Riassunto Polillo - Facile da usare

Progettazione Dell'Interazione Con L'Utente + Laboratorio (Università degli Studi di Bari Aldo Moro)

Riassunto Progettazione dell'interazione con l'utente

INDICE:

- Capitolo 1: Sistemi interattivi e interfacce d'uso

Per sistema interattivo intendiamo qualsiasi combinazione di componenti hardware e software
che ricevono un input da un utente umano e gli forniscono un output, allo scopo di supportare l'attuazione di un compito. Vengono quindi esclusi quei sistemi che interagiscono a loro volta con altri sistemi.
Per interfaccia d'uso intendiamo l'insieme di tutti i componenti di un sistema interattivo software o hardware che forniscono all'utente informazioni e comandi per permettergli di effettuare specifici compiti attraverso il sistema.
Per task infine si intende qualsiasi insieme di attività richieste per raggiungere un risultato.
Le interazioni che avvengono tra l'utenza e i sistemi interattivi sono definiti secondo l'ISO 9241 dialoghi.

La complessità di un sistema interattivo può dipendere da due fattori chiave: è costituito da molte componenti che interagiscono tra loro, oppure è destinato a dare l'opportunità di svolgere diverse attività. Si parla quindi di complessità interna (strutturale) ed esterna (funzionale). Non necessariamente queste due metriche sono correlate: esistono infatti sistemi funzionalmente complessi ma strutturalmente semplici e viceversa. Ovviamente tali complessità vanno misurate con diverse metriche; ad esempio in un moderno sistema software, la complessità strutturale può essere misurata contando le linee del codice

sorgente mentre quella funzionale attraverso il concetto di punto funzione, ossia l'insieme delle funzionalità presentate all'utente a prescindere dal linguaggio di programmazione usato basandosi solo sul progetto logico.
Ulteriore grado di complessità di un sistema interattivo è la complessità d'uso, cioè il grado di semplicità con cui l'utente può svolgere i task offerti dal sistema.

Va precisato che la complessità d'uso è strettamente collegata alla diversità dell'utenza: ogni utente tanto quanto differisce da un altro per fattori come lingua, cultura ecc. tanto cambia per il suo modo di rapportarsi al sistema.
I sistemi inoltre tendono ad evolversi, comportando il problema del *iperfunzionalismo*.
Donald Norman presentò il modello evolutivo tipico dei prodotti di alta tecnologia secondo il quale nella prima fase di vita di ogni prodotto le sue prestazioni sono inadeguate rispetto ai bisogni degli utenti. In questa fase il prodotto è ancora immaturo. Se si riesce a mantenerlo sul mercato raggiunge il "punto di pareggio", nel quale le prestazioni eguagliano i bisogni del suo utente tipico, cioè il suo target principale. In seguito il prodotto eccederà nelle prestazioni cercando di soddisfare un bacino di utenza sempre più ampio. Ovviamente nelle fasi finali chi ne risente è l'utente "medio" che deve farsi strada tra funzionalità per lui inutili. Si aggiunge inoltre il rischio di instabilità del prodotto.
Il compito principale dell'interfaccia d'uso è proprio quello di filtrare la complessità derivante dall'utilizzo di un sistema interattivo permettendo un dialogo tra sistema e utente più semplice e veloce.
Negli anni '90 nasce la disciplina denominata *Human-Computer Interaction* (HCI) che si occupa di progettare, valutare e realizzare sistemi interattivi basati su computer destinati all'uso umano e allo studio dei principali fenomeni che li circondano.

NO

- Capitolo 2: Evoluzione dei paradigmi d'interazione

L'evoluzione della tecnologia ha permesso nel corso degli anni di aumentare il numero di paradigmi di interazione che a loro volta hanno favorito un'ulteriore avanzamento delle

nuove tecnologie. Schematizzando i punti salienti di questa evoluzione si possono individuare 5 punti:

- il terminale scrivente (teletype) (Paradigma "scrivi e leggi")
- il terminale video (Paradigma "indica e compila")
- il personal computer (Paradigma "non dirlo, fallo")
- il broswer web (Paradigma "point & click")
- il mobile (Paradigma "alzati e cammina")

- Il teletype era essenzialmente un apparato composto da tastiera e stampante integrata, a foglio continuo. Tipicamente, il calcolatore segnala all'utente il suo stato di attesa comandi; l'utente digita il comando, cui segue la risposta dell'elaboratore e il prompt successivo, il tutto stampato sul rullo di carta. Il dialogo è quindi controllato dalla macchina e l'utente si limita a fornire le risposte richieste. Esempi sono i sistemi esperti e in generale quei sistemi in cui è il calcolatore ad avere competenze el problema.
- Con l'introduzione del terminale video il tabulato viene sostituito dallo schermo video. La tastiera acquisisce nuove funzioni che attivano servizi software presenti sulla macchina, e altro aspetto innovativo è la presenza di un cursore spostabile mediante tasti appositi. Questo permetteva all'utente di indicare una posizione precisa all'interno del video. Si introducono quindi sistemi software che permettono l'interazione tramite menu e form da compilare organizzati talvolta con strutture gerarchiche. Questo nuovo paradigma semplifica notevolmente l'interazione anche se però limita le funzionalità a quelle visibili sullo schermo.
- Le novità introdotte con i personal computer sono potenza di calcolo locale, archiviazione dei dati su dischetti e stampa locale. La vera particolarità però è l'aumento della tipologia di utenza: l'utilizzo viene elargito anche a utenti non informatici con lo sviluppo di nuovi software alla portata praticamente di tutti. Inoltre diventa possibile utilizzare i software contemporaneamente con l'uso delle "finestre".
- Con l'introduzione dei browser web avviene un ulteriore cambiamento: l'interazione avviene attraverso l'esplorazione di documenti ipertestuali svincolando il lettore da un'interazione sequenziale. L'esplorazione può anche essere semanticamente non conforme, effettuata solo sulla base degli interessi. Ovviamente la navigazione doveva avere un certo tipo di controllo per indirizzare l'utente verso i documenti giusti; nascono i primi motori di ricerca che indicizzano le pagine e processano i vari documenti fornendo quelli più adeguati alla ricerca effettuate dall'utente.
- Con il mobile si arriva alle interazioni dei nostri giorni tramite laptop, smartphone ecc.

Una certa importanza va attribuita anche al social computing che permette interazioni complesse tra interlocutori distanti spazialmente e temporalmente.

Negli ultimi anni un ramo importante verso cui procede la rete è quello della "intelligenza ambientale", ossia ambienti con macchine intelligenti interconnesse che possono interagire con ambienti in cui vi è la presenza delle persone. Il paradigma dell'intelligenza ambientale si fonda su 5 tecnologie cardine:

- tecnologia embedded: dispositivi interconnessi e integrati nell'ambiente;
- context aware: i dispositivi possono in grado di percepire informazioni provenienti dall'ambiente in cui si trovano e di interpretarle in base al contesto;
- personalizzate: i dispositivi si possono configurare in base alle necessità dell'utenza;

- adattive: i dispositivi sono in grado di apprendere e di evolversi;
- anticipatorie: i dispositivi possono prevedere i desideri e le necessità dell'utente.

Modello di Normann
Modello approssimativo che può essere applicato a qualsiasi tipo di azione.
1. Formare lo scopo: decidiamo quale scopo vogliamo raggiungere;
2. Formare l'intenzione: decidiamo che cosa intendiamo fare per raggiungere lo scopo prefissato;
3. Specificare un'azione: pianifichiamo nel dettaglio le azioni specifiche da compiere;
4. Eseguire l'azione: eseguiamo effettivamente le azioni pianificate;
5. Percepire lo stato del mondo: osserviamo come sono cambiati il sistema e il mondo circostante dopo le nostre azioni;
6. Interpretare lo stato del mondo: elaboriamo ciò che abbiamo osservato, per dargli un senso;
7. Valutare il risultato: decidiamo se lo scopo iniziale è stato raggiunto;

- Capitolo 3: Usabilità

vedere nell'altro pdf

L'interazione che avviene tra l'utente e un sistema viene concettualizzato per la prima volta da Norman. Suddivise l'operare sugli oggetti in 7 passi principali:

- Formare lo scopo

Modello approssimativo che può essere applicato a qualsiasi tipo di azione.
1. Formare lo scopo: decidiamo quale scopo vogliamo raggiungere;
2. Formare l'intenzione: decidiamo che cosa intendiamo fare per raggiungere lo scopo prefissato;
3. Specificare un'azione: pianifichiamo nel dettaglio le azioni specifiche da compiere;
4. Eseguire l'azione: eseguiamo effettivamente le azioni pianificate;
5. Percepire lo stato del mondo: osserviamo come sono cambiati il sistema e il mondo circostante
dopo le nostre azioni;
6. Interpretare lo stato del mondo: elaboriamo ciò che abbiamo osservato, per dargli un senso;
7. Valutare il risultato: decidiamo se lo scopo iniziale è stato raggiunto;

- Formare l'intenzione
- Specificare un'azione
- Eseguire l'azione
- Percepire lo stato del mondo
- Interpretare lo stato del mondo
- Valutare il risultato

Ovviamente i passi sono approssimativi, il vero scopo di questo modello è quello di evidenziare i momenti in cui possono presentarsi problemi. Infatti è possibile che si incontrino difficoltà nel passare da uno stato all'altro (cioè nell'attraversare i *golfi* che li separano). In particolare nel "golfo dell'esecuzione" (separa le intenzioni dalle azioni che permettono di realizzarle. Si supera identificando le azioni che mi permettono di raggiungere lo scopo) e nel "golfo della valutazione" (legato alla difficoltà dell'utente che deve superare per interpretare lo stato fisico del sistema dopo le azioni effettuate, e successivamente comprendere se ha raggiunto o meno lo scopo prefissato).

Si introduce ora il termine *affordance* che denota la proprietà di un oggetto di influenzare, attraverso la sua apparenza visiva, il modo in cui viene usato. È molto importante come concetto in quanto un oggetto che possiede un buon livello di affordance invita chi lo guarda a utilizzarlo nel modo corretto, cioè nel modo in cui è stato concepito. Gli oggetti ad alta tecnologia sono spesso privi di affordance. Una buona affordance riduce il golfo di esecuzione, mentre per ridurre il golfo della valutazione gli oggetti dovranno fornire un feedback che indichino chiaramente quali modifiche, le azioni, abbiano prodotto sullo stato del sistema.
Il *feedback* deve essere ben comprensibile e specifico e l'utente deve essere in grado di interpretarlo senza fatica. Fattore importante è la sua tempestività in modo che l'utente lo pone in relazione con l'azione a cui si riferisce. Se passa troppo tempo tra l'azione e il feedback è opportuno inserire dei feedback intermedi (discreti o continui) con lo scopo di segnalare all'utente il progredire dello stato del sistema verso lo stato finale desiderato.
In conclusione, compito principe del progettista è quello di progettare oggetti con buona affordance (per ridurre ampiezza del golfo di esecuzione) e con buon feedback (per ridurre ampiezza del golfo della valutazione).

Norman però non definisce formalmente l'usabilità e ciò che serve è proprio un modo per quantificarla. Nello standard ISO 9241 l'usabilità è definita come: *L'usabilità di un prodotto è il grado con cui esso può essere usato da specificati utenti per raggiungere specifici obiettivi con efficacia, efficienza e soddisfazione in uno specificato contesto.*
Questa definizione offre l'opportunità di creare delle metriche, in particolare scompone il concetto di usabilità su 3 assi:

- L'efficacia: è quella che viene definita come l'accuratezza e la completezza con ciò gli utenti raggiungono specifici obiettivi.
- L'efficienza è definita come la quantità di risorse spese in relazione all'accuratezza e alla completezza con cui gli utenti raggiungono gli obiettivi.
- La soddisfazione è definita come la libertà dal disagio e l'attitudine positiva verso l'uso del prodotto.

Questa definizione è ancora rudimentale in quanto non tiene in conto che un utente può subire un'evoluzione di apprendimento nel tempo. Nella progettazione di un sistema, il progettista ha di fronte a sé diverse scelte possibili:

- considerare come principali destinatari del prodotto gli utenti occasionali;
- progettare in primo luogo per gli utenti continuativi, cioè per coloro che lo utilizzeranno in modo frequente e continuativo.

