

Interazione uomo macchina

- **Obiettivi dell'analisi dei requisiti**

Stabilire la necessita dell'utente e analisi dei task per assicurare adeguate funzionalità

Stabilire le necessita dell'utente vuol dire definire quali task e sotto task devono essere eseguiti e tenendo presente anche task che sono eseguiti occasionalmente.

Inoltre, bisogna comprendere che le funzionalità dovranno riflettere quelle che sono le necessita raccolte in fase di raccolta dei requisiti.

Affidabilità, disponibilità, sicurezza e integrità dei dati

Quando si parla di integrità dei dati vuol dire che le azioni dovranno funzionare come specificato, i dati visualizzati devono riflettere il reale contenuto del database sottostante e mitigare il senso di sfiducia da parte dell'utente.

Quest'ultima caratteristica significa incoraggiare l'utente a usare il sistema anche nell'esplorare operazioni ancora non note poiché il sistema lo protegge da eventuali errori con messaggi di warning.

Per disponibilità e affidabilità si intende che il sistema dovrebbe essere disponibile più frequentemente possibile e non dovrebbe introdurre errori, per quanto riguarda la sicurezza invece bisogna assicurare la privacy dell'utente dando la sicurezza all'utente che non verranno divulgati i propri dati.

Standardizzazione, Integrazione, Consistenza

Standardizzazione: usare standard industriali per-esistenti per aiutare nell'apprendimento ed evitare errori.

Integrazione: il prodotto deve essere eseguito all'interno di un ambiente ben definito ovvero all'interno di diversi strumenti e pacchetti software.

Consistenza: può riguardare la compatibilità tra diverse versioni dello stesso prodotto, oppure tra sistemi su carta e non ancora disponibili su computer, utilizzare un uso di comuni sequenze di azioni, termini, unita, colori all'interno del programma.

Pianifica e Budget

Bisogna completare il progetto in tempo e rientrare nei costi poiché consegnare un prodotto in ritardo vuol dire un fallimento e può indicare un meno soddisfacimento del cliente.

- **Comunità utente**

Una delle prime cose da fare nell'analisi dei requisiti è definire la comunità utente 'target' associata all'interfaccia tenendo anche conto che le comunità si evolvono e cambiano.

La comunità può essere valutata mediante cinque fattori:

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:** frequenza d'uso e la facilità di apprendimento che favoriscono una miglior iterazione dell'utente
5. **Soddisfazione soggettiva:** consentire feedback da parte dell'utente tramite interviste, commenti liberi e scale di soddisfazione

- **Requisiti di usabilità**

L'usabilità richiede il project management (organizzazione e metodologia) e un'attenta cura dell'analisi dei requisiti, in relazione alla comunità target di utenti e al contesto in cui il sistema dovrà essere utilizzato, con un testing che miri ad avere obiettivi ben definiti.

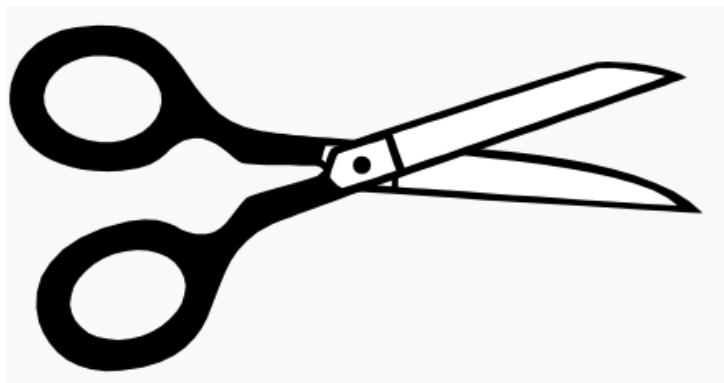
- **Modelli Mentali**

Le persone possiedono un modello mentale di come funzionano le cose:

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

Poiché tali modelli gli permettono di fare previsioni su come funzioneranno le cose.

I Modelli Mentali vengono costruiti a partire da (prendiamo anche in esempio le forbici per capire meglio cosa significa ognuno di essi):



- **Affordances:** ciò che l'ambiente offre, ovvero quelle caratteristiche dell'oggetto che offrono delle funzionalità. Nel caso delle forbici le

caratteristiche dell'oggetto che offrono delle funzionalità sono: fori per inserire le dita e lame per tagliare.

- **Vincoli:** sono degli indizi che servono a comprendere come un oggetto può essere utilizzato e limitano le azioni possibili. Possono essere fisico, culturale, semantico e logico. Nel caso delle forbici i vincoli che possiamo individuare sono: foro grande per più dita, foro piccolo per il pollice.
- **Mapping:** rapporto tra azioni e risultati, tra comandi e i loro effetti.
- **Positive transfer:** conoscenze acquisite in situazioni analoghe. Nel caso delle forbici, sappiamo che il loro utilizzo viene appreso da bambini.
- **Associazioni/Standard culturali**
- **Istruzioni**
- **Interazioni:** Comunicazioni tra l'utente e il sistema, e consigliabile l'utilizzo di un framework poiché permette la contestualizzazione e presenta una vista globale

Il modello concettuale quindi delle forbici è chiaro in quanto le implicazioni su come le parti delle forbici operano sono, appunto, chiare.

• **Ciclo di Norman**

L'azione umana ha due aspetti, esecuzione e valutazione.

Esecuzione vuol dire fare qualcosa, mentre valutazione è il confronto tra ciò che è accaduto e ciò che si desiderava accadesse.

Diciamo che la prima ha effetti sul mondo mentre la seconda valuta la risposta del mondo.

L'esecuzione ha tre stadi:

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

Successivamente si esegue l'azione e in fine si effettua una valutazione.

La valutazione ha tre stadi:

1. Percepisco il mondo;
2. Interpreto ciò che ho percepito;
3. Confronto con le intenzioni di partenza.

• Framework dell'interazione

Il framework dell'interazioni di Norman è un modello basato su azioni (ciclo esecuzione-valutazione):

1. L'utente stabilisce un obiettivo;
2. Formula l'intenzione;
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 (Golfo dell'esecuzione);
7. Valuta lo stato del sistema rispetto all'obiettivo (Golfo della valutazione).

Il Golfo dell'Esecuzione quantifica il grado con cui il sistema risponde alle azioni dell'utente ovvero "La formulazione delle azioni da parte dell'utente trova corrispondenza con le azioni permesse dal sistema?".

Se le azioni permesse dal sistema coincidono con quelle delle intenzioni dell'utente, l'intenzione sarà effettiva.

Il Golfo della valutazione quantifica lo sforzo richiesti a una persona per interpretare, lo stato fisico del Sistema e fino a che punto le aspettative e le intenzioni sono state soddisfatte.

Sia per il golfo dell'esecuzione e il golfo della valutazione si preferisce avere un piccolo golfo.

• Fattori Umani

La maggior parte degli odierni sistemi sono progettati male dal punto di vista dell'interazione uomo-macchina.

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

Considera cose come:

- La disposizione dei controlli e dei display;
- L'ambiente circostante;
- Questioni di salute (postura, luminosità, rumore ecc.);
- L'uso dei colori.

L'interazione può essere vista come un dialogo tra il computer e l'utente.

Alcune applicazioni hanno tipi di interazione molto diverse, ma possiamo identificare alcuni stili comuni.

- **Interfaccia a linea di comando**

E' un modo per esprimere istruzioni al computer direttamente. Possono essere tasti di funzione, singoli caratteri, abbreviazioni, parole intere o una combinazione.

E' indicata per task ripetitivi e possiede alcune proprietà:

- Meglio per utenti esperti che per principianti
- Offre un accesso diretto alle funzionalità del sistema
- I nomi/abbreviazioni di comandi dovrebbero essere significativi

- **Menu**

Abbiamo un insieme di opzioni disposte sullo schermo.

Le opzioni sono visibili e richiedono quindi meno memoria poiché contano sul riconoscimento e quindi su nomi significativi.

Spesso le opzioni sono raggruppate gerarchicamente e quindi è necessario un lavoro delicato di raggruppamento.

I sistemi a menu possono essere:

- Puramente basati sul testo, con opzioni presente come scelta numerata
- Possono avere un componente grafico, con il menu che compare in un box e le scelte vengono effettuate o digitando la lettera iniziale oppure muovendo tasti freccia.

- **Form-Fill e Spreadsheets**

I Form-Fill riproducono lo schema di un modulo di 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. L'utente può immettere ed alterare i valori delle celle ma la formula rimarrà sempre corretta.

- **Linguaggio Naturale**

Si utilizza il riconoscimento vocale o della scrittura in quanto familiare, ma esso porta a molti problemi in quanto è vago ed ambiguo.

- **Interfacce question/answer e Interfacce di interrogazione**

Utilizzando interfacce question/answer l'utente è condotto attraverso l'interazione mediante una serie di domande.

Tale iterazione è:

- Adatta per utenti inesperti ma di funzionalità ristrette;
- Spesso viene usata nei sistemi informativi.

Un'interfaccia d'interrogazione invece utilizza dei linguaggi di interrogazione (es. SQL) usati per ritrovare informazioni in un database.

Nell'ottenere un'efficiente iterazione però richiede la comprensione della struttura del database e della sintassi del linguaggio, quindi richiede qualche livello di esperienza.

- **Interfacce <<From Filling>>**

Consistono nel riempimento di un modulo, principalmente per l'immissione o il ritrovamento di dati.

Lo schermo simula il modulo di carta e i dati vanno collocati in posti rilevanti.

Tale interazione richiede una buona progettazione e ovvie facility per la correzione.

- **WIMP**

- **Windows**

Aree dello schermo che si comportano come se fossero terminali indipendenti. Possono contenere testo o grafica, possono essere spostate o ridimensionate, possono sovrapporsi e oscurarsi l'un l'altra oppure essere disposte l'uno a fianco all'altra. Le barre di scorrimento consentono all'utente di scorrere il contenuto della finestra verticalmente o orizzontalmente e la barra dei titoli descrivono il nome della finestra.

