Interazione
Uomo Macchina

Enrico Mensa,

Immagini tratte dalle slide del prof. Giovanni Maria Sacco

Interazione Uomo Macchina
Indice degli argomenti

Introduzione

Elementi di psicologia cognitiva

User - Centered Design

Accessibilità

Yale Web Style Guide

Introduzione

1) Interazione uomo macchina: perché?

1.1) L'importanza di IUM

L'interazione uomo macchina è importante per diversi fattori:

- **Business**: sfruttare le risorse umane in maniera più efficiente è fonte di risparmio.
- **Mercato**: la gente, non avendo tempo e voglia di mettersi a studiare un sistema complesso (seppur eventualmente con una ottima base ingegneristica) propenderà più probabilmente all'acquisto di applicazioni più accattivanti ed intuitive.
- **Fattori umani**: gli umani compiono errori, e non tenerne conto potrebbe costare sia economicamente che in termini di vite.
- **Società**: l'espansione del personal computer ha resto questo oggetto parte integrante dell'esistenza di moltissimi sul pianeta, quindi, non si può richiedere una "preparazione elevata" agli utenti.
- **Futuro professionale**: l'informatico deve sempre più tenere conto dell'IUM per i motivi sopracitati (più del 50% del codice di un'applicazione è oramai dedicato all'IUM).

2) Prima un po' di storia...

Per immergerci nel mondo dell'IUM, facciamo una rapida carrellata storica degli eventi fondamentali che ci hanno portato alle interfacce che oggi utilizziamo quotidianamente.

2.1) Fino agli anni '70: batch

Sistemi batch, si utilizzavano le schede perforate. L'output era costituito da tabulati e tra macchina e utente praticamente non vigeva interattività.

2.2) Anni '70: sistemi "conversazionali"

Si passa all'idea di sistemi time-sharing, con interazione tramite dialoghi (riga di comando). Si ha comunque una interazione modale, ovvero l'utente dopo aver comunicato col computer non può far altro che attendere la risposta, senza poter compiere ulteriori azioni.

2.3) Inizio anni '80: CRT come output e l'avvento dei PC

Il PC, sempre vissuto come strumento di lavoro, è ora più diffuso. Inoltre viene adoperato un vetro che sostanzialmente sostituisce la carta degli anni '70. Non c'è una grande miglioria, tranne per un menù (i famosi tasti F1, ..., F12) che fornisce alcune funzioni predefinite.

2.4) Anni '80 fino ad oggi: GUI

Definitivo cambiamento, come output abbiamo i video grafici e usiamo tastiera e mouse come input. Sistema a finestre (fondamentale, non c'è più interazione modale).

2.5) Anni '90 fino ad oggi: il computer come mezzo multimediale

Si aggiungono immagini, suoni, video, ma questo ovviamente crea problematiche da risolvere come la composizione di pagine per essere standardizzate (layout, ecc.) piuttosto che l'organizzazione dell'informazione e della navigazione.

Elementi di psicologia cognitiva

1) Lo schema di interazione

Dobbiamo cercare di sviluppare prodotti ben progettati, incentrati sull'uomo.
Quindi si rende necessario uno studio di come l'uomo vede e percepisce le cose che lo circondano. Questo studio fa parte della psicologia cognitiva. L'essere umano segue questo schema in seguito ad un input sensiorale:

- input sensoriale
- codifica
- comparazione dello stimolo con stimoli pregressi grazie all'uso della memoria
- selezione della risposta
- esecuzione della risposta
- output

2) La percezione

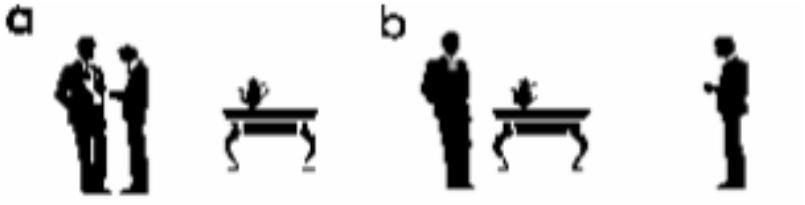
Parliamo prima di tutto della percezione. È necessario distinguere la **percezione** dalla **visione**. Il primo è un fenomeno *attivo*, quindi che richiede l'utilizzo di attenzione e la scelta di percepire, mentre il secondo è un fenomeno *passivo*, che accade "senza pensarci".

Abbiamo due teorie che ci introducono alla percezione, vediamole separatamente.

2.1) Teoria costruzionista / gestalt

Secondo questa teoria la percezione è costruita tramite gli stimoli dall'ambiente e dai ricordi precedenti. Troviamo diversi schemi percettivi, vediamoli nel dettaglio.

Prossimità



Nel caso a percepiamo due uomini e un tavolo, nel caso b percepiamo un uomo, un tavolo ed un altro uomo. Oggetti vicini vengono quindi "accomunati", pertanto, dovendo aggregare dei tasti, sarà buon uso mettere vicini i tasti con funzioni simili o comunque accomunabili fra loro.

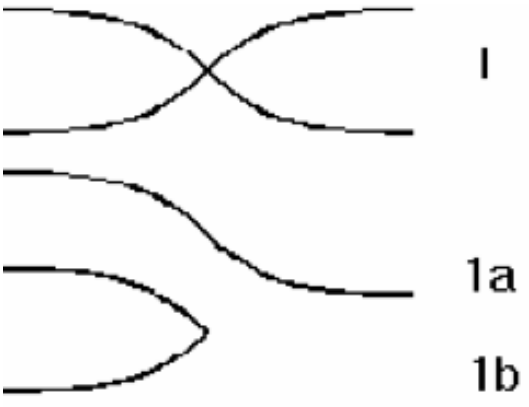
Similarità

A sinistra vediamo una X costituita da pini, a destra invece vediamo una L rovesciata. Pertanto tendiamo a vedere elementi omogenei in relazione fra loro, un po' come la prossimità.

Chiusura

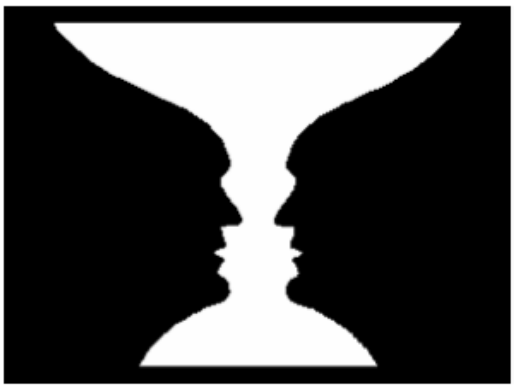
In questo caso tendiamo invece a "chiudere" le linee, quindi percepiremo la scritta "Washo" interamente, così come il quadrato. Questo può essere utile nel disegnare piccole icone. La non interezza attira l'attenzione.

Continuità



Per natura tenderemo ad interpretare l'immagine 1 come due linee del tipo 1a intersecate e mai come 1b. Sempre utile per il disegno di interfacce grafiche.

Consideriamo infine questo esempio:



Lo sfondo è il nero oppure il bianco?

Nel disegno di interfacce grafiche dobbiamo porre attenzione alla scelta dei colori.

2.2) Teoria ecologica

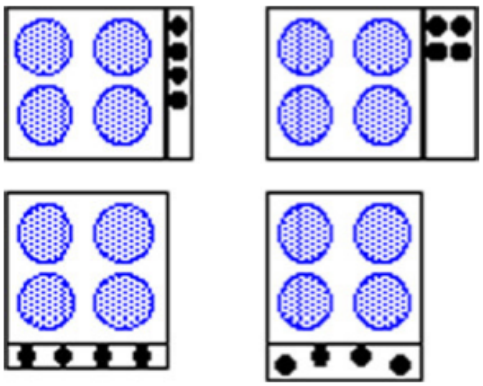
Secondo la teoria ecologica noi tendiamo a vedere gli oggetti basandoci sulla loro funzionalità. Questo è derivato dall'uomo primitivo che vedeva le cose per la loro utilità, come ad esempio un sentiero era una via per procurarsi il cibo. Si pone attenzione all'**affordance**.

L'**affordance percettiva** (ciò che viene percepito) si basa su due principi (oltre al banale fatto che un bottone venga visto come generatore di azioni): i vincoli ed il mapping.

I vincoli

Abbiamo vincoli **fisici** (una maniglia non può essere aperta al contrario), vincoli **semantici** (comando grigio non si può usare) vincoli **culturali** (in certi paesi si usano le scarpe sulle toilettes) e vincoli **logici** (i tasti di un menù sono raggruppati per logica e non sparsi a casaccio).

