Interazione uomo macchina

Jacopo De Angelis

3 maggio 2021

Indice

Capitolo 1

Modulo 1: Concetti di base

1.1 Cos'è l'interazione uomo macchina?

Definizione standard

"HCI (Human-Computer Interaction) è una disciplina che si occupa della progettazione, realizzazione e valutazione di sistemi interattivi con capacità computazionali destinati all'uso umano e dello studio dei principali fenomeni che li circondano" - Associacion for Computing Machinery

L'usabilità di un sistema è spesso trascurata all'interno dell'ambito lavorativo italiano, non per faciloneria ma perchè le risorse da adibire a questo ambito sono una spesa che non viene vista come essenziale nella produzione del valore per andare avanti.

Il problema viene aggravato dall'outsourcing verso paesi dove il costo del lavoro sia inferiore e, ultimamente, anche da sistemi di ML che sono in grado, con grado di precisione sempre maggiore grazie al deep learning, di sviluppare codice funzionante. Ma cosa viene assegnato ai lavoratori esterni? Ciò che è altamente formalizzato, in modo da lasciare poche possibilità di variazione dalle richieste dei clienti.

Cosa rimane difficile da esternalizzare? Il contatto del cliente, sia durante la prima raccolta di requisiti funzionali, sia le successive interazioni con esso per cambiamenti incrementativi, variazioni di funzionalità o feedback.

L'usabilità è quella caratteristica che rende "facile la vita" al cliente.

Piccola digressione

Perchè la concorrenza moderna rende sempre più importante l'analisi dell'interazione uomo macchina?

In un contesto monopolistico le aziende non sono invogliate a produrre la soluzione "migliore"[1] ma solo quella più efficiente a livello di costo marginale.

Cosa vuole dire questo? Che nella moderna concorrenza derivante da sistemi di sviluppo sempre più semplici e un'offerta più rapida tramite internet, per ottenere quote di mercato le aziende devono iniziare a pensare non solo al "funziona?" ma anche al "come lo faccio funzionare?"

"Qui non si impara a fare delle interfacce usabili, si impara a riconoscere l'usabilità delle interfacce" ovvero impariamo strumenti che ci possono portare a fare delle belle interfacce, certo, ma soprattutto ci permette di riconoscere cosa renda buona un'interfaccia.

La disciplina nasce negli anni '80 ma l'interazione con le macchine (intese come calcolatori) esiste dagli anni'40, semplicemente prima l'utente era ultra specializzato mentre ora quasi tutti possono accedere ad un PC e con questo interagire tramite un'interfaccia.

Dal 1983 si tiene la conferenza annuale ACM CHI. In questi casi tre aree disciplinari si incontrano:

[1] Dove per migliore si intende quella che prende in considerazione più metriche come usabilità, efficienza, efficacia, design, ecc

- ergonomia
- informatica
- psicologia comportamentista

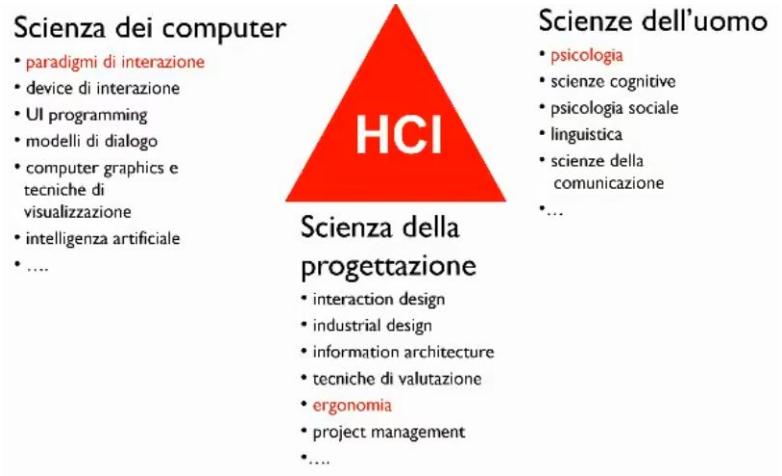


Figura 1.1: HCI e le sue componenti

Dobbiamo ricordare sempre una cosa: dobbiamo lasciarci alle spalle l'utente ideale, l'utente senza faccia e senza capacità, e iniziare a pensare all'utente reale, ovvero a chi, idealmente, è diretta la nostra interfaccia. L'utente non siamo noi. Proprio per questo le interviste, i test con esterni, i mockup sono utili, perchè ci permettono di vedere la nostra idea attraverso gli occhi di altri. Esempio banale: la nostra interfaccia basata sul colore potrebbe essere altamente confusionaria per un daltonico. Un'applicazione mobile dove tutti i comandi sono sulla destra potrebbe essere difficile da usare per un mancino.

Ergonomia cognitiva: studio dell'interazione tra l'uomo e gli strumenti per l'elaborazione di informazioni studiando i processi cognitivi coinvolti (percezione, attenzione, memoria, pensiero, linguaggio, emozioni) e suggerendo delle soluzioni per migliorare tali strumenti.

1.2 Perchè è difficile progettare perchè l'interazione sia "buona"?

Ci sono tre ragioni, idealmente:

1. **La varietà dei sistemi interattivi**: cellulari, computer, cloche, macchine da cucina, tablet...
2. **La varietà degli utenti**: fasce d'età, background culturale, condizioni mediche...
3. **La varietà degli scopi e degli usi**: contesti formativi, ludici, lavorativi e usi tramite dispositivi diversi, in luoghi differenti...

Noi studieremo come progettare per la varietà e al volto delle procedure.

1.3 Temi dell'HCI

- Criteri, metodi e strumenti per la **progettazione dell'interazione** fra esseri umani e sistemi interattivi. L'interazione è il contesto nel quale interagiamo;
- Criteri, metodi e strumenti per la **valutazione dell'usabilità** dei sistemi interattivi;
- Progettazione di nuove tecniche di interazione (dall'hololens ai sistemi che sfruttano i sistemi nervosi);
- Valutazione dell'**impatto dell'automazione** nei contesti umani
- **Sistema socio-tecnico**

Per valutazione dell'**impatto** ci si riferisce a due tipi di impatto:

- a breve termine: sui singoli e nel qui ed ora (usabilità)

- a medio-lungo termine: sono conseguenze inattese sulla collettività

Errore comune

L'usabilità è un tipo di effetto/impatto sugli utenti (sui singoli utenti), NON è una caratteristica intrinseca di un sistema). Per questo non si può valutare l'usabilità senza il coinvolgimento degli utenti (di vario tipo) e sarebbe meglio valutarla in un contesto più simile possibile a una situazione reale.

Ricordiamo che l'usabilità non è solo un concetto di "correttezza funzionale" ma anche di "facilità d'uso".

Non si può parlare di impatto se non si parla di su che sistema socio-tecnico dovrà avere un impatto.

1.4 Sistema socio-tecnico

Banalmente possiamo dire che è un contesto di umani che lavorano assieme tramite delle tecnologie. Quindi:

- è un sistema, "un insieme di elementi interrelati ed eventualmente mutuamente dipendenti che, agli occhi di un osservatore esterno, appaiono come un'entità unitaria ma collettiva, con caratteristiche e comportamento proprio, solitamente autonomo ed intenzionale (cioè volto ad un obiettivo)";
- Un sistema in cui la componente umana (sociale) e quella tecnica (tecnologica) sono inestricabilmente legate tra loro e la loro interazione porta a fenomeni emergenti imprevedibili. Attenzione che tecnica è un termine generico proprio per intendere tutti gli strumenti, che siano fisici o dell'ingegno. In più si parla di interdipendenza perchè lo strumento

è fermo senza qualcuno che lo usa, l'umano è fermo se non ha uno strumento per agire;

- è un concetto di invarianza di scala, ovvero il concetto non cambia in base alla grandezza dell'ambiente sociale

Come progettisti dobbiamo essere pronti all'imprevisto, ovvero all'utente che usa lo strumento non nel modo prescritto.

STS[2] thinking: un approccio che è consapevole che le due componenti si integrano bene (fit) e si "ottimizzano" solo congiuntamente in configurazioni subottime (joint optimization). Bisogna quindi pensare al sistema nella sua interezza e come le parti si integreranno.

Non si deve pensare solo alla qualità degli strumenti ma anche al contesto nel quale vanno usati. Ad esempio la soddisfazione del lavoratore può migliorare l'interazione. Questo va a contribuire alla qualità della configurazione.

L'introduzione di una tecnologia in un contesto (sociale) è parte di un processo di cambiamento operante su più piani. Seguendo ciò che è detto prima si capisce che bisogna pensare non solo alla nostra tecnologia a livello funzionale (sistema tecnico) quando pensiamo all'interazione ma anche a dove andrà inserita e come migliorare quel contesto (sistema sociale).

1.4.1 Proprietà emergenti

Sono di due tipi:

- **Sono proprietà funzionali**, riguardano il funzionamento dell'intero sistema una volta che tutte le sue parti, assembrale come devono, funzionano bene.
- **Sono proprietà non funzionali** che riguardano quanto bene opera il sistema in un determinato ambiente/contesto. Un elenco non esaustivo è affidabilità, sicurezza, performance, sicurezza, usabilità, comfort. (E.g. le informazioni

[2]Socio-technical system

sono giuste ma non aggiornate in un sistema medico, perchè? Perchè non era comodo farlo, quindi veniva visto più come un obolo rispetto ad uno strumento utile, vuole dire che è progettato male a livello di usabilità)

L'emergenza è quando la somma delle parti vale più (a livello di usabilità) delle parti singole (e.g. una bici e le sue parti)

1.4.2 Conseguenza inattesa

Iniziamo con un esempio, l'effetto Peltzman: l'introduzione obbligatoria del casco per i ciclisti rendeva questi più spericolati e anche i conducenti di macchina vedendone uno col casco. In più nei luoghi dove non c'era una distinzione netta tra marciapiede e carreggiata, gli automobilisti rallentavano automaticamente, riducendo il rischio di incidenti, questo perchè non potevano ideare una "zona sicura" da evitare ma per il resto vivere la strada come loro. Tutto ciò perchè gli umani tendono a bilanciare il rischio.