- **Icons**

Piccola figura o immagine, usate per rappresentare qualche oggetto dell'interfaccia. Le finestre possono essere ridotte a questa piccola rappresentazione (iconizzate), permettendo di accedere a più finestre.

Le icone possono essere molte e varie: altamente stilizzate o rappresentazioni realistiche.

- **Menus**

Scelta offerta sullo schermo di operazioni o servizi che possono essere eseguiti. Vi sono diversi tipi di menu:

- 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.

- **Pointers**

È una componente importante, poiché lo stile WIMP si basa sul puntamento e la selezione di cose.

- **Widgets**

- Bottoni: regioni individuali ed isolate all'interno di un display, che possono essere selezionate per invocare un'azione;

- Bottoni radio: insieme di scelte mutuamente esclusive;

- Check box: insieme di scelte non esclusive;

- Palette: indicano un insieme di modalità possibili disponibili, oltre a quella corrente. In genere si presentano come una collezione di icone disposte a mattonella.

- **Paradigmi per l'usabilità**

- **Paradigmi**

Esempi di sistemi interattivi di successo. La storia del progetto di sistemi interattivi fornisce paradigmi per progetti usabili.

- **Metafore**

Mette la computazione in relazione con altre attività del mondo reale e può essere considerata effettivamente una tecnica di apprendimento.

Per esempio la gestione dei file su una scrivania d'ufficio, l'elaborazione di testi come una macchina da scrivere e la realtà virtuale.

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

I vantaggi sono:

- Apprendimento più facile;

- Possono essere innovative in modo da rendere il mondo del computer più accessibile;

- Aiutano gli utenti a comprendere il modello concettuale sottostante.

Gli svantaggi sono:

- Possono violare le regole convenzionali e culturali (cestino sul desktop);

- Conflitti con i principi di progettazione (forzare troppo la metafora rende il sistema poco chiaro);

- Si possono utilizzare dei cattivi design esistenti (utilizzo dei cattivi esempi nella metafora);

- Limitano l'immaginazione del progettista (limitarsi alla metafora esclude soluzioni innovative);

- Gli utenti comprendono il sistema solamente in termini della metafora (si tenta di rapportare la stessa metafora su sistemi diversi).

○ **Manipolazione Diretta (DM)**

Metafora del mondo ideale: l'interfaccia non è un mediatore tra utente e sistema, ma dal punto di vista dell'utente è il sistema stesso. Le operazioni eseguite sull'interfaccia evitano la necessità di comprenderne il significato ad un livello più profondo del sistema.

Non c'è una netta distinzione fra un linguaggio di input e un linguaggio di output. Le espressioni di output vengono infatti utilizzate per articolare espressioni di input. Ad esempio, l'icona del documento è l'output, ma viene utilizzata come input per l'operazione di spostamento. I widgets sono così soggetti di output e input.

Correlato alla DM è il paradigma "What You See Is What You Get" dove ciò che conta nelle interfacce è la semplicità e l'immediatezza della

corrispondenza tra rappresentazione e prodotto finale.

I suoi principi di base sono:

- Continua rappresentazione degli oggetti e azioni d'interesse;
- Azioni fisiche e presenza di pulsanti anziché impartire comandi con una sintassi complessa;
- Rapide azioni reversibili con feedback immediato.

Shneiderman definisce la DM 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.

Vantaggi:

- Gli utenti inesperti imparano rapidamente le funzionalità di base ed acquisiscono sicurezza;
- Gli utenti esperti portano avanti in modo molto veloce più task contemporaneamente;
- Messaggi di errore raramente necessari.

Svantaggi:

- Non tutte le azioni possono essere eseguite direttamente;
- Alcuni task sono svolti in maniera più efficace se si delega il sistema.

● **Paradigma delle azioni**

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.

Le interfacce a manipolazione diretta implicano il paradigma delle azioni.

- **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 queste istruzioni vengano eseguite.

A questo paradigma linguistico si possono associare due interpretazioni significative:

- La prima richiede che l'utente capisca il funzionamento del sistema di base e l'interfaccia come interlocutore non debba eseguire molta traduzione. Questa interpretazione di paradigma linguistico è simile al tipo d'interazione che esisteva prima che fossero introdotte le interfacce di manipolazione diretta.
- La seconda non esige che l'utente comprenda la struttura di base del sistema 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.

- **Computer Supported Cooperative Work (CSCW)**

Con l'arrivo delle reti di computer fu possibile la comunicazione tra macchine diverse. Un risultato di ciò fu l'affermarsi delle tecniche di collaborazione tra singoli via computer, il cosiddetto **Computer Supported Cooperative Work** (CSCW). La principale differenza fra i sistemi CSCW ed 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, un'altra 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 di più di settimane.

- **World Wide Web (WWW)**

Il web è costruito sopra 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.

- **Multimodalità**

Combina molte modalità di input simultanee che possono presentare le informazioni in molte modalità di output diverse.

Un modo è un canale di comunicazione umano. Particolare enfasi va posta sull'uso simultaneo di più canali di comunicazione per l'input e per l'output.

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.
Il computer come partner di dialogo: le modalità multiple sono usate per aumentare l'antropomorfismo della macchina (riconoscimento parlato).
Output come teste parlanti e altre modalità "Human like".

- **Aspetti Fisici e Cognitivi**

L'aspetto cognitivo (la componente mentale) interviene in situazioni che danno luogo ad emozioni, mentre l'aspetto fisico (la componente fisica) è una risposta fisiologica che avviene con un'emozione o la segue immediatamente.

L'aspetto cognitivo viene scatenato con:

- Interfacce espressive;
- Aspetti negativi;
- Antropomorfismo e agenti di interfaccia;
- La progettazione di personaggi sintetici.

- **Interfacce Espressive**

Colori, icone, elementi grafici e animazioni sono usate per rendere attraente il "look and feel" di un'interfaccia, comportando così uno stato emotivo.
Ciò può influire sull'usabilità di un'interfaccia.

- **Agente di Interfaccia**

L'agente di interfaccia amichevole è un tipo di interfaccia espressiva (es. Microsoft-Clippy).

Il ruolo di un agente di interfaccia è quello di mettere gli utenti più a loro agio e non farli sentire in difficoltà, ma alcuni utenti non li amano.

- **Antropomorfismo**

L'antropomorfismo ha lo scopo di attribuire qualità "umane" a oggetti inanimati. È un fenomeno ben noto nella pubblicità e molto sfruttato nell'interazione uomo macchina poiché rende più divertente l'esperienza dell'utente, lo rende più motivato e riduce l'ansietà.

Vantaggi:

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

Svantaggi:

- 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.

Reeves e Naas nel 1996 hanno dimostrato che i computer che adulano e si complimentano con l'utente nei programmi per l'education hanno un impatto positivo su di loro, ma non è sempre così poiché in alcuni casi possono far provar ansietà agli utenti.

La gente tende a non amare personaggi dello schermo che puntano il dito verso l'utente ma molti preferiscono un aspetto più impersonale.

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.

• **Personaggi Virtuali**

Sono sempre più diffusi sui nostri schermi (web, personaggi dei videogiochi, compagni di apprendimento...), forniscono una persona che dà il benvenuto, ha personalità e coinvolge l'utente.

Tuttavia, hanno un forte svantaggio:

- Portano la gente a un falso senso di confidenza, a parlare di fatti personali con personaggi virtuali;
- A volte noiosi e frustranti;
- Non Affidabili.

Gli agenti rientrano nei personaggi virtuali e si possono classificare sul grado di antropomorfismo che esibiscono:

- **Personaggi sintetici**
Autonomo, con stati interni e in grado di rispondere a eventi esterni.
- **Agenti animati**
Giocano un ruolo collaborativo sull'interfaccia.
- **Agenti emotivi**
Personalità predefinita e un insieme di emozioni che l'utente può cambiare.
- **Agenti conservativi emotivi**
Caratteristiche fisiche “Human Like” e sofisticate intelligenze artificiali per permettere l'interazione.

• **Punti chiave sull'antropomorfismo**

- Gli aspetti “affettivi” riguardano il tipo di risposta emotiva che provocano nel gene i sistemi interattivi.
- Interfacce ben disegnate comportano sensazioni positive negli utenti.
- Interfacce espressive possono rassicurare.
- Interfacce mal progettate possono provocare rabbia e frustrazione negli utenti.
- Antropomorfismo sempre più usato sotto forma di agenti e personaggi virtuali.

• **Messaggi di errore**

La scelta dello stile di interazione più appropriato non basa a garantire l'usabilità dell'interfaccia, la maggior parte delle frustrazioni dell'utente nascono quando l'applicazione non funziona o va in crash, oppure quando un sistema non fa ciò che l'utente vuole che faccia, quando compaiono messaggi di errore vaghi, ecc...

Le linee guida di Shneiderman per i messaggi di errore includono:

- Evitare termini come FATAL, INVALID, BAD;
- Tono positivo e suggerimenti costruttivi;
- Evitare MAIUSCOLE e codici con lunghi numeri;
- I messaggi dovrebbero essere precisi piuttosto che vaghi;
- Fornire aiuto relativo al contesto.

• **Fase di Analisi**

Per decidere il design di un sistema dobbiamo tener conto di chi sono i nostri utenti, che tipo di attività svolgono nel contesto in cui operano e dove ha luogo l'attività. In modo tale da cercare di progettare l'esperienza interattiva in maniera che sia efficace e soddisfacente il sistema.

Il processo di sviluppo per queste interfacce parte dalla fase di analisi in cui:

- capiremo chi sono gli stakeholders e in particolare chi sono i nostri utenti;
- cercheremo di tracciare degli scenari che ritraggono il contesto attuale e li analizzeremo per individuare le difficoltà che gli utenti incontrano nell'eseguire le attività, ovvero capire cosa potrebbe essere migliorato con l'introduzione del nostro sistema e inquadrare gli aspetti positivi da mantenere nel nostro sistema.