Il mapping



Non è banale capire a che controller corrisponde quale fornello. Il mapping si occupa proprio di questo.

Il mapping si può basare su **indizi spaziali** (ad esempio nel caso di tasti su layers diversi, un tasto sul layer più "a fondo" creerà azioni per tutta la finestra rispetto a uno sul layer più "in alto"), **etichette** (che indicano semplicemente la funzione tramite il testo) e le **icone**.

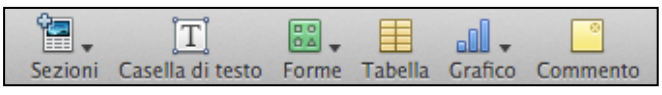
Le icone sono molto importanti perché forniscono leggibilità immediata, il testo invece, richiedendo la lettura, è più impegnativo e quindi meno immediato.
Infatti i segnali stradali, leggibili ad alta velocità, sono puramente simbolici.

Il mapping può risultare più o meno facile da implementare a seconda di:
- **contesto** (immediato e socioculturale - la scarpa da donna sul bagno della toilette)
- **compito** (warning in un messaggio, tipo di file in una lista)
- **forma di rappresentazione** (icone):
  - *icone per similitudine*: si rappresenta l'immagine reale, come una frana nei segnali stradali.
  - *icone per esempio*: si fa una sineddoche, una parte per il tutto, le forchette al ristorante.
  - *icone simboliche*: si prende un simbolo per rappresentare una situazione, quindi il bicchiere rotto per dire "fragile": la scatola non contiene bicchieri rotti.
  - *icone arbitrarie*: si devono spesso rappresentare concetti astratti, quindi si crea una icona "arbitraria", ma **deve essere imparata**.
- **natura del concetto rappresentato** (astratto è più difficile da rappresentare!)

È possibile associare

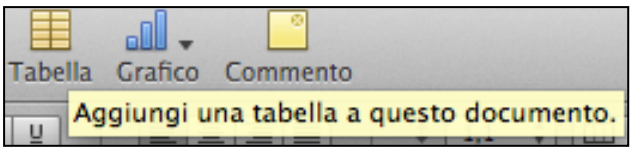
2.3) Descrizioni sotto le icone e tooltips

Per semplificare l'utilizzo dell'interfaccia è oltremodo possibile aggiungere delle descrizioni alle icone. È possibile aggiungere la **descrizione sotto alle icone:**



questo ha il vantaggio di aiutare l'utente poiché la descrizione è costantemente presente sotto alle icone, ma è anche vero che quelle descrizioni occupano spazio oltre a rischiare di non essere sufficentemente descrittive.

Una seconda tecnica consiste nell'aggiungere i **tooltips**, piccole finestre che compaiono quando il cursore del mouse è su uno strumento della toolbar.



Queste finestre sono a scomparsa e possono essere di dimensioni variabili quindi i problemi di prima sono risolti. Tramite i tooltips, però, l'utente dovrà esplorare tutta l'interfaccia almeno una volta per conoscere ogni funzione.

2.4) Le toolbars

Le **toolbars** sono barre che si trovano nei software ed hanno la funzione di raggruppare alcuni pulsanti per un utilizzo più immediato e veloce. Sono quindi funzioni che sono utilizzabili anche tramite il menù classico, ma che vengono messe su barre per velocizzarne l'uso. Alcune linee guida per la progettazione delle toolbars:

- icone descrittive
- icone facilmente distinguibili fra loro
- uso corretto del colore (senza esagerare)
- evitare l'affollamento (porta confusione)
- la toolbar deve contenere solo le funzioni usate più di frequente
- i pulsanti devono essere raggruppati logicamente (gestalt)
- uso di tooltips
- eventuale opportunità di personalizzare la toolbar

3) Visione e percezione: considerazioni fisiche

Vediamo ora dal punto di vista più fisico la visione e la percezione di immagini e testo.

3.1) Patologie della vista

Astigmatismo
Difficoltà nel seguire linee. Si sconsigliano lunghe linee di testo.

Presbiopia
Difficoltà a leggere da vicino. Si consiglia di implementare la possibilità di ingrandire il testo.

Daltonismo
Difficoltà/incapacità di distinguere alcuni colori. Si sconsiglia di basare l'interfaccia unicamente su codifiche di colore.

Cataratta
Diminuzione della quantità di luce che passa all'interno dell'occhio. Si consiglia di mantenere un alto contrasto fra gli elementi.

Si noti che tutti questi disturbi (a parte il daltonismo) sono molto frequenti e aumentano anche con l'avanzare degli anni.

3.2) L'accessibilità: un diritto per il disabile

L'accessibilità è la capacità dei sistemi informatici di erogare un servizio che fornisca informazioni senza discriminazioni, quindi anche fornendo eventuali configurazioni o funzionalità particolari per gli utenti disabili.

3.3) Il contrasto e la leggibilità

Nella scelta dei colori da adottare in un sito web, ad esempio, dobbiamo porre attenzione a non scegliere particolari configurazioni estremamente difficili da leggere se non addirittura fastidiose.

Qui possiamo vedere come "Blu su nero" e "Blu su rosso" siano poco leggibili.

Formule per luminosità e contrasto

Vi sono alcune utili formule per capire se la configurazione scelta sia appropriata o meno. Red_x sta per il valore RGB del colore rosso per il colore x-esimo.

Luminosità:

```
L(a) = ((Red_a X 299) + (Green_a X 587) + (Blue_a X 114)) / 1000
```

|L(1) - L(2)| deve essere **>= 125**.

Contrasto:

```
(MAX(Red_a, Red_b) – MIN(Red_a, Red_b)) +
(MAX(Green_a, Green_b) – MIN(Green_a, Green_b)) +
(MAX(Blue_a, Blue_b) – MIN(Blue_a, Blue_b)) +
```

deve essere **>= 500**.

Si sconsiglia l'uso di **immagini come background** se non in rari casi (titoli di coda).

3.4) Il movimento dell'occhio

L'occhio umano non è in grado di percorrere lunghe distanze durante la lettura, poiché nel momento del cambio di linea deve "cercare" la linea successiva se la riga appena letta dovesse essere troppo lunga. Ecco perché i quotidiani sono scritti in colonne.

Questo testo:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet.

Risulta molto più leggibile se scritto **in colonne** così:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim.

In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet.

Si noti che nel caso di utilizzo di scrollbar, la lettura potrebbe diventare complessa (finita una colonna si deve tornare da capo a inizio pagina).

3.5) Leggibilità di dati tabellari

PAZ	Inizio	Fine	Reparto	COD	Nome	Primario
A102	02/05/94	09/05/94	A	A	Chirurgia	203
S555	05/10/94	01/11/94	A	A	Chirurgia	203
B444	01/12/94	02/01/95	B	B	Pediatria	574
A102	02/12/94	02/01/95	P	-	-	-

Questa tabella risulta difficile da leggere. Potremmo allora inserire delle colorazioni differenti per le linee orizzontali. Si sfrutta il gestalt di chiusura per le bordature verticali.

PAZ	Inizio	Fine	Reparto	COD	Nome	Primario
A102	02/05/94	09/05/94	A	A	Chirurgia	203
S555	05/10/94	01/11/94	A	A	Chirurgia	203
B444	01/12/94	02/01/95	B	B	Pediatria	574
A102	02/12/94	02/01/95	P	-	-	-

3.5) Cenni di tipografia elettronica

Alcune definizioni:
- *Font topografico*: è una rappresentazione dei caratteri dell'alfabeto.
- *Corpo topografico*: dimensione del carattere misurata in punti.
- *Punto topografico*: 1/72 di pollice.
- *Spaziatura:* proporzionale o non. La spaziatura proporzionale avvicina le lettere più o meno a seconda di quali lettere sono, mentre una spaziatura non proporzionale riserva sempre lo stesso quantitativo di spazio per ogni lettera. Es: Spaziatura - `Spaziatura`

Due famiglie: serif e sans-serif

I serifs sono piccoli **abbellimenti** aggiunti alle lettere per farle apparire come fossero in corsivo e quindi più legate fra loro, per aumentare la leggibilità.
Per lunghi testi è quindi estremamente consigliabile l'utilizzo di un carattere serif. Nel caso il carattere sia però molto piccolo, si suggerisce invece l'uso di un carattere sans serif poiché i monitor non hanno abbastanza risoluzione per rendere apprezzabili i serif che quindi rovinano le lettere rendendole quindi meno leggibili. Vediamo qui confrontati un carattere sans-serif con uno serif.

at at

Caratteri anti-aliased

Per aumentare la leggibilità si usa la tecnica della **scala di grigi** che rende più "fluido" il testo e meno spigoloso. Vediamo qui confrontate (e zoommate) le due scritte prima con l'uso dell'anti-aliased e poi senza.