Apprendibilità e memorabilità sono così importanti che diversi autori li considerano talmente influenti da incorporarli nella stessa definizione di usabilità.
Nielsen definisce l'usabilità come la somma dei 5 attributi seguenti:

- Apprendibilità: il sistema dovrebbe essere facile da imparare, in modo che l'utente possa rapidamente iniziare ad ottenere qualche risultato dal sistema;
- Efficienza: il sistema dovrebbe essere efficiente da usare, in modo che, quando l'utente ha imparato a usarlo sia possibile un alto livello di produttività;
- Memorabilità: il sistema dovrebbe essere facile da ricordare, in modo che l'utente occasionale sia in grado di ritornare al sistema dopo un periodo di non-utilizzo, senza dover imparare tutto di nuovo;
- Errori: il sistema dovrebbe rendere difficile sbagliare, in modo che gli utenti facciano pochi errori durante l'uso e in modo che, se ne fanno, possono facilmente recuperare;
- Soddisfazione: il sistema dovrebbe essere piacevole da usare in modo che gli utenti siano soggettivamente soddisfatti dopo l'utilizzo.

Per favorire l'apprendimento all'utenza vengono messi a disposizione diversi *sussidi*. Per esempio i tutorial, gli help online, gli help desk oppure i manuali stessi del prodotto. Questi sussidi hanno lo scopo di assistere l'utente in vari modi: alcuni (starter kit e tutorial) lo accompagnano nell'uso iniziale del sistema, quando non lo conosce ancora, altri (manuali utente) hanno lo scopo di assisterlo durante l'uso successivo.
In sintesi, l'usabilità di un prodotto va valutata considerando il sistema complessivo dei suoi sussidi, che spesso non sono distinguibili dal prodotto stesso. un sistema interattivo ben fatto dovrebbe fornire *al suo interno* tutti gli strumenti per accompagnare l'utente dall'uso iniziale a un uso evoluto. un sistema usabile dovrebbe mettere in grado i suoi utenti di utilizzarlo senza alcun sussidio esterno al sistema stesso. in questo senso va interpretata la frase di Norman: *ho una regola semplice per individuare il cattivo design. tutte le volte che trovo indicazioni su come usare qualcosa, si tratta di un oggetto progettato male.*
ovviamente vanno considerati anche i contesti d'uso e le tipologie di utenti. ciò significa che l'usabilità, così come gli altri parametri di qualità di un software, è relativa.

L'*accessibilità* è un concetto strettamente correlato all'usabilità ed è definita come la capacità dei sistemi informatici, nelle forme e nei limiti consentiti dalle conoscenze tecnologiche, di erogare servizi e fornire informazioni fruibili, senza discriminazioni, anche da parte di coloro che a causa di disabilità necessitano di tecnologie assistive o configurazioni particolari.
Per tecnologie assistive la legge intende: gli strumenti e le soluzioni tecniche, hardware e software, che permettono alla persona disabile (mimmo), superando o riducendo le

condizioni di svantaggio, di accedere alle informazioni e ai servizi erogati dai sistemi informatici.

- Capitolo 5: Progettare per l'utente

Nell'attività di progettazione si parte da un esame della situazione attuale per riconoscerne difetti o limiti per concepire e specificare il risultato finale. Pertanto progettare è diverso da realizzare. Quest'ultima infatti è un'attività concreta: si parte da un progetto e lo si attua concretamente.

La progettazione di sistemi usabili richiede un drastico cambiamento di mentalità rispetto all'approccio di progettazione tradizionale. Nella progettazione tradizionale, l'oggetto principale dell'attenzione è il sistema da progettare (partendo dai requisiti funzionali fino a essere realizzato). In questo approccio il progettista concentra la sua attenzione sulle funzionalità e sugli aspetti tecnici connessi all alla loro realizzazione, per arrivare a soddisfare le specifiche con un rapporto costo/qualità accettabile.

Se l'obiettivo è la progettazione di un sistema usabile, questo approccio non funziona. In questo caso, il progettista dovrà porre la sua attenzione sull'utente e dovrà studiarne le caratteristiche, le abitudini, e le necessità in relazione all'uso del sistema:

1. dovrà preconfigurare i vari contesti in cui il sistema sarà utilizzato, e i suoi diversi casi d'uso;
2. dovrà analizzare in dettaglio i compiti che l'utente svolgerà con il sistema.

Secondo questa impostazione il compito del progettista sarà quello di progettare l'interazione fra il sistema e il suo utente. Si parla così di *interaction design* e di *progettazione centrata sull'essere umano* (human centred design HCD).

La progettazione centrata sull'essere umano è l'oggetto di un altro standard molto importante, l' ISO 13407: *Human-centred design processes for interactive systems.* E' un approccio allo sviluppo dei sistemi interattivi specificamente orientato alla creazione di sistemi usabili. E' un'attività multidisciplinare che incorpora la conoscenza e le tecniche dei fattori umani e dell'ergonomia (per applicarla si deve tener conto della capacità, delle abilità, delle limitazioni e delle necessità umane). Queste tecniche potenziano la progettazione dei sistemi interattivi nell' efficienza, efficacia e migliora le condizioni del lavoro umano. Inoltre contrasta i possibili effetti avversi dell'uso sulla salute, sulla sicurezza e sulle prestazioni. I benefici human-centred posso includere una maggiore produttività, una migliore qualità del lavoro, riduzione dei costi di supporto e di addestramento e una migliore soddisfazione dell'utente.
La progettazione human-centred è un approccio generale, che può essere sviluppato in molti modi, in funzione della natura dei prodotti da realizzare e delle caratteristiche dell'organizzazione che ospita il progetto. L'utilizzo di un approccio human-centred è caratterizzato da:

a. il coinvolgimento attivo degli utenti e una chiara comprensione dei requisiti degli stessi e dei compiti;
b. un'assegnazione appropriata delle funzioni fra utenti e tecnologia;
c. l'iterazione delle soluzioni di progetto;
d. una progettazione multidisciplinare.

L'ultimo punto è il cambiamento più rilevante rispetto a un approccio progettuale tradizionale, poiché richiede un'organizzazione diversa dei gruppi di progetto. Infatti oltre all'ingegnere sono necessarie altre competenze, relative alle scienze dell'uomo e non alla tecnologia. Queste competenze sono:

- l'analisi e la comprensione dei comportamenti e delle loro motivazioni;
- l'analisi e la conoscenza dei processi percettivi e cognitivi coinvolti nell'interazione con i sistemi;
- la comprensione delle problematiche ergonomiche;
- la competenza sulle diverse modalità della comunicazione umana.

Tutto questo deve inevitabilmente far parte dei processi che portano alla progettazione e alla realizzazione di sistemi usabili. Lo HDC produce risultati completamente diversi da quelli ottenuti con l'approccio tradizionale. Le funzioni che interessano agli utenti non vengono fornite dai singoli componenti modulari, ma dalla loro cooperazione. Si dovrebbe partire dall'utente nella progettazione di qualsiasi strumento, semplice o complesso.

Un *caso d'uso*, di grande importanza nella progettazione human-centred, può essere definito come un insieme di interazioni tra l'utente e il sistema, finalizzate a uno scopo utile per l'utente.
Nota molto importante è che non bisogna confondere i casi d'uso con le funzionalità del sistema. Nel primo il soggetto è l'utente (ex. Ascoltare CD, Guardare DVD), nel secondo è una prestazione realizzata dal sistema (ex. Caricamento del CD, Espulsione del CD).
Semplificando al massimo, si può dire che il progettista orientato al sistema si occupa di progettare funzioni, lasciando all'utente il compito di "metterle nella sequenza giusta", per ottenere ciò che gli serve (segue un approccio bottom-up).
Il progettista orientato all'utente desidera conoscere perché l'utilizzatore adopererà il sistema, e vuole permettergli di raggiungere questi obiettivi nel modo più semplice e lineare (segue approccio top-down: non parte dalle funzioni, ma dagli obiettivi).
L'identificazione dei casi d'uso è un'attività fondamentale nella progettazione human-centred: disporre di un elenco ben fatto dei casi d'uso del sistema costituisce un primo passo indispensabile per poterlo progettare. Per eseguirlo correttamente, occorre superare diverse difficoltà:

1. esiste sempre il rischio di scambiare i casi d'uso con le funzionalità e ricadere nelle pratiche della progettazione orientata al sistema;
2. occorre individuare il giusto livello di astrazione;
3. l'elenco dei casi d'uso deve essere il più possibile completo (ciò permette di avere un quadro preciso del rapporto fra utente e sistema e della sua complessità funzionale.

Si vuole poter costruire sistemi che risultino usabili non solo per una specifica persona o gruppo di persone, ma per ogni categoria di utenti. La progettazione di sistemi universalmente usabili è stata chiamata *progettazione universale* o design for all (DfA) che è la progettazione di prodotti e ambienti usabili da tutte le persone senza la necessità di adattamenti o progettazioni speciali. Questa filosofia del design universale non riguarda solo i sistemi interattivi ma può applicarsi a molti ambiti diversi. Un gruppo di architetti, ingegneri, designer industriali, e ricercatori, alla fine degli anni 90, ha elaborato 7 principi generali per orientare i progettisti indipendentemente dal particolare ambito di progettazione:

1. Equità d'uso: prodotto utile e vendibile a persone con abilità diverse;
2. Flessibilità d'uso: prodotto supporta ampio spettro di preferenze e abilità individuali;
3. Uso semplice e intuitivo: l'uso del prodotto facile da comprendere indipendentemente dalle caratteristiche dell'utente;
4. Informazione percepibile: prodotto comunica efficacemente l'informazione necessaria all'utente (indipendentemente dalle condizioni ambientali o abilità sensoriali dell'utente);
5. Tolleranza agli errori: il prodotto minimizza i rischi e conseguenze di azioni avverse e accidentali o non intenzionali;
6. Ridotto sforzo fisico: prodotto può essere usato in modo efficace, confortevole e con sforzo minimo;
7. Dimensione e spazio adatti all'uso e all'approccio: forniti dimensioni e spazi appropriati per avvicinamento, manipolazione e uso indipendentemente dalla corporatura, postura o mobilità dell'utente.

Nel caso dei prodotti interattivi, la progettazione universale rappresenta una sfida difficile per il progettista. Le difficoltà derivano sostanzialmente da 3 ragioni:

1. Diversità delle tecnologie disponibili ai diversi utenti. Gli ecosistemi tecnologici sono in continuo cambiamento;
2. Diversità degli individui. Queste differenze rientrano approssimativamente in 3 grandi categorie: fisiche, cognitive, socio-culturali;
3. Diversità nella capacità d'uso della tecnologia.

Dal punto di vista generale, ci sono 2 strade per produrre un design universale.
La prima rappresenta l'approccio tradizionale: consiste nel definire un utente "medio" del sistema (cioè al quale il prodotto si indirizza prioritariamente), mentre per gli altri utenti si realizzano dei componenti ad hoc (separati dal sistema). Questo approccio però pone diversi problemi:

- il progettista non è in grado di tener conto fin dall'inizio delle necessità degli utenti diversi (adattamenti quindi potranno essere complessi da realizzare);
- altra difficoltà dovuta alla necessità di mantenere la compatibilità fra il sistema e le diverse tecnologie assistive destinate agli utenti speciali.

Per risolvere queste difficoltà si è fatta strada una filosofia di progettazione molto diversa: questa non privilegia l'utente medio, ma tiene in considerazione fin da subito le necessità di tutte le categorie di utenti. E' evidente che i due approcci sono molti diversi. Il primo semplifica l'attività di progettazione; il secondo richiede che tutti i problemi siano affrontati e risolti in modo sistematico fin dall'inizio (strategia più complessa).
Esiste infine, una terza possibilità (ancora più sofisticata). In questo caso il sistema, oltre ad essere personalizzabile in fase di configurazione iniziale, è in grado di monitorare con continuità i comportamenti e le modalità d'uso dell'utente, e di adattarvisi modificando il proprio comportamento. In sostanza, il sistema impara durante l'uso (sistemi si chiamano adattativi).
I tre approcci non sono alternativi, ma complementari. Possono essere in qualche misura compresenti in uno stesso sistema. In definitiva, potremmo definire la progettazione universale come l'approccio secondo il quale i sistemi sono progettati per essere sufficientemente intelligenti da adattarsi alle richieste o alla modalità di utilizzo dei loro diversi utenti o da permettere un facile interfacciamento con adattatori speciali.