Conseguenza inattesa

Una conseguenza inattesa è quella cosa che può anche andare in maniera contro intuitiva rispetto alla progettazione, e.g. un social network fatto per conoscersi che diventa la base per le proteste di Washington.

Un tipo di conseguenza inattesa è il overreliance, ovvero il fatto di affidarsi maggiormente allo strumento a causa dei feedback positivi. Ci sono due tipi:

- **Overdependence**: mancanza di autonomia, abuso e uso al di là dei bisogni effettivi, mancanza o ignoranza di un piano di contingenza, ovvero come eseguire un compito senza l'uso della tecnologia in esame.

- **Overconfidence**: pensare che non ci saranno mai problemi, non ci saranno mai danni derivanti dall'uso e non si potrà mai sbagliare.

La **complacency è la fiducia che il sistema tecnico funzionerà sempre come è stato progettato**, riducendo così l'attenzione durante l'utilizzo (e.g. le macchine a guida autonoma che richiedono l'attenzione del conducente ma questo le ignora perchè sicuro del funzionamento). è legata ai processi di monitoraggio.

L'**automation bias è l'eccessiva fiducia nella risposta** del supporto alle decisioni e quindi causa di errori di omissione o di azione quando i sistemi sono imperfetti (e.g. la calcolatrice che deve rispondere correttamente, non può essere altrimenti). è legato ai processi decisionali.

Le conseguenze inattese non sono intrinsecamente positive o negative, è solo la registrazione di un effetto non previsto.

Progettare sistemi usabili è progettare per l'uso, quindi per qualcosa che il progettista non controlla e che dipende dall'utente e da quello che fa. Per progettare sistemi usabili non esistono metodologie, è meglio:

- imparare per imitazione e per esperienza, diventando così capaci di basarsi sulla seconda per allontanarsi dalla prima
- essere creativi ma non troppo per non disorientare l'utente che non avrebbe basi conoscitive
- valutare il proprio sistema coinvolgendo utenti veri
- essere progettisti responsabili, ovvero ragionare sulle conseguenze impreviste,Essere progettisti responsabili significa sapere che il proprio sistema sarà il componente di un sistema socio-tecnico che può stravolgere o comunnque modificare.
- aspettarsi che il lavoro possa avere conseguenze inattese e lavorare per minimizzarne l'impatto

1.5 Usabilità

1.5.1 Bassa usabilità = danni e problemi

Vari tipi di bassa usabilità:

- gli utenti non capiscono come svolgere i propri compiti con il sistema
- il sistema presenta un'eccessiva quantità di funzionalità e opzioni (low use)
- gli utenti non capiscono cosa il sistema stia facendo (poca trasparenza)

1.5.2 La colpa è dell'utente o del progettista?

La prima idea è che se il progettista ha creato il sistema a prova di stupido, ci siano tutte le guardie, gli avvisi e le informazioni necessarie allora la colpa sia dell'utente se qualcosa va storto. Nella storia di Dedalo e Icaro viene automatico pensare che Dedalo (il progettista) ha ragione, ha detto tutto a suo figlio e l'altro comunque è andato contro di ciò.

Certo, la colpa è qualcosa che può essere arginato tramite i termini d'uso, "qualsiasi utilizzo fuori da queste linee guida non è nostro problema", per la legge saremmo a posto ma ciò non cambia un dettaglio: se l'utente ha sbagliato c'è la possibilità che siamo stati noi a progettare male l'esperienza. "Mica gli ho detto io di mettere le dita così vicino alla lama dell'affettatrice" ma hai pensato "se il salume diventasse troppo piccolo le dita si avvicinerebbero troppo alla lama, meglio inserire un sistema comodo per spostarlo"?

Gli oggetti ben progettati sono facili da interpretare e comprendere: contengono indizi visibili (affordances) del loro funzionamento. Vedremo successivamente cosa voglia dire. Un esempio è l'indicatore in tempo reale di robustezza della password.

Affordance (invito alluso)

La relazione tra le azioni possibili di un agente in un determinato ambiente.

Gli oggetti progettati male possono essere difficili e frustranti da usare: non offrono indizi o ne danno di sbaglianti, oppure sono stati progettati curando l'estentica più che la funzionalità.

1.5.3 Come capire a livello generico se l'interfaccia è chiara?

Uno dei metodi più semplici è tracciare una linea tra le interazioni per vedere quanto rispetti un ordine di lettura. Possiamo anche basarci su quanti ostacoli alla comprensione ci siano, ad esempio pulsanti che non fanno capire cosa facciano, spiegazioni verbose e non basate su simboli, rendendo difficile l'interpretazione per una persona che non sa leggere la lingua.

1.5.4 Concetti estremi di (non) usabilità

Porta di Norman

https://www.youtube.com/watch?v=yY96hTb8WgI

Una porta di Norman è una porta il cui design suggerisce un'azione contraria a quella da fare o che richiede un segnale per spiegare come usarla.

Uno dei principi che possono andare in correzione qua è la "visibilità", ovvero la possibilità di scoprire quali azioni possano essere eseguite. Prendiamo un attimo come esempio i touchpad del computer: guardandoli non permettono di comprendere cosa

un singolo, doppio o triplo tocco possano fare, non c'è quindi scopribilità.

L'altro è il "feedback", ovvero un segnale che permetta la comprensione di cosa sia successo.

"La porta ideale è quella per la quale non devo pensare come aprirla"

Porta di Norman

La tendenza del progettista a concepire oggetti o sistemi che non invitano all'uso (e all'uso corretto) sulla base di elementi visuali o indicazioni chiare (affordance), affidandosi quindi solo alla memoria, all'esperienza o all'intuizione e incontrando spesso il disorientamento o il fraintendimento dell'utente.

Discoverability (visibilità)

Permette allutente di individuare facilmente qual è la funzione delloggetto, e quindi riconoscere se è adatto allo scopo prefissato. È il risultato dellapplicazione di tutti gli altri principi elencati.

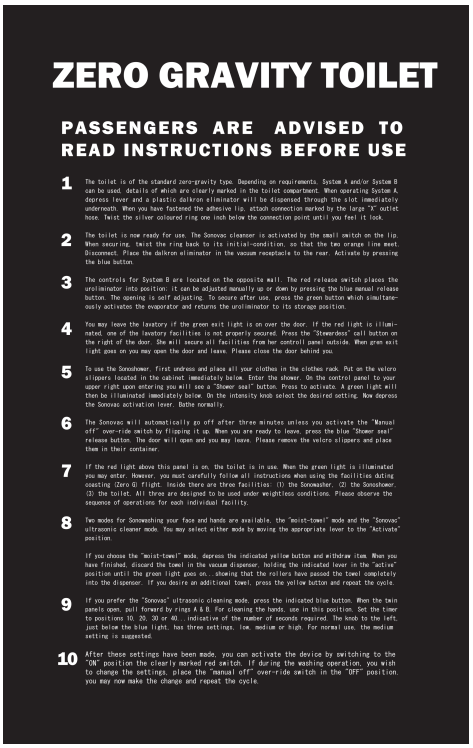
Feedback (reazione)

É linsieme di risposte che il sistema comunica allutente, in base alle azioni effettuate fino a quel momento.

Toilette di Floyd

É la serie di istruzioni della toilette in Odissea nello Spazio.

Una toilette di floyd è un qualcosa di progettato in maniera così poco intuitiva per il quale devo fornire una serie di istruzioni.



ZERO GRAVITY TOILET

PASSENGERS ARE ADVISED TO READ INSTRUCTIONS BEFORE USE

1 The toilet is of the standard zero-gravity type. Depending on requirements, System A and/or System B can be used, details of which are clearly marked in the toilet compartment. When operating System A, depress lever and a plastic dalkron eliminator will be dispensed through the slot immediately underneath. When you have fastened the adhesive lip, attach connection marked by the large "A" outlet hose. Twist the silver coloured ring one inch below the connection point until you feel it lock.

2 The toilet is now ready for use. The Sonovac cleanser is activated by the small switch on the lip. When securing, twist the ring back to its initial-condition, so that the two orange line meet. Disconnect. Place the dalkron eliminator in the vacuum receptacle to the rear. Activate by pressing the blue button.

3 The controls for System B are located on the opposite wall. The red release switch places the urolminator into position: it can be adjusted manually up or down by pressing the blue manual release button. The opening is self adjusting. To secure after use, press the green button which simultaneously activates the evaporator and returns the urolminator to its storage position.

4 You may leave the lavatory if the green exit light is on over the door. If the red light is illuminated, one of the lavatory facilities is not properly secured. Press the "Stewardess" call button on the right of the door. She will secure all facilities from her controll panel outside. When gren exit light goes on you may open the door and leave. Please close the door behind you.

5 To use the Sonoshower, first undress and place all your clothes in the clothes rack. Put on the velcro slippers located in the cabinet immediately below. Enter the shower. On the control panel to your upper right upon entering you will see a "Shower seal" button. Press to activate. A green light will then be illuminated immediately below. On the intensity knob select the desired setting. Now depress the Sonovac activation lever. Bathe normally.

6 The Sonovac will automatically go off after three minutes unless you activate the "Manual off" over-ride switch by flipping it up. When you are ready to leave, press the blue "Shower seal" release button. The door will open and you may leave. Please remove the velcro slippers and place them in their container.

7 If the red light above this panel is on, the toilet is in use. When the green light is illuminated you may enter. However, you must carefully follow all instructions when using the facilities during coasting (Zero G) flight. Inside there are three facilities: (1) the Sonowasher, (2) the Sonoshower, (3) the toilet. All three are designed to be used under weightless conditions. Please observe the sequence of operations for each individual facility.