Il tono della comunicazione

Ogni font trasmette subliminalmente un "**tono**" di conversazione. Vediamo alcuni esempi.

Times New Roman:	molto formale (ufficiale)
Avenir:	semplice ed elegante (schede tecniche)
Helvetica/Arial:	neutro
Verdana:	informale (email fra amici)
Comic Sans MS:	molto informale (fumetti)
Giddyup Std:	stravagante (titoli)

Si suggerisce comunque l'uso di font diffusi in modo che l'utente non debba installare sulla propria macchina il font necessario.

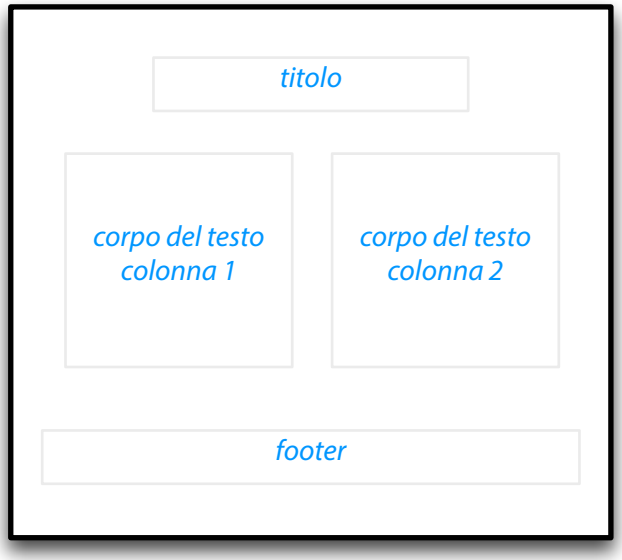
Il sovraccarico (di testo e colore)
Durante la scrittura di testi è sconsigliato l'uso massiccio di grassetto, corsivo, sottolineato, poiché si ottiene l'effetto inverso a quello desiderato: invece di evidenziare si rende il testo confusionario.
Si adoperino al più due font diversi.

4) Composizione della pagina

Il **layout** della pagina è, come si può immaginare, fondamentale. Esso non solo deve essere "ben fatto" per permettere all'utente una facile lettura, ma, nel caso di siti piuttosto che di libri, fornire anche una "firma" grafica, ovvero deve svolgere la funzione di **riconoscimento** immediato del prodotto.

4.1) La gabbia

La gabbia è una griglia che funge da template per ogni pagina di un sito web. La composizione della griglia è fondamentale poiché regola le proporzioni fra spazi vuoti e colonne di testo, oltre che a fornire, come detto sopra, una firma che permetta all'utente di riconoscere lo stile di una certo prodotto.



4.2) Allineamenti di testo

L'allineamento a **sinistra** può essere a bandiera (primo esempio) oppure giustificato (secondo esempio):

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim.

Come si vede, il testo giustificato risulta più leggibile. Per questo è il più usato per testi lunghi.

L'allineamento al **centro**, sconsigliato per il testo (primo esempio), ottimo per i titoli (secondo esempio):

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

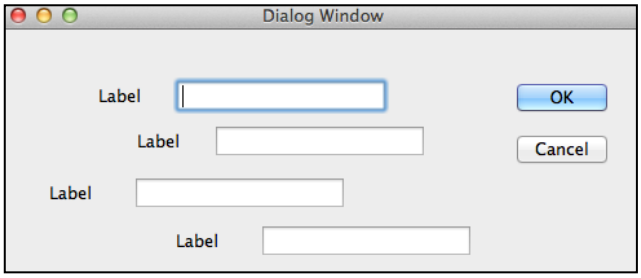
TITOLO

L'allineamento a **destra**, usato molto raramente, risulta poco leggibile:

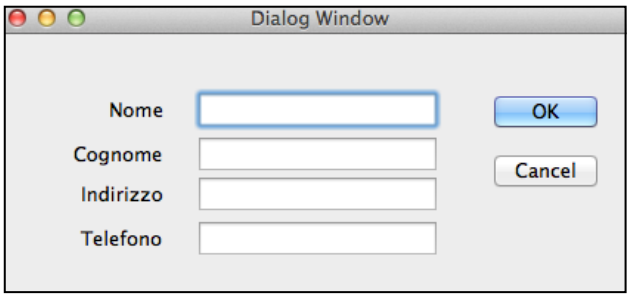
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

4.3) Disegno di interfacce grafiche (finestre di dialogo)

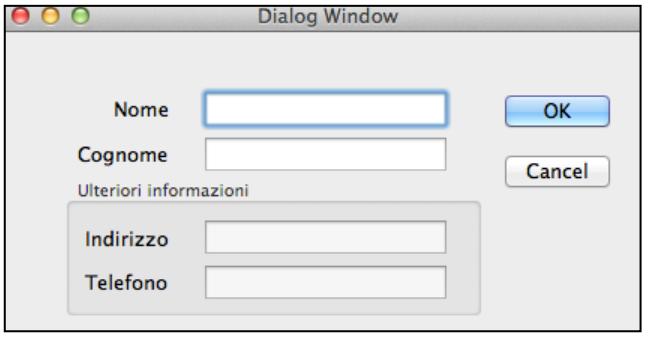
È necessario l'utilizzo di una chiara organizzazione, logica, degli elementi in una finestra per permettere all'utente.



Vediamo una finestra mal organizzata e non chiara. Organizzando per sezioni la pagina, invece, rendiamo molto più leggibile.



Ulteriore miglioria potrebbe essere l'aggiunta di box che raggruppano alcune celle (si adopera anche il gestalt).



5) Attenzione e memoria

La nostra memoria è paragonabile ad un sistema di piccole memorie gerarchiche che gradualmente contengono e conservano più informazioni.

5.1) Diversi tipi di memoria

- **Memoria sensoriale**: ha permanenza di qualche decimo di secondo (ad esempio quando tocchiamo qualcosa di freddo)
- **Memoria di breve periodo**: ha permanenza di qualche secondo, immaginiamola come una cassettiera ove è possibile memorizzare dai cinque ai nove oggetti che possono essere cifre piuttosto che concetti (identificati tramite un'etichetta).
- **Memoria di lungo periodo**: grande permanenza ma è soggetta a decadimento.

Perché è importante per noi? Se l'astrazione viene adoperata in maniera corretta possiamo stimolare la memoria a breve periodo e quella a lungo periodo e migliorarne così il rendimento.

5.2) L'attenzione

Innanzi tutto dobbiamo considerare che è possibile avere un'**attenzione focalizzata** oppure una **divisa**. Compiere più task insieme è pericoloso poiché i task secondari interrompono il primario.
Abbiamo alcune tecniche per focalizzare l'attenzione.

- Strutturare (un buon quantitativo di informazioni ordinate e chiare. Con troppe informazioni avremmo confusione, con troppe poche non si coglierebbe il contesto)
- Indizi spaziali (posizionare i tasti sempre nello stesso posto nelle message box: consistenza)
- Indizi temporali (progress bar)
- Codifiche di colore (reverse quando si seleziona del testo)
- Tecniche di avvertimento (flash, beep, ecc.)

Diversi messaggi, diversa priorità

Relativamente ai messaggi che vengono dati all'utente, il modo con cui vengono presentati è indicativo della loro criticità. Un message box (pop up) è molto importante, una status bar fornisce informazioni utili ma non fondamentali (sotto a VLC ad esempio), e infine abbiamo informazioni ottenibili solo su richiesta, come potrebbero essere i file di log.

Icone vs Comandi

È più semplice riconoscere di ricordare, quindi le icone avranno sempre un grande vantaggio rispetto ai comandi (pensiamo al terminale di unix).

5.3) L'automatic processing

Relativamente alla memoria è fondamentale ricordare un processo che il nostro cervello compie continuamente. Azioni ripetute spesso e per lunghi periodi tendono a diventare automatiche. Pensiamo ad esempio al premere sempre "ok" in fase di installazione, piuttosto che l'utilizzo delle shortcut. L'automatic processing può essere utilizzato per migliorare molto l'interazione con il prodotto, ma allo stesso tempo è necessario ricordare che:

- Non è richiesto sforzo per effettuare automatic processing
- Non ci sono limiti cerebrali
- Le abitudini prese sono molto difficili da cambiare

Si pensi al fallimento delle tastiere ergonomiche oppure al fatto che i piloti spengano automaticamente le sirene di allarme poiché scattano in momenti non previsti, vanificando l'effetto dell'allarme.