Durante la fase di analisi ci sono cinque questioni chiave da considerare:

1. **Fissare gli obiettivi** concordando che cosa è importante osservare/studiare e decidere come analizzare i dati una volta raccolti;
2. **Identificare i partecipanti** ovvero capire chi sono le persone rilevanti che possono fornirci i dati utili a identificare le esigenze di quel determinato contesto;
3. **Stabilire il tipo di relazione con i tipi di partecipanti allo studio**, mantenendo una relazione chiara e professionale, rendendo chiaro l'obiettivo di progetto;
4. **Adottare tecniche di triangolazione nell'analisi**, guardando i dati da più prospettive, documentandosi con fonti diverse e confrontare i dati con le fonti diverse, incrociando questi ultimi con le documentazioni disponibili;
5. **Stabilire uno studio pilota** (indagine di contesto), prima di partire col processo di analisi, stabiliamo un'indagine di contesto che ci serve a non sprecare tempo e capire se su un piccolo gruppo stiamo effettivamente

raccogliendo i dati che ci servono a soddisfare i nostri obiettivi principali. Nel condurre tale studio dobbiamo cercare di comprendere le esigenze degli utenti, partendo dalle osservazioni di, (che cosa le persone sanno fare o non sanno fare?), cercando di capire cosa potrebbe aiutarli a migliorare le attività che già svolgono.

L'ideale fondamentale dell'analisi che andremo a fare, e quella di poter osservare le persone mentre svolgono i loro compiti e poter dialogare con loro mentre lo fanno. Tutto ciò ci permette di attuare un approccio user-friendly

Nell'identificare gli utenti bisogna tener conto della diversità della natura umana:

- Vincoli tra capacità fisiche e luoghi in cui lavorano
- Capacità cognitive e percettive
- Differenze nella personalità
- Diversità culturale e internazionale
- Utenti disabili
- Utenti anziani

Inoltre, è importante considerare il contesto sociale/organizzativo in cui l'iterazione deve avere luogo, capendo se certe attività devono svolgersi con presenza di altre persone e se tali persone influenzano l'esecuzione dell'attività.

Tutto ciò serve a progettare il nostro sistema, in modo tale da non creare nell'utente frustrazioni e migliorare la sua esperienza col sistema a lungo termine.

• **Analisi degli stakeholder**

Nella progettazione dell'interfaccia dovremo considerare tutte le possibili necessità delle persone e di chi abbia interesse che il sistema venga messo in uso, le quali potrebbero essere influenzate dalle decisioni di progetto.

Tra gli stakeholder ci saranno gli utenti finali, ovvero il gruppo critico su cui si focalizzerà il team di progetto per individuare il target di applicazione.

Tra gli utenti finali rientrano:

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

• **Personas**

I profili utenti prendono il nome di personas, ovvero personaggi inventati che servono ad avere un intero cast di profili utente, rispetto ai quali si possono testare idee e concetti del design.

Una personas è descritta attraverso una mini-biografia che ne identifica: l'età, il genere, le capacità formative, il livello d'istruzione, la dimestichezza con la tecnologia. Inoltre, descrive una situazione realistica nel contesto che abbiamo osservato includendo obiettivi e motivazioni.

Utilizzare tale descrizione incoraggia il team di progettisti a mettersi nei panni dell'utente e fare delle scelte che potrebbero accontentare quella persona.

Tipicamente vanno identificati diversi tipi di personas, ciascuno dei quali può rappresentare diversi gruppi d'utenti finali, non preoccupandoci di essere politically correct.

Nella descrizione del personaggio oltre alla mini-biografia dovremo includere un comportamento e le motivazioni di tale comportamento, dando dei tratti che ci permettono di comprendere le reazioni dell'utente rispetto al sistema in modo da poter valutare decisioni di design.

La costruzione dei personaggi deve avvenire a valle del processo di definizione che avviene durante l'indagine di contesto. Identificando il range dei possibili utenti, parlando con stakeholder o analizzando la documentazione disponibile.

Dopo di che andremo a restringere la lista assicurandoci di interagire con l'intera gamma di potenziali utenti, creando dei gruppi e definendo un personaggio per ciascun gruppo.

Ciascun gruppo avrà in comune gli obiettivi di esperienza, finali, collegiali, pratici, falsi obiettivi che andranno catturati nella descrizione del personaggio

• **Sorgenti per la raccolta dei dati**

La raccolta dei dati può essere attuata mediante diverse tecniche:

Visite sul posto

Pianificazione:

1. Obiettivi ed argomenti: Cosa si intende apprendere attraverso la visita sul luogo;
2. Partecipanti: Chi si vuole osservare o intervistare. Quanti partecipanti includere. Quale tipo di utente vorrebbe partecipare;
3. Sedi: Dove svolgere l'intervista o l'osservazione;
4. Programmazione delle visite: Quando si potrebbe andare sul posto. Quanto tempo si prevede che si possa dedicare a ciascuna visita;
5. Selezione: Come trovare le persone che si ha bisogno di incontrare? Come convincerle a partecipare;
6. Tecniche di raccolta dei dati: Quali tecniche utilizzare per raccogliere i dati. Si parteciperà attivamente alla sessione o si osserverà solamente?
7. Media: Useremo videoregistratori, registratori audio, carta, penna, moduli prestampati?

Attuazione: chiedere all'utente di pensare ad alta voce, osservare, parlare, ascoltare l'utente ed annotare le caratteristiche interessanti per il progetto:

- Prendere nota dell'ambiente di lavoro di ciascun utente;
- Capire gli obiettivi degli utenti;
- Capire i compiti degli utenti;
- Annotare quali fattori determinano l'esecuzione dei compiti, la situazione all'inizio dei compiti;
- Separare le osservazioni dall'inferenza durante l'osservazione;
- Fare attenzione alle interazioni con altre persone, altri programmi, documenti;
- Osservare cosa succede quando l'utente ha terminato il compito.

Interviste

Il tipo di informazione da raccogliere è un fattore rilevante per determinare come strutturare l'intervista o come formulare le domande.

Bisogna identificare i bisogni degli utenti e i loro modelli di comportamento

Andremo ad intervistare gli utenti finali del prodotto utilizzando interviste singole per ottenere risposte più dettagliate e interviste di gruppo consentendo dialoghi tra gli utenti. Effettuandole nell'ambiente di lavoro dell'utente poiché permette all'intervistatore di osservare direttamente il contesto d'uso.

User survey

Metodo appropriato per reperire informazioni utili in diverse fasi di sviluppo di un'applicazione. considerato il punto di partenza per il processo di analisi dei requisiti, prima ancora dell'osservazione diretta dell'utente.

- **Questionari**
- **Indagini per telefono**
- **Indagini via Web**
- **Analisi documentazioni esistenti**

• Check list dei profili utenti

Gli individui apprendono in maniera diversa l'uno dall'altro secondo le modalità e le strategie con cui elaborano 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, la memoria e le preferenze di risposta

CHECKLIST DEI PROFILI UTENTE 1/3

Caratteristiche Psicologiche

Stile Cognitivo	Attitudine	Motivazione
Sistematico/Intuitivo	Positivo	Alta
Globale/Analitico	Neutrale	Moderata
Impulsivo/riflessivo	Negativo	Bassa
Verbale/Spaziale		

Conoscenze ed esperienza

Livello di lettura	Esperienza battitura	Istruzione
Minore al 5° grado	Bassa	Media Inf
Tra il 5° e il 12° grado	Media	Media Sup
Oltre il 12° grado	Alta	Università
Esperienza col sistema	Esperienza sul task	Esperienza di applicazioni
Esperto	Novizio del settore	Nessun sistema simile
Intermedio	Intermedio	Un sistema simile
Novizio	Esperto del settore	Alcuni sistemi simili
Linguaggio	Uso di altri sistemi	Conoscenza di informatica
Italiano	Poco o nessuno	Alta
Inglese	Frequente	Media
Altri		Bassa

CHECKLIST DEI PROFILI UTENTE 2/3

Caratteristiche di Lavoro e Task

Frequenza d'uso	Addestramento	Uso del Sistema
Bassa	Nessuno	Obbligatorio
Media	Solo da manuale	Discrezionale
Alta	Formale facoltativo	
	Formale obbligatorio	
Categorie di job	Turnover	Altri strumenti
Dirigente	Alto	Telefono
Ingegnere	Moderato	Calcolatrice
Segreteria	Basso	Altre macchine
Impiegato		
Importanza del task	Complessità del task	
Alta	Alta	
Bassa	Media	
	Bassa	

CHECKLIST DEI PROFILI UTENTE 3/3

Caratteristiche fisiche

Distingue i colori	Predominanza	Sesso
Si	Destro	Donna
No	Mancino	Uomo
	Ambidestro	

• Analisi dei task

Quando osserviamo i nostri futuri utenti, dobbiamo cercare di capire per ogni attore che tipo di compiti svolge, quando li svolge e con quali obiettivi.

Nella definizione di un compito è importante avere in mente:

- L'obiettivo del compito ovvero, uno stato auspicato del mondo che l'utente vuole ottenere.
- Il compito è una attività intrapresa da una persona per raggiungere un obiettivo usando un particolare strumento o artefatto.
Potrebbero esserci uno o più task alternativi per raggiungere lo stesso obiettivo oppure il task può essere suddiviso in più sotto-task.
- Il dominio ovvero il contesto in cui viene eseguito il task.

Nel momento in cui si esegue un'analisi dei task si comincia nell'associare ad un obiettivo l'elenco delle attività che portano al compimento dell'obiettivo;

Successivamente il task viene decomposto gerarchicamente e l'analisi di quest'ultimo serve a stabilire la frequenza e la sequenza con cui questi task vengono portati al termine.