8 Two modes for Sonowashing your face and hands are available, the "moist-towel" mode and the "Sonovac" ultrasonic cleaner mode. You may select either mode by moving the appropriate lever to the "Activate" position.

If you choose the "moist-towel" mode, depress the indicated yellow button and withdraw item. When you have finished, discard the towel in the vacuum dispenser, holding the indicated lever in the "active" position until the green light goes on, showing that the rollers have passed the towel completely into the dispenser. If you desire an additional towel, press the yellow button and repeat the cycle.

9 If you prefer the "Sonovac" ultrasonic cleaning mode, press the indicated blue button. When the twin panels open, pull forward by rings A & B. For cleaning the hands, use in this position. Set the timer to positions 10, 20, 30 or 40, indicative of the number of seconds required. The knob to the left, just below the blue light, has three settings, low, medium or high. For normal use, the medium setting is suggested.

10 After these settings have been made, you can activate the device by switching to the "ON" position the clearly marked red switch. If during the washing operation, you wish to change the settings, place the "manual off" over-ride switch in the "OFF" position, you may now make the change and repeat the cycle.

Figura 1.2: Lista di istruzioni della toilette

Toilette di Floyd

Tendenza del progettista a complicare affari semplici e a pretendere che basti descrivere una procedura per renderla ovvia a qualsiasi utente.

1.6 Sei concetti centrali

Definizione classica di interazione (ISO 9241) **Un sistema interattivo** è una combinazione di componenti hardware e software che ricevono input da un utente umano e gli forniscono un output allo scopo di supportare l'effettuazione di un compito.

Figura 1.3: Mappa sullo scopo e sul metodo di completamento di un'azione

Un'interfaccia è l'insieme dei componenti di un sistema interattivo (software o hardware) che forniscono all'utente informazioni e comandi per permettergli di effettuare specifici compiti attraverso il sistema.

In comune hanno il concetto di "effettuazione di un compito", da ciò si capisce che il sistema interattivo permette all'utente di conseguire un'obiettivo (compito).

1.6.1 Visibilità

Permette allutente di individuare facilmente qual è la funzione delloggetto, e quindi riconoscere se è adatto allo scopo prefissato. È il risultato dellapplicazione di tutti gli altri principi elencati.

1.6.2 Modello concettuale

É il modello che lutente ha delloggetto in questione; il modello può essere superficiale e riferirsi alla sola conoscenza di relazione tra input e output, ma può anche essere più approfondito, arrivando a conoscere anche tutti i passaggi intermedi a livello macchina. Più il modello concettuale è fedele al funzionamento reale, maggiore è la probabilità di successo nellinterazione tra utente e macchina.

1.6.3 Affordance (invito alluso)

Vedere pagina 21.

Gibson: è la risorsa o supporto che l'ambiente offre all'utente

Norman: è una qualsiasi proprietà o qualità di un oggetto che definisce i suoi possibili utilizzi o rende chiaro come possa o debba essere usato.

Una affordance è qualsiasi proprietà di un oggetto che invita una persona competente all'azione mediata da tale oggetto.

1.6.4 Significanti

Tutto ciò che aggiunge ulteriori indicazioni sullesecuzione dellazione, come direzione di movimento, senso di rotazione, ecc.

1.6.5 Feedback (reazione)

É linsieme di risposte che il sistema comunica allutente, in base alle azioni effettuate fino a quel momento.

1.6.6 Mapping

Vedere pagina 22.

É il rapporto fra i comandi, il loro azionamento ed i risultati che ne derivano nel mondo esterno; permette allutente di creare un collegamento diretto fra i comandi di controllo e le parti delloggetto di cui modificano rispettivamente lo stato.

1.7 Affordance

Affordance (invito alluso)

Gibson: è la risorsa o supporto che l'ambiente offre all'utente

Norman: è una qualsiasi proprietà o qualità di un oggetto che definisce i suoi possibili utilizzi o rende chiaro come possa o debba essere usato.

Una affordance è qualsiasi proprietà di un oggetto che invita una persona competente all'azione mediata da tale oggetto.

Ci sono vari tipi di affordance:

- **Cognitive**: aiuta gli utenti attraverso le loro capacità cognitive (pensare, decidere, imparare, ricordare e conoscere)
- **Fisiche**: aiuta l'utente attraverso le sue azioni fisiche (cliccare, toccare, puntabile, gesticolare, muovere)

- **Sensoriali**: aiuta l'utente attraverso le sue azioni sensoriali (vedere, sentire e percepire
- **Funzionali**: aiuta l'utente a realizzare il suo compito e a comprendere come usare lo strumento
- **Emotiva**: aiuta l'utente ad esperire certe emozioni (il caso più intuitivo è quello delle emoticon e delle sue evoluzioni)
- **Sociale**: aiuta l'utente ad esperire certe funzioni sociali

Le affordance non sono esclusive, posso copartecipare all'intuizione, il problema è quando sono in contraddizione. Nel caso la progettazione non sia intuitiva gli utenti possono aggiungerle (ad esempio inserendo un cartello).

1.8 Mapping

Mapping

É il rapporto fra i comandi, il loro azionamento ed i risultati che ne derivano nel mondo esterno; permette allutente di creare un collegamento diretto fra i comandi di controllo e le parti delloggetto di cui modificano rispettivamente lo stato.

Come possiamo vedere nell'immagine 1.4 il mapping topologico

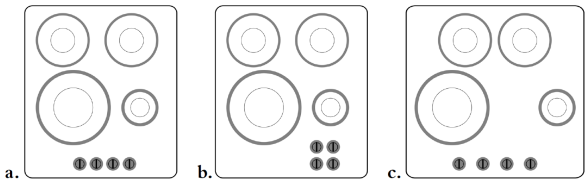


Figura 1.4: Esempi di mapping topologici mancati e riusciti

è buono ma quello funzionale, in certi casi, può essere migliorato: cosa ha a che fare la rotazione con l'intensità della fiamma e quindi il calore?

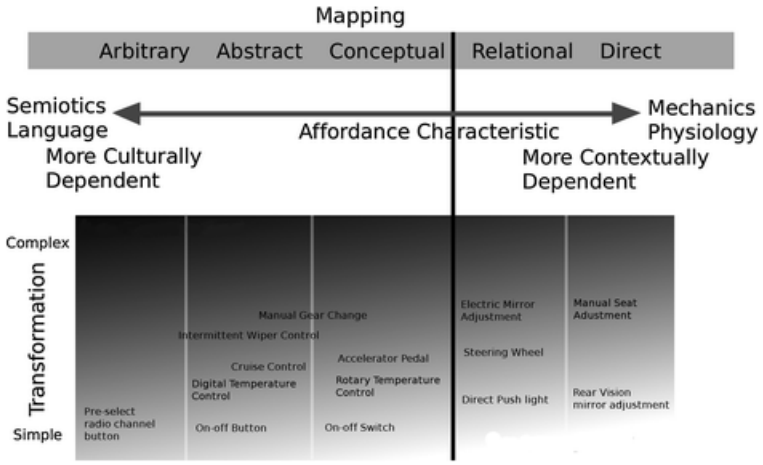


Figura 1.5: Grafico mapping-affordance

Anche per il mapping possiamo averne di semplici e complessi. Dall'immagine 1.5 possiamo vedere i tipi di mapping individuati da McLuhan:

- **Arbitrario**: una relazione del tutto immaginata dal progettista e che l'utente adotta solo per prove ed errori e poi impara a memoria, senza riferimenti culturali o indicazioni fisiche

- **Convenzionale**: la relazione tra controllo ed effetto è stabile, quasi ovvia, ma legata a delle convenzioni. Ad esempio solo una convenzione ci permette di capire che la seconda marcia, che in molte automobili si trova nella fila inferiore e comunque sotto la prima marcia, in altre invece sopra, è quella che riduce la coppia alla ruota a parità di coppia del motore

- **Naturale**: la relazione è meno immediata che nel caso diretto (ma i concetti sono molto simili)

- **Diretto**: c'è una relazione fisica, visibile, spesso legata alla posizione, tra controllo (che afforda una azione) e l'effettore controllato che produce un effetto nel mondo

Ricordiamo che non si può avere mapping senza affordance ma si può avere affordance senza mapping.

Un'affordance non è solo qualcosa che suggerisce una azione su di essa ma anche qualcosa che suggerisce un possibile effetto sul mondo. è proprio il mappinga, che è una caratteristica dell'affordance, che si occupa di rendere questo collegamento o relazione più o meno semplice da capire.

1.8.1 La legge di Fitt

Legge di Fitts

$$MT = a + b \log_2(\frac{D}{W} + 1)$$

dove MT è il tempo di movimento, D è la distanza dall'obiettivo e W è la grandezza dell'obiettivo

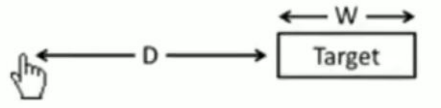


Figura 1.6: Legge di Fitt

Nella progettazione si può pensare di rendere più efficienti per la legge di Fitt le funzioni più frequenti e rendere meno efficienti quelli per i quali serve una forma di sicurezza.

1.9 Scheumorfismo

Uno scheumorfismo è un ornamento fisico o grafico apposto su un oggetto allo scopo di richiamare le caratteristiche estetiche di un altro. Un esempio è la calcolatrice digitale che cerca di replicare gli schemi che ritroviamo su quelle fisiche, oppure all'applicazione di un e-reader che simula il girare fisicamente la pagina.

Figura 1.7: Esempio di scheumorfismo

Perchè oggi non viene più usato così pervasivamente? Perchè spesos questo approccio può essere estremamente utile per permettere all'utente un'interazione veloce ma porta con sè eventuali costi di performance. Oggi è stato sostituito da un approccio più flat.

Lo scheumorfismo può anche portare il problema di essere vincolati troppo dall'oggetto fisico.