Talvolta si può scegliere di disegnare contro l'automatic processing, per far "cadere in un tranello" l'utente e renderlo più attento.

6) Rappresentazione della conoscenza e modelli mentali

Vi sono due macro-tipi di rappresentazione:
- **Simbolica**: strutture simboliche sono manipolate da regole
    - Analogica: le strutture sono principalmente immagini
    - Proposizionale: la presentazione è costituita da etichette "language-like"
- **Sotto simbolica**: reti neurali (riconoscimento del viso)

Abbiamo poi due meccanismi per organizzare:
- **Reti semantiche**: ogni nodo è un concetto. Fra loro i nodi sono correlati da interrelazioni.
- **Schemi**: i comportamenti da seguire per raggiungere uno scopo (ad esempio come acquistare un biglietto).

6.1) Le reti semantiche

I nodi (concetti) sono legati da due tipi di interrelazioni: **IS-A** (un cane è un animale) e **PART-OF** (una mano è una parte di corpo).

Le interrelazioni IS-A estendono il nodo precedente ereditandone le caratteristiche, ed eventualmente aggiungendone delle peculiarità. Si crea così la **generalizzazione** e la **specializzazione** (a seconda che si guardi in alto o in basso dell'albero).
Questa tecnica è assolutamente congeniale ad una programmazione Object Oriented.

6.2) Gli schemi

Si tratta di sequenze di comportamento utili al raggiungimento di un determinato obbiettivo. V'è un'organizzazione gerarchica.
Lo script del ristorante (Schank)
Mangiare al ristorante
    Entrare
        Entrare nel ristorante
        Cercare un tavolo
        Decidere dove sedersi se piu' tavoli vuoti Andare al tavolo
        Sedersi
    Ordinare
        ...
    Mangiare
        ...
    Lasciare il ristorante
        ...

La scrittura di scripts è essenziale e se ben fatta permette un risparmio di spazio/tempo (per stampare è necessario un solo script, inutile che ogni programma si faccia il suo!).

6.3) I modelli mentali

I modelli mentali sono piccoli mondi astratti che sono utilizzati con lo scopo di predire lo svolgimento di una determinata situazione. È quello che manca agli script, la dinamicità!
I **modelli mentali**, data la loro ambiziosa mansione, sono spesso errati o incompleti poiché non è possibile "predire" ogni situazione possibile.

Modelli strutturali
I modelli strutturali vedono il sistema per **come esso è composto** e per **come funziona**. Pertanto, tramite un modello strutturale possiamo vedere il meccanismo di accensione della televisione tramite il telecomando come un agglomerato di componenti elettrici dei quali è chiaro lo scopo.

Modelli funzionali
Si basano invece sul semplice **utilizzo del sistema**, ed è il modello che più spesso **adotta l'utente**. È più semplice e richiede meno dettagli, ma allo stesso tempo aumenta le possibilità di stabilire analogie con modelli errati e rende più difficile specificare dei nessi logici fra le componenti del sistema.
Questo modello vede il sistema del telecomando che accende la televisione come un aggeggio che accende, appunto, la televisione.

Quando si parla di analogie errate possiamo pensare a questo semplice esempio:
Un rubinetto dell'acqua aperto a diversa inclinazione, aumenta/diminuisce la quantità di acqua passante per il rubinetto stesso.
Un rubinetto del termostato invece, che sceglie la gradazione, non fa aumentare il tempo con cui i termosifoni arriveranno alla gradazione desiderata, bensì li farà staccare a quella certa temperatura.

6.4) Utilizzo e trasferimento della conoscenza
Quindi si può utilizzare la conoscenza a tre livelli differenti:
- Livello di abilità: auto processing (estremamente familiare)
- Livello di regole: scripts (familiare)
- Livello di conoscenza: situazioni nuove ed inaspettate (modelli mentali)

Il trasferimento della conoscenza
L'ideale è trasferire la conoscenza che l'utente già possiede e sfruttarla all'interno della propria applicazione. Trattasi di un trasferimento **positivo**, evitando quindi di avvicinare modelli che sono incoerenti fra loro e che confonderebbero l'utente (trasferimento **negativo**).
Per effettuare il trasferimento si fa uso di **metafore**. Le metafore devono necessariamente essere "adatte", ovvero devono portare più similitudini che incongruenze fra i due modelli! Vediamo i due tipi di metafore applicabili.

Metafore verbali
È possibile ad esempio descrivere verbalmente un word-processor come una macchina da scrivere. Per fare ciò è necessario essere consistenti anche nella terminologia. Le differenze sono poche e non sostanziali, come ad esempio il tasto backslash che nelle macchine da scrivere faceva tornare indietro (senza cancellare nulla), mentre sui programmi WP permette di cancellare un carattere.

Metafore di interfaccia
Invece di usare la metafora per creare il modello mentale, si sceglie di usare la metafora come modello mentale. Un esempio classico sono i programmi come Paint, che forniscono interazione diretta e paragonabile al reale.
Anche il filesystem è rappresentato come il sistema di archiviazione di un ufficio.

Gli oggetti rappresentati sullo schermo dovrebbero essere manipolabili così come lo sono nella vita reale, quindi la penna tira la linea e non devo usare dei comandi per tracciarla. L'utente dovrebbe essere facilitato con operazioni di **undo** per via del fatto che la nuova interfaccia aumenta la possibilità di fare errori. Gli oggetti di interesse sono subito evidenti (tramite icone).

Abbiamo perciò i vantaggi:
- utenti novizi che imparano più in fretta,
- gli esperti che lavorano più in fretta,
- gli utenti occasionali fissano i concetti di base,
- c'è feedback istantaneo all'azione,
- il sistema nella totalità è più comprensibile
- c'è meno ansia: operazione di undo

E gli svantaggi:
- Non tutti i task possono essere rappresentati con metafore (ricerca nel database)
- Spesso l'interfaccia rende più difficile la selezione di certi elementi (scroll richiesto)
- Non è sempre presente la reversibilità delle azioni

Un classico esempio di metafora è il **Drag and Drop**.

7) L'apprendimento

L'apprendimento è un processo attivo che può essere stimolato tramite una chiara affordance (vedere che cosa è in grado di fare il programma) ed una chiara correlazione del principio causa/effetto.
Ogni utente progredisce attraverso i tre step:
- Fase **cognitiva** (utente novizio): conoscenza dichiarativa, tramite parole
- Fase **associativa** (utente occasionale): le associazioni causa/effetto vengono rafforzate
- Fase **autonoma** (utente esperto): le informazioni vengono conservate nella memoria a lungo termine e l'utente sfrutta l'auto processing.

Alcune problematiche dell'apprendimento:

- **Gli utenti imparano con difficoltà**, tendenzialmente guardano altri fare e non leggono i manuali (che sono comunque mal scritti!).
- Inoltre gli utenti non devono necessariamente avere nozioni concernenti il computer, quindi i messaggi per interloquire con l'utente devono essere esposti in modo chiaro e non tecnico.
- È facile provocare interpretazioni errate causando così misunderstanding di metafore.
- È difficile notare che un problema può generarne un'altro (interazione).
- L'interfaccia è troppo complessa: niente feedback sulle azioni.
- Manuali fatti male.

8) Case study: il pannello autostradale

Un sistema adottato negli anni '70 per la regolazione dei cartelli autostradali è gestito da un operatore.
In seguito ad una non immissione di un messaggio di "allerta nebbia" ci sono diversi incidenti e anche alcuni morti.
La colpa viene data all'operatore. Esaminiamo la situazione.

8.1) I dettagli del software

Le caratteristiche del software sono le seguenti:
- Deve essere inserito un codice XR300/1 nel dispositivo, con interpretazione: "X" come imposta, "R" come autostrada M5, "300" come numero di pannello e "1" come nebbia.
- Il comando viene eseguito su una shell che non da risposta in caso di esito positivo dell'operazione.
- I messaggi di errore non sono comprensibili (Error 7).
- Il nastro della teletype utilizzata era usurato.
- L'operatore aveva anche altri compiti.