Nella base per un buon design ci sono:

- Una giusta allocazione dei task tra uomo-sistema poiché serve ad ottenere dei buoni requisiti di performance in modo tale da arrivare ad una buona specifica di requisiti funzionali e no;
- Ordinare e raggruppare le funzioni del sistema;
- Quali oggetti rappresentare;
- Quali dispositivi input/output e stili di dialogo usare;
- Quali conoscenze/competenze necessitano gli utenti per completare i task.

Quando parliamo di analisi dei compiti significa distinguere tra:

- **task fisici** ovvero la conoscenza o rappresentazione di un insieme appropriato di azioni o comportamenti fisici;
- **task mentali** la conoscenza o rappresentazione di un insieme appropriato di azioni o comportamenti mentali basandoci sulla conoscenza di questi task significa ovvero:
 1. Elencare tutti gli oggetti e le azioni coinvolte nel task;
 2. Elencare tutte le conoscenze di cui necessitano gli utenti per compiere ciascun'azione;
 3. Raggruppamento in tassonomia per poter identificare la conoscenza di cui ha bisogno l'utente per eseguire il task;
 4. Definire le competenze per preparare il training.

Tutto questo ha un impatto sul design quindi è importante catturare i giusti requisiti, poiché, può accadere che c'è dell'incompatibilità tra il modello mentale dell'utente e le rappresentazioni del computer:

- potremmo ritrovarci di fronte a una rappresentazione del dominio sbagliata;
- non si è in grado di eseguire task o eseguirlo in maniera sgradevole;
- oppure succedere una scorretta allocazione del task tra utente e sistema: un eccessivo carico mentale o fisico per l'utente.

Il design deve essere guidato dagli obiettivi degli utenti reali e quindi non dobbiamo rifarci al task vero e proprio ma all'obiettivo, documentando l'obiettivo dell'analisi dei task poiché quest'ultimo è più duraturo, mentre i task sono provvisori, e quindi possono cambiare più facilmente. Gli utenti eseguono i task per raggiungere l'obiettivo, quindi, è importante identificare gli obiettivi dell'attività degli utenti insieme ai task poiché quest'ultimo diventa la base per identificare le funzionalità che dovrebbe fornire il sistema ed inoltre, bisogna identificare il dominio per poter facilitare la rappresentazione del task nell'interfaccia utente.

MANCA LA LEZIONE NUMERO 9: Empowerment – Potenziamento personale

• Regole di Design

Per ottenere da una progettazione la massima usabilità di un sistema si utilizzano:

- **Principi di usabilità:** regole di design astratte che hanno un'ampia generalità. Una volta assimilati questi principi possono essere applicati in qualsiasi fase di un processo iterativo user-friendly, poiché aiutano noi progettisti a fare delle scelte in modo tale da avvicinare il sistema prodotto al modello mentale delle persone stesse. Possiedono una bassa autorità poiché sono principi generali, ovvero indicazioni utili;
- **Standard:** sono delle regole più rigide e specifiche di determinati domini, che hanno un'alta autorità ed hanno un'applicabilità limitata;
- **Linee Guida:** sono una via di mezzo e vengono fornite in base all'esperienza dello sviluppatore.

• Principi di Usabilità

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

Rientrano tra i principi di ausilio all'usabilità:

1. Apprendibilità o Learnability: facilità con cui i nuovi utenti possono iniziare un'interazione efficace e raggiungere le massimi prestazioni. Si suddivide in:

- **predicibilità**, per determinare l'effetto di azioni future è sufficiente ragionare sulla base della storia delle interazioni precedenti.
Un'interfaccia è predicibile se l'utente che esegue un task riesce a interagire con il sistema intuendo cosa usare per progredire nell'esecuzione del suo compito, ovvero in ogni momento l'interfaccia riesce a comunicare all'utente quello che può essere l'effetto della sua azione sulla base di una storia passata;
- **sintetizzabilità**, che non si tratta di ottenere un'interfaccia sintetica ma sintesi è la capacità di dedurre l'effetto di azioni passate. L'utente deve poter assestare 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 immediatamente non richiedendo alcuna ulteriore interazione dell'utente (onestà immediata). In altri casi può arrivare alla fine dopo esplicite direttive dell'utente (onestà ritardata). Ad esempio, basta fare il confronto del "copia e incolla" su sistemi a manipolazione diretta e in sistemi a riga di comando;
- **familiarità**, cioè il grado con cui la conoscenza e l'esperienza dell'utente anche in altri domini o in generale nel mondo reale possano essere applicate all'interazione con il nuovo sistema. 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**, fornire supporto poiché l'utente possa estendere la conoscenza di una specifica interazione in applicazioni che già conosce a nuove situazioni. 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 testo che tentano di fornire le operazioni di copia e incolla nello stesso modo;
- **consistenza**, riguarda la similarità nel comportamento di input e di output che può derivare da situazioni simili che si ritrova l'utente davanti. È un concetto relativo a diverse proprietà dell'interazione e può determinare la consistenza del linguaggio, i layout delle singole interfacce ecc.

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

- **iniziativa al dialogo**, ovvero l'assenza di vincoli imposti dal sistema sul dialogo d'input.
L'utente deve avere l'impressione che abbia sotto controllo il dialogo con il sistema. L'obiettivo è quello di massimizzare la predominanza dell'utente e minimizzare quella del sistema permettendo comunque al sistema di poter aiutare l'utente a recuperare dagli errori.
Il sistema può iniziare il dialogo e l'utente rispondere semplicemente alle richieste di informazioni (dialogo **System pre-emptive**). Il sistema non può essere introdotto. Una finestra mondiale fa sì che l'utente possa comunicare con il sistema solo tramite la finestra. In alternativa, l'utente può essere libero di interrogare il sistema e di intraprendere qualsiasi azione su di esso (dialogo **User pre-emptive**). In generale si tenta di aumentare la capacità dell'utente di interrompere il sistema e ridurre le possibilità che il sistema interrompa l'utente. Alcune situazioni possono però richiedere un dialogo system pre-emptive, ad esempio, la cancellazione di testo in un editor cooperativo;
- **multithreading**, la capacità del sistema di supportare un'interazione che riguarda anche più di un compito alla volta (più task). Per progettarlo dobbiamo fornire all'utente la capacità di passare da un task all'altro. 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 o **interfogliato** per permettere una sovrapposizione temporale tra compiti separati ma in ogni dato momento il dialogo si limita ad un solo compito. Un sistema a finestra supporta un dialogo interfogliato, mentre un sistema multimodale potrebbe supportare un dialogo concorrente;
- **migrabilità di un task**, ovvero 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**, ovvero la capacità di modificare l'interfaccia da parte dell'utente stesso o 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 una preferenza spostando magari dei pulsanti. L'adattività invece è la personalizzazione automatica dell'interfaccia sulla base dell'osservazione dei compiti dell'utente;
- **sostituibilità**, ovvero la possibilità di sostituire tra loro i valori di input e 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 misure e per l'**output** avendo molteplicità nella rappresentazione degli stati del sistema. A questo punto è legato il concetto di **pari opportunità** dove non riscontriamo differenza tra input e output. Ad esempio, 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.

- **osservabilità**, capacità dell'utente di valutare lo stato interno del sistema dalla sua rappresentazione percepibile. Essa può essere ottenuta tramite:
 - esportabilità o navigabilità: consente all'utente di esaminare lo stato corrente del sistema tramite visualizzazione fornita dall'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 degli errori e vogliono ripristinare lo stato precedente del sistema. La ripristinabilità 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): tentando di annullare gli effetti di un'interazione appena eseguita per tornare 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 eseguire;

- **risposta**, come l'utente percepisce la comunicazione con il sistema in termini di tempo che il sistema impiega ad esprimere cambiamenti di stato. Misurare la rapidità di comunicazione tra il sistema e utente;
- **conformità dei task**, il grado con cui i servizi offerti dal sistema supportano tutti i task degli utenti comprendendo completezza dei task e adeguabilità dei task. Poiché lo scopo di un sistema interattivo è consentire a un utente di eseguire vari compiti per raggiungere i suoi obiettivi all'intero 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.

• Standard

Gli standard rappresentano delle regole di design più specifiche. Essi vengono stabiliti da organismi nazionali o internazionali per assicurare la conformità da parte di una vasta comunità di progettisti e richiedono una teoria di base corretta ed una tecnologia che cambi lentamente.

Lo standard che riguarda la progettazione del software interattivo è legato in particolare al concetto di usabilità.

Standard ISO-9241 (norma uomo-sistema guida sull'usabilità). Lo standard ISO-9241 dà la definizione di usabilità, come efficacia, efficienza e soddisfazione con cui determinati utenti eseguono compiti in un determinato contesto d'uso. Tale standard è una **norma uomo-sistema guida sull'usabilità**, che fornisce un quadro di riferimento necessario per comprendere il significato del termine usabilità per poterlo applicare successivamente in un determinato contesto d'utilizzo. L'ultima versione di tale standard risale al 2018, ed è una norma di carattere generale che specifica quali siano i processi specifici o i metodi da prendere in esame per garantire l'usabilità ma specifica quale deve essere il risultato finale e a cosa bisogna fare attenzione per ottenere tale risultato. Nella parte 110 di tale standard si parla esplicitamente di una serie di euristiche di usabilità che si possono applicare all'interazione tra persone e sistemi informatici. Questa parte dello standard si riferisce all'interazione uomo-macchina come un dialogo e stabilisce sette principi di dialogo generale:

- **Idoneità al compito**, il dialogo dovrebbe essere adatto al compito e al livello di abilità dell'utente;
- **Auto descrittività**, il dialogo dovrebbe chiarire che cosa l'utente dovrebbe fare dopo;
- **La controllabilità nel dialogo**, l'utente dovrebbe essere in grado di controllare il ritmo e la sequenza dell'interazione;
- **Conformità alle aspettative dell'utente**, si deve avere un dialogo coerente dove l'utente non dovrebbe essere spiazzato rispetto a quello che si aspetta di dover fare;

- **Tolleranza agli errori**, il dialogo deve supportare l'utente negli errori che commette e lo guida a rimediare a tali errori;
- **Idoneità all'individualizzazione**, il dialogo dovrebbe essere personalizzabile per adattarsi all'esigenze dell'utente.;
- **Idoneità all'apprendimento**, il dialogo stesso dovrebbe supportare l'apprendimento;

Standard ISO-9126 (standard sull'Ingegneria del Software). Nella parte I di tale standard si parla del modello di qualità specificando una serie di norme che forniscono esempi di metriche da utilizzare per poter misurare la qualità del sistema rispetto ai tre punti di vista:

- esterno
- interno
- in uso,

La qualità in uso permette di abilitare determinati utenti ad ottenere determinati obiettivi con:

- **efficacia**: rappresenta la capacità del software nel permettere agli utenti di raggiungere gli obiettivi specificati con completezza e accuratezza;
- **produttività**: la capacità di mettere in grado gli utenti di spendere una quantità di risposte appropriate in relazione all'efficacia ottenuta;
- **sicurezza**: è la capacità del prodotto di raggiungere livelli accettabili di rischi;
- **soddisfazione**: è la capacità del prodotto software di soddisfare gli utenti.