1.10 Vincoli e workaround

Un vincolo può essere:

- **passivo/concettuale**: ad esempio concepisco un form con tre campi e non quattro, per vincolare l'inserimento di soli

tre tipi di dato; non prevedo un campo "note", non prevedo un canale di ritorno (feedback) tra consumatore e azienda

- **attivo/funzionale**: ad esempio concepisco dei controlli per cui l'utente non può proseguire se non compila tutti i campi, o non li compila "correttamente" sulla base di regex come ad esempio per l'indirizzo mail o il codice fiscale)

I vincoli vanno sempre bene? No, se l'utente si sentisse troppo vincolato potrebbe iniziare a creare dei workaround. Questi vengono chiamati "desire path".

Desire Path

Qualsiasi azione relativa all'esecuzione di un processo o di un compito (supportati dal sistema informatico) che non è prevista o descritta nei manuali di uso del sistema informatico e/o nei manuali che descrivono tale processo o procedura e che può bypassare l'uso del sistema o piegarlo ai propri fini.

"Percorso del desiderio", sono quei percorsi di interazione che descrivono il percorso ideale da parte dell'utente.

1.11 Progettare per affordance

- Seguire le maggiori convenzioni già stabilite per immagini e azioni ed adottare un mapping naturale il più spesso possibile
- Se appropriato, usare parole in aggiunta alle icone e alle grafiche
- usare metafore riconoscibili (ad esempio il cestino dei rifiuti per indicare il cancellare un file)

- Essere consistenti e coerenti nell'uso dei modelli concettuali usati durante la fase di design (banalmente l'usare sempre la stessa dicitura per il tasto di conferma)

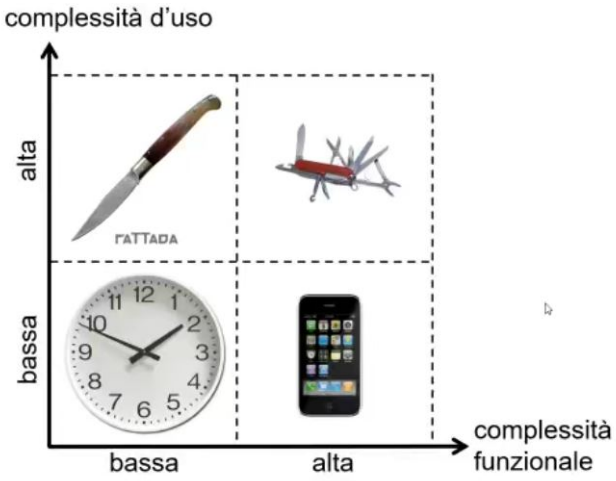


Figura 1.8: Mappa complessità funzionale e strutturale

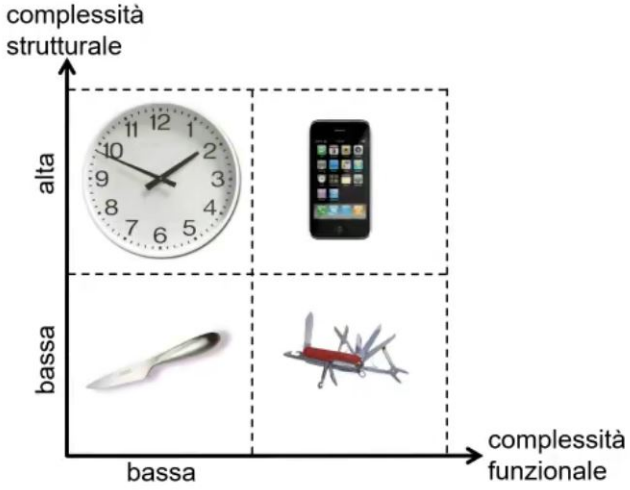


Figura 1.9: Mappa complessità funzionale e d'uso

- **Complessità funzionale**: indica quante funzioni possono essere svolte tramite l'interfaccia in analisi (un cacciavite a punta singola ha una bassa complessità funzionale,

un coltellino svizzero ha un'alta complessità funzionale in quanto in base a come viene usato può svolgere numerose funzioni in virtù delle sue componenti)

- **Complessità strutturale**: indica, genericamente, quanto sia complessa la struttura di uno strumento (un cucchiaio ha una complessità strutturale bassa, uno smartphone una complessità strutturale alta)
- **Complessità d'uso**: indica quando sia semplice utilizzare l'interfaccia, ha un forte elemento basato anche sulle capacità personali (una maniglia ha complessità d'uso bassa, una macchina con cambio manuale ha complessità d'uso alta)

Il nostro obiettivo è raggiungere alta complessità funzionale con bassa complessità d'uso. Ci sono degli accorgimenti che ci permettono di diminuire la complessità d'uso, ad esempio una progettazione oculata.

1.11.1 Perchè è necessario semplificare l'uso?

Prima di tutto la pervasività della tecnologia al giorno d'oggi ci richiede ciò, se fosse complicata da usare ci sarebbero più difficoltà nella sua implementazione nella vita di tutti i giorni.

L'accessibilità è un altro capo saldo, renderla usabile e accessibile a tutti, non solo in termine di rimozione di barriere architettoniche ma in generale come possibilità d'accesso, non bloccata da limiti fisici o di disponibilità.

C'è anche la necessità di comprendere ruoli e possibilità della tecnologia per migliorare la qualità della vita.

Le interfacce non sono solo il mezzo interattivo con i sistemi ma sono anche un filtro della complessità d'uso, una buona interfaccia semplifica l'utilizzo e da qua l'integrazione. La capacità di creare un sistema con una minore complessità d'uso ci permette di essere competitivi.

Capitolo 2

Valutazione di usabilità

2.1 In cosa consiste il progetto?

Chi è coinvolto? Noi e utenti reali

Che tipi di analisi ci sono?

- Confronto longitudinale: nel tempo
- Confronto trasversale: tra due opzioni

Quali sono gli step? Valutazione euristica: 3 utenti esperti di dominio + 3 utenti esperti di usabilità (cioè noi). Per utenti esperti di dominio si intendono utenti che usano abitualmente il tipo di interfaccia in analisi.

User test: i 12 utenti tra i 24 che saranno coinvolti negli user test devono essere più rappresentativi possibili della popolazione utilizzatrice.

Durante lo user test bisogna registrare l'interazione (c'è un modulo apposito per la privacy).

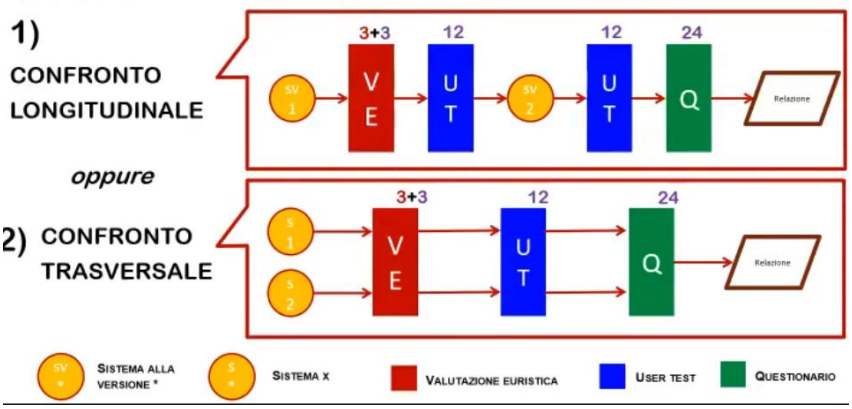


Figura 2.1: Gli step per i due confronti

2.2 Valutazione euristica (qualitativa)

Si coinvolgono utenti (esperti del task) e/o esperti di usabilità (inclusi noi stessi).

Per valutare sistemi interattivi, a qualsiasi livello di prototipazione e maturità alla luce dei principi di buona progettazione.

Qualitativa non si intende in modo vago ma dove non viene prodotto un numero come output ma invece farsi un'idea sufficientemente completa dei possibili problemi di usabilità.

Per tutti gli utenti bisogna includere all'interno della relazione una mappa che interpoli expertise di dominio e sulla usabilità e posizioni gli utenti coinvolti su di questa (una mappa come nella figura 2.2.

L'expertise sull'usabilità è quella riguardante cosa si conosce dei buoni principi di progettazione, quella sul dominio è l'expertise nell'applicazione specifica.

La valutazione euristica è la valutazione (di usabilità) svolta alla luce di un determinato insieme di euristiche (ben definite, possibilmente ben note, prese a riferimento) per identificare soluzioni di design che o si conformano o violano una o più euristiche

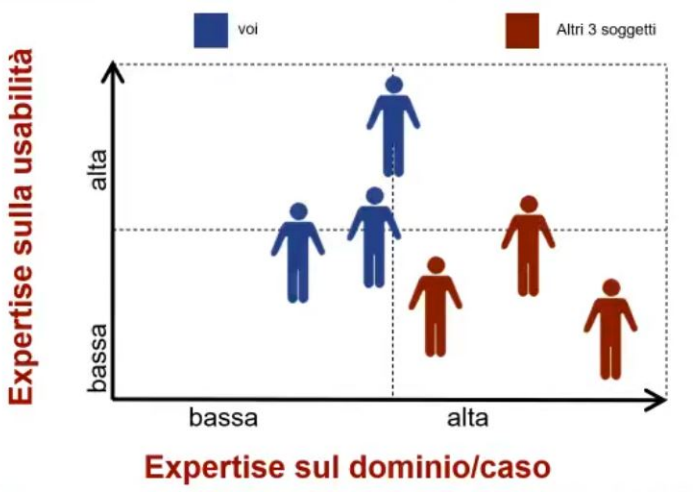


Figura 2.2: Mappa expertise

di tale insieme.