8.2) Riflessioni

Notiamo che si può delineare un pattern di progettazione per evitare i problemi che hanno portato al disastro.
- **Analisi dei task** (quali sono fondamentali, quali critici, quanti sono, ecc.)
- **Coinvolgimento dell'utente** (è l'utente a sapere cosa è più frequente e cosa meno e soprattutto può fornire un punto di vista differente e più importato all'interazione facilitata con la macchina: è esperto del problema)
- **Prototipazione** (fornire diversi prototipi per coinvolgere l'utente)
- **Personalizzazione** (interfacce diverse per diversi utenti, Model View Controller)
- **Minimizzare gli errori** (ingrigire i comandi poco sensati, controllo sull'input -7000 km di coda-)
- **Ambiente fisico in cui il lavoro si svolge** (illuminazione, ergonomia)
- **Limitazioni dell'operatore** (difficoltà fisiche, di orientamento)
- **Ri-organizzazione del lavoro** (decentralizzare i fattori critici sui caselli, ad esempio, automatizzare!)

9) I modelli di interazione

9.1) Modello di Shneiderman

È necessario effettuare una scissione fra i due tipi di conoscenza.

Conoscenza Semantica

La conoscenza semantica è a sua volta composta da:
- **Semantica del task**: è necessario comprende il compito che deve essere eseguito. A che livello di astrazione è? È indipendente dalla tipologia di computer? È indipendente addirittura dal computer stesso? (ad esempio si può scrivere anche a mano!)
- **Semantica del computer**: è necessario apprenderla, ma deve essere fatta bene per poter essere applicabile su ogni device. Ad esempio diversi sistemi operativi salvano i file in modo diverso, ma il concetto di salvataggio di file è sempre lo stesso e quindi l'utente non rimane destabilizzato. Si tratta quindi di un grado di astrazione più alto del singolo concetto di macchina/task.

Conoscenza Sintattica

Ben diversa è la conoscenza sintattica, che è device dependent. L'effettivo svolgimento dei task dipende dal sistema operativo, e quindi deve essere appreso.

I mappaggi

Quindi, lo scopo viene tradotto in un insieme di comandi (task) che sono dipendenti dalla semantica macchina, dalla semantica del computer e infine dalla sintassi.
I tre livelli devono essere il più uniforme possibile, in modo da facilitare l'apprendimento.

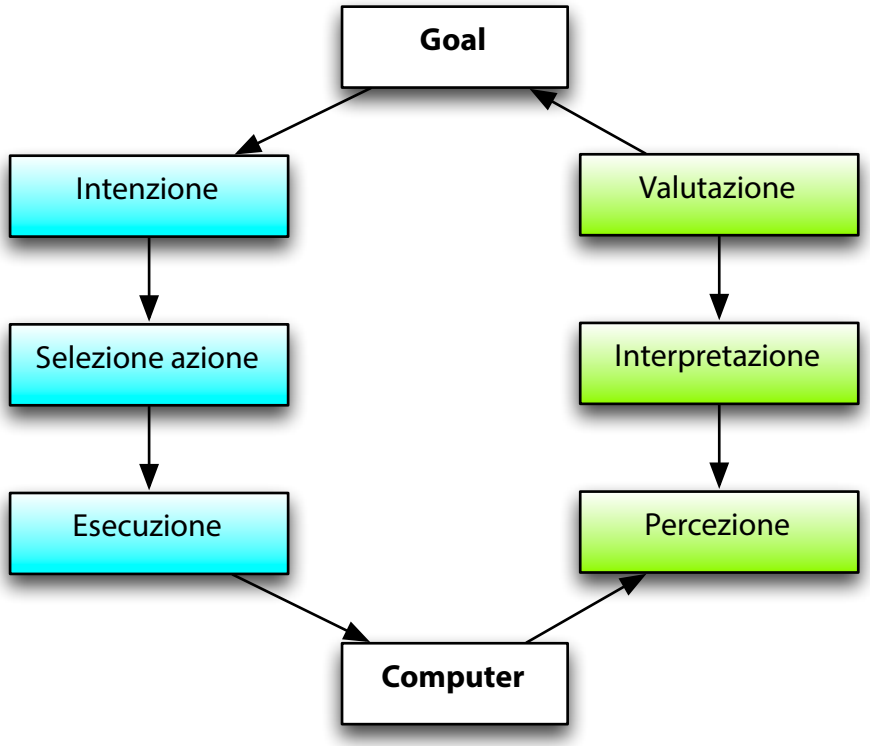
Semantica del task -> Semantica del computer -> Sintassi del computer

Per il primo passaggio è ottimo l'utilizzo delle metafore.
Nei manuali è poi suggerita una spiegazione chiara e senza fronzoli, come ad esempio il concetto di "scrivere una lettera" deve essere tradotto in "apri il Word Processor, scrivi, salva". Senza ulteriori aggiunte.

Il secondo passaggio può rappresentare difficoltà in quanto lo stesso task astratto può essere rappresentato in modi diversi a seconda dell'applicazione, oppure una semplice azione astratta viene descritta sintatticamente con molti più passaggi (complessi). I comandi, infine, possono risultare non chiari o ambigui.

9.2) Modello di Norman

Il modello di Norman completa il modello di Shneiderman per quanto riguarda il feedback all'utente.
Ogni goal (obbiettivo dell'utente), infatti, dovrebbe seguire questo sviluppo:



- **Goal**: obbiettivo dell'utente [scrivere una lettera a Bill]
- **Intenzione**: Semantica del task (massima astrazione) [scrivere una lettera]
- **Selezione azione**: Mappaggio tra semantica del task e semantica del computer [aprire il programma WP]
- **Esecuzione**: Mappaggio tra semantica e sintassi [click su World]
- **Percezione**: Osservo il cambiamento di azione prodotto [si apre una finestra]
- **Interpretazione**: Interpreto l'evento appena accaduto secondo il mio modello mentale [la finestra si chiama Excel]
- **Valutazione**: Il risultato mi ha aiutato ad arrivare al goal? [No, ho attivato lo spreadsheet]

Cosa sono i golfi?
Quindi tra il computer ed il goal si frappongono diversi livelli. Possiamo vedere quei livelli come "**Golfo dell'esecuzione**" e come "**Golfo di valutazione**".
Il primo rappresenta la difficoltà nel tradurre i goal in azioni (semantica del task -> sintassi del computer).
Il secondo raffigura invece la difficoltà nel valutare la risposta in relazione al goal desiderato (feedback).

Si vogliono quindi **eliminare i golfi**.

Il golfo di esecuzione
L'idea migliore è quella di adattare il più possibile la semantica del computer alla semantica del task (lo vediamo tramite il task analisys). Per fare questo, un mappaggio totale è certamente il metodo migliore, ovvero, si cerca di rendere l'intenzione e l'esecuzione un tutt'uno.
La riduzione di sintassi è certamente uno dei metodi migliori per perseguire il nostro scopo.
Perciò, per stampare una lettera sarà meglio adottare un drag and drop sulla stampante (richiede solamente la sintassi del mouse) piuttosto che una linea di comando come era in unix (sintassi arbitraria con nomi arbitrari).

Il golfo di valutazione
Il feedback ricevuto deve essere chiaro e visibile, ossia **percepito**. Per questo si possono adoperare dei gestalt.
Inoltre, deve essere chiaro (WYSIWYG contro i tag espliciti).
Nell'insieme si richiede consistenza nei nessi causa-effettuo affinché l'utente sappia come raggiungere i goal e non sia confuso sulle funzionalità del software.

Quattro punti contro i golfi
1) **Sviluppare un buon modello concettuale**
Rispettando la semantica del task, si può cercare l'utilizzo di metafore (se sufficentemente appropriate!). Dare sempre importanza alla coerenza totale del sistema.

2) **Visibilità alle funzioni del software**
Le funzionalità del software sono chiare ed evidenti? Lo stato del sistema è visibile?

3) **Un buon mappaggio**
Le relazioni causa/effetto devono essere chiare, ogni controllo deve avere una funzione non equivoca.

4) **Feedback**
L'utente deve avere un riscontro continuo delle azioni che compie.

Con i golfi ridotti al minimo, l'interfaccia non è più un livello frapposto fra il goal e l'utente, ma diventa **trasparente** e permette all'utente di raggiungere lo scopo in maniera efficiente e senza sforzo.

10) Linee guida per il disegno dell'interazione

Molti, erroneamente, pensano allo sviluppo dell'interfaccia solamente in ultima fase di progettazione.
Ma l'interfaccia è ciò che l'utente vede prima di tutto ed è la facciata del nostro software. Anche con un grande core, una applicazione con una cattiva interfaccia non verrà considerata.

