altro secondo un ordine logico, browser che non consentono alternative da tastiera ai comandi del mouse;
- per **persone non udenti**: uso di frasi lunghe, di un vocabolario inusuale o complesso, mancanza di didascalie o di trascrizioni audio/video, mancanza di immagini legate ai contenuti in pagine pieno di testo, che possono rallentare la comprensione per le persone che usano il linguaggio dei segni piuttosto che una lingua scritta/parlata;
- per **persone con difficoltà cognitive**: mancanza di modalità alternative per le informazioni, testo alternativo, didascalie per l’audio.

Un **grandissimo errore** è creare siti pensati per determinati browser: scripting, Java, plug-in e cookies non funzionano con browser ad accesso speciale.

10.2 Tecnologia assistiva

Attualmente gli strumenti informatici sono resi accessibili con adattamenti applicativi come l'aggiunta di un "lettore di schermo" con riga braille per i ciechi e con l'aggiunta di un programma di ingrandimento dei contenuti dello schermo per gli ipovedenti.

Il termine "**dispositivo di tecnologia assistiva**" si riferisce a qualsiasi oggetto, attrezzatura, o prodotto HW/SW, sia questo di nuova acquisizione, modificato o adattato, che venga usato per aumentare e mantenere o migliorare le capacità funzionali di individui diversamente abili.

Abbiamo **4 principi** nel design per l'accessibilità:

1. **Percettibile**: *l'utente riesce a vedere le informazioni?*
2. **Operabile**: *vi sono alcuni elementi all'interno dell'interfaccia che sono interattivi?*
3. **Comprensibile**: *le informazioni presenti sono facili da capire?*
4. **Robusto**: *il contenuto e l'interfaccia possono essere facilmente interpretati da interfacce personalizzate?*

10.3 La legge Stanca

Tutela e garantisce il diritto di accesso ai servizi informatici e telematici della pubblica amministrazione e ai servizi di pubblica utilità da parte delle persone disabili, in ottemperanza al principio di uguaglianza:

"tutti i cittadini hanno pari dignità sociale e sono eguali davanti alla legge, senza distinzione di sesso, di razza, di lingua, di religione, di opinioni politiche, di condizioni personali e sociali".

La legge definisce "**accessibilità**" la capacità dei sistemi informatici di erogare servizi e fornire informazioni fruibili, senza discriminazioni, anche da parte di coloro che a causa di disabilità necessitano di tecnologie assistive o configurazioni particolari.

La legge definisce "**tecnologie assistive**" gli strumenti e le soluzioni tecniche, hardware e software, che permettono alla persona disabile, superando o riducendo le condizioni di svantaggio, di accedere alle informazioni e ai servizi erogati dai sistemi informatici.

10.4 Il W3C e la Web Accessibility Iniziative (WAI)

Il **World Wide Web Consortium (W3C)** è l'organismo internazionale, senza fini di lucro, che dal 1994 ha il compito di definire i linguaggi e le procedure standard per rendere il web uno strumento democratico e universale.

W3C ha promosso la **WAI** (Web Accessibility Initiative). Lo scopo è quello di individuare e suggerire criteri per la realizzazione dei siti web, tali da permetterne l'accesso anche ad utilizzatori disabili.

Le componenti della **WAI** sono:

- **Web Content Accessibility Guidelines (WCAG)**: tali linee guida per l'accessibilità dei contenuti Web spiegano come realizzare contenuti per il Web in modo che siano accessibili a persone diversamente abili;
- **Authoring Tool Accessibility Guidelines (ATAG)**: i documenti sulle ATAG spiegano come rendere i tool di authoring accessibili a persone diversamente abili. Gli **authoring tool** sono strumenti usati per produrre pagine web e contenuti web. Un obiettivo primario di ATAG è di definire come tali strumenti aiutino a produrre contenuti web conformi alle *Web Content Accessibility Guidelines*;

- **User Agent Accessibility Guidelines (UAAG)**: i documenti della UAAG spiegano come rendere gli user agent accessibili alle persone diversamente abili, in particolare per aumentare l'accessibilità ai contenuti web. Gli ‘*user agents*’ includono i browser web, i media player e la tecnologia assistiva, che alcune persone con disabilità usano per interagire con i computer;
- **Evaluation and Report Language (EARL)**: l’EARL è un linguaggio general-purpose per esprimere risultati di test. Gli strumenti per la valutazione dell’accessibilità del web possono usare questo linguaggio come uno standard per esprimere i risultati di un test in un formato indipendente dalla piattaforma;
- **Accessibility Information for Specific Technologies**: collegamenti alle informazioni sull’accessibilità dell’*XML*, *SVG*, *SMIL*, ed altre tecnologie specifiche.

Per ciascuna linea guida, il **WAI** definisce delle regole più dettagliate da applicare per soddisfarla (“checkpoint”). Essi sono in tutto 65 e sono classificati in tre livelli di priorità:

Priorità 1: chi realizza il sito deve soddisfare questi checkpoint;

Priorità 2: chi realizza il sito dovrebbe soddisfare questi checkpoint;

Priorità 3: chi realizza il sito potrebbe soddisfare questi checkpoint.

Se il livello **WAI** è:

A – tutti i checkpoint di priorità 1 sono soddisfatti;

AA - tutti i checkpoint di priorità 1 e 2 sono soddisfatti;

AAA - tutti i checkpoint di priorità 1, 2 e 3 sono soddisfatti.

10.5 Relazione tra accessibilità e usabilità

Usabilità: si riferisce alla progettazione/creazione di siti web e applicazioni interattive che siano intuitivi da navigare o da utilizzare, consistenti nell’aspetto e che diano informazioni rilevanti;

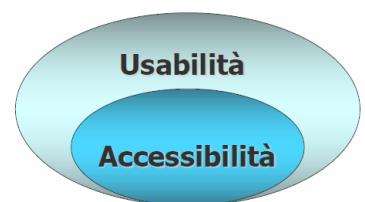
Accessibilità: si riferisce alla progettazione/creazione di siti web e applicazioni interattive fatta in modo che possano essere visti e compresi da tutti gli utenti eventualmente su diverse piattaforme tecnologiche.

Un **sistema interattivo accessibile** è più usabile e un **sistema interattivo usabile** è più accessibile perché entrambe promuovono un buon design.

Usabilità e **accessibilità** sono approcci alla progettazione compatibili.

Condividono la preoccupazione per un design universale come base per una buona progettazione. Esse condividono anche alcuni metodi e tecniche ma potrebbero e dovrebbero essere maggiormente in relazione tra loro.

L’**accessibilità** può essere vista come un sottoinsieme dell’**usabilità**.



Falsi miti sull’Usabilità e sull’Accessibilità

Creare un equivalente ‘text-only’ è sufficiente per l’accessibilità del web: fornire agli utenti diversamente abili un sito accessibile ‘separatamente’ può diventare un altro modo per far sentire loro emarginati rispetto al resto della società.

Siti web accessibili non sono attraenti e user friendly: un sito web accessibile e ben pianificato non deve alterare in alcun modo il buon design di un sito web.

Le persone diversamente abili non sono la mia ‘*target audience*’: rendere siti web o applicazioni web accessibili non risponde solo alla necessità di raggiungere gli utenti diversamente abili. L’accessibilità piuttosto migliora l’usabilità e fornisce un’eccellente esperienza per i vostri utenti.