Tra i vari fattori che contribuiscono a tali qualità in uso vi è l'usabilità, descritta in termini di:

- **comprensibilità**, esprime la comprensione del prodotto;
- **apprendibilità**, è la capacità di ridurre l'impegno richiesto dagli utenti per imparare ad usare l'applicazione;
- **operabilità**, è la capacità di permettere agli utenti di farne uso
- **attrattività**, la capacità del software di essere piacevole verso l'utente che ne fa uso

Nel modello di qualità di tale standard si definisce la capacità di un prodotto software di essere compreso, appreso, utilizzato e attraente per l'utente, se utilizzato in determinate condizioni

Questa stessa definizione rispetto alla prima (Usabilità ISO-9241) non è in contrasto ma conferma, l'importanza di chi siano gli utenti e soprattutto l'importanza del particolare contesto in cui quel prodotto software viene usato.

• Linee Guida

Le linee guida sono regole più generali che possono essere più o meno astratte (principi) o possono essere più concrete (guide di stile).

Sapere quali siano le motivazioni delle linee guida può aiutare a capire quando applicarle e in caso di conflitti quali applicare.

Tra le linee guida possiamo individuare le cosiddette euristiche o regole d'oro.

Le regole d'oro sono regole di granularità più grosse, che possono fornire una checklist utile come sintesi dei consigli di design disponibili a partire dall'esperienza di chi le ha redatte.

Regole d'oro ed euristiche. Tutte le regole descritte in precedenza richiedono al progettista un certo impegno per tenere traccia delle linee guida appropriate oppure per interpretare i principi. Molti sostenitori del design centrato sull'utente hanno presentato delle serie di "regole d'oro" o euristiche. Queste sono ovviamente regole imperfette, che possono non essere applicabili in tutte le situazioni, ma forniscono una lista di controllo utile che costruisce una sintesi dei consigli di design disponibili. È chiaro che un progettista che segue queste norme creerà un sistema migliore di chi le ignora. Le euristiche più famose sono le dieci di Nielsen, le otto regole d'oro di Shneiderman, e i sette principi di Norman.

• Le 8 regole d'oro di Shneiderman

1. **Preservare la coerenza:** le sequenze di operazioni simili dovrebbero essere effettuate sempre con lo stesso tipo di azioni.

La coerenza può riguardare non solo le sequenze di operazioni ma anche: le convenzioni, la terminologia, il font, i menu, l'aspetto ecc...;

2. **Consentire agli utenti abituali di usare comandi rapidi**

Con l'aumento della frequenza d'uso riducendo il numero delle interazioni e aumentando quindi la velocità, tramite l'uso di macro e abbreviazioni, comporta che task che vengono eseguiti ogni giorno siano velocizzati. Inoltre, tutto ciò serve ad abbreviare anche i tempi di risposta;

3. **Offrire un feedback informativo**

Ad ogni azione dell'utente dovrebbe corrispondere un feedback del sistema.

Inoltre, il responso che corrisponde ad azioni frequenti dovrebbe essere modesto mentre, il responso che corrisponde ad azioni occasionali o primarie dovrebbe essere più dettagliato;

4. **Progettare dialoghi previsti di chiusura**

Chiusura significa organizzare le sequenze di azioni in gruppi, preservando il feedback informativo alla fine di ciascun gruppo di azione. Tutto ciò serve a dare all'utente la sensazione di poter scaricare la mente alla fine di una sequenza di azioni per dedicarsi interamente al task successivo;

5. Offrire una prevenzione e una gestione semplice degli errori

Progettando l'interfaccia in modo che sia quanto più difficile commettere gli errori, e in presenza di essi fare in modo che il sistema li riconosca e fornisca istruzioni semplici, costruttive per risolverli.

Nel progettare le interfacce dobbiamo considerare che azioni errate, dovrebbero lasciare immutato lo stato del sistema, oppure il sistema stesso dovrebbe fornire informazioni su come ripristinare lo stato.

Shneiderman dice di aiutare gli utenti ad evitare gli errori seguendo 3 possibili tecniche:

- 1. Coppie corrispondenti corrette**, ad esempio, la mancata chiusura di una parentesi aperta con un editor che lo fa;

- 2. 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.

Per poter scegliere quali azione raggruppare è necessario uno studio di quali azioni verranno compiute maggiormente col sistema;

- 3. Comandi corretti**

Altra causa di errori è l'uso di un linguaggio di comandi in cui si possono commettere errori di battitura. Il completamento automatico dei comandi può evitare questo tipo di errori, limitando inoltre l'uso della tastiera.

6. Permettere un'inversione semplice delle azioni

Rendere le azioni reversibili il più possibile, incoraggiando quindi l'utente a esplorare operazioni sconosciute dando la sensazione di poter tornare indietro senza danno.

La responsabilità del progettista è quella di scegliere opportunamente le unità di azioni reversibili;

7. Supportare il controllo interno

Dare all'utente esperto la sensazione di essere responsabile del sistema, che non farà altro che rispondere alle sue azioni, evitando reazioni sorprendenti dell'utente, sequenze di azioni ripetitive, difficoltà ad ottenere le informazioni richieste o ad eseguire azioni desiderate.

L'utente deve sempre poter controllare il dialogo;

8. Ridurre il carico della memoria a breve termine

Mantenendo le visualizzazioni quanto più semplici possibili. Questa regola si basa sul fatto che la memoria umana a breve termine può elaborare un numero limitato d'informazioni. Questo significa mantenere il display semplice, ridurre la frequenza di spostamenti di finestre e dare il tempo necessario per allenarsi.

- **I 7 principi di Norman**

1. **Bisogna usare sia la conoscenza presente del mondo sia la conoscenza mentale:** invita quindi a riflettere su come è opportuno progettare l'esperienza dell'utente con il sistema, in maniera da favorire la costruzione di un adeguato modello mentale a partire da quello che è la conoscenza del mondo esterno da parte dell'utente e a partire da quello che gli viene presentato dal sistema stesso.
Questo principio fa molto leva sul concetto di Affordances, ovvero l'aspetto di un oggetto che suggerisce come tale oggetto deve essere utilizzato;
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 devono rendere le cose visibili:** bisogna gettare un ponte sul golfo dell'esecuzione e sul golfo della valutazione;
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 chiarito quindi quale controllo fa cosa e in che misura lo fa;
5. **Si deve sfruttare il potere del vincolo sia naturali che artificiali:** i vincoli sono elementi nel mondo che consentono di eseguire solo l'azione corretta nel modo corretto. Un esempio semplice è il puzzle, in cui i 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 creino degli standard:** se non esistessero corrispondenze naturali, quelle arbitrarie dovrebbero andare standardizzate. È questo principio di standardizzazione che permette ad una persona di guidare qualsiasi macchina con pochissime difficoltà.

• Prototipazione e design

I prototipi sono uno strumento, che permettono:

- Di inserire l'uomo all'interno del ciclo, quindi optare un approccio con un feedback continuo con il cliente.
- Di valutare anche precocemente ciò che stiamo facendo,

Un prototipo si definisce come specifiche o un'implementazione parziale di un sistema che consentono di investigare e decidere alternative di progettazione, simulando o animando alcune caratteristiche del sistema che si ha in mente.

Si creano per avere:

- un feedback continuo sul design
- sperimentare design alternativi
- eliminare problemi prima di scrivere il codice.
- Tenere il design centrato sull'utente

È uno strumento che consente di adottare un processo ciclico interattivo al design, si parte con lo sviluppare il design, successivamente creiamo un prototipo che verrà valutato (valutazione formativa) dall'utente successivamente. Tale ciclo si ripete finché non si arriva a una valutazione dell'usabilità totalmente soddisfacente.

Gli obiettivi che ci spingono ad utilizzare i prototipi sono:

- Utilizzare un approccio partecipativo (participatory design) che ci permette di coinvolgere gli utenti sin dall'inizio, nella specifica ed evoluzione di sistemi interattivi;
- Stabilire un processo di ingegnerizzazione del software che aiuti a uno sviluppo/consegna incrementale;
- Sviluppo e condivisione tra utenti e sviluppatori;
- Avere una fonte in comune per l'analisi, sviluppo, documentazione, training.

Il design iterativo risolve i problemi inerenti a una specifica dei requisiti incompleta ma può causare problemi da un punto di vista della gestione poiché il tempo richiesto per la costruzione di prototipi potrebbe far crescere troppo i costi oppure, portare ad una difficile pianificazione e stima dei costi di un processo di design.