Lo human-centred design può essere considerato un approccio maturo, che contiene al suo interno le problematiche tecniche del system-centred design e ci permette di comprendere in modo più approfondito le finalità del sistema. Possiamo classificare le attività di progettazione in differenti *livelli di maturità:*

- Primo livello di maturità: il prodotto funziona. Il progettista si occupa principalmente della risoluzione di problemi di natura tecnologica, e si accontenta che le funzioni previste nel sistema siano operative, e non ci siano errori di funzionamento.
- Secondo livello di maturità: il prodotto fornisce le funzionalità necessarie. Il sistema non soltanto funziona, ma realizza tutte le funzionalità ritenute necessarie per gli scopi per cui è concepito.
- Terzo livello di maturità: il prodotto è facile da usare. Questo è il livello della progettazione human-centred. Non solo il prodotto funziona e offre tutte le

funzionalità richieste, ma le organizza in modo adeguato rispetto alle tecnologie e alla necessità dei suoi utenti, nei diversi contesti d'uso.

- Quarto livello di maturità: il prodotto è invisibile durante l'uso. In questo caso il prodotto funziona, fornisce tutte le funzionalità richieste, è usabile e si integra in modo così armonico e poco intrusivo con i comportamenti del suo utente che questi non si accorge di usarlo.

Le competenze richieste a un *interaction designer* sono molto diverse da quelle richieste a un system designer. Mentre quest'ultimo dovrà possedere essenzialmente competenze di natura tecnologica nel dominio cui appartiene il sistema da progettare e progettuale, il primo dovrà essere in grado di analizzare e comprendere le caratteristiche e i bisogni dell'utente per definire, a partire da queste, le modalità d'interazione più opportune.

- Capitolo 6: L'ingegneria dell'usabilità

Il termine *ingegneria* può essere definito in molti modi. Vuole sottolineare l'approccio pragmatico e basato su fondamenti scientifici di questa disciplina, che si propone di dare indicazioni concrete e operative a chi abbia il compito di progettare e sviluppare sistemi interattivi.
L*'ingegneria del software* si occupa dei metodi e delle tecniche per la progettazione e realizzazione di sistemi software di qualità, senza sprechi (si è occupata tradizionalmente di sistemi software molto complessi il cui sviluppo coinvolge decine di persone o più.
Più recentemente è entrato nell'uso il termine *ingegneria dell'usabilità* (usability engineering) per indicare la disciplina che studia le tecniche, i metodi e i processi che possono essere utilizzati per progettare e sviluppare sistemi usabili. E' un processo che si basa

sull'ingegneria classica e consiste nello specificare quantitativamente e in anticipo quali caratteristiche e in qual misura il prodotto finale da ingegnerizzare dovrà possedere.
Inizialmente, l'ingegneria dell'usabilità si è focalizzata sul design dell'interfaccia utente dei sistemi software. Oggi, comprende la totalità delle pratiche utilizzate nel processo di progettazione e sviluppo dei sistemi interattivi, a partire dalla raccolta e analisi iniziale dei requisiti. I principi cardine di questa disciplina possono considerarsi ben consolidati e si possono riassumere in 3 punti chiave:

1. Focalizzazione sull'utente, all'inizio e durante tutto il processo di progettazione;
2. Prove con l'utente durante l'intero processo di progettazione, con analisi qualitative e misure quantitative;
3. Modello di progettazione e sviluppo iterativo, per prototipi successivi.

Quando la disciplina dell'ingegneria del software era agli esordi si pensava che per realizzare un progetto di successo fosse necessario procedere per fasi logiche ben sequenziate, ognuna delle quali ponesse le basi per la fase successiva. In poche parole, dalla fase iniziale di un progetto si arriva, passo passo, al rilascio del sistema senza tornare mai sui passi precedenti (*Modello a cascata,* waterfall model).
Per costruire qualcosa bisogna prima decidere che cosa si vuole ottenere e descriverlo dettagliatamente; poi si passerà alla sua realizzazione, quindi al collaudo finale e alla consegna al richiedente. Ci si accorse però che non sempre funzionava così: nella pratica, in nessun progetto reale le cose procedevano in maniera così semplice e lineare. Bisognava ritornare sui passi precedenti, per rivedere e modificare decisioni già prese, anche se erano ritenute assolutamente consolidate. Le cause potevano essere molteplici:

- il committente richiedeva delle varianti che modificavano le specifiche già approvate;
- i progettisti scoprivano difficoltà tecniche inattese;
- nella fase di rilascio del sistema, i primi utenti segnalavano delle difficoltà nell'uso che non erano state previste e richiedevano cambiamenti consistenti.

Quindi ci si rese conto che nessun sistema complesso può essere realizzato con il modello della cascata, perché impossibile specificarne tutti gli aspetti all'inizio e poi realizzarlo senza modificarne nulla. Ragioni di questa impossibilità sono sia di carattere pratico, sia teorico-concettuale.
Pratico perchè molto difficile prevedere "sulla carta" tutti gli aspetti di un sistema complesso. Teorico-concettuale perché per soddisfare le nostre necessità produciamo strumenti, che a loro volta, generano nuovi bisogni. Allora si costruiscono nuovi strumenti o si modificano quelli già disponibili in un ciclo evolutivo infinito, al quale è stato dato il nome di *task-artifact cycle (*ciclo compito-artefatto). In sostanza, non è possibile valutare completamente l'adeguatezza dello strumento ai suoi utenti, prima che questi lo usino effettivamente.

Se il modello a cascata è inadeguato si ha bisogno di un modello che coinvolga gli utenti fin da subito per sperimentare l'uso di versioni preliminari del sistema e aiutarci, con le loro reazioni e le loro indicazioni in un processo di prove e aggiustamenti successivi. Quindi ecco come il *modello iterativo* prende il sopravvento. L'idea è di procedere con la realizzazione di una serie di *prototipi,* via via più vicini al sistema finale. Si inizia con un prototipo preliminare, a costi ridotti e lo si sottopone all'utente che prova a utilizzarlo (prima prova limitata perché il sistema sarà molto semplificato ma sufficiente a verificare alcune

assunzioni di partenza ed eventualmente aggiustare il tiro. Si realizza poi un altro prototipo, sempre incompleto, ma più vicino alla versione finale e lo si sottopone ancora alla prova degli utenti e così via per approssimazioni successive fino alla conclusione del progetto. Si nota quindi che le prove d'uso diventano parte integrante del processo di progettazione. Ovviamente, nelle varie iterazioni le diverse attività avranno pesi diversi. Infatti il primo prototipo sarà piuttosto rudimentale (mock up): effettuare un primo confronto con gli utenti e il committente. Il processo di progettazione per prototipi successivi è il modello concettualmente corretto per la realizzazione di sistemi complessi. A fronte di questi vantaggi esiste il rischio che il processo diverga a causa di richieste di modifiche che nascono durante le attività di valutazione dei vari prototipi. Per evitare ciò, per ogni progetto sarà quindi necessario pianificare il processo iterativo di progettazione e sviluppo in modo che non degeneri.

Il modello iterativo nell'ambito dell'ingegneria dell'usabilità assume una particolare autorevolezza. La descrizione che ne da lo *standard ISO 13407* ha lo scopo di fornire una guida alle attività di progettazione centrata sull'essere umano lungo il ciclo di vita dei sistemi interattivi basati sui computer.

Il contesto in cui il sistema sarà utilizzato è definito dalle caratteristiche degli utenti, dei compiti e dell'ambiente fisico e organizzativo. E' importante identificare il contesto per orientare le decisioni iniziali del progetto e per fornire una base per la loro successiva convalida. Tutto questo sia per un nuovo progetto che per una modifica di un progetto già esistente. Nel secondo caso, possono essere molto utili le reazioni degli utenti provenienti dai rapporti dell' help desk o da specifiche indagini esplorative.
La descrizione del *contesto d'uso* del sistema dovrebbe comprendere i seguenti argomenti:

- le caratteristiche degli utenti;
- i compiti che gli utenti dovranno eseguire;
- l'ambiente nel quale gli utenti utilizzeranno il sistema.

Lo standard ricorda esplicitamente che tutte queste descrizioni non potranno essere congelate in un documento immutabile, ma in un documento di lavoro che sarà corretto, revisionato e ampliato più volte in accordo con la natura iterativa del processo di progettazione e sviluppo.

Nella maggior parte dei processi di progettazione, esiste una consistente attività per la *specifica dei requisiti del prodotto o sistema.*
Nella progettazione human-centred, quest'attività dovrebbe essere ampliata, per descrivere i requisiti in relazione al contesto d'uso più sopra specificato.
Si dovrebbero considerare i seguenti aspetti:

- le prestazioni richieste al nuovo sistema in relazione agli obiettivi operativi ed economici;
- requisiti normativi e legislativi rilevanti, compresi quelli relativi alla sicurezza e alla salute;
- comunicazione e cooperazione fra gli utenti e altri attori coinvolti;
- attività degli utenti;
- ripartizione dei compiti fra esseri umani e sistemi tecnologici;

- prestazioni dei diversi compiti;
- progettazione delle procedure di lavoro e dell'organizzazione;
- gestione del cambiamento;
- fattibilità delle diverse operazioni;
- progettazione dei posti di lavoro e interfaccia uomo-computer.

I requisiti e obiettivi dovrebbero essere stabiliti operando opportuni compromessi fra eventuali requisiti tra loro conflittuali. Organizzati inoltre per livelli di priorità e formulati in modo da permettere la loro successiva convalida mediante opportuni test.

Produrre soluzioni di progetto: Lo standard identifica le seguenti attività:

- utilizzare le conoscenze disponibili per sviluppare proposte di progetto con un approccio multi-disciplinare;
- rendere le soluzioni di progetto più concrete, utilizzando simulazioni, modelli e prototipi di vario tipo;
- presentare le soluzioni di progetto agli utenti, permettendogli di eseguire i compiti che il sistema è destinato a supportare;
- modificare il progetto in conseguenza delle reazioni degli utenti, e ripetere questo processo fino a che gli obiettivi della progettazioni non siano raggiunti;
- gestire l'iterazione delle soluzioni di progetto (registrando i risultati delle attività precedenti).

La *valutazione* è un passo essenziale nella progettazione human-centred e dovrebbe essere compiuta in tutte le fasi del ciclo di vita del sistema. Lo standard identifica le seguenti attività:

- produrre il piano di valutazione. Le tecniche di valutazione variano secondo i casi. La scelta è determinata dalla natura del sistema, dai vincoli economici e di tempo e dalla fase del ciclo di sviluppo in cui si svolge la valutazione.
- fornire feedback per la progettazione. La valutazione dovrebbe essere condotta in ogni fase del ciclo di vita del sistema;
- verificare se gli obiettivi sono stati raggiunti. La valutazione dovrebbe utilizzare metodi appropriati, con un campione rappresentativo di utenti che eseguono compiti realistici;
- valutazione sul campo. Lo scopo sul campo è provare il funzionamento del sistema finale durante l'uso effettivo, per assicurare che esso soddisfi i requisiti degli utenti, dei compiti e dell'ambiente;
- monitoraggio di lungo termine. Consiste nel raccogliere input dagli utenti, con modalità differenti, lungo un certo periodo di tempo;
- documentazione dei risultati. Allo scopo di gestire il processo di progettazione iterativo, i risultati delle valutazioni dovrebbero essere registrati in modo sistematico.

Al modello dell'ISO 13407 sono stati definiti e applicati vari approcci che differiscono fra loro nei dettagli e nel *ruolo che l'utente assume durante il processo della progettazione.*
L'approccio più conosciuto è noto come *progettazione centrata sull'utente* (user-centred design (UCD)): l'utente ha un ruolo fondamentale nell'acquisizione dei requisiti del sistema e nell'effettuazione delle prove d'uso dei diversi prototipi prodotti nelle varie iterazioni del

progetto. Non è coinvolto, invece, nelle attività di progettazione e realizzazione dei vari prototipi che sono condotte dai progettisti e dagli sviluppatori del sistema.
In altri approcci, il coinvolgimento dell'utente avviene con modalità diverse. Nella *progettazione partecipativa* (participatory design) l'utente viene coinvolto anche nelle attività di progettazione, partecipando attivamente alla realizzazione dei prototipi. Nello *usage-centred design* il centro dell'attenzione è sull'uso e non sull'utente, che viene coinvolto nel processo di progettazione in modo molto limitato.