2.2.1 Quali indicazioni per il design dell'interfaccia utente?

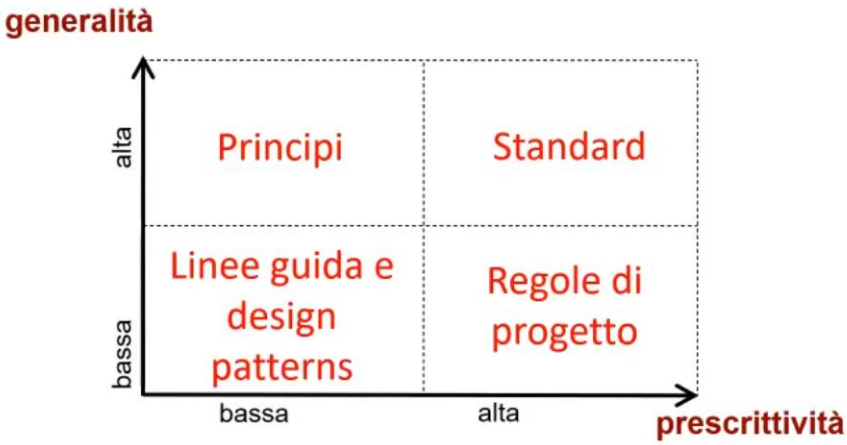


Figura 2.3: Tipi di principi e linee guida

Si chiamano anche principi euristici o semplicemente "euristiche". Una euristica è quindi un insieme di concetti, riferimenti e, soprattutto, strategie che si sono rilevate particolarmente adatte a risolvere un determinato problema, nel nostro caso la progettazione di sistemi interattivi usabili.

Principi

Indicazioni generali per la progettazione di interfacce utente usabili, basate su evidenza scientifica o sul generale consenso. Derivano dalla conoscenza degli aspetti psicologici, computazionali e sociali e sono indipendenti dalla tecnologia. Sono espressi spesso in forma molto generale.

Standard

Insieme di regole da applicare nel progetto di una classe di sistemi. possono essere vincolanti. Sono di norma emesse da un Ente di standardizzazione (e.g. ISO). La conformità allo standard deve essere valutabile in modo preciso. (Vedere 19

Linee guida e design patterns

Insieme di raccomandazioni per il progetto dell'interfaccia utente per una particolare classe di sistemi, espresse in modo generale ma meno astratte dei principi, con esempi e motivazioni. Non sono mai vincolanti, sta al progettista decidere sulla opportunità per applicarle caso per caso.

Il concetto di "design pattern" nasce in architettura. Si tratta di riutilizzare buone pratiche (appunto raccomandazioni) per non dovere reinventare tutto da capo. Per i programmatori i più famosi sono i design pattern pensati dalla GoF. Diffondono standard testati e facilitano la formazione di un linguaggio comune.

Regole di progetto

Insieme di regole o specifiche da applicare nel progetto di un particolare sistema. Sono vincolanti all'interno del progetto, ad esempio perchè dettate da precisi requisiti del cliente/committente o per motivi di budget e vincoli progettuali.

Si usano i principi e non le linee guida perchè non vogliamo essere vincoli eccessivi.

2.2.2 Livelli di autorevolezza dei principi

Ci sono diversi livelli di autorevolezza dei principi (o forza dell'evidenza):

1. Completamente supportati da risultati di ricerca e dati empirici
2. Basati su pratica generalmente accettata (in modo documentato)
3. Non ben documentate, ma supportate dall'opinione di professionisti esperti
4. Opinione individuale

Solo i primi due gradi sono recepiti. I principi di Nielsen sono tra B e C, alcune soluzioni specifiche (ispirate a quei principi) anche di livello A.

2.2.3 Le euristiche usate più frequentemente

La prima checklist è la IXD checklist, utile per la valutazione di siti web che vogliono essere anche gradevoli visivamente. Viene creata una lista di ambiti e in ognuno di questo ci sono vari domini ai quali si applicano:

- Euristiche

  - Affordance
  - Feedback
  - Simplicity
  - Structure
  - Consistency
  - Tolerance
  - Accessibility

- Domini
  - Interface
  - Iconography
  - Typography
  - Interaction
  - Navigation

Questa checklist è altamente generica ma permette di raggruppare le idee e scovare problemi di usabilità da migliorare.

Un altro standard è l'ISO 9241-110 che prescrive 7 principi del dialogo uomo macchina:

- Adeguatezza del compito
- Autodescrizione (capacità del sistema di dire in che stato sia)
- Conformità alle aspettative dell'utente
- Adeguatezza all'apprendimento (facilitare l'adozione facilitando il processo di apprendimento)
- Controllabilità
- Tolleranza verso gli errori
- Adeguatezza alla personalizzazione (inteso come fine tuning)

Euristiche di Nielsen

L'insieme più importante e a cui viene attribuita una buona autorevolezza sono le euristiche di Nielsen.
Possono essere tenute in sede d'esame. Basati su un'analisi fattoriale di 249 problemi di usabilità con cui si è derivato l'insieme minimo per massimizzare la potenza esplicativa.

In rete ci sono numerosi template per tenere traccia dei problemi.

Una rielaborazione del professor Mussio:

Percezione

- fare vedere lo stato del sistema (feedback)
- adeguare il sistema al mondo reale (parlare il linguaggio dell'utente)
- controllo dell'utente e libertà (uscite indicate chiaramente)

Cognizione

- assicurare consistenza (nell'applicazione, sistema, ambiente)
- riconoscimento piuttosto che uso della memoria dell'utente
- assicurare flessibilità ed efficienza d'uso (accelleratori)
- visualizzare tutte e sole le informazioni necessarie (estetica e design minimalista)

Errori

- prevenire gli errori
- permettere all'utente di correggere gli errori e non solo di rilevarli
- help contestuale e documentazione

2.2.4 Spiegazione euristiche

Fare vedere lo stato del sistema (feedback)

- Il sistema deve informare continuamente l'utente di ciò che sta facendo e come sta interpretando gli input dell'utente.
- L'utente deve poter capire l'effetto dell'azione passata sullo stato presente e individuare le possibili nuove azioni.
- Fornire indicazioni sui malfunzionamenti e le ipotesi plausibili delle cause. Non bisogna segnalare solo per errori ma anche se l'applicazione è in uno stato di transizione, ad esempio cambiare il cursore durante un'elaborazione
- Espresso in maniera concreta, non astratta o in termini generali (ad esempio copia file ma quale file da dove a dove)
- Il feedback può avere differenti gradi di persistenza. Esempi:
  - **Persistenza breve**: un messaggio che la stampante ha finito la carta
  - **Persistenza media**: sta sullo schermo finchè l'utente esplicitamente non lo riconosce, ad esempio l'avviso di reindirizzamento su di un'altra stampante
  - **Persistenza alta**: rimane una parte permanente dello schermo, ad esempio la batteria
- è importante nel caso di operazioni che richiedono una lunga elaborazione

Adeguare il sistema al mondo reale (parlare il linguaggio dell'utente)

- Permettere all'utente di sfruttare la sua esperienza nell'interagire con il sistema
- Familiarità: adotta le notazioni di utente, convenzioni sui nomi

- Proiezione mondo reale: metafore familiari

- I dialoghi, se possibile, devono essere nella lingua nativa dell'utente così come le icone devono rispettare la cultura (affordance percepita)

Controllo dell'utente e libertà (uscite indicate chiaramente)

- Gli utenti non si devono sentire "intrappolati" dai computer, deovno sentire che sono loro che controllano il dialogo: ad esempio usare le finestre modali solo quando servono

- Offrire sempre all'utente un modo semplice per uscire dalla situazione corrente, dare delle chiare vie d'uscita

- Offrire sempre delle possibilità di "Undo", ovvero tornare indietro sui propri passi

- vie d'uscita e undo permettono all'utente di imparare esplorando il sistema (flip undo è il singolo step, multiple undo è quello che permette di navigare in vari step e solitamente è associato ad un redo)

- permettere di interrompere o cancellare operazioni che durano troppo

Assicurare consistenza (nell'applicazione, sistema, ambiente)

Avere la stessa interfaccia nei vari ambiti permette una veloce navigazione dell'applicazione. Avere i pulsanti ok, cancel, help messi sempre in posizione differente possono disorientare.

Nell'ambito del gruppo di sviluppo è importante adottare o definire convenzioni interne di rappresentazione e di interazione con rigore e farle rispettare in modo da avere una consistenza interna.

Riconoscimento piuttosto che uso della memoria dell'utente

Un comportamento corretto è quello di rimuovere la necessità per l'utente di ricordare le cose a memoria quando possono essere mostrate direttamente.

Se si richiede all'utente un formato specifico è meglio ricordarglielo e non lasciare che usi un sistema di try and error. Un modo è mostrare il pattern, un altro è riempire il campo con un placeholder. Nel caso ci sia un intervallo legale specificarlo.

Assicurare flessibilità ed efficienza d'uso (accelleratori)

L'utente esperto deve poter svolgere velocemente le operazioni più frequenti, ad esempio l'uso di shortcuts (accelleratori).

Flessibilità nel senso che il sistema deve permettere di fare le stesse cose in maniere diverse (ad esempio per un utente novizio o esperto)

Altri esempi:

- Segnalibri
- Breadcrumb trail
- Uso della storia dell'interazione (e.g. file aperti di recente)
- Definizione di templates e macro
- Type-ahead (poter inserire il prossimo input prima che il computer sia pronto ad accettarlo)
- Fornire valori di default

Visualizzare tutte e sole le informazioni necessarie (estetica e design minimalista)

L'interfaccia deve essere semplice e gradevole e rispecchiare nella maniera più naturale possibile i compiti dell'utente. Presentare solo l'informazione:

- che l'utente vuole
- quando l'utente la vuole
- come l'utente la vuole
- dove l'utente la vuole

e non accompagnata da informazioni non rilevanti o fuorvianti.

Raggruppare poi le informazioni che vengono usate insieme e presentarle in modo da suggerire la sequenza di azioni.