In generale possiamo classificare i prototipi a seconda di come vengono costruiti:

- **Throw-away**, il prototipo è di bassa fedeltà e serve a discutere possibili idee di design, ma viene successivamente gettato via per dare posto all'implementazione vera e propria;
- **Incrementale**, parte da un core di funzionalità fondamentali e irrinunciabili per il nostro sistema e successivamente viene implementato per arrivare al prodotto finale;
- **Evolutivo**, è un prototipo che evolve, diventa la base per ogni successiva iterazione del design. Si passa da un prototipo in bozza che evolve fino ad arrivare al prototipo completo che può essere anche testato con gli utenti

I prototipi si sviluppano con due diversi obiettivi:

- **Esplorativo**, quando il prototipo aiuta a sviluppare una visione comune e a dedurre i requisiti utente;
- **Sperimentale**, quando il prototipo consente studi di fattibilità.

Nella costruzione di questi prototipi possono esserci diversi livelli di dettaglio che determina il livello di fedeltà stessa del prototipo:

- **Mock-up**, è un prototipo dell'interfaccia uomo-macchina senza alcuna funzionalità;
- **Prototipo funzionale**, è un'implementazione di un sistema parziale;
- **Sistema pilota**, è un sistema con funzionalità limitate (Use Case).

Solitamente quando si lavora con un prototipo si va a creare un modello che catturi l'uso del sistema dal punto di vista dell'utente (ad esempio, use case), poi si vanno a creare delle viste interattive prototipali che possono animare i principali scenari d'uso, le quali permettono di valutare e migliorare il prototipo con diversi utenti. Tutto ciò fa evolvere il modello per fornire specifiche di design, documentazioni, casi di testing del sistema e materiale illustrativo.

- **Paper sketches (disegni su carta)**

I paper sketch offrono il vantaggio:

- Di supportare le attività di brain storming all'interno del team di progetto, il vantaggio di avere attività di brain storming all'interno del team di progetto e quello di rendere visibili il design rapidamente e confrontare subito idee diverse;
- Non richiedono la definizione di dettagli;
- Non richiedono particolare abilità tecniche.

Gli svantaggi invece sono:

- non si modificano facilmente, poiché vengono disegnati su carta;
- vanno tradotti manualmente in forma elettronica;
- non interagiscono direttamente con l'utente.

Nel caso di paper sketches si parla sempre di design con “bozza”, quindi, non è necessario coprire tutti i casi, ma è possibile concentrarsi solamente sui task necessari. Inoltre, il rapido sviluppo di essi permette di mostrare diverse alternative al cliente

- **Paper mock-ups**

I paper mock-up sono schermate principali, disegnate a mano per mostrare come si intende fare apparire inclusi i dialoghi.

Essi sono un modo semplice per verificare le idee con gli utenti, sono buoni per la spetto grafico ma limitati sulle iterazioni se non vengono animati.

Nel caso in cui tali mock-up vengono disegnati con strumenti fatti apposta per consentire una visione più dettagliata, l'utente avrà un'idea più chiara e precisa e tali mock-up e potranno essere resi interattivi per consentire un testing precoce con l'utente.

- **Storyboarding**

Questi mock-up vengono utilizzati (paper sketches in genere) per costruire i cosiddetti storyboard, ovvero un'altra tecnica basata su carta che presenta più dettagli.

Di solito vengono usati per illustrare i flussi interattivi per compiere determinati task ed implica l'uso di figure e testo per illustrare l'aspetto grafico delle schermate. L'aspetto peculiare di tali storyboard è illustrare sequenze di schermate che corrispondono a dei task interattivi ovvero i casi d'uso che si sono ritenuti fondamentali.

Lo storyboard non deve essere particolarmente elaborato o totalmente accurato, poiché aspira a mostrare l'idea di base.

Gli storyboard, non necessariamente sono computer-based e possono essere animati per consentire la simulazione di un numero limitato di funzionalità.

Essi vengono divisi in:

- Lo-Fi: c'è una bassa fedeltà del prototipo e mancano dettagli rilevanti;
- Hi-Fi: assomigliano al prodotto finale.

Per poter realmente illustrare la reale esperienza dell'utente col sistema c'è bisogno però dei cosiddetti facilitatori, ovvero alcune funzionalità del sistema devono essere simulate dai progettisti. Una tecnica per simulare tali funzionalità è detta la tecnica del Mago di Oz.

Quando parliamo del design iterativo che si avvale dei cosiddetti storyboard, può accadere che i progettisti rimangano vincolati alle scelte di design fatte all'inizio del processo iterativo anche quando sono sbagliate poiché non vanno alla ricerca dei reali problemi di usabilità.

○ **Scenari**

Gli scenari consentono ai progettisti di esplorare il proprio pensiero, ma è necessario identificare i profili utenti e le attività di un task. Il progettista costruirà un chiaro scenario per le attività di quel task in modo tale da assestare le idee sui risultati, anticipando i possibili problemi o riflettere su di essi per evidenziare problemi.

Gli scenari possono essere sviluppati usando i flowchart. Essi permettono di identificare la sequenza e la struttura, mostrando i percorsi di interazione all'utente. Hanno senso se insieme ad essi abbiamo la possibilità di associare dei paper mock-up per dare un quadro completo di come viene strutturata l'applicazione.

○ **Simulazione di funzionalità limitate**

Poter simulare le funzionalità consente di portare avanti delle attività di testing, ma per poter far testare il prototipo all'utente, quest'ultimo deve poter interagire con esso.

Questa simulazione di funzionalità limitate è un approccio semplice che consiste nel rendere il mock-up computerizzato e renderlo interattivo.

Sappiamo che per poter realmente illustrare la reale esperienza del utente col sistema c'è bisogno però dei cosiddetti facilitatori, ovvero alcune funzionalità del sistema devono essere simulate dai progettisti. Una tecnica per simulare tali funzionalità è detta la tecnica del Mago di Oz.

Con la 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 e il sistema in modo da aumentare le funzionalità percepite del sistema e fa sì che la valutazione si concentri su come reagirebbe l'utente di fronte al sistema completo. Da tale testing ne derivano suggerimenti su come migliorare il prototipo nelle versioni successive.

• Dal design concettuale a quello concreto

Qualsiasi prototipo presenta dei compromessi che possono riguardare il grado di modificabilità rispetto alla robustezza ovvero più si va verso un livello di fedeltà elevato minore sarà la possibilità di modificarli.

Abbiamo due tipi di compromessi:

- **Orizzontali**, dove si cerca di ricoprire una vasta gamma di funzionalità ma con pochi dettagli (ad esempio, paper sketch), in questo caso parliamo di prototipi throw-away o evolutivi.
- **Verticale**, dove si arriva a fornire molti dettagli ma si limita solo ad alcune funzioni, in questo caso parliamo di prototipi incrementali.

○ Design Concettuale

Qualsiasi prototipo presenta dei compromessi che possono riguardare il grado di modificabilità rispetto alla robustezza ovvero più si va verso un livello di fedeltà elevato minore sarà la possibilità di modificarli.

Il design concettuale è un approccio che ci permette di costruire prodotti che rispettano le regole di design e quindi l'usabilità.

La prima attività da compiere è andare a ricercare i requisiti e le necessità degli utenti, che sono individuati nell'analisi del problema e trasformarli in un modello concettuale.

Il 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. Uno strumento che viene usato per costruire il modello concettuale è il mood board che può essere usato per catturare la sensazione dell'utente.

In generale quello che dobbiamo fissare come concetti di base per un modello concettuale che serve per un design concettuale sono:

- **metafora e analogie**, che comunicano alle persone come capire a cosa serve un prodotto e come usarlo per un'attività;
- 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 da essi;
- le **relazioni** tra questi oggetti;
- le **mappature** tra i concetti e le esperienze utente che il prodotto è progettato per supportare o richiamare.

Per le metafore si chiede se esiste una metafora adatta, poiché vengono utilizzate in maniera molto ampia nel modello concettuale per favorire la capacità di apprendimento.

Per poter identificare una metafora si eseguono tre passaggi: comprendere la funzionalità, identificare potenziali aree problematiche e successivamente si genera la metafora.

Tale metafora verrà poi valutata:

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

Un altro aspetto importante del modello concettuale è il tipo di interazione che si può supportare con quella metafora, ovvero come l'utente invoca le azioni quindi che tipo di stile d'interazione usa l'interfaccia: dare istruzioni, conversare, manipolare, etc...

- **Istruzioni** (l'interfaccia che da istruzioni e guida l'utente);
- **Conversazionale** (il sistema agisce come se fosse un partner nel dialogo, quindi è progettato per rispondere all'utente come se fosse un partner umano e si basa sul riconoscimento vocale. Ad esempio, Siri);
- **Manipolazione diretta**;
- **Esplorativa** (permette all'utente di interagire fisicamente con l'ambiente);
- **Rispondere** (è il tipo d'interazione che prevede l'iniziativa dal sistema, abbiamo la predominanza del sistema poiché controlla il dialogo. Tale stile può risultare fastidioso ad alcune persone).

Non è detto che bisogna utilizzare solo un tipo d'interazione anzi i nostri design concettuali includeranno una combinazione di interazioni.

Anche il tipo d'interfaccia può aiutare a completare il modello concettuale poiché può guidare nell'interazione che l'utente può fare con quell'interfaccia. La scelta del tipo di interfaccia può influenzare la costruzione del modello concettuale quindi quando si crea un prototipo bisogna già considerare un tipo di interfaccia o interfacce alternative.

Una volta costruito il modello concettuale si cerca di espanderlo iniziando a capire:

- Quali funzioni questo prodotto dovrà supportare? Ovvero cosa farà il prodotto e cosa farà l'uomo;
- In che modo le funzioni sono correlate tra loro? In modo sequenziale o parallelo, classificando tutte le azioni e raggruppandole;
- quali informazioni sono necessarie? Quali dati sono necessari per eseguire l'attività e come devono essere trasferiti al sistema.