Nel contesto dei processi dell'ingegneria dell'usabilità, alcune professionalità specifiche assumono un ruolo rilevante. Possono venire raccolte sotto la generica etichetta di *usability professional.*

I primi *costi* da affrontare sono quelli relativi alla trasformazione di un progetto tradizionale in uno che utilizza principi dell'ingegneria dell'usabilità. Questo richiede attività di addestramento (per progettisti e responsabili di progetto) e reclutamento di nuove risorse come ad esempio consulenti esperti di usabilità. Inoltre vanno quantificati i costi delle attività esclusive per rendere il progetto usabile, attività che in un progetto tradizionale non verrebbero eseguite. Un'idea secondo Nielsen sarebbe quella di realizzare due prodotti "equivalenti", differenti però solo per l'approccio ingegneristico: uno tradizionale e uno human-centred in modo da valutare i costi aggiuntivi. Ovviamente il tutto è puramente teorico e per le situazioni reali si utilizzano tecniche euristiche.

- Capitolo 7: I requisiti

L'attività di definizione dei requisiti è articolata in tre fasi principali:

- Fase esplorativa: si raccolgono i requisiti attraverso interviste, questionari ecc.
- Fase organizzativa: le osservazioni raccolte vengono organizzate in modo coerente risolvendo eventuali conflitti;
- Fase di revisione: il documento prodotto viene sottoposto a revisione da parte di tutti gli stakeholder (soggetti coinvolti direttamente o indirettamente in un progetto) del sistema e approvato dal committente. (Questo documento mai da considerarsi finale in un modello iterativo).

Nella fase di esplorazione si possono verificare problemi di 3 tipi:

- Problema di dominio. Si rischia di ampliare eccessivamente il campo di esplorazione o viceversa di restringerlo troppo e quindi di non raccogliere la giusta quantità di informazioni necessarie. È importante non cercare di iniziare a progettare il prodotto in questa fase e lasciare ciò alle fasi successive.
- Problemi di comprensione. In primo luogo l'utente ha una comprensione spesso parziale dei propri bisogni e una conoscenza limitata dalle possibilità offerte dalla tecnologia. Dall'altro lato invece chi raccoglie le informazioni utilizza un linguaggio differente sia dagli utenti sia dagli stakeholder: ogni interlocutore tende quindi a tralasciare aspetti per lui ovvi.
- Problemi di conflitto. Stakeholder possono avere punti di vista diversi sul sistema che dovrà essere progettato. I problemi vanno risolti nel documento dei requisiti finale.
- Problemi di volatilità. I requisiti evolvono nel tempo e spesso anche molto rapidamente. Questi cambiamenti possono derivare da diverse condizioni di mercato, ricambio del management, ristrutturazione delle organizzazioni ecc.

Metodologie di raccolta delle informazioni:

- Le *interviste individuali* sono la tecnica principale per analizzare i singoli problemi in profondità. Gli intervistatori formulano le loro domande in colloqui individuali con ciascuno stakeholder raccogliendo le risposte, annotando esigenze, suggerimenti, desideri e lamentele. Interviste sono fatte con cura rispettando le risorse e il tempo disponibili, ma senza tralasciare persone che potrebbero avere qualcosa di importante per la progettazione. Possono essere:
  - non strutturate: sono di carattere esplorativo, assomigliano a delle conversazioni sugli argomenti di interesse;
  - strutturate: prevedono un insieme di domande predefinite, utili soprattutto quando gli obiettivi del colloquio siano stati ben identificati (quindi si possono fare domande precise e ricevere risposte precise).Queste domande poste in forma identica a tutti gli intervistati;
  - semistrutturate: collegano sia domande libere, sia domande specifiche.

  Occorre evitare di influenzare l'intervistato formulando domande che non contengono già la risposta.
- I questionari permettono di raccogliere informazioni in forma strutturata, elaborabili con metodi statistici. Normalmente però il tasso di risposta (*redemption*) è basso e si utilizzano tecniche per elaborare al meglio le informazioni ottenute. La più diffusa è quella di Likert. Il questionario è composto da una serie di affermazioni collegate alle opinioni su cui si vuole indagare per ciascuna delle quali sono possibili 5 risposte: *completamente d'accordo, d'accordo, incerto, in disaccordo, in completo disaccordo.* A ciascuna risposta è associato un numero compreso tra 1 e con i quali sarà possibile effettuare dei calcoli.
- I focus group sono interviste di gruppo, che hanno lo scopo di mettere a fuoco uno specifico argomento e di far emergere i diversi punti di vista dei partecipanti. È necessaria la presenza di un supervisore che garantisca a tutti i partecipanti la possibilità di esprimere il proprio punto di vista e controllando che il gruppo non dilaghi.

- Talvolta è l'utenza stessa che mette a disposizione dei feedback tramite forum ecc. e tali opinioni sono molto importanti per la corretta evoluzione del prodotto.
- Un'altra fase importante è l'analisi dei prodotti concorrenti. La concorrenza può essere più o meno ampia e può essere molto complesso e costoso analizzare ogni punto di forza e debolezza di altri prodotti in modo da caratterizzare il proprio e poterlo lanciare sul mercato. Le prime fonti di analisi sono spesso commenti dell'utenza che confronta i vari prodotti in circolazione esprimendo possibili desideri di miglioramento o esigenze che mancano altrove.

Uno *scenario d'uso* è una narrazione in linguaggio comune, di una possibile storia d'uso del sistema da parte di uno specifico utente, fittizio, ma in qualche modo tipico. Esso serve principalmente come mezzo di comunicazione con i diversi stakeholder e in seguito con i progettisti e gli sviluppatori. È molto importante definire delle identità seppur fittizie dei soggetti degli scenari d'uso; questo perchè in casi pratici ogni vero utente tende a descrivere lo scenario basandosi sulle proprie abilità di utilizzo di uno strumento e quindi del prodotto in questione. Ciò ovviamente risulta sbagliato e andrebbe ad inficiare sull'efficacia dello scenario.
La persona inventata inoltre deve essere creata con criterio: si valutano risultati di ricerche etnografiche, si descrive un profilo in cui si evidenziano competenze, obiettivi e comportamenti

Un *caso d'uso* può essere definito come un insieme di interazioni finalizzate a uno scopo utile tra uno o più utenti e il sistema. Esso ha un nome che è composto da un verbo e da un complemento, oppure da una frase che descrive sinteticamente lo scopo dell'interazione. Viene invocato da un attore per un particolare scopo e si conclude quando si è raggiunto tale scopo. Ogni attore rappresenta un particolare ruolo nell'interazione con il sistema. Ciascuno di questi attori invocherà casi d'uso specifici per il suo ruolo. Se un caso d'uso coinvolge più attori, quello che persegue lo scopo del caso d'uso sarà considerato l'attore principale (solitamente è quello che da inizio al caso d'uso con la sua invocazione).

Nel documento dei requisiti è consigliabile aggiungere all'elenco dei casi d'uso un diagramma riassuntivo detto *diagramma dei casi d'uso* che mostra le relazioni fra gli attori e i casi d'uso del sistema. In questi diagrammi gli attori vengono rappresentati con omini stilizzati e i casi d'uso con ellissi (gli attori coinvolti in un caso d'uso non devono essere necessariamente umani). L'associazione tra attore e caso d'uso è rappresentata da un segmento che li congiunge e può indicare una o più situazioni tra le seguenti:
- l'attore esegue il caso d'uso;
- l'attore fornisce informazioni al caso d'uso;
- l'attore riceve informazioni dal caso d'uso.

Un diagramma che rappresenta tutti i casi d'uso di un sistema si chiama *diagramma di contesto del sistema*, perché indica i confini dello stesso e tutti gli attori che lo utilizzano. In questo caso, il sistema si indica con un riquadro che circonda i casi e ne riporta il nome.

Nel documento dei requisiti a ogni caso d'uso mostrato nel diagramma dovrebbe essere associata una *descrizione* per far comprendere al lettore di che cosa si tratta. Essa dovrebbe essere espressa in un linguaggio semplice e informale, comprensibile a chiunque (infatti il

documento dei requisiti viene usato non solo dai progettisti del sistema ma anche dagli stakeholder).
Anche se non sono state definite delle regole standard, si usa la prassi di descrivere un caso d'uso specificandone i diversi scenari d'uso. Quest'ultimi sono astratti e ridotti ai minimi termini, che servono esclusivamente a chiarire il significato che il redattore del documento dei requisiti attribuisce ai vari casi d'uso. Alcuni di questi scenari permettono di raggiungere lo scopo, altri portano alla conclusione del caso d'uso senza che lo scopo sia raggiunto.
La forma più comune della descrizione di un caso d'uso è che prima si descrive lo *scenario principale di successo* (quello che è ritenuto più frequente o importante e che si conclude con il successo del caso d'uso). Questo descrive il flusso principale degli eventi d'interazione (sequenza di passi numerati) che corrisponde a un'interazione tra uno o più attori e il sistema.
Seguono poi gli altri scenari, *scenari alternativi*. Sono delle estensioni di quello principale (per indicarle si scrive la condizione che determina il verificarsi di una sequenza d'interazioni alternativa rispetto a quella dello scenario principale).
Ogni passo di uno scenario dovrebbe essere espresso con una frase semplice senza indicare i dettagli delle azioni e descrivere i particolari dell'interfaccia utente. Quindi un caso d'uso specifica chi (attori/e), che cosa (interazione) e perché (scopo), senza entrare nel merito del funzionamento interno del sistema.
Se un caso d'uso ne richiama un altro, il nome di quest'ultimo viene sottolineato (collegamento ipertestuale), chiamasi *inclusione.*
L'inclusione può essere utile per esprimere con un singolo passo una sequenza di passi più elementari oppure per raccogliere una sequenza di passi che si ripetono più volte nello stesso o in diversi casi d'uso.
Nella descrizione dei casi d'uso non è consigliabile scendere a un livello di dettaglio troppo basso, ciò che interessa è dare al lettore un'immagine abbastanza chiara.
Non bisogna confondere gli scenari d'uso con i casi d'uso. I primi hanno lo scopo di illustrare situazioni tipiche di uso del sistema, per farne comprendere la portata e far emergere eventuali requisiti impliciti. I casi d'uso, invece, sono collezioni di scenari ridotti ai minimi termini, che hanno lo scopo di fissare gli aspetti principali del flusso dell'interazione con il sistema, sviluppati poi nella fase di progettazione.

Il *documento dei requisiti* è lo stadio finale dell'analisi in cui è possibile presentare agli stakeholder un documento strutturato con richieste ben formate. Si osservano:

- Analisi degli utenti: categorie d'utenza di destinazione, caratteristiche dell'utenza, categorie prioritarie;
- Analisi dei bisogni: necessità per ogni categoria di utenza;
- Analisi del contesto d'uso: contesti d'uso del prodotto, contesti prioritari.

La struttura del documento può essere suddivisa come segue:

1. Sommario
2. Generalità:
   - Scopo del prodotto: specifica sintetica degli obiettivi del prodotto, specificando i primari e i secondari;

- Situazione attuale: specifica se il prodotto è una nuova realizzazione o un miglioramento di uno già esistente;
- Caratteristiche degli utenti: specifica le categorie degli utenti con relative abilità, competenze ed esperienze ecc.
- Contesti d'uso: specifica i contesti e ambienti d'uso. Questi possono includere eventuali standard adottati, il contesto normativo e l'ambiente organizzativo;
- Scenari d'uso: specifica sinteticamente gli scenari d'uso tipici e significativi evidenziando le caratteristiche del prodotto collocati nel loro contesto;
- Fattibilità tecnologica: specifica le tecnologie da utilizzare per realizzare e usare il prodotto.

3. Posizionamento:
   - Analisi della concorrenza: analisi dei prodotti di concorrenza con relativi pregi, difetti, storia e documentazione accessibile;
   - Posizionamento competitivo: specifica eventuali punti di forza e debolezza del prodotto in relazione ai prodotti concorrenti.
4. Casi d'uso:
   - Diagramma dei casi d'uso;
   - Descrizione dei casi d'uso: descrizione in forma verbale di ogni caso d'uso specificandone casi di di successo e scenari alternativi.
5. Altri requisiti:
   - Requisiti per l'esperienza utente: descrive i requisiti relativi all'interfaccia utente, con allegati esempi o figure dove necessario. Specifica inoltre come dovrà essere valutato il soddisfacimento di tali requisiti;
   - Requisiti prestazionali.
6. Allegati:
   - Glossario: eventuali termini tecnici utilizzati nel documento;
   - Altri allegati;
   - Riferimenti.