Una credenza errata "Più opzioni offro e più modi di fare le cose metto a disposizione, più utenti soddisfo" (noto come Bells and whistles bias). Ogni volta che si aggiunge un'opzione l'utente ha una cosa in più da ricordare e da usare in maniera scorretta.

Un'alternativa: offrire diversi "modi utente", ad esempio per novizio, esperto, operatore di linea e manutentore...

Il fenomeno del low use riguarda il rapporto tra funzioni usate spesso o a volte e quelle che non sono note o sono note ma inutilizzate. Si può calcolare un "use index" così:

$$UI = \frac{(N_S * 1) + (N_A * 0,5) + (N_M * 0)}{N_S + N_A + N_M}$$

Dove:

N_S sono le funzioni usate spesso

N_A sono le funzioni usate a volte

N_M sono le funzioni mai usate

I low users sono quelli con

$$UI \leq 0,5$$

Può essere usato per capire quali funzioni promuovere e, nel caso, potenziare.

Importanza della grafica, del layout e del colore (estetica) Mettere "prima" le informazioni che devono essere lette prima.

Oggetti lampeggianti o CARATTERI MAIUSCOLI, solo se necessario, anche perchè rallentano del 10% la velocità di lettura.

In generale si considera una lettura media di 200-300 pam (parole al minuto) per la lettura generica (rounding) ma la velocità dipende dal task (memorizzazione 100, skimming 700), dall'illuminazione, dai font, dal livello di scolarità.

Considerare alcuni problemi aiuta, ad esempio per il daltonismo:

- Non usare più di 5-7 colori
- Rendere l'interfaccia distinguibile anche senza colori
- Non dedicare troppo spazio al titolo, nome del venditore, numero di versione e informazioni simili.

Prevenire gli errori

- Individuare le situazioni di possibile errore e cercare di aiutare l'utente. Cerare meccanismi di prevenzione, ad esempio selezionare i valori da un menù e non tramite digitazione libera.
- Usare le richieste di conferma.
- Evitare comandi troppo simili con significati diversi.

- Creare meccanismi di comunicazione utente - progettista: errori rilevati dall'utente e inviabili tramite log al progettista.

Tecniche di prevenzione

- Diversificare le azioni dell'utente
- Amodalità: si intende fare sì che i comandi non abbiano significato differente in base alla modalità utente
- Funzioni vincolate
- Avvertimenti e richieste di conferma
- Default inoffensivi
- Bypass sicuri (e.g. password dimenticata)

Permettere all'utente di correggere gli errori e non solo di rilevarli

Dare buoni messaggi di errore è essenziale, devono essere chiari, senza codici, dare indicazioni precise, proporre una soluzione e, soprattutto, essere gentili.

La recoverability è la capacità di raggiungere il risultato voluto dopo avere commesso l'errore. Se un'azione non è reversibile sarebbe meglio renderla difficilmente conseguibile (inefficienza programmata).

Granularità dei messaggi d'errore Meglio un messaggio d'errore alla volta quando viene commesso l'errore o tutti insieme alla fine?

Help contestuale e documentazione

Il sistema perfetto è così facile da usare che non ha bisogno di help e manuali ma poichè non esiste è sempre bene prevederli.

Bisogna anche tenere a mente che spesso gli utenti non leggono i manuali, preferiscono usare il sistema senza avere letto alcuna spiegazione, quindi mai dare per presupposto che l'utente abbia letto il manuale e fornire aiuti in linea con documentazione raggiungibile in modo preciso.

L'help deve essere minimale e contestuale (Clippy non era male).

2.2.5 AB testing

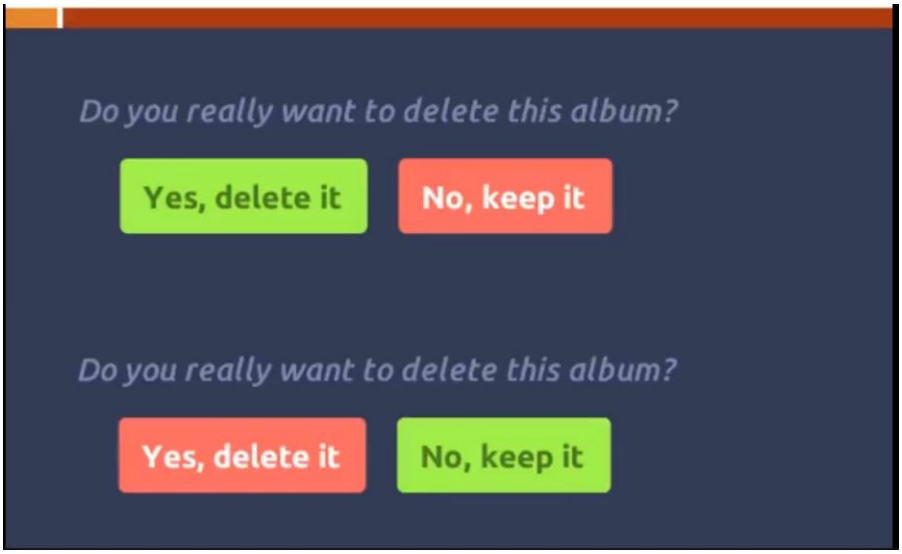


Figura 2.4: Esempio di AB testing

Presentare due alternative di design e fare scegliere agli utenti quale preferiscano permette di capire quale sia l'interazione statisticamente preferita.

AB suggerisce due possibilità ma in realtà possono esserci N combinazioni di elementi sotto esame, ad esempio combinazioni di media, label dei pulsanti, colori ecc. che vengono proposti agli utenti e poi valutati in base ad un feedback misurabile.

2.3 Valutazione euristica di usabilità

è una tecnica sistematica di ispezione predittiva (non misurazione).

Un gruppo di valutatori (e/o esperti) esamina il sistema e identifica eventuali problemi di usabilità rispetto ad un insieme prefissato di principi, le cosiddette euristiche. La valutazione può essere:

- olistica (complessiva e non guidata) o task oriented[1] (l'utente deve conseguire un obiettivo specifico)
- con o senza osservatori

Il risultato di una valutazione euristica è un elenco di problemi di usabilità, associati ad una o più euristiche, pesati per gravità.

2.3.1 Sequenza di passi

1. scegliere un'applicazione
2. concepire 2 o 3 task
3. ogni persona nel gruppo base esplora l'applicazione (valutazione esperta olistica) con uno dei form a portata di mano
4. ogni persona, indipendentemente dagli altri, scrive un elenco di problemi, riconducendoli alla violazione di una o più euristiche
5. ora viene presentata l'applicazione a tre utenti esperti di domini. A loro viene dato un tempo di 3-4 minuti per esplorare l'applicazione (valutazione utente olistica). Deve essere comunicato che l'utente verrà registrato (con presentazione di modulo dedicato) e di pensare ad alta voce

[1] La differenza tra task e scenario è che un task è un compito da adempiere ma in maniera generale, scenario è invece un insieme prescrittivo di passi (nel Behavios-driven development è la sequenza given when then)

mentre usa l'applicazione (think aloud protocol analysis), esprimendo quello che gli piace e vuole fare e quello che non gli piace e non riesce a fare. Uno del gruppo base gli deve stare vicino, senza prendere appunti. Un altro prende appunti, segnandosi tempi ed espressioni verbali particolari (queste espressioni sono da inserire all'interno delle slide).

6. presentare un task di quelli definiti al passo 2. Meglio farlo con una presentazione o filmaot che spieghi obiettivo e macropassi.
7. fare eseguire all'utente il task (non più di 3-4 minuti, valutazione utente task based) e continuare con la protocol analysis. Se ha bisogno di aiuta la persona vicina gli dia suggerimenti
8. dalla sessione di uso di ciascun utente estrarre collaborativamente un elenco di problemi di usabilità
9. mettere insieme le liste individuali e quelle che sono state prodotte in sessione con gli utenti, decidendo cosa togliere e cosa lasciare
10. predisporre un questionario con il quale prioritizzare noi e agli utenti i problemi trovati
11. associare un problema ad una o più euristiche

I task devono essere realistici, per fare alcuni esempi:

- user goal: esplorare le offerte dei prodotti e acquistarne uno
- task concepito male: acquista un paio di scarpe da corsa della Nike arancioni
- task migliore: compra un paio di scarpe sotto i 40 euro

I task devono essere eseguibili:

- user goal: cercare un film e i suoi orari

- task concepito male: vuoi vedere un film di domenica pomeriggio. Vai sul sito X e dimmi dove cliccheresti come prima cosa
- task migliore: usa il sito X per trovare un film che ti piacerebbe vedere di domenica pomeriggio

Evitare suggeriemnti e descrizioni dei singoli passi:

- user goal: consultare i voti online
- task concepito male: vuoi vedere i voti del tuo esame IUM. Vai sul sito del corso, loggati e dimmi dove cliccheresti per vedere il tuo voto
- task migliore: cerca sul sito del corso il risultato dell'esame IUM

A fine progetto bisogna anche includere una matrice con una serie di dati che mettano assieme i problemi trovati e con che frequenza per poi identificare gli esaminatori migliori e i problemi più difficili:

Scala di valutazione dei problemi

0 Non sono d'accordo che questo sia un problema di usabilità

1 **è solo un problema "cosmetico"**: non deve essere risolto a meno che nel progetto non sia disponibile del tempo extra

2 **problema secondario**: alla sua risoluzione bisognerebbe dedicare bassa priorità

3 **problema rilevante**: è importante risolverlo, bisognerebbe dare alta priorità alla sua risoluzione

4 **catastrofe di usabilità**: è imperativo risolverlo prima che il prodotto possa essere rilasciato

Con questo specchietto si può prioritizzare la lista di problemi. Nel rapporto di severità bisogna riportare anche il numero di persone coinvolte in questa valutazione della severità.