- **Design Concreto**

La differenza sostanziale è semplicemente uno spostamento dell'enfasi da quelle che sono le questioni concettuali a dettagli più concreti evidenziati quando si passa alla progettazione del design concreto.

In questo caso per iniziare a produrre un prototipo bisogna iniziare a prendere decisioni concrete anche su aspetti grafici delle nostre interfacce: colore, icone, pulsanti, dispositivi di interazione ecc.

Oltre a queste ci possono essere altre caratteristiche importanti da considerare, l'aspetto dell'accessibilità e l'aspetto del design interculturale (background culturale).

- **Scenari e Use-Case**

Un modo per arrivare alla costruzione di un prototipo che tenga conto di tutti i vincoli catturati già a livello concettuale e l'uso degli scenari, poichè riescono ad esprimere situazioni proposte o immaginare relative all'uso del contesto d'interazione dell'utente con il sistema

Gli scenari sono usati nel design in vari modi:

- come base per la progettazione complessiva;
- script per la valutazione utente dei prototipi;
- come esempi concreti di compiti;
- come mezzo di cooperazione oltre i confini professionali.

Molto spesso sono usati anche per esplorare casi estremi del nostro programma.

Un'alternativa agli scenari per generare prototipi card base e partire dagli use-case. Gli use-case sono un modo più dettagliato di descrivere l'interazione dell'utente con il sistema.

• Collaborazione

Cos'è la collaborazione? Software progettato specificatamente per supportare il lavoro di gruppo, avendo in mente requisiti di cooperazione.

Non solo strumenti di comunicazione. La collaborazione può essere classificata in base a:

- quando e dove i partecipanti stanno lavorando;
- la funzione che svolge per il lavoro cooperativo.

Per analizzare la collaborazione, facciamo uso di una matrice spazio-tempo.

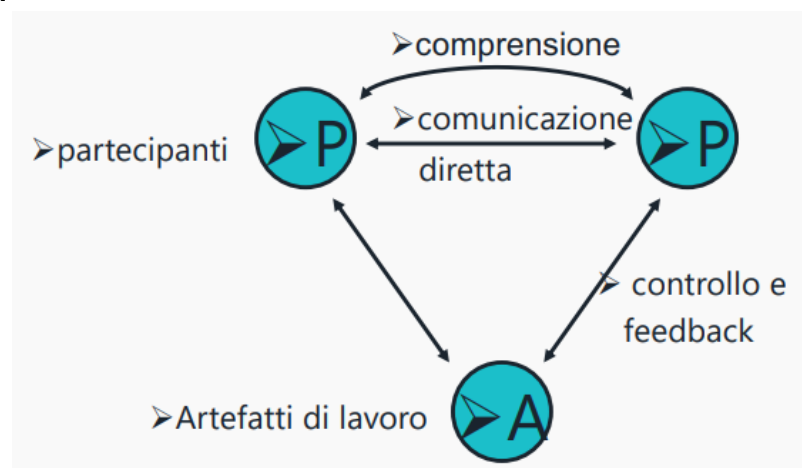
	Stesso Tempo	Tempi Diversi
Stesso luogo	face to face (classrooms, meeting rooms)	interazione asincrona (scheduling di progetto, strumenti di coordinamento)
Luoghi Diversi	sincrona distribuita (editori , video, finestre condivisi)	asincrona distribuita (email, listservs, conferenze)

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 invece prendiamo in esempio 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 sincrone 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.

• Classificazione per funzione

Il lavoro cooperativo coinvolge i **partecipanti** che lavorano e gli **artefatti** sui quali lavorano.



- **Supporto a meeting e decisioni**

Nel design, nella gestione e nella ricerca vogliamo:

- Generare idee;
- Sviluppare idee;
- Registrare idee.

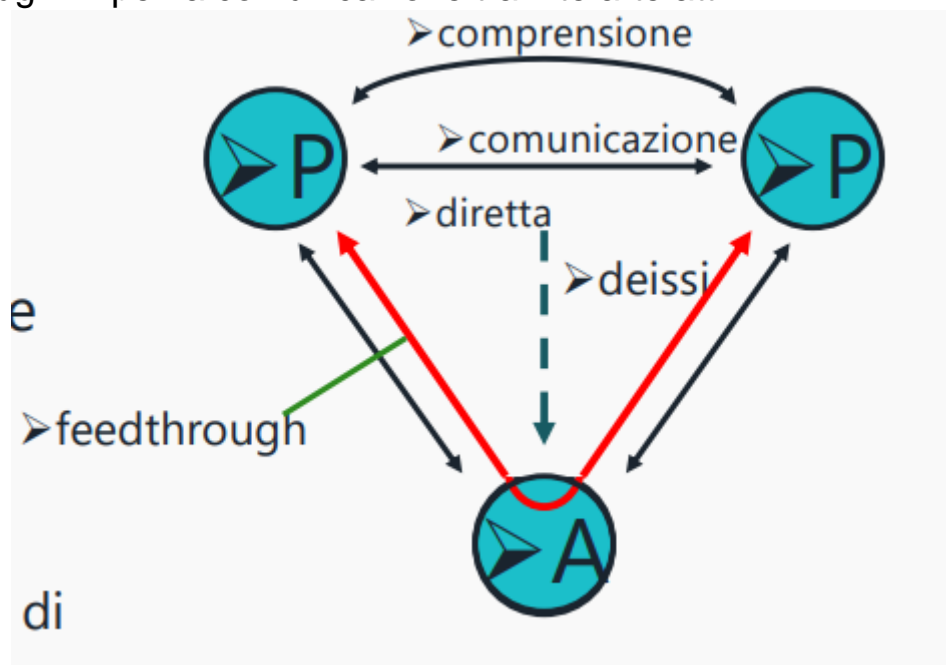
Vi sono tre tipi di sistemi:

- Strumenti di argomentazione: asincroni co-situati. Registrazione delle argomentazioni per le decisioni di design. Hanno due scopi: ricordare ai progettisti le ragioni delle loro decisioni, 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;
- 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 complicità del tipo, chi può scrivere e quando sulla lavagna?
- 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.

- **L'integrazione della comunicazione e del lavoro**

Aggiunto:

- deissi -> riferimenti agli oggetti di lavoro;
- feedthrough -> per la comunicazione tramite artefatti.



• **Progettare l'esperienza utente**

Il sistema deve essere progettato in modo da invogliare le persone ad usarlo, rendendolo efficiente e cercando di disegnare l'esperienza in maniera tale che l'utente sia invogliato ad utilizzarlo.

Il flow è un concetto molto sfruttato nella teoria dell'apprendimento.

Viene utilizzato per definire quel momento di completo assorbimento in una attività che corrisponde al momento esatto in cui l'esperienza può diventare divertente.

Esso si concentra sullo stato mentale di una persona nel momento in cui si verificano tali proprietà:

- Un'attività stimolante che richiede delle abilità;
- La combinazione di azione e consapevolezza;
- Concentrazione sul task sottostante;
- Il venir meno del disagio;
- La trasformazione del tempo;

Il piacere arriva in un ben preciso momento: ogni volta che le opportunità di azioni percepite dall'individuo sono uguali alle sue capacità. Tale momento è detto flow.

• **Analisi dei task**

Quando progettiamo la user experience ci poniamo l'obiettivo di indurre nell'utente momenti esperienziali simili al flow attraverso l'uso della tecnologia informatica.

Per fare questo bisogna garantire il pieno assorbimento nell'attività, che implica garantire le seguenti proprietà:

- L'impiego delle proprie abilità
- La consapevolezza di ciò che accade in ogni istante
- L'assenza di disagio
- La trasformazione del tempo

E garantire la sensazione di piacere che si ha quando l'esperienza diventa naturale e ci si dimentica delle abilità che ci hanno permesso di eseguire il task.

Quindi ciò che vogliamo è ancora una volta l'empowerment del nostro utente.

- **Design pattern**

L'importanza dei design pattern nasce dall'esigenza di codificare la conoscenza del design così da poter guidare i progettisti che hanno meno esperienza. Avere un design efficace e flessibile è difficile da ottenere al primo colpo, un progettista esperto riesce a realizzare un buon progetto in poco tempo, ma un principiante di solito viene sopraffatto da tutte le opzioni di design disponibili. I progettisti esperti non risolvono il problema partendo da zero ma riutilizzando soluzioni che hanno funzionato in passato, in esperienze di progetto precedenti. Quando trovano una buona soluzione la riusano più volte, tale esperienza li classifica come esperti. Questo tipo di esperienze prendono in nome di design pattern. L'idea di fornire dei design pattern di successo non è nuova, viene utilizzata anche in altri ambiti, gli artisti, i romanzieri e nella teoria dei giochi vengono utilizzati spesso i pattern.

- **Differenza design pattern e linee guida di design**

Le linee guida possono essere:

- Astratte, non suggeriscono come risolvere un problema;
- Concrete, troppo “cucite addosso” a una specifica interfaccia.

Le linee guida di solito assumono una validità assoluta e sono più utili per descrivere i requisiti, mentre i design pattern enfatizzano l'efficacia in un particolare contesto e sono utili per tradurre i requisiti in soluzioni software specifiche.

- **La struttura dei pattern di Alexander (Design Pattern Architetture)**

- **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, da poter 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**, le motivazioni alla base;
- **Dichiarazione problema;**
- **Soluzione;**
- **Altri schemi da considerare.**

Alexander ha sottolineato l'importanza dei linguaggi dei pattern. Un linguaggio più che semplici raccolte di pattern 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.

• Design Pattern HCI

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

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 a 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), forze;
- Esempi (examples);
- La soluzione (solution);
- Uno schizzo (sketch o diagramma);

- Riferimenti (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 adottandole alle esigenze del proprio problem space.

• Information Design

Definire e organizzare gli elementi visuali (o multimodali) di un'interfaccia utente: layout dello schermo, progettazione delle icone, selezione del vocabolario, ma anche la cosiddetta "big picture" o modello globale dell'informazione, modelli di percezione, etc...