- Capitolo 8: Ingegneria e creatività

È da premettere in primis che la progettazione non è un algoritmo: a parità di requisiti, due progettisti è assai probabile che concepiranno un prodotto molto diverso; il prodotto finale dipende molto da una profonda conoscenza dell'utenza.
Un buon progettista deve avere la capacità di inventare nuove soluzioni, di rappresentarle utilizzando notazioni rigorose e di valutarne criticamente la validità.
Di seguito si elencano i vari processi cognitivi che un progettista deve tenere in conto durante la progettazione del prodotto:

- *Mimesi:* riproduzione di un prodotto già esistente, con tecnologie diverse, che risolve il problema. Questa è la via più semplice ed è quella che richiede il minor sforzo creativo per il progettista. Il procedimento della mimesi funziona bene quando le azioni che l'utente compie sull'oggetto reale hanno un corrispettivo "naturale" sulla sua rappresentazione virtuale, che ovviamente può essere poi arricchito con ulteriori funzionalità (es. mimesi: calcolatrice trasformata in calcolatrice sw);
- *Ibridazione:* si fondono le caratteristiche funzionali di due o più prodotti già esistenti, creandone uno del tutto nuovo (es. ibridazione: orologio, calendario di windows che rappresenta una fusione software di più oggetti esistenti, oppure l'applicazione djay 3 per Mac);
- *Metafora:* è il processo più complesso che produce però i risultati più interessanti. Si trasferiscono nel proprio progetto soluzioni adottate in diversi domini applicativi (es. metafora: il concetto di "desktop" secondo cui lo schermo del computer È la scrivania dell'utente, aprendo le porte a nuovi concetti tradotti poi in strumenti software);
- *Variazione:* si progetta il sistema prendendo come riferimento un modello noto, introducendo varianti per l'appunto. Il nuovo prodotto potrà essere un sistema concorrente oppure un semplice upgrade di quello esistente (es. variazione: upgrade grafico di un software, oppure prodotti di concorrenza come le suites office);
- *Composizione di design pattern:* si tratta di un riuso di soluzioni di design già sperimentate provenienti da ambienti diversi, estraendo i pattern che più sono interessanti per il progetto. I design pattern hanno molteplici utilità tra cui suggerire ai progettisti meno esperti le migliori pratiche da adottare, raccolgono in forma più organica lo stato della pratica corrente, contribuiscono alla formazione di un linguaggio comune facilitando la comunicazione tra i professionisti, riducono sprechi di tempo e risorse, facilitano l'individuazione di di soluzioni più adatte al contesto (es. design pattern: pattern per le funzioni di ricerca di un sito web).

- Capitolo 9: I prototipi

Secondo lo standard ISO 13407 si definisce prototipo:

una rappresentazione di un prodotto o di un sistema, o di una sua parte, che, anche se in qualche modo limitata, può essere utilizzata a scopo di valutazione.

L'uso di prototipi porta i seguenti benefici:

- rende le decisioni di progetto più esplicite;
- consente ai progettisti di esplorare numerosi design concept prima della scelta finale;
- permette di incorporare nel progetto i feedback degli utenti fin dalle prime fasi del ciclo di progettazione;
- rende possibile valutare numerose varianti del progetto;

- migliora la qualità e la completezza delle specifiche del progetto.

La creazione di un prototipo ha come scopo quello di identificare i problemi più critici, ossia quelli per i quali esistono più soluzioni possibili, fra le quali i pro e i contro si bilanciano, oppure per i quali i rischi derivanti da una cattiva progettazione sono più elevati.
Per le diverse tipologie di prototipi vd. pag. 184 fig 155.

Dal punto di vista del loro scopo si possono classificare i prototipi in 3 grandi categorie:

1. prototipi che servono a valutare il *ruolo* del prodotto nella vita del suo utente;
2. prototipi che servono a valutare l'*interfaccia* del prodotto, intesa come modalità di interazione tra l'utente e il prodotto;
3. prototipi che servono a valutare aspetti tecnici relativi all'*implementazione* del prodotto, per esempio particolare algoritmi utilizzati dal software.

Dal punto di vista della modalità d'uso un prototipo può essere:

1. statico (es. immagini tridimensionali);
2. dinamico (es. video illustrativi);
3. interattivo.

Dal punto di vista della fedeltà i prototipi sono resi il più somiglianti possibili al prodotto finale (*hi-fi prototype*). Ovviamente per non aumentare troppo i costi esistono prototipi meno accurati costruiti con materiali più economici (*lo-fi prototype*).

Dal punto di vista della completezza funzionale un prototipo può essere:

1. orizzontale: fornisce molteplici funzionalità, realizzate però in modo schematico. Ad esempio l'interfaccia finale senza però le sue vere funzioni, in modo da fornire le caratteristiche complete del prodotto.
2. verticale: al contrario, realizza compiutamente un insieme limitato di funzionalità

Dal punto di vista della durata un prototipo può essere:

1. Usa e getta (*throw-away prototype*); prototipi a bassa fedeltà
2. Prototipo evolutivo: viene integrato nel prodotto finale.

Varie tecniche di prototipizzazione sono l'uso di *schizzi, storyboard*. Sono però chiaramente imprecise ed è necessario uno strumento che permetta di rappresentare tutte le possibili sequenze d'interazione: i *diagrammi per macchine a stati* (statechart). Questi permettono di descrivere un sistema in modo gerarchico (per livelli di astrazione successivi). Ciò è molto importante per mantenere entro i limiti accettabili la complessità dei diagrammi. Sono strumenti semplici, ma flessibili e potenti che servono a descrivere il comportamento di sistemi di ogni tipo.
Essenzialmente uno statechart è costituito da nodi e archi:

- ogni nodo rappresenta uno stato del sistema: uno stato del dialogo con l'utente (rappresentato con un rettangolo dai bordi arrotondati contenente il nome dello stato);
- ogni arco rappresenta una transizione da uno stato all'altro (è etichettato con il nome dell'evento/condizione/azione associati).

La transizione è innescata da un evento, normalmente corrisponde ad un'azione dell'utente, e può causare l'esecuzione di un'azione del sistema. Può essere subordinata al verificarsi di una condizione. Con questi semplici diagrammi possiamo descrivere bene interazioni complesse che permettono di mettere in evidenza aspetti critici nella realizzazione del caso d'uso, che possono avere conseguenze importanti sull'usabilità.
Un grosso vantaggio degli statechart è che permettono di rappresentare il sistema per livelli di astrazione successivi. Inoltre sono utili per definire percorsi che corrispondono a situazioni d'errore (è molto meglio analizzare e risolvere subito tutti i problemi, per evitare che emergano nelle fasi successive).

I *prototipi iniziali* hanno lo scopo, nelle prime fasi del progetto, di esplorare più di una soluzione prima di scegliere quella che sarà sviluppata nei dettagli. E' opportuno che siano realizzabili molto velocemente e a costi molto contenuti. Le tecniche possibili sono varie:

- *prototipi di carta*. Permettono di provare l'interazione con l'utente. L'interfaccia del sistema viene disegnata a bassa fedeltà su fogli di carta che vengono usati per effettuare una simulazione manuale del sistema con utenti-cavia;
- *prototipi wireframe*. Prendono il nome dai modelli wireframe della grafica computerizzata. Sono prototipi interattivi a bassa fedeltà nei quali la grafica è estremamente semplificata e mostra solo i contorni degli oggetti. Permettono di sperimentare le modalità principali di interazione, prima che i dettagli della grafica siano definiti;
- *prototipi ipertestuali*.Il prototipo è costituito da una serie d'immagini (snapshot) che rappresentano l'aspetto del prodotto in corso di progettazione. Le varie snapshot sono legate fra loro da link ipertestuali, cliccandoli l'utente passa da una snapshot all'altra, simulando così l'interazione con il prodotto.

I prototipi iniziali, sono spesso usa e getta: si costruiscono con le tecnologie più semplici, allo scopo di avere dei rapidi feedback sulle idee iniziali della progettazione.
Quando il design concept è definito, potrà iniziare la realizzazione effettiva del sistema (attraverso un numero adeguato di iterazioni). Da questo momento in poi si cercherà di sviluppare i prototipi utilizzando le tecnologie finali; in questo modo il sistema evolve per ampliamenti successivi e per modesti rifacimenti a partire da una base di codice iniziale.
Le strategie che guidano il processo dovranno essere definite di volta in volta, a seconda del particolare tipo di sistema.
I *prototipi intermedi* permettono di provare specifici aspetti del prodotto, ma non ancora le sue funzioni complessive, che potranno essere esercitate soltanto alla fine del processo.
Se il processo è stato condotto bene, nelle fasi finali potremo avere una ragionevole certezza che le prove d'uso condotte sui diversi prototipi hanno guidato la progettazione in modo corretto. Quindi le *prove finali* sono molto importanti perché è solo quando il sistema è sostanzialmente finito che si potranno condurre prove complete per i compiti per i quali è stato costruito.

*- Capitolo 10: **Principi e linee guida***

Le indicazioni per il progettista possono essere suddivise in quattro grandi categorie in funzione del loro livello di *generalità* (applicabili in ogni situazione) e *coercitività* (suggerimenti che possono essere seguiti oppure no dal progettista oppure ancora dei vincoli):

- Regole di progetto: sono le regole che *devono* essere applicate nell'ambito di uno specifico progetto. Hanno una bassa generalità ed un alta coercitività, sono imposte dal committente e sono vincolanti per il progettista (regole piuttosto dettagliate). ESEMPIO - Regole che definiscono l'apparenza grafica per l'interfaccia utente.
- Standard: sono norme di tipo generale emesse da organismi internazionali e definiscono le regole da applicare nei progetti di determinate classi di sistemi (vincolanti per tutti i progetti conformi allo standard). ESEMPIO - L'ISO o gli standard emessi dal W3C.

- Linee guida: sono delle raccomandazioni per il design dell'interazione di specifiche classi di sistemi, spesso corredate di esempi e motivazioni e non sono vincolanti per il progettista. ESEMPIO - Linee guida per le interfacce utente di Windows, Apple, GNOME.
- Principi: sono indicazioni generali per la progettazione di interfacce utente usabili, basate su evidenze scientifiche o il generale consenso. Derivano dalla conoscenza degli aspetti fisiologici, psicologici e sociali degli utenti e dall'esperienza accumulata nella pratica della progettazione dei sistemi usabili. Sono indipendenti dalla tecnologia.

L'ente principale responsabile della preparazione degli standard è l'ISO (associazione non governativa di enti nazionali di standardizzazione). I prodotti principali dell'ISO sono documenti chiamati *International Standard (IS).* Lo standard internazionale dovrebbe rappresentare le conoscenze e le pratiche sulle quali si raccoglie il massimo consenso tra gli esperti dei vari paesi. Pertanto, questo viene elaborato in diverse fasi iniziando con numerose bozze sottoposte al commento e all'elaborazione delle varie parti coinvolte fino al Draft International Standard (DIS), che deve essere approvato per votazione da almeno il 75% dei membri.
Il sotto-comitato di maggior interesse è il TC 159/SC 4 che dichiara il seguente scopo: *standardizzazione ergonomica dell'interazione fra i sistemi e le persone che li progettano, fabbricano, usano e mantengono. Le aree di standardizzazione comprendono l'ergonomia dell'hardware, del software e i metodi e processi dello human-centred design.*
I principali standard prodotti sono i seguenti: ISO 13407 (Human-Centred design processes for interactive systems), ISO 9241, ISO 14915 (software for ergonomics for multimedia user interfaces).
Gli standard sono di vario tipo: ce ne sono per processi, prodotti, altri hanno lo scopo di definire la terminologia da usare in un determinato ambito.

Tra i numerosi documenti che compongono l'ISO 9241 vogliamo occuparci di un documento breve, chiamato *Dialogue principles*, che descrive sette principi chiave del dialogo, ovvero sette caratteristiche che ogni dialogo fra un utente e un sistema interattivo dovrebbe avere. Questi sono:

- Adeguatezza al compito: un sistema interattivo è adeguato al compito se supporta l'utente nel completamento del compito, cioè quando le funzionalità del sistema e il dialogo sono basati sulle caratteristiche del compito, piuttosto che sulla tecnologia scelta per eseguirlo.
- Auto-descrizione: un dialogo è auto-descrittivo se agli utenti risulta evidente in ogni momento in che dialogo si trovano, quali azioni possono compiere e come possono essere eseguite.
- Conformità alle aspettative dell'utente: un dialogo è conforme alle aspettative dell'utente se corrisponde alle necessità dell'utente, prevedibili in base al contesto e a convenzioni comunemente accettate.
- Adeguatezza all'apprendimento: un dialogo è adeguato all'apprendimento se supporta e guida l'utente nell'apprendimento del sistema.