10.1) Una semplice interazione

L'interazione con l'utente deve essere chiara e semplice, mai ridondante, con il minimo delle informazioni ed il massimo della chiarezza. Quindi:
- Usare il **modello** concettuale dell'**utente** e la semantica del **task**
- Presentare solo le **informazioni** veramente **utili** (lo schermo è prezioso!)
- **Details on demand** (informazioni aggiuntive possono essere accessibili tramite appositi menù)
- Strutturare correttamente l'informazione (**gestalt di prossimità**)
- Utilizzare **widges** per ottimizzare gli spazi (splitters, tabbed windows e property sheets, personalizzazione e docking windows)
- Minimizzare la **navigazione** fra finestre per non confondere l'utente

10.2) Utilizzare il linguaggio dell'utente

- Utilizzare la tecnologia **semantica del task** (invece di "scambio tramite POP3", "contattare l'ufficio postale")
- Se si adoperano metafore, usare SOLAMENTE i **termini della metafora** (non disco, file, ecc.)
- **Evitare neologismi** e tecnicismi informatici
- **Manuali** scritti in maniera adatta: periodi brevi, pochi sinonimi

10.3) Minimizzare il carico mnemonico dell'utente

- Evitare l'utilizzo pesante di righe di testo, sono suggerite invece **immagini** e **icone**

- Facilitare l'**input** di dati (particolarmente se formali). Ad esempio in un field per la data, fornire una data di esempio o meglio ancora fornire un calendario clickabile
- **Polimorfismo** degli operatori, il cut, il paste o il drag&drop sono applicabili ad oggetti diversi
- Meccanismi di **ripristino del contesto**

10.4) Consistenza

Le inconsistenze generano confusione nell'utente quindi vanno eliminate.
- **Evitare sinonimi** (l'utente potrebbe non capire la correlazione)
- Mantenere lo **stesso layout** per le finestre: i tasti ok e cancella non dovrebbero cambiare continuamente posizione. Si ricorda che il rischio di automatic processing è alto
- **Stessa** modalità di **interazione**, ovvero se un file viene spostato con drag&drop, dovrà essere sempre spostato tramirte drag&drop. Eventualmente possono essere aggiunte altre funzionalità, ma quella principale deve essere sempre presente per non disorientare l'utente
- Utilizzare gli **standard della piattaforma**

10.5) Feedback
- Chiarezza sullo **stato** del **sistema** (selezione dello strumento su photoshop)
- **Reazione** del sistema dato un certo stato (cursore che cambia icona)
- Specificità dell'**informazione**: ad esempio durante il compiersi di un'azione lunga, fornire una status bar con i dettagli su che cosa stia accadendo. In merito a questo si considerino questi tempi:
    - 1/10 sec, azione istantanea, nessun feedback richiesto
    - 1 sec, il ritardo è colto ma l'attenzione è focalizzata, cursore a clessidra
    - 10 sec, perdita di attenzione, cursore a clessidra
    - > 10 sec, panico, progress bar

10.6) Fornire uscite chiare

L'utente è molto frustrato nel caso in cui non si trovino uscite chiare dal programma.
- Fornire un tasto **cancel** per operazioni molto lunghe
- Fornire un tasto **reset** per la compilazione di form
- **Uscita immediata** dall'applicazione senza doversi curare delle finestre
- **Undo** per permettere all'utente di tornare indietro
- Fornire un log delle azioni, eventualmente, e la possibilità di fare **backup**

10.7) Fornire scorciatoie
- **Shortcut** per utenti esperti
- **Doppio click**
- **Toolbars**
- **Tasti di scorrimento** (task oriented)
- L'utente dovrebbe poter modificare le shortcut
- Minimizzare l'input presentando **valori di default**, e con meccanismi di **MRU** (most recently used)
- Meccanismi di **history** e **bookmarking**

10.8) Gestione degli errori

Chi fa, sbaglia. Dobbiamo tener conto del fatto che l'utente possa sbagliare. Egli può compiere un **errore**, ovvero un'azione conscia errata, effettuata a causa di un modello mentale erroneo, oppure uno **slip** cioè un errore di distrazione, magari causato da un'interferenza.

La tecnica generale di controllo degli errori si confà nell'utilizzo di UNDO (permettendo all'utente di tornare sui suoi passi) e di messaggi di alert prima di compiere azioni irreversibili o pericolose.

Errori di slip
- Errore di **cattura**, un'azione erronea si sovrappone all'azione corretta. Spesso dettata dall'auto processing. (es: digitare :wq invece che :q in VIM). La *soluzione* è quella di discernere il più possibile le azioni.
- Errore di **descrizione**, un'azione corretta compiuta sul soggetto sbagliato. Situazione classica è il click di un'icona a fianco a quella desiderata. (es: trascinare un file sul cestino invece che sulla stampante). La *soluzione* è quella di evitare la prossimità se può condurre ad errori gravi.
- Errore di **modalità**, un'azione corretta ma effettuata in modalità errata del sistema. (es: digitare comandi mentre si è in modalità di insert in VIM). *Soluzione*: eliminare il sistema delle modalità oppure rendere chiaro in quale modalità si è.
- Errore di **attivazione**: si dimentica lo scopo di una sequenza di azioni (es: la semantica del computer è molto più complessa della semantica del task). *Soluzione*: i goal frequenti o critici dovrebbero essere eseguiti direttamente, eventualmente con l'ausilio di un Wizard.
- Errore **pilotato dai dati** o di **associazione**: a causa di un interferenza esterna si compiono azioni che poi non si ricorderanno (es: rinominare un file con un nome appena sentito, che poi si dimentica). *Soluzione*: log delle azioni.

Le tre strategie contro gli errori
Vi sono tre "macro" strategie adottabili per contrastare il fenomeno degli errori.
1 - **Prevenire**: impedire che l'errore avvenga ingrigendo i tasti non utili, cambiando il cursore in maniera contestuale, non accettare dati non corretti in input, evitare la digitazione diretta.
2 - **Impedire la prosecuzione**: se viene immessa una password errata, non proseguire!
3 - **Verificare ed avvertire**: mediante suoni o message box, anche se non troppo frequenti per evitare l'auto processing dell'utente. Bisogna essere gentili nei confronti dell'user!

Altre strategie
- **Do What I Mean**: spelling checker (solo se efficiente e non invasivo).

10.9) Help
Vi sono due tipi di assistenza che dovrebbe essere fornita: una per gli utenti inesperti che si approcciano per la prima volta al programma, e l'altra per i power user che vogliono tendenzialmente avere informazioni su task meno usuali.
Si tenga presente che gli **utenti non leggono i manuali**, se li leggono **non li capiscono**, e se li capiscono **non sono aggiornati**.
È possibile fornire manuali cartacei (ma sono poco convenienti sia come costi che come aggiornabilità) oppure on-line (che possono includere sezioni how to ma soprattutto funzioni di ricerca per parola).

Si suggerisce poi l'integrazione di **tutorials**, di **training online**, di **baloon** (punto di domanda in alto), di **tooltips**, di **wizard** e di **tip of the day**.

10.10) Personabilità
Si possono adoperare **aiutanti** (Clippy su Microsoft Word) purché non siano tedianti, ridondanti e devono essere disattivabili.
Fornire poi la possibilità di personalizzare l'**interfaccia**, con finestre amovibili o raggruppabili, con dock personalizzabili, ecc.
Infine dare la possibilità di creare **macro** e l'utilizzo di **modelli predefiniti**.

User - Centered Design

1) Il coinvolgimento dell'utente

Analizziamo alcuni punti che, in fase di progettazione, è importante ricordare.

1.1) Chi utilizzerà il sistema?

Avremo **utenti esperti**, **utenti inesperti** e **utenti intermittenti**.

1.2) Che cosa farà l'utente?

La Task Analysis è fondamentale. Si suggerisce una categorizzazione dei task fra task **critici** (controlli di correttezza), task **frequenti** (ottimizzati sia nell'accesso che nell'esecuzione) ed infine task **non frequenti** (accessibili ma non direttamente).

1.3) L'esperienza pregressa dell'utente

L'uso di metafore e il trasferimento positivo della conoscenza sono importanti.

1.4) Limitazioni personali

Handicap, limitazioni di percezione e nella cognizione.

1.5) Ambiente

L'utente lavora spesso in open space: rumore ed illuminazione possono essere fastidiosi.