Ingegnerizzare l'information design: assicurarsi di ciò che le persone vedono (sentono, etc...) ciò che ha senso per loro e li aiuta a perseguire obiettivi significativi; dipende da cosa stanno facendo e di qui l'importanza e il collegamento con la progettazione dei task interattivi.

Percezione: ci permette di organizzare e codificare dati percettivi nella mente: linee, forme, colori vengono "estratti" ... Unità di basso livello poi raggruppate e organizzate, percepite come righe, colonne, griglie, figure, si vedono le relazioni tra i diversi elementi.

L'obiettivo di design è rendere il processo percettivo quanto più rapido e accurato possibile.

Parliamo quindi dei **principi di percezione della Gestalt**:

- Prossimità: elementi vicini tra loro tendono a essere visti come un gruppo.

Possiamo applicarlo ovunque dalla navigazione, alle gallerie, alle liste, al corpo di un testo, all'impaginazione. **La distanza relativa tra gli oggetti in un display influenza la nostra percezione di sé e come gli oggetti sono organizzati in sottogruppi;**

- Similitudine: elementi che condividono caratteristiche visuali (forma, colore, etc...) tendono a essere visti come un gruppo. Possiamo applicarlo ai link di navigazione, ai pulsanti e a molti altri widgets. **Le similitudini tra gli oggetti in un display influenzano la nostra percezione di quali oggetti formano gruppi collegati tra loro;**

- Chiusura: c'è la tendenza a organizzare gli elementi in figure complete chiuse. **L'occlusione parziale di alcuni oggetti sull'interfaccia non pregiudica la loro percezione;**

- Area: c'è la tendenza a raggruppare elementi per creare la più piccola figura possibile;

- Simmetria: c'è la tendenza a vedere elementi simmetrici come parti della stessa figura. **Gli elementi simmetrici sono semplici, armoniosi e piacevoli alla vista. I nostri occhi cercano quegli attributi, insieme all'ordine e alla stabilità. È uno strumento utile per comunicare informazioni in modo rapido ed efficiente, soprattutto su interfacce particolarmente affollate;**

- Continuità: C'è la tendenza a raggruppare elementi in contorni continui o in pattern che si ripetono;

- Figura-Sfondo: c'è la tendenza a distinguere le figure sulla base del loro contorno, il resto è sfondo.


Ogni elemento del display aggiunge complessità al progetto dell'interfaccia. Un design elegante sfrutta la posizione, la ripetizione tematica, schemi di colori a

bassa gradazione e spazio bianco, piuttosto che linee, riquadri ed etichette per organizzare le informazioni.

Interpretazione: in questa fase vengono riconosciuti i risultati della codifica percettiva: un'icona del desktop da selezionare, un pulsante da premere, un campo di testo da editare, un messaggio che spiega qualcosa.

Il riconoscimento è in genere un misto di elaborazione "bottom-up" e "top-down":

- più rapido e più accurato nel riconoscere ciò che ci si aspetta;

-  è sempre interpretato come un pulsante per controllare la stampa.

Obiettivo di design: rendere il processo di interpretazione rapido e accurato. Si scelgano vocaboli di interfaccia che le persone sono abituati a leggere e a vedere.

Sfruttare le Affordances. Un'affordance è l'aspetto di un oggetto che suggerisce l'uso che se ne può fare e a cosa serve. Comuni nel mondo reale e nelle interfacce utente: maniglia di una porta, volante, penna, scale, seduta di una sedia, etc ... Possono diventare parte dell'interazione uomo-sistema.

Modelli informativi: uno "spazio informativo" che gli utenti navigano.

L'integrazione delle informazioni, un aspetto chiave del modello mentale.

Vogliamo una struttura che sia semplice e coerente ma che allo stesso tempo sia completa e flessibile. Molte tecniche per disegnare modelli informativi:

- gerarchia: menu systems, folders, index pages;

- grafo orientato: hypertext, associative links;

- struttura spaziale: tabella, mappe, strutture 3D.

Compromesso tra flessibilità e complessità: di nuovo, è essenziale una buona comprensione delle necessità dei task.

• **Paradigmi e tecniche di 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:

Indagini di Laboratorio

Dove 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.

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.

La valutazione del design viene effettuata in fase di design prima che sia iniziata la fase d'implementazione in modo tale che possono essere evitati grandi sprechi di risorse. Solitamente 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 ce 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:

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

• **COGNITIVE WALKTHROUGH (SONDAGGIO COGNITIVO)**

Basato sulla teoria dell'apprendimento esplorativo di Polson:

- Valuta quando il design supporta l'utente nell'apprendimento dei task;
- Usualmente 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 delle cognitive walkthrough, eseguito da una squadra attentamente gestita. Il collegio di esperti inizia a lavorare separatamente, segue una discussione organizzata che porta a una decisione collegiale.

Tale approccio si presenta bene al participatory design.

- **VALUTAZIONE EURSTICA**

Proposto da Nielsen e Molich (1990).

Ha lo scopo di verificare se l'interfaccia del prodotto rispetta i principi fondamentali dell'usabilità. 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.

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.

Nielsen afferma che 5 esperti diagnosticano circa il 75% dei problemi.

Le 10 Euristiche di Nielsen

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. Avviene una sessione informativa per dire agli esperti cosa fare
2. Ce 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 REVISIONI**

Alcuni modelli cognitivi e di design forniscono un mezzo per combinare specifiche di progetto e valutazione nello stesso framework.

- Il modello GOMS (Goals Operators Methods and Selection), che predice le prestazioni dell'utente con un particolare interfaccia e può essere usato per filtrare delle particolari scelte di progetto
- Il modello "keystroke-level", che fornisce previsioni del tempo che l'utente impiegherà a eseguire task fisici di basso livello.

- **Accessibilità e Usabilità**

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

Due principali difficoltà:

- difficoltà tecnologica: accesso da parte di vari utenti o "interoperabilità";
- difficoltà sociali: accesso usabili per persone diversamente abili; 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 rivolgerci 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.

Barriere:

- **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:** significato che è suggerito soltanto dal colore, 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 la 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.

Tecnologia assistiva: attualmente gli strumenti informatici sono resi accessibili con adattamenti applicativi a posteriori, come, ad esempio, 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 ...

Linee guida per l'accessibilità:

- alternative testuali: il testo è accessibile alla maggior parte degli utenti. Fornire testo alternativo per descrivere pagine, immagini, fotografie o altri elementi;
- navigazione: aiutare l'utente a creare un "modello mentale" della struttura del sito. La navigazione dovrebbe essere consistente e prevedibile. Fornire informazioni per l'orientamento, descrivere la navigazione con del testo ...;
- link: il modello mentale è creato con i collegamenti; devono essere descrittivi ed avere senso al di fuori del contesto;

- tabelle: molto problematiche per i lettori di schermo; è difficile per un utente diversamente abile ricordare i titoli della riga e della colonna quando legge; bisogna fornire descrizioni sintetiche, usare i titoli delle colonne e delle righe, rendere sensibile la lettura linea per linea;
- movimento: molto problematico per gli utenti con difficoltà cognitive; inaccessibile alla maggior parte dei lettori di schermo; finché gli utenti non sono in grado di disabilitare il movimento, non bisogna usare affatto il movimento nelle pagine, quindi evitare lampeggiamenti, auto-refreshing etc ... Fornire degli equivalenti in testo che siano sincronizzati col movimento;
- stile di scrittura: suggerimenti per rendere i contenuti più facili da leggere a tutti; titoli informativi e precisi, titoli con collegamento, etichette di pulsanti; la piramide invertita: dichiarare l'argomento all'inizio dell'articolo, l'argomento all'inizio del paragrafo e limitare ogni paragrafo a una singola idea; layout della pagina consistente, contenuti facili da comprendere, evitare gergo o slang, favorire parole di uso comune, evitare strutture di frasi complesse e disposizione di testo inusuale.

Quattro principi nel design per l'accessibilità:

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

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, nelle forme e nei limiti consentiti dalle conoscenze tecnologiche, 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;
- "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.

Criteri e principi generali per l'accessibilità:

- accessibilità al contenuto del servizio da parte dell'utente;
- fruibilità delle informazioni offerte;
- compatibilità con le linee guida sull'accessibilità indicate dall'Unione europea,

con le normative internazionalmente riconosciute e in considerazione degli standard internazionali:

- International Organization for Standardization (ISO);
- World Wide Web Consortium (W3C);
- specifiche regole tecniche che disciplinano l'accessibilità, da parte degli utenti, agli strumenti didattici e formativi.

Si riconduce direttamente al concetto di usabilità come definito da ISO

9241, Ergonomic requirements for office work with visual display, Part 11.

“efficacia, efficienza e soddisfazione con i quali gli utenti raggiungono determinati obiettivi in determinati ambienti”.

È caratterizzata da:

- consistenza: ... assicurando ... che le azioni da compiere per ottenere servizi di informazioni siano sempre uniformi tra loro;
- efficienza d'uso, espressa soprattutto in termini di flessibilità dell'interfaccia: ... assicurando ... la separazione tra contenuto, presentazione e modalità di funzionamento delle interfacce, nonché la possibilità di rendere disponibile l'informazione attraverso differenti canali sensoriali (le interfacce multimodali);
- efficacia nell'uso e rispondenza alle esigenze dell'utente, assicurando, fra l'altro, che le azioni da compiere per ottenere in modo corretto servizi e informazioni siano indipendenti dal dispositivo utilizzato per l'accesso;
- soddisfazione nell'uso, assicurando, fra l'altro, l'accesso al servizio e all'informazione senza ingiustificati disagi o vincoli per l'utente.

Il World Wide Web Consortium (W3C): 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.

I livelli di conformità del WAI: 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.

• **Relazione tra Usabilità e Accessibilità**

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.