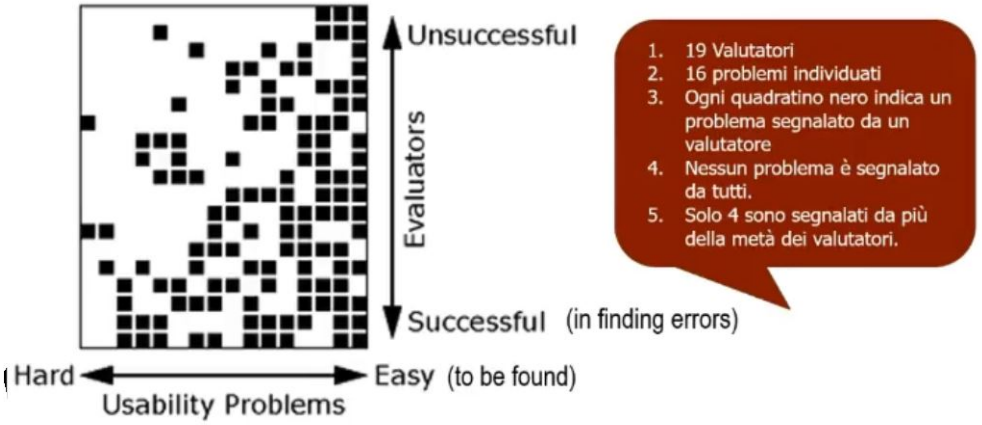


Figura 2.5: Matrice problemi/utente

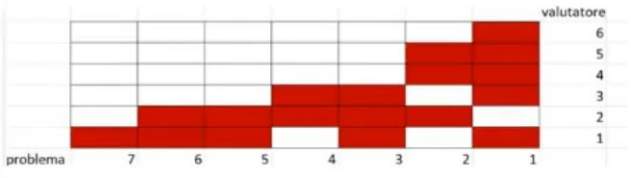


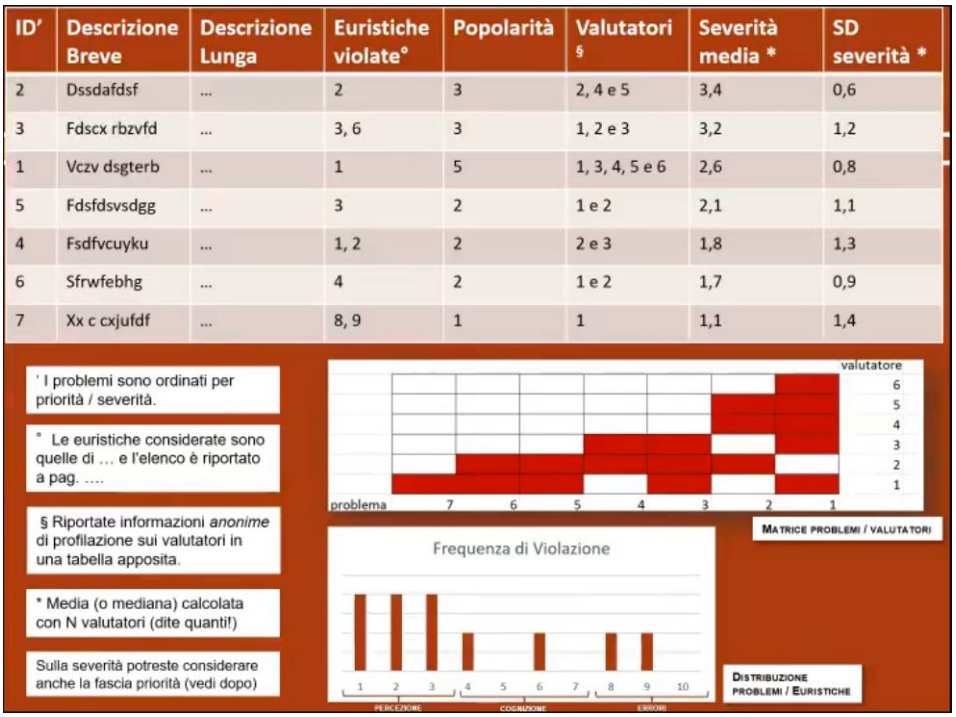
Figura 2.6: Matrice problema/utente

Alcuni indicatori per la valutazione della gravità sono:

- La frequenza o probabilità di occorrenza: è comune o raro?
- L'impatto del problema SE esso occorre: è facile o difficile da superare? Quali sono le conseguenze all'atto pratico?
- La persistenza: è un problema in cui lo stesso utente può incappare più volte anche essendone consapevole e a conoscenza o è di fatto una singola occorrenza?

Perchè fare una valutazione euristica?

è una tecnica dell'approccio più generale detto di "discount usability engineering" (veloce, efficiente, efficace). Nielsen ha riscon-



ID'	Descrizione Breve	Descrizione Lunga	Euristiche violate°	Popolarità	Valutatori §	Severità media *	SD severità *
2	Dssdafdsf	...	2	3	2, 4 e 5	3,4	0,6
3	Fdscx rbzvfd	...	3, 6	3	1, 2 e 3	3,2	1,2
1	Vczv dsgterb	...	1	5	1, 3, 4, 5 e 6	2,6	0,8
5	Fdsfdsvsdgg	...	3	2	1 e 2	2,1	1,1
4	Fsdfvcuyku	...	1, 2	2	2 e 3	1,8	1,3
6	Sfrwfebhg	...	4	2	1 e 2	1,7	0,9
7	Xx c cxjufdf	...	8, 9	1	1	1,1	1,4

Figura 2.7: Esempio di riassunto dei riscontri

trato spesso rapporti benefit-cost > 1:50 (ovvero la valutazione costa 10k ma i benefici attesi sono stimati a 500k). Questi benefici derivano dalla non segnalazione di difetti, minore numero di rinunce da parte degli utenti.

2.3.2 Valutazione euristica vs. test utente

Pro della valutazione euristica:

- è più veloce (quindi meno costosa)
- i risultati sono già in forma interpretata

Contro:

- meno accurata: può non individuare dei problemi o produrre falsi positivi
- non tiene conto degli utenti veri e dei loro compiti situati

2.3.3 Valutazione euristica: risorse necessarie

- Un sistema funzionante oppure un prototipo o un mock-up.
- n valutatori
- un osservatore se vengono coinvolti utenti esperti
- tempo:
  - In ambito professionale una sessione individuale di osservazione dura tra i 30 e i 90 minuti ma noi siamo autorizzati a farle durare 5-15
  - ogni valutatori dovrebbe riesaminare l'interfaccia più volte

2.3.4 Valutazione euristica: pianificazione

2.3.5 Alcuni errori frequenti

- Condurre la valutazione aggregando i problemi e non analizzandoli uno ad uno: **i problemi di usabilità vanno descritti uno ad uno**
- Nella discussione di un problema proporre soluzioni: **questo lavoro va fatto nella fase DOPO la valutazione euristica, non ne fa parte**
- Considerare un principio alla votla e indicare i problemi che violano tale principio: **in realtà ogni problema può violare più principi, si parte dai problemi**
- Mischiare problemi e soluzioni: **l'output della valutazione è un elenco di problemi prioritizzati**

Assessing appropriate ways to use heuristic evaluation, 4 people @ 2 hours	8
Having outside evaluation expert learn about the domain and scenario	8
Finding and scheduling evaluators, 1.8 hours + 0.2 hours per evaluator	4
Preparing the briefing	3
Preparing scenario for the evaluators	2
Briefing, 1 system expert, 1 evaluation expert, 11 evaluators @ 1.5 hours	19.5
Preparing the prototype (software and its hardware platform) for the evaluation	5
Actual evaluation, 11 evaluators @ 1 hour	11
Observing the evaluation sessions, 2 observers @ 11 hours	22
Debriefing, 3 evaluators, 3 developers, 1 evaluation expert @ 1 hour	7
Writing list of usability problems based on notes from evaluation sessions	2
Writing problem descriptions for use in severity-rating questionnaire	6
Severity rating, 11 evaluators @ 0.5 hours	5.5
Analyzing severity ratings	2
Total	105

Figura 2.8: Esempio di pianificazione di Nielsen

2.3.6 Come si prioritizza?

Per ogni problema identificato dal gruppo, far valutare indipendentemente la severità, e coinvolgere anche una decina di colleghi della azienda e una decina di potenziali utenti (che non siano colleghi dell'azienda).

In poche parole, cercare di ottenere 20-30 valutazioni ordinali di severità per ciascun problema

Il fine è identificare i problemi più urgenti da indirizzare con le risorse a disposizione.

A questo punto si estrae dal sistema di rilevazione (il questionario online) la tabella delle valutazioni (i problemi indicati sulle colonne, le valutazioni di severità nelle righe, una per singolo valutatore).

Per ciascuna riga considerate la classifica di severità adottando la strategia di competizione standard (ovvero un pareggio conta a livello di ordine come due livelli separati, non 1223 ma 1224), costruendo una nuova tabella delle posizioni (a partire da quella delle valutazioni): il o i problemi in testa sono quelli che

hanno ottenuto il punteggio più alto.

Considerare 2 fasce: quella dei problemi a priorità più alta considera le prime ne posizioni, con n circa 20% del numero totale dei problemi, t e poi la fascia delle restanti t-n posizioni. Solitamente nella prima fascia si considerano le prime 3 o 5 posizioni, cioè il podio o una specie di podio allargato. è ragionevole pensare che questi problemi siano responsabili di circa l'80% di tutti i problemi.

A questo punto contare, per ogni problema, il numero di volte che è in prima fascia (p) e in seconda fascia (s) per ogni utente.

Assegnare il problema definitivamente alla fascia p o s in base al confronto della misura precedente: se p > s allora sarà in p, altrimenti in s.

Eseguire un test binomiale per capire se la differenza tra p ed s è dovuta al caso. In base a questo test si stabiliscono tre fasce in base al P value del test:

1. Fascia alta con priorità significativa
2. Fascia intermedia di non significatività
3. Fascia bassa con priorità significativa

Nel caso da questo test non sia possibile mettere nessun problema in fascia alta il gruppo può decidere di staccare manualmente dalla fascia intermedia alcuni problemi considerati come di fascia alta.