1.6) Organizzazione pregressa

Se il sistema che stiamo sviluppando si trova a sua volta in un'organizzazione, bisogna ponderare attentamente se sia il caso di "riorganizzare" il tutto.

L'utente deve essere quindi coinvolto durante la progettazione, poiché è l'*esperto* del settore. Egli non ne sa nulla di IUM ma esaminandone i modi di fare e le necessità si può disegnare un'interfaccia molto più efficiente.
Gli utenti sono inoltre i clienti del sistema, quindi seguendone le necessità la vendibilità del prodotto aumenta.
Di contro, l'utente spesso non conosce esattamente cosa vuole e non essendo esperto può fornire consigli travianti.

2) Le fasi di progettazione

1.1) Fase iniziale

Compiere interviste agli utenti ed una volta raccolte le idee, tramite il brainstorming, provare a miscelarle e ottenere così rappresentazioni diverse del problema.
Si definisce poi un'interfaccia basilare selezionando una delle rappresentazioni.

1.2) Fase intermedia

L'interfaccia viene chiarificata, non è importante la posizione dei vari item quanto la scelta dell'interazione che si vuole fare (pulsanti, menù, drag&drop, ecc.).

1.3) Fase finale

Si passa quindi al testing, con test di usabilità, test sul campo ed infine Alpha/Beta test.

1.4) I diversi prototipi

Come detto, è necessario prototipare molto durante la progettazione in modo da avere sempre chiaro il punto della situazione e anche in modo da avere qualcosa da mostrare all'utente, che, eventualmente, può correggere/dare consigli.
Il prototipo è, inoltre, fondamentale per quanto riguarda l'aspetto dei costi. Un prototipo riesce a dare una fotografia istantanea della progettazione e quindi permette un costante feedback da parte dell'utente. Più la fase di progettazione avanza, maggiore saranno i costi di modifica.

Prototipo a bassa fedeltà
- **Sketch**: viene disegnato su carta l'aspetto base dell'interfaccia. Costo minimo.
- **Storyboard**: le varie finestre dell'interfaccia vengono logicamente connesse. Si ha quindi una sequenza di sketch. Costo minimo.

Prototipo a media fedeltà
Si dà all'utente un'idea più realistica del sistema dell'interazione, magari con piccoli prototipi già programmabili. Il costo aumenta. Questo prototipo può mettere in luce problemi più sottili.
Abbiamo **prototipi orizzontali** (implementare tutta l'interfaccia senza funzionalità alcuna), **prototipi verticali** (funzionalità completa), **prototipi a scenario** (vengono implementati scripts fissi per mostrare le funzionalità senza implementarle veramente - ad esempio una ricerca "statica").

Dal prototipo al prodotto
Il prototipo si sviluppa in prodotto in tre modalità differenti:
- **Prototipo "usa e getta"**: ogni prototipo viene gettato via, utilizzato solo per test.
- **Prototipo incrementale**: ogni componente è prototipata, testata e poi unita al sistema totale.
- **Prototipo evolutivo**: il prototipo è lentamente migliorato e diventa il sistema finale.

Prototipazione alla mago di Oz
Quando ci sono difficili funzioni da implementare, in fase di testing l'intelligenza artificiale è sostituita dall'intelligenza reale (un addetto).

2) Metodi qualitativi di valutazione dell'usabilità

Come verificare il livello di usabilità del nostro prodotto?
Vi sono diversi metodi.

2.1) Introspezione del progettista

Tecnica economica e dagli ottimi risultati, quasi mai applicata. Il progettista si deve mettere nei panni dell'utente (non è così facile) e usare il sistema per un po', provando a capire le difficoltà dal punto di vista dell'usabilità.
Si riescono a sgrossare gli errori più grandi.

2.2) Valutazione euristica

Un piccolo gruppo di esperti di IUM (5 persone riescono a correggere il 75% degli errori) esamina il sito alla ricerca di problematiche.

2.3) Osservazione diretta

Il valutatore osserva gli utenti mentre adoperano il sistema, tramite **osservazione semplice**, chiedendo agli utenti di **parlare ad alta voce** (l'utente può essere però distratto da questa tecnica!), oppure tramite **interazione costruttiva** (due utenti vengono messi sullo stesso task e se ne ascolta il dialogo: magari utenti esperti ed inesperti insieme per evidenziare la differenza fra gli approcci).

2.4) Questionari

Sono spesso **costosi** e gli utenti non hanno **voglia** di farli (si può incentivare magari con qualche sconto su versioni successive). I questionari sono ovviamente direttamente dipendenti dalle domande.
Con **domande aperte** possiamo ottenere grandi suggerimenti / critiche, ma ci vuole qualcuno che legga e discerna ogni risposta. Con **domande chiuse**, di contro, non possiamo avere l'opinione totale dell'utente ma essa sarà necessariamente filtrata a seconda delle domande che poniamo. In un certo senso dobbiamo già conoscere il problema per porre una domanda chiusa su quella problematica.

2.5) Valutazione continua

Implementando un monitoraggio del sistema possiamo notare i bug e i problemi e fixarli nelle prossime release.
I feedback per i progettisti sono ottenibili tramite **email**, piuttosto che con un **Help Desk** oppure con un **forum**.

3) Oltre l'user-centered design

Donald Norman (lo stesso del modello di interazione Norman) fornisce una nuova interpretazione del design, che non deve essere necessariamente o solamente centrato sull'utente.

3.1) Tre livelli di design

La sua idea si basa sul concetto che **gli utenti rendono di più lavorando con prodotti piacevoli**.
Quindi distingue tre differenti livelli di design:
- **Comportamentale**: un oggetto è molto funzionale, semplice da adoperare, è facile interfacciarsi con esso e così ha successo nonostante il suo aspetto non sia dei migliori (es. Maggiolino Wolksvagen).
- **Viscerale**: un oggetto è molto bello, accattivante, gradevole. Ha quindi successo nonostante sia poco funzionale o addirittura costi sforzo all'utente imparare ad adoperarlo (es. Jaguar E).
- **Riflessivo**: un oggetto non porta soltanto con se il concetto di oggetto stesso, con le sue funzionalità ed il suo aspetto, bensì anche una marca, ovvero uno status symbol sociale (es. iPhone).

Norman nota, poi, che la maggior parte degli oggetti che usiamo oggi non sono affatto progettati secondo una metodologia UCD, basti pensare all'automobile!

3.2) Il disegno centrato sulle attività

Così, egli propone un'alternativa al UCD, si tratta di una architettura gerarchica:
Attività > Task > Azioni > Operazioni.
Il concetto di attività è quello di riunire in uno stesso oggetto molti task differenti, come un cellulare che è in grado di adempiere al task di telefonare ma anche quello di fare da rubrica.

Secondo Norman non è sempre vero che è il sistema a doversi adattare all'utente ma bensì viceversa: se pensiamo agli strumenti musicali oppure all'automobile possiamo vedere come sia l'essere umano a dover apprendere nuovi metodi e nuove tecniche per diventare "capace" nell'utilizzo dei sopracitati oggetti.
Lo sforzo richiesto è notevole ma il successo è stato ampio poiché gli strumenti sono estremamente adatti all'attività ed una volta capita l'attività lo strumento diventa comprensibile.

3.3) User-centered design: il rovescio della medaglia

Talvolta, progettare facendo uso di UCD può essere addirittura dannoso!
- Concentrarsi troppo su certe fasce di utenti preclude l'utilizzo del sistema ad altri. Non è neanche così facile capire quali sono gli utenti "target".
- Le necessità degli utenti possono portare a una frammentazione del sistema, che non sarà più coeso nella sua visione d'insieme ma apparirà invece esploso e non chiaramente amalgamato.

Accessibilità

1) Perché è importante?

L'accessibilità è il disegno che permette alle persone disabili di avere accesso a siti web ed applicazioni.
Lo strumento della tecnologia può essere impiegato per dare nuove opportunità ai disabili, che, altrimenti, non avrebbero alternative per svolgere determinate azioni.
Non si tratta solo di un **obbligo etico** ma anche di un **obbligo legale** (almeno per la pubblica amministrazione).