- Controllabilità: un dialogo è controllabile se l'utente è in grado di iniziare a tenere sotto controllo la direzione e i tempi dell'interazione fino al raggiungimento dell'obiettivo.
- Tolleranza verso gli errori: un dialogo tollera gli errori se, nonostante evidenti errori negli input, i risultati desiderati possono essere ottenuti senza o con minime azioni correttive. La tolleranza per gli errori si ottiene attraverso il controllo degli errori, la correzione degli stessi e la loro gestione.
- Adeguatezza all'individualizzazione: un dialogo è adeguato all'individualizzazione se l'utente può modificare l'interazione e la presentazione dell'informazione per adattarle alle proprie necessità e capacità individuali.

Questi principi sintetizzano, secondo il punto di vista adottato nello standard, gli elementi chiave che determinano l'usabilità di un sistema interattivo senza precludere però la possibilità di altre formulazioni degli aspetti chiave. Essi non sono indipendenti tra loro e possono a volte esserci delle sovrapposizioni tanto da poter causare dei conflitti, sarà dunque compito del progettista trovare il compromesso più opportuno. In sostanza questi principi definiscono un *modello di qualità* che può essere utilizzato per valutare l'usabilità del sistema assegnando dei voti ad ogni principio, poi rappresentabili in un diagramma a stella. Per ciascuno dei sette principi l'ISO 9241-110 fornisce una lista di raccomandazioni che dovrebbero essere tenute in considerazione senza pretendere di essere esaustive. Nel seguito proponiamo per ciascun principio dei piccoli sunti delle raccomandazioni dello standard.

Il principio di *adeguatezza al compito* afferma che le funzionalità e modalità di interazione del sistema devono essere modellate sulle caratteristiche del compito che l'utente compie con il supporto del sistema e non sulla tecnologia o caratteristiche del sistema stesso. Cioè il dialogo dovrà essere progettato a partire dai casi d'uso identificati nell'analisi dei requisiti (ogni caso d'uso dovrà permettere all'utente di svolgere i compiti e le azioni necessarie nella sequenza più naturale per lui e non per il software che gestisce l'interazione).
Alcune linee guida coerenti con questo principio generale sono le seguenti:

- *dialogo adeguato al compito.* Inclusi tutti i passi necessari ed evitati i passi non necessari*;*
- *informazione adeguata al compito.* Il sistema deve presentare all'utente tutte le informazioni utili per lo svolgimento del compito.
- *dialogo essenziale.* Il sistema dovrebbe evitare di presentare all'utente informazioni ridondanti (ogni duplicazione genera dubbi nell'utente e ne distoglie l'attenzione dal compito principale);
- *dispositivi di input e output adeguati al compito.* La tecnologia mette a disposizione molti dispositivi di input e output per realizzare il dialogo con l'utente (quelli utilizzati scelti in funzione del compito specifico e non viceversa). A volte il compito può richiedere l'utilizzo contemporaneo di più dispositivi (multi-modalità);
- *formati di input e output adeguati al compito;*
- *default tipici.* La corretta impostazione dei valori di default può semplificare notevolmente il dialogo (impostati in modo da riflettere le scelte più comuni). Un'adeguata impostazione dei valori di default è particolarmente importante in quelle

funzionalità che richiedono all'utente di specificare molti parametri (l'utente meno esperto si affida in questi casi ai valori di default quando non conosce il significato dei parametri);
- *compatibilità con i documenti.*

Il principio di auto-descrizione richiede che il sistema comunichi all'utente, in ogni momento, che cosa egli possa fare, come può farlo e cosa sta accadendo. Nella maggior parte dei casi si tratta di una comunicazione visiva, data da informazioni grafiche e testuali. Di seguito alcune linee guida da seguire per la progettazione di un dialogo autodescrittivo:

- Guida per l'utente: ogni passo del dialogo dovrebbe fornire all'utente ogni informazione necessaria per proseguire le sue attività. Le informazioni possono essere sostanzialmente di tre tipi: operative ("che cosa puoi fare ora"), sullo stato del sistema ("ora sto facendo questo") e informazioni feedback ("ciò che hai fatto ha avuto questo effetto"). Ovviamente il dialogo dovrebbe essere il più flessibile possibile in modo da adattarsi alle esigenze dell'utente;
- Interazione evidente: questa linea guida è strettamente collegata al concetto di affordance. Raccomanda che i dialoghi siano progettati in modo che l'interazione con il sistema risulti evidente;
- Descrizione dell'input atteso: ogni volta che il sistema richiede un input all'utente, dovrebbe fornirgli informazioni adeguate su ciò che si aspetta, e sul suo formato;
- Stato visibile: è una delle raccomandazioni più importanti. L'utente dovrebbe sempre essere informato sullo stato del sistema, sia quando questa è in attesa di input, sia quando ha elaborato l'input immesso. Ciò è fondamentale perchè se l'utente non conosce lo stato del sistema, non può prevedere come risponderà alle sue azioni;
- Formati descritti: il sistema dovrebbe fornire all'utente ogni informazione sui formati e sulle unità di misura utilizzate;
- Manualistica minima: durante l'interazione si dovrebbe minimizzare la necessità di consultare i manuali d'uso o altre informazioni presenti fuori dal sistema.

La *conformità alle aspettative* afferma che il dialogo deve essere conforme a ciò che l'utente si aspetta e alle convenzioni comunemente adottate. (principio molto generale, applicabile a molte situazioni diverse). Le implicazioni sono:

- linguaggio familiare. Il sistema dovrebbe usare un linguaggio che conosce bene (un uso improprio del linguaggio può compromettere gravemente l'usabilità);
- aderenza alle convenzioni. Il dialogo dovrebbe seguire le convenzioni comunemente adottate nello specifico contesto (ex. sito web occidentale, il testo è allineato a sinistra);
- organizzazione abituale. La struttura del dialogo e l'organizzazione dei dati dovrebbero permettere all'utente di effettuare le operazioni secondo la modalità a lui ordinarie;
- dialogo consistente. Tutti i dialoghi realizzati da uno stesso sistema dovrebbero avere aspetto e comportamento consistenti. (ex. i bottoni o voci di menu che servono per attivare le stesse funzioni dovrebbero sempre trovarsi nella stessa posizione);
- feedback conforme alle aspettative. Il feedback deve essere ben comprensibile e specifico in modo tale che l'utente lo interpreta senza fatica;

- tempi di risposta conformi alle aspettative. L'utente si forma delle aspettative sul tempo di esecuzione delle elaborazioni richieste. Se la risposta del sistema ritarda troppo, o se il sistema resta muto, può sorgere il dubbio che si sia bloccato per qualche errore. L'utente in questo caso rinuncia interrompendo l'operazione. E' utile tenere presente che il tempo percepito dall'utente non sempre coincide con il tempo reale;
- messaggi adeguati al contesto. Non tutti i messaggi sono adatti a ogni situazione, anche se il loro contenuto è pertinente;
- messaggi in posizione appropriata. I messaggi di feedback e le spiegazioni fornite all'utente dovrebbero apparire dove si trova il focus dell'attenzione dell'utente, per non interrompere il flusso dell'interazione;
- input in posizione attesa. Il sistema dovrebbe richiedere l'input all'utente nella posizione in cui questi si aspetta di doverlo fornire;
- stile coerente dei messaggi;

L'adeguatezza all'apprendimento si riferisce alla learnability. Ci si aspetta che il dialogo con il sistema indirizzi l'utente all'apprendimento del sistema stesso.
Di seguito alcune linee guida:

- Bassa soglia di apprendimento: ogni sistema dovrebbe essere utilizzabile, sia pure in modo elementare, anche con un livello di apprendimento minimo. L'utente inesperto quindi dovrebbe essere in grado di usare le funzioni base con addestramento minimo, o meglio ancora senza. Si è osservato che l'utenza difficilmente legge i manuali, preferisce piuttosto sperimentare direttamente nuove funzioni esplorando il sistema. Un esempio pratico è la presenza di pannelli che suddividono le varie funzioni del sistema in base al loro grado di complessità di utilizzo, partizionando quindi l'utenza in base alle proprie abilità;
- Aiuto alla familiarizzazione: il sistema dovrebbe aiutare l'utente a prendere familiarità con il dialogo, fornendo tutti gli aiuti necessari (Un esempio sono i tutorial che seguono la registrazione dell'utente al sistema);
- Aiuto online: negli esempi precedenti era il sistema che forniva indicazioni all'utente per il corretto utilizzo del sistema e per far prendere familiarità. L'help online invece consente all'utente di chiedere aiuto là dove lo desidera, tramite i numerosi servizi presenti sul web;
- Feedback intermedi: il dialogo dovrebbe fornire dei feedback sui risultati intermedi e finali di un compito, in modo che l'utente possa imparare dei compiti portati a termine con successo;
- Modello concettuale evidente: il sistema dovrebbe aiutare l'utente a costruirsi un modello concettuale appropriato del sistema, mostrando la sua logica interna. La tecnica più semplice è di fornire un sistema gerarchico delle funzioni o ancora fornire dei tab con all'interno delle icone che le rappresentano;
- Sperimentazione sicura: come già accennato l'utenza preferisce esplorare il sistema e sperimentarlo. In virtù di ciò, è necessario rendere il sistema robusto in modo da prevenire utilizzi non corretti del sistema (es. funzione *undo* che annulla l'ultima azione eseguita);

- Riapprendimento facilitato: è possibile che nel sistema siano presenti funzioni usate raramente come le compilazioni di bilancio aziendali usate una volta all'anno. Il sistema dovrebbe quindi facilitare il riutilizzo di queste funzioni semplificando il più possibile il processo di riapprendimento. Ciò è estremamente importante special modo se le funzioni sono sì di raro utilizzo ma con elevata criticità.

Il *principio di controllabilità* afferma che è l'utente a dover guidare il sistema. Egli infatti dovrebbe poter decidere di sospendere il dialogo quando lo desidera, per riprenderlo successivamente, e di fornire le informazioni richieste dal sistema nell'ordine che più gli è congeniale. Dovrebbe inoltre poter cambiare idea durante l'interazione potendo modificare gli input da lui forniti senza vincolo alcuno. Nelle interfacce semplificate del paradigma "scrivi e leggi" dei primi elaboratori questo principio era spesso violato. Si realizzavano infatti dialoghi in cui il sistema poneva delle domande alle quali l'utente doveva rispondere nell'ordine stabilito, senza deroghe. Oggi questo stile per fortuna è scomparso anche se a volte il progettista può introdurre della rigidità per ridurre la complessità del software da realizzare. Per evitarlo è importante seguire le linee guida seguenti:

- Tempi dell'interazione controllati dall'utente: l'utente dovrebbe *poter impiegare tutto il tempo che desidera per effettuare i vari passaggi del dialogo*, senza che il sistema gli ponga dei vincoli. I time-out andrebbero introdotti solo il caso di effettiva necessità ad esempio per esigenze di sicurezza come per le sessioni di una banca online. Per i messaggi invece è pratica corretta far si che questi restino visibili fin quando l'utente non ne comunichi l'avvenuta lettura attraverso, per esempio, un apposito bottone, in quanto potrebbe essere distratto o impegnato in altre attività in quel momento.
- Proseguimento del dialogo controllato dall'utente: l'utente dovrebbe *poter decidere come proseguire nel dialogo*, senza che il sistema imponga vincoli rigidi, ad esempio permettendo di effettuare uno o più compiti secondari durante l'esecuzione del compito principale (vedasi uno smartphone durante una chiamata).
- Punto di ripartenza controllato dall'utente: se il dialogo è stato interrotto per qualche motivo, l'utente dovrebbe *poter scegliere il punto dal quale riprenderlo*, se ciò è compatibile con il compito. A volte il motivo dell'interruzione del dialogo è dovuto al fatto che l'utente si accorge di dover modificare qualche input da lui fornito. Anche in questo caso il sistema dovrebbe permettergli di ripartire dal punto desiderato senza costringerlo a ripartire da capo.
- Reversibilità delle operazioni: se le operazioni sono reversibili e se il contesto d'uso lo permette, dovrebbe essere sempre possibile *annullare almeno il passo più recente del dialogo*. La disponibilità di una funzione *undo* e *redo* migliora sensibilmente l'usabilità del sistema.
- Modalità di visualizzazione dei dati controllata dall'utente: è utile che l'utente *possa tenere sotto controllo non soltanto la sequenza dei passi del dialogo, ma anche le modalità di visualizzazione dei dati necessari al compito*. È particolarmente importante nel caso in cui questi dati siano numerosi: l'utente dovrebbe essere in grado di controllare quali e quanti dati gli vengono mostrati.
- Dispositivo d'interazione scelto dall'utente: all'utente dovrebbe essere permesso di *scegliere uno qualsiasi dei dispositivi di input o di output disponibili, se compatibili*

con il compito. Per esempio poter effettuare una ricerca sia col tasto cerca che con enter.