Tutti i problemi nella fascia di alta priorità significativa dovranno essere risolti prioritariamente e scegliere poi alcuni problemi dalla fascia di non significatività.

Nella tabella dei problemi aggiungere due colonne: in una indicare la fascia di priorità, nell'altra la mediana/media dell'indice di severità indicando anche la deviazione standard.

Per chiarezza, ordinare i problemi mettendo prima quelli che sono in prima fascia e risultati anche significativamente seve-

ri. Riportare come nota il numero delle persone coinvolte nella prioritizzazione.

ID'	Descrizione Breve	Descrizione Lunga	Euristiche violate°	Popolarità	Valutatori §	Priorità	Severità media (SD)
2	Dssdafdsf	...	2	3	2, 4 e 5	A	3,4 (0,6)
3	Fdscx rbzvfd	...	3, 6	3	1, 2 e 3	A	3,2 (1,2)
1	Vczv dsgterb	...	1	5	1, 3, 4, 5 e 6	B	2,6 (0,8)
5	Fdsfdsvsdgg	...	3	2	1 e 2	B	2,1 (1,1)
4	Fsdfvcuyku	...	1, 2	2	2 e 3	B	1,8 (1,3)
6	Sfrwfebhg	...	4	2	1 e 2	C	1,7 (0,9)
7	Xx c cxjufdf	...	8, 9	1	1	C	1,4 (1,1)

Figura 2.9: Un esempio

2.3.7 Cosa contiene alla fine la valutazione euristica?

- Descrizione del design sperimentale: cosa avete valutato, perchè, coinvolgendo quante persone, considerando quali task, aerogramma di profilazione, diagramma cartesiano delle expertise...
- Elenco finale dei problemi prioritizzati: tabella dei problemi, screenshot dei problemi più gravi, matrice problemi/-valutatori, distribuzione problemi/euristiche
- Considerazioni finali: legate al numero totale stimato di problemi, alla distribuzione dei problemi, alle citazioni notevoli dalle sessioni olistiche ed esperte

L'importante è la prioritizzazione dei problemi di due fasce (priorità più alta, intermedia, più bassa).

Le n schede di analisi da cui è stato redatto l'elenco finale come allegato/appendice.

Meglio uno slideware, non un documento scritto.

Nella cartella di progetto mettere anche le registrazioni e i moduli di consenso.

2.4 Statistica inferenziale (approccio quantitativo)

2.4.1 Come potremmo definire il successo di un sistema interattivo?

- gli utenti scelgono il sistema
- gli utenti sono disposti a pagare bene per il sistema
- la fetta di mercato cresce
- il fatturato/ricavi crescono
- si viene comprati/copiati da facebook
- gli utenti ne parlano bene e lo conosigliano
- gli utenti considerano il sistema parte della loro vita
- gli utenti preferiscono usare il sistema rispetto altri

Ma soprattutto è:

- **Utile**: deve fare quello che viene richiesto
- **Usabile**: deve consentire di fare quello che viene richiesto in modo naturale, senza aumentare i rischi di errore, ecc.
- **Usato**: deve rendere le persone desiderose di usarlo, quindi essere interessante, divertente, piacevole

Capitolo 3

Lessico

Ergonomia Disciplina scientifica che si occupa dei problemi relativi al lavoro umano in rapporto alla progettazione delle macchine e agli ambienti di lavoro, al fine di individuare le soluzioni più idonee alle esigenze psicofisiche dei lavoratori e al contempo a quelle della produzione.

Sistema socio tecnico Pagina 11.

- è un sistema, "un insieme di elementi interrelati ed eventualmente mutuamente dipendenti che, agli occhi di un osservatore esterno, appaiono come un'entità unitaria ma collettiva, con caratteristiche e comportamento proprio, solitamente autonomo ed intenzionale (cioè volto ad un obiettivo)";

- Un sistema in cui la componente umana (sociale) e quella tecnica (tecnologica) sono inestricabilmente legate tra loro e la loro interazione porta a fenomeni emergenti imprevedibili. Attenzione che tecnica è un termine generico proprio per intendere tutti gli strumenti, che siano fisici o dell'ingegno. In più si parla di interdipendenza perchè lo strumento è fermo senza qualcuno che lo usa, l'umano è fermo se non ha uno strumento per agire;

- è un concetto di invarianza di scala, ovvero il concetto non cambia in base alla grandezza dell'ambiente sociale

Affordance Pagina 21.

Gibson: è la risorsa o supporto che l'ambiente offre all'utente

Norman: è una qualsiasi proprietà o qualità di un oggetto che definisce i suoi possibili utilizzi o rende chiaro come possa o debba essere usato.

Una affordance è qualsiasi proprietà di un oggetto che invita una persona competente all'azione mediata da tale oggetto.

Mapping Pagina 22.

É il rapporto fra i comandi, il loro azionamento ed i risultati che ne derivano nel mondo esterno; permette allutente di creare un collegamento diretto fra i comandi di controllo e le parti delloggetto di cui modificano rispettivamente lo stato.

Porte di Norman Pagina 16.

Una porta di Norman è una porta il cui design suggerisce un'azione contraria a quella da fare o che richiede un segnale per spiegare come usarla.

Toilette di Floyd Pagina 17.

Una toilette di floyd è un qualcosa di progettato in maniera così poco intuitiva per il quale devo fornire una serie di istruzioni.

Sistema interattivo e interfaccia Pagina 19.

Per interfaccia d'uso (o interfaccia utente, user interface) intendiamo l'insieme di tutti i componenti di un sistema interattivo (software o hardware) che forniscono all'utente informa-

zioni e comandi per permettergli di effettuare specifici compiti attraverso il sistema.

Automation bias/overreliance Pagina 14.

è l'eccessiva fiducia nella risposta del supporto alle decisioni e quindi causa di errori di omissione o di azione quando i sistemi sono imperfetti (e.g. la calcolatrice che deve rispondere correttamente, non può essere altrimenti). è legato ai processi decisionali.

Errori di giustapposizione Quando a causa della posizione specifica di un elemento si preme un pulsante differente rispetto a quello desiderato (e.g. due elementi in un menù a tendina che sono molto vicini tra loro)

Workaround Pagina 25.

Qualsiasi azione relativa all'esecuzione di un processo o di un compito (supportati dal sistema informatico) che non è prevista o descritta nei manuali di uso del sistema informatico e/o nei manuali che descrivono tale processo o procedura e che può bypassare l'uso del sistema o piegarlo ai propri fini.

Complessità strutturale, funzionale e d'uso Pagina 26.

- **Complessità funzionale**: indica quante funzioni possono essere svolte tramite l'interfaccia in analisi (un cacciavite a punta singola ha una bassa complessità funzionale, un coltellino svizzero ha un'alta complessità funzionale in quanto in base a come viene usato può svolgere numerose funzioni in virtù delle sue componenti)
- **Complessità strutturale**: indica, genericamente, quanto sia complessa la struttura di uno strumento (un cucchiaio ha una complessità strutturale bassa, uno smartphone una complessità strutturale alta)

- **Complessità d'uso**: indica quando sia semplice utilizzare l'interfaccia, ha un forte elemento basato anche sulle capacità personali (una maniglia ha complessità d'uso bassa, una macchina con cambio manuale ha complessità d'uso alta)

Euristica e valutazione euristica Pagina 30.

Una euristica è un insieme di concetti, riferimenti e, soprattutto, strategie che si sono rilevate particolarmente adatte a risolvere un determinato problema, nel nostro caso la progettazione di sistemi interattivi usabili.

La valutazione euristica è la valutazione (di usabilità) svolta alla luce di un determinato insieme di euristiche (ben definite, possibilmente ben note, prese a riferimento) per identificare soluzioni di design che o si conformano o violano una o più euristiche di tale insieme.

Segno, simbolo, indice, icona

Significante, significato, oggetto

Metacomunicazione

Deputy del progettista L'oggetto creato rappresenta l'artista che lo crea.

Captologia

Soggettivismo - oggettivismo

3.1 "Metodologia"

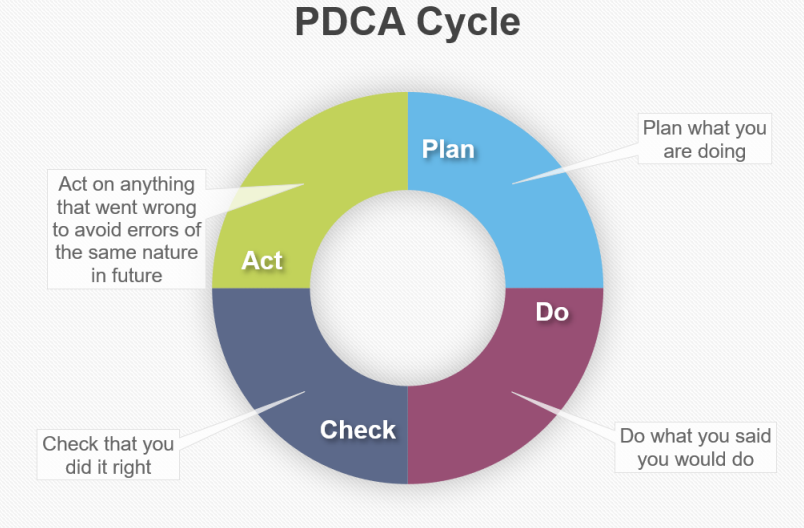


Figura 3.1: PDCA cycle

- Affina sensibilità per il brutto e il cattivo
- Progetta in equilibrio tra familiarità e originalità
- Sappi che parli all'utente anche di te stesso
- Non guardarti l'ombelico ma coinvolgi gli utenti
- Trova i problemi, prioritizzali e risolvili
- Valuta usabilità in termini di efficienza, efficacia e soddisfazione
- Migliora il tuo artefatto
- Torna agli utenti