2) Tipi di disabilità

Vediamo una carrellata delle disabilità che possono colpire un utente (ovvero un cliente!):

- **Disabilità della vista**
    - Cecità
    - Ipovisione
    - Daltonismo
        - Deuteranopia (non si percepisce il verde ed il rosso)
        - Protanopia (non si percepisce il rosso)
        - Trinatopia (non si percepisce il blu ed il giallo)
        - Acromatopsia (non si percepiscono i colori, si vede in scala di grigio)

- **Disabilità dell'udito**

- **Disabilità motorie**
    - Paralisi parziali o totali
    - Morbo di Parkinson
    - Artrite

- **Disabilità cognitive**
    - Perdita della memoria a breve periodo
    - Morbo di Alzheimer

- **Difficoltà di apprendimento**
    - Perdita di attenzione
    - Dislessia

Si noti che su una popolazione di 750 milioni di individui nel mondo, ben 54 milioni sono disabili! Questo significa una larga fetta di utenza. Progettare tenendo conto dell'usabilità è senz'altro un costo, ma aumenta il pubblico possibile.

3) Devices

3.1) Devices di input

- Trackball gigante
- Tastiera adattata (tasti in rilievo per aiutare il puntamento)
- Mouth stick (puntare facendo uso della bocca)
- Fascia per la testa (puntare facendo uso della testa)
- Switch (pulsantone a fianco della testa)
- Switch per inspirazione/espirazione (azionato mediante il respiro)

- Eye Tracking (puntare con gli occhi)
- Controllo vocale

3.2) Devices di output
- Audio screen readers
- Braille devices

4) Le regole del W3C

4.1) Percepibilità
Il testo deve essere percepibile dai sensi. Questo è attuabile tramite:

Alternative testuali
Ogni contenuto non testuale deve avere una alternativa in formato testo, così da essere letto da devices quali quelli per il braille. Assolutamente consigliato permettere di aumentare la dimensione del testo.

Multimedia
Fornire alternative per il contenuto multimediale, come sottotitoli, trascrizione del testo, ecc.

Adattabilità
Il contenuto deve essere rappresentato in modo da essere visibile in più layout senza perdita di informazione.

Distinguibilità
Rendere più semplice la visione/percezione degli elementi fra loro, quindi ad esempio tra foreground e background.

4.2) Operabilità
Le componenti dell'interfaccia e la navigazione devono essere utilizzabili

Accesso via tastiera
È possibile effettuare tutte le operazioni adoperando solamente la tastiera.

Tempo sufficiente
Evitare count-down o alert a tempo: permettere all'utente di leggere nel tempo di cui ha bisogno.

Evitare crisi epilettiche
Evitare quindi cicli di flash, con frequenza maggiore di tre volte al secondo.

Navigabilità
Fornire strumenti per dare un'idea all'utente di dove si trovi, come tornare indietro, come trovare il contenuto desiderato.

4.3) Compensibilità
Il contenuto e l'interfaccia devono essere comprensibili

Leggibilità
Testo comprensibile (font), con l'eventuale aggiunta della selezione della lingua. Fornire spiegazioni per acronimi o termini tecnici.

Prevedibilità
Consistenza nella navigazione e nella identificazione dei componenti.

Assistenza agli utenti
Aiutare gli utenti a non commettere ed eventualmente correggere gli errori.

4.4) Robustezza

Compatibilità
Massimizzare la compatibilità con agenti esterni, passati presenti e futuri.

Questa tabella dell'IBM riassume parte di quanto detto.

Keyboard Access	✓ Provide keyboard equivalents for all actions. ✓ Do not interfere with keyboard accessibility features built into the operating systems.
Object Information	✓ Provide a visual keyboard focus that is programmatically exposed to assistive technology ✓ Provide semantic information about user interface objects including text for images. ✓ Associate labels with controls, objects, icons, and images. Use images consistently. ✓ Electronic forms must be usable with assistive technology.
Sounds & Multimedia	✓ Provide an option to display a visual cue for all audio alerts. ✓ Provide accessible alternatives to significant audio and video. ✓ Provide an option to adjust the volume.
Display	✓ Provide text through standard system function calls or through an API which supports interaction with assistive technology. ✓ Use color as an enhancement, not as the only way to convey information or indicate an action. ✓ Support system settings for high contrast for all user interface controls and client area content. ✓ When color customization is supported, provide a variety of color selections capable of producing a range of contrast levels. ✓ Inherit system settings for font, font size, and color for all user interface controls. ✓ Provide an option to display animation in a non-animated presentation mode.
Timing	✓ Provide an option to adjust the response times on timed instructions or allow the instructions to persist. ✓ Avoid the use of blinking text, objects or other elements.

Yale Web Style Guide

1) Progress

Il processo si distingue in diverse fasi di sviluppo. Nel dettaglio:

Definizione e pianificazione
Viene definito il **team** (in-house, outsourcing), vengono analizzate le **tecnologie** (browsers, caratteristiche più o meno avanzate, scelta del database, scelta del web-server), analisi del **budget**, elezione del **coordinatore**.

Architettura dell'informazione
Viene stilato un inventario dell'informazione che deve essere rappresentato, con specifiche annesse.

Disegno del sito
Vengono tracciati il contenuto e l'organizzazione definitiva dei dati, gli accessi al database, i vari template grafici, l'accessibilità.

Costruzione del sito
Il sito viene effettivamente scritto e reso operativo, si inizia il beta-testing.

Marketing
Si introduce il sito in tutto il materiale stampato relativo all'azienda, i motori di ricerca indicizzano il nostro sito ed è possibile integrare con pubblicità online (legata alle keywords di ricerca, magari).

Tracking, valutazione, manutenzione
Si cerca di capire "come va" il sito tramite i log (accessi in certe fasce orarie, ecc.), questionari sull'usabilità e accessibilità (che portano i problemi citati nei capitoli precedenti), aggiornamento di contenuti e del look periodicamente.

2) Disegno dell'interfaccia

Vediamo i dettagli in merito al disegno dell'interfaccia del sito web.

Composizione della pagina
Ogni pagina deve essere stand-alone poiché può essere intercettata dai browser e quindi l'utente potenzialmente può connettersi direttamente ad ogni pagina del sito. Deve essere poi fornito un titolo chiaro (bookmark), l'autore del testo, la data di creazione, un link all'homepage.

Menù
I menù è meglio organizzarli in poche categorie, eventualmente espandibili. Si possono adottare tab e liste.

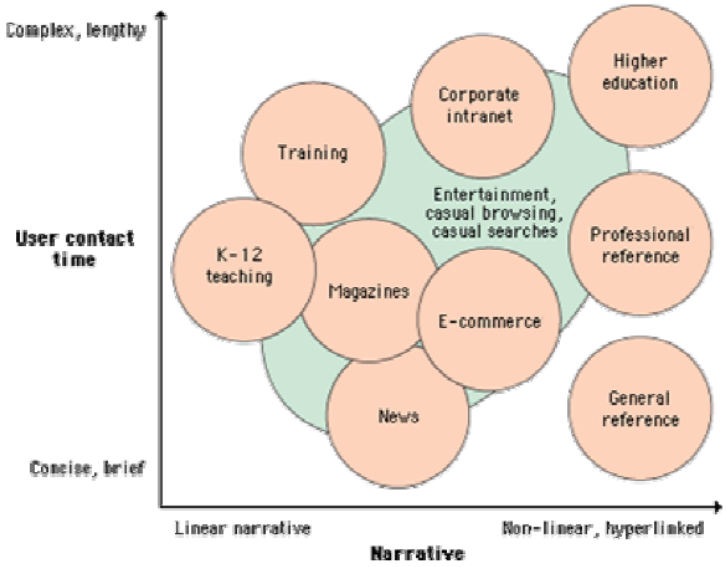
Navigazione
È importante fornire un contesto per far capire all'utente dove siamo. Si adotta la tecnica delle "briciole".

3) Disegno del sito

Organizzazione del contenuto
L'informazione deve essere organizzata in piccole unità logiche, gerarchicamente ordinate e in relazione fra loro.

Organizzazione della forma
Utilizzo di griglie e di templates.

Temi
A seconda del tema che il sito tratta, avremo questo tipo di rapporto contenuto lineare/tempo di accesso dell'utente.



Elementi
Gli elementi la cui introduzione è suggeribile in un sito web sono: home page, menu & sottositi, lista delle risorse del portale, guida al sito, "What's New"?, search (funzione di ricerca), contatti mappe e direzioni per raggiungere l'eventuale sede, feedback, bibliografia e appendici, FAQ, pagine di errore chiare e specifiche.

4) Disegno della pagina

Si adotti un corretto uso degli spazi e delle intestazioni in modo da condurre l'occhio, con paragrafi non troppo larghi e allineamento tendenzialmente a sinistra.

Headers e Footers devono portare informazioni utili (briciole di pane).