- Personalizzazione dei valori di default: l'utente dovrebbe essere in grado di definire nuovi valori di default in accordo alle proprie personali esigenze, se compatibili con il compito.
- Disponibilità dei dati originali: dopo la modifica, i dati originali dovrebbero rimanere disponibili all'utente se necessari per il compito.

Un sistema si dice *tollerante verso gli errori* quando fornisce i risultati desiderati anche in presenza di errori dell'utente, senza (o minime) azioni correttive da parte sua. Ciò è possibile attraverso una giusta prevenzione, un'adeguata segnalazione se gli errori avvengono, suggerimenti di azioni correttive appropriate da intraprendere. Le raccomandazioni sono contenute nella ISO 9241 e sono le seguenti:

- Assistenza all'utente: Il sistema dovrebbe aiutare l'utente a evitare di commettere errori negli input da lui forniti, e a scoprire quelli che comunque vengono commessi;
- Verifica e convalida dei dati: prima di procedere all'elaborazione dell'input, il sistema dovrebbe verificarlo e convalidarlo;
- Prevenzione di azioni non lecite: il sistema dovrebbe evitare che un'azione dell'utente possa causare una caduta o uno stato indefinito del sistema;
- Richieste di conferma: Prima di eseguire azioni che possano produrre conseguenze gravi e irreversibili, il sistema dovrebbe chiedere conferma all'utente;
- Spiegazione dell'errore: il sistema dovrebbe fornire una spiegazione adeguata, indicando la causa e possibili azioni correttive;
- Spiegazioni aggiuntive: informazioni che dovrebbero essere fornite su richiesta dell'utente;
- Assistenza per il recupero;
- Minimo sforzo di correzione: il sistema dovrebbe far si che le azioni correttive siano il più semplici possibili;
- Correzione differibile: il sistema dovrebbe permettere di poter rimandare la correzione dell'errore a un momento successivo a meno che non sia necessario che venga corretto per poter proseguire nel dialogo con il sistema;
- Correzione automatica e modificabile: quando il sistema è in grado di correggere automaticamente un errore commesso dall'utente, dovrebbe avvisarlo della correzione effettuata e permettergli di modificarla.

L'*individualizzazione* può riguardare numerosi aspetti diversi, quali la lingua, le preferenze in relazione ai compiti da effettuare, le modalità di rappresentazione dei dati e così via. Alcune linee guida da tener presente in relazione a questo principio sono:

- Scelta di rappresentazioni alternative: il sistema dovrebbe permettere all'utente di scegliere fra varie forma di rappresentazione, adatte alle diverse necessità individuali (ad esempio sistemi di misura, valuta ecc.). Particolarmente importanti sono le possibilità di rappresentazioni alternative che rendano il sistema accessibile a utenti con disabilità di ogni tipo.

- scelta dei formati dei dati input e output: l'utente dovrebbe poter scegliere le rappresentazioni più appropriate per il formato dei dati elaborati nello specifico contesto applicativo;
- Vocabolario personalizzabile: in molti casi, è utile poter arricchire il vocabolario usato dal sistema, per aggiungere eventuali termini utilizzati nel contesto specifico.
- scelta del livello delle spiegazioni. Il livello di dettaglio e/o la forma delle spiegazioni dovrebbe essere modificabile in funzione del livello di conoscenza dell'utente;
- Personalizzazione dei tempi di risposta. L'utente dovrebbe essere in grado di modificare i tempi di risposta dei dispositivi di input e di output, per adattarli alle proprie personali esigenze;
- Scelta del metodo d'interazione: quando appropriato, l'utente dovrebbe poter scegliere fra diverse tecniche di dialogo o metodi d'interazione.
- Personalizzazione del dialogo: l'utente dovrebbe poter modificare alcune componenti del dialogo, per adattarlo a specifiche necessità nell'effettuazione dei compiti (personalizzazione può essere più o meno spinta).
- Ripristinabilità dei valori precedenti: i sistemi interattivi più evoluti forniscono di solito ampie possibilità di personalizzazione. Questo può creare delle difficoltà all'utente, che potrebbe dimenticare quali personalizzazioni ha attivato. È quindi importante che l'utente possa ripristinare facilmente le impostazioni di default o quelle da lui stesso definite precedentemente.

- Capitolo 11: Progettare per l'errore

In questa sezione verranno elencati i vari tipi di errori che possono essere commessi con le relative prevenzioni da adottare.
Il principio di prevenzione segue la regola secondo la quale *l'utente sbaglia perchè è il sistema che gli consente di sbagliare* e questo è associabile a cattiva progettazione.
Volendo definire il significato di errore si utilizza la definizione di James Reason:
Errore sarà inteso come un termine generico per comprendere tutti i casi in cui una sequenza pianificata di attività fisiche o mentali fallisce il suo scopo, e quando questo fallimento non possa essere attribuito all'intervallo di qualche agente casuale.
Si classificano gli errori in quattro categorie principali:

- Azione intenzionale ma errata (*mistake*): questo tipo di errore si verifica quando l'utente ha agito con intenzione, l'azione si è verificata ma non ha ottenuto lo scopo prefissato;
- Azione non intenzionale (*lapsus*): questo tipo di errore si verifica quando si compie un'azione al posto di un'altra;
- Azione spontanea: in questo caso l'azione è compiuta intenzionalmente, ma senza che l'utente avesse precedentemente l'intenzione di agire. Va precisato che un'azione come questa non necessariamente va classificata come errore ma è tale solo se produce effetti collaterali;
- Azione involontaria: in questo caso l'azione è del tutto non intenzionale.

Prevenire l'errore significa progettare il sistema in modo che la possibilità di errori da parte dei suoi utenti sia minima. Alcune tecniche molto diffuse per prevenire gli errori sono le seguenti:

1. Diversificare le azioni dell'utente;
2. Evitare comportamenti modali;
3. Usare funzioni obbliganti;
4. Imporre input vincolanti;
5. Non sovraccaricare la memoria a breve termine dell'utente;
6. Richiedere conferme;
7. Usare default inoffensivi.

1 - Diversificare le azioni dell'utente: questa tecnica serve a prevenire i lapsus. Si tratta delle azioni che un utente deve eseguire per effettuare compiti diversi siano ben diversificati, in modo da minimizzare la probabilità che l'utente ne esegua inavvertitamente una al posto dell'altra (ex. distanziare fisicamente i pulsanti). A volte questa indicazione è in conflitto con quella di raggruppare tra loro i comandi semanticamente correlati.

2 - Evitare i comportamenti modali: chiamiamo modale un sistema che, a fronte di una stessa azione dell'utente, si comporta diversamente a seconda dello stato in cui si trova *e questo stato non è facilmente riconoscibile dall'utente*. Questo comportamento andrebbe però sempre evitato. Un esempio è quello dell'inserimento di una password. Spesso nei sistemi non è facilmente visibile lo stato del Caps Lock dunque l'utente di fronte ad un rifiuto della password sarà portato a pensare di aver digitato in maniera errata la password provando a ridigitarla più volte.

3 - Usare funzioni obbliganti: Donald Norman definisce *obbligante* una funzione in cui le azioni dell'utente sono vincolate in modo tale che la mancata esecuzione di un passaggio impedisca il successivo. In tal modo egli è obbligato a compiere le azioni nella sequenza corretta. Si tratta di una tecnica molto efficace di prevenzione degli errori. Ad esempio nei sistemi desktop dove è necessario prima selezionare l'oggetto e poi l'azione da compiere per far si che il sistema possa disabilitare tutte quelle funzioni che non ha senso compiere su quel determinato oggetto. Al contrario invece, selezionando prima l'azione e poi il suo argomento, questo non sarebbe possibile.

4 - Imporre input vincolati: questa tecnica, che generalizza quella delle funzioni obbliganti, consiste nel permettere all'utente di fornire solo valori di input corretti.

5 - Non sovraccaricare la memoria a breve termine: Dialoghi che sovraccaricano la memoria a breve termine risultano faticosi per l'utente, aumentando quindi la possibilità che vengano commessi errori. Un esempio di cattiva progettazione sono le chiamate pre-registrate dei call-center che enumerano le possibilità di scelta le quali risultano alcune volte troppo lunghe; l'utente a quel punto è probabile che dimentichi le scelte iniziali e digiti il numero sbagliato.

6 - Richiedere conferme: il sistema dovrebbe sempre avvertire l'utente quando questi richiede l'esecuzione di azioni irreversibili o comunque potenzialmente pericolose, e domandare conferma. Queste richieste di conferma devono essere formulate in modo semplice e non ambiguo.

7 - Usare default inoffensivi: Questa tecnica consiste nell'usare, per quanto possibile, valori di default inoffensivi. Il sistema, in altre parole, non dovrebbe mai intraprendere per default l'azione più pericolosa tra quelle possibili in un determinato contesto. Per esempio, la richiesta di stampa di un documento in Microsoft Word 2007 ha come default l'opzione Pagine da stampare - Tutte. Non viene richiesta alcuna conferma, nemmeno nel caso di documenti molto lunghi.

Anche se il progettista adotta le tecniche di prevenzione più appropriate, resterà sempre la possibilità che l'utente commetta un errore. Pertanto, il sistema deve sempre controllare l'input e dovrà fornire all'utente una spiegazione adeguata che gli permetta di recuperare la situazione in modo rapido. Sono tre le funzioni che un messaggio di errore ben progettato deve svolgere:

- *Allertare,* cioè segnalare che qualcosa non va;
- *Identificare,* cioè indicare che cosa non va, e perchè;
- *Dirigere*, cioè spiegare all'utente i passi che deve compiere per ripristinare una situazione corretta: "ora devi fare questo".

In base al tipo di errore è possibile che esso venga corretto dall'utente, dal sistema o da entrambi in modo cooperativo. A partire dallo stato di errore del sistema, il processo di correzione può avvenire con due strategie diverse:

- Backward recovery: secondo questa strategia, si tratterà di annullare le conseguenze negative dell'errore commesso, e riportare il sistema nello stato iniziale, dal quale l'utente potrà compiere, questa volta in modo corretto l'azione che aveva sbagliato. Il processo si chiama *ripristino*;
- Forward recovery: esistono situazioni in cui la correzione dell'errore avviene con un processo diverso. Si cerca di raggiungere lo stato finale direttamente dallo stato di errore, senza prima tornare indietro. (Può essere effettuato automaticamente dal sistema o richiedere l'intervento dell'utente). Un sistema che attua sistematicamente strategie di forward recovery si dice *error tolerant*.

Il senso di questo capitolo è che, nel dialogo con un sistema interattivo, non esiste una dicotomia netta tra comportamento errato e comportamento corretto. Tutta l'interazione uomo-macchina dovrebbe essere trattata come una procedura cooperativa tra utente e sistema, dove gli equivoci possono nascere da entrambi le parti, e devono essere risolti con chiarezza e serenità. L'errore è parte integrante del comportamento umano, e come tale deve essere previsto e accettato.

- Capitolo 12: Progettare la grafica

Si introducono in questo capitolo le leggi della Gestalt che descrivono le modalità con le quali l'apparato visivo umano segmenta il campo visivo raccogliendo in gruppi gli elementi visivi che lo compongono.
La comunicazione visiva ha un ruolo fondamentale nell'interazione uomo-macchina: la maggior parte delle informazioni trasmesse da un sistema avvengono tramite display video di varia forma e dimensione. L'usabilità di tali strumenti dipende dalla loro interfaccia grafica.
L'interfaccia può essere orientata a diversi obiettivi quali usabilità, comprensibilità delle informazioni, gradevolezza complessiva, originalità e capacità di suscitare emozioni. Tali

obiettivi è importante che vengano prefissati già nei requisiti in quanto indipendenti tra loro e ciascuno richiede approcci diversi. L'importanza di queste caratteristiche deriva dalla complessità con la quale interagiscono nel sistema. Lo stesso Norman sottolinea quanto le emozioni possano influire in un dialogo separatamente rispetto alla buona progettazione forma-funzione (vedi pag. 254-255). Secondo lo standard ISO 9241-12 nel progettare l'informazione visiva, si considerino le seguenti caratteristiche:

- Chiarezza: il contenuto informativo è veicolato velocemente e accuratamente;
- Discriminabilità: l'informazione visualizzata si distingue con chiarezza;
- Concisione: agli utenti viene fornita solo l'informazione necessaria;
- Consistenza: la medesima informazione è presentata al sistema conformemente alle aspettative dell'utente;
- Scopribilità: l'attenzione dell'utente è diretta verso l'informazione necessaria;
- Leggibilità: l'informazione è facile da leggere;
- Comprensibilità: il significato è comprensibile, non ambiguo e riconoscibile.

Un altro aspetto da tenere in conto per un progettista, oltre al funzionamento del sistema cognitivo, sono le consuetudini individuali e cognitive che associano significati e valori diversi alle immagini.

L'idea portante della psicologia della Gestalt è che non è corretto dividere l'esperienza umana nelle sue componenti elementari, da analizzare separatamente, perchè *un insieme è più della somma delle sue parti.* Nel 1923 Max Wertheimer descrive le *leggi dell'organizzazione figurale* e sono le seguenti:

- *Legge della vicinanza:* a parità di tutte le altre condizioni, gli elementi del campo visivo che sono tra loro più vicini tendono a essere raccolti in unità (es. grafico pg. 257);
- *Legge della somiglianza:* a parità di tutte le altre condizioni, gli elementi che sono tra loro simili tendono a essere raccolti in unità (es. grafico pag. 258);
- *Legge della chiusura:* a parità di tutte le altre condizioni, le linee delimitanti una superficie chiusa si percepiscono come unità più facilmente di quelle che non si chiudono (es. grafico pag. 258);
- *Legge della continuità in direzione:* a parità di tutte le altre condizioni, le linee che vanno nella stessa direzione si costituiscono in unità più facilmente di altre (es. grafico pag. 259);
- *Legge della buona forma:* a parità di tutte le altre condizioni, il campo percettivo si segmenta in modo che risultino entità per quanto possibile equilibrarte, armoniche, costituite secondo un medesimo principio in tutte le loro parti (es. grafico pag. 260);
- *Legge dell'esperienza passata:* a parità di tutte le altre condizioni, gli elementi di un campo visivo che danno origine a una figura familiare o dotata di significato tendono a formare un'unità (es. grafico pag. 261).

gli elementi che hanno una buona forma si percepiscono come unità più facilmente rispetto agli altri.

(L'approfondimento delle leggi si ha con altri esempi grafici che comprendono tutto il resto del capitolo).

Un'importante attenzione è da dedicare al percorso visivo che l'utente compie quando esamina una schermata: è diffusa la convinzione per la quale esaminando una schermata, i nostri occhi seguano un percorso regolare, iniziando dalla posizione di home (angolo in alto

a sinistra) e procedendo da sinistra a destra e dall'alto in basso. come quando si legge un testo scritto. Ciò non ha alcun fondamento e la situazione risulta essere molto più complessa. Ci sono dispositivi di eye-tracking che tracciano il percorso effettuato dal nostro sguardo chiamato *scanpath*. Questi dispositivi mostrano che il movimento dei nostri occhi è molto irregolare: lo sguardo si fissa per un certo tempo su un determinato punto, per acquisire l'informazione visiva chiamata *fissazione*, e quindi sposta l'asse visivo su un altro punto, con un movimento rapidissimo chiamato *saccade* durante il quale non viene acquisita alcuna informazione visiva; la saccade ha un movimento di circa 2 gradi che comprende in media 8 caratteri, lo spazio necessario quindi per passare da una parola all'altra. Capita alle volte che su termini difficili le fissazioni siano più lunghe e che certi blocchetti vengano riesaminati se per qualche motivo la lettura non va a buon fine. Circa il 15% del testo complessivo è riutilizzato in queste riletture.
In media vengono eseguite tre-quattro fissazioni al secondo.
Gli studi mostrano che il percorso dello sguardo, anche sulla stessa immagine è molto variabile e dipende non solo dalle caratteristiche dell'immagine stessa, ma anche, e soprattutto, dagli obiettivi di chi guarda (es. grafico pag. 275).
Sommando tra loro gli scanpath percorsi da numerosi utenti, è possibile costruire le cosiddette *heatmap*, che mostrano le aree della pagina sulle quali gli sguardi si sono, in media, maggiormente soffermati. Analizzando un gran numero di pagine, Nielsen e Pierce hanno osservato una configurazione prevalente a forma di F:

- gli utenti inizialmente tendono a esaminare, con movimento orizzontale degli occhi, la parte superiore dell'area dei contenuti: questo rappresenta il tratto orizzontale della F;
- poi, lo sguardo esplora la pagina un pò più sotto, anche qui con una scansione orizzontale, ma più breve: il tratto orizzontale corto della F;
- quindi, la pagina viene esaminata con un movimento verticale, tendenzialmente sulla sinistra: il tratto verticale della F.

Si tratta ovviamente di una configurazione media che può tornare utile per strutturare il layout dei sistemi come ad esempio i siti web.

- *Capitolo 13: Progettare il testo*

In questo capitolo si analizza l'usabilità dei testi utilizzati nei sistemi interattivi. Un testo una possibile interpretazione, un testo risulterebbe efficace quando il lettore è in grado di acquisirne tutti i contenuti ("completezza") in dettaglio ("accuratezza"). L'efficienza invece può essere misurata dal tempo impiegato dal lettore per raggiungere determinati obiettivi di accuratezza e completezza. Ovviamente anche in questo caso la diversificazione del bacino di utenza influisce sul grado di usabilità di un testo. Si passa ad analizzare il testo sotto tre punti chiave:

distinguere

- *Legibility:* è la facilità con cui riusciamo a discriminare le singole lettere che lo compongono. L'analisi considera la struttura tipografica del testo: forma, dimensione,

colore e caratteri, la loro posizione all'interno della pagina. Sotto questo punto di vista, non ci si occupa del suo significato, né tantomeno della facilità con cui il lettore apprende o meno il suo significato, ma solo dalla rappresentazione grafica e della riconoscibilità in rapporto al suo sistema visivo. L'analisi è relativamente semplice in quanto dipende da misure oggettive, i soggetti analizzati devono essere selezionati sulla base di una normale acuità visiva;

- *Readability:* è la leggibilità complessiva. In questo caso ciò che viene considerata è la struttura linguistica: l'ampiezza del lessico utilizzato, la complessità delle strutture sintattiche e semantiche. Analisi di questo tipo sono più complesse in quanto influisce anche la dimestichezza del lettore con il lessico e i costrutti utilizzati nel testo. Il motivo principale di questa differenza è che i processi cognitivi coinvolti sono ancora poco noti;
- *Struttura para-testuale:* indica qualunque elemento che è di "ausilio" al testo. Questo tipo di analisi si concentra su elementi come l'organizzazione del testo in capitoli e di questi in sezioni, l'esistenza e la forma dei titoli, riassunti, tabelle, schemi, figure, decorazioni, prefazioni ecc. La particolarità di questi elementi è che oltre a orientare la fruizione del testo, la rendono possibile.

Un *tipo di carattere* o *font* è un insieme di caratteri con un certo stile grafico. Esso contiene caratteri *alfabetici*, in versione maiuscola e minuscola, *cifre e caratteri speciali*. Le minuscole hanno altezze diverse, per la presenza di *ascendenti* (esempi sono le lettere b, d, t) e *discendenti* (esempi sono le lettere g, p, q, y).
I caratteri appartenenti a un certo font possono essere rappresentati con glifi diversi, segno secondo le seguenti proprietà:

- *Stile (font-style): normale (normal), corsivo (italic), obliquo (oblique)*. Di solito, lo stile obliquo è ottenuto con algoritmi che trasformano i glifi dello stile normale inclinandoli verso destra, mentre lo stile corsivo utilizza glifi disegnati appositamente;
- *Variante (font-variant): normale (normal), maiuscoletto (small-caps)*. Nel maiuscoletto, le minuscole sono simili alle maiuscole, ma con un pò più piccole e con proporzioni diverse;
- *Peso (font-weight):* tutti i tratti dei glifi possono essere di spessore *normale* o più spessi *(bold)* in diversi gradi;
- *Dimensione (font-size):* è uguale alla distanza verticale fra il margine superiore dell'ascendente più alto e il margine inferiore del discendente più basso. Si possono utilizzare diverse unità di misura. La più utilizzata è il *punto tipografico*, indicato con la sigla pt. Vale 1/72 di un pollice, pari a 0,35 mm circa.

Fra le proprietà globali del testo vi sono:

- *Decorazione (text- decoration): sottolineato (underline), cancellato (line-through), lampeggiante (blink);*
- *Spaziatura delle lettere (letter-spacing):* può essere quella di default per il font utilizzato, o una spaziatura aggiuntiva, di specificato valore;
- *Spaziatura delle parole (word-spacing)*: quella di default per il font, o una spaziatura aggiuntiva;

- *Allineamento (text-align): a bandiera sinistra (left), a bandiera destra (right), centrato (center), giustificato (justify);*
- *rientro (text-indent):* il valore di rientro della prima riga di ogni paragrafo;
- *interlinea (line-height):* è la distanza fra le linee di base di due righe.

I font hanno un'ulteriore suddivisione in due categorie principali: *graziati (serif) o senza grazie (sans-serif).* I primi presentano particolari terminazioni dei tratti delle lettere, chiamati appunto grazie. L'uso delle grazie deriva dei caratteri lapidari romani, dove era molto difficile scalpellare nel marmo angoli di novanta gradi necessari a terminare le aste; i secondi non presentano le grazie e sono anche chiamati bastoni.
Un'altra importante distinzione è quella tra *print-font e screen-font*. I primi sono i font tradizionali, disegnati principalmente per la stampa. I secondi sono più recenti e sono essenzialmente disegnati per una resa ottimale per i monitor.

Di seguito i font più importanti e usati:

- *Times New Roman:* è il più graziato sulla carta stampata. Aveva lo scopo di essere ben leggibile anche con caratteri di piccole dimensioni stampate sulla carta di cattiva qualità. È caratterizzato da caratteri alti e stretti con lo scopo di ridurre gli spazi bianchi derivanti dall'allineamento giustificato;
- *Georgia:* screen-font graziato, progettato per schermi anche molto piccoli, molto simile al times new roman con qualche piccola miglioria: le linee che compongono le lettere sono leggermente più spesse e il loro spessore varia meno all'interno di uno stesso carattere. A parità di dimensione del font, le lettere sono un pò più larghe e alte, le grazie anche sono un pò più larghe con tratti meno obliqui;
- *Arial:* è un font senza grazie adatto sia ai monitor sia alla carta stampata;
- *Verdana:* è uno screen-font senza grazie diventato quasi uno standard. Ha i caratteri larghi e ben spaziati, minuscole alte e ben leggibili, ed ha il vantaggio di differenziare bene i caratteri simili;

Un font con glifi di larghezza variabile è detto *proporzionale*, mentra un font con larghezza fissa è detto *non-proporzionale*. Un esempio di font a spaziatura fissa è il *Courier* disegnato principalmente per le macchine da scrivere.

Da molti anni è diffusa la condizione che la lettura a video sia più faticosa e più lenta di quella su carta stampata. Questa condizione deriva da esperimenti condotti negli anni '80, i quali indicavano come la lettura su monitor fosse più lenta del 25% a causa della diversità delle due tecnologie su caratteristiche fisiche, possibilità di regolazione, angolo di lettura ecc. Per esempio, su carta il lettore può seguire il testo con dito e il video ha una risoluzione molto inferiore rispetto a quella della stampa. Successivi studi hanno dimostrato che questa affermazione è piuttosto dubbia: la lettura su video può infatti essere altrettanto veloce e accurata rispetto a quella su carta. Ciò che è cruciale è in realtà la qualità dell'immagine presentata al lettore. Da allora sono stati realizzati font specificatamente progettati per il video (screen-font) e la tecnologia dei monitor ha fatto significativamente passi avanti tanto da rendere poco rilevanti le differenze tra i due tipi di lettura.

Ulteriori precisazioni vanno fatte su altri fattori che prescindono dal font e influenzano l'esperienza di lettura:

- *Maiuscole e minuscole:* i caratteri minuscoli risultano più leggibili grazie ai discendenti e agli ascendenti che favoriscono l'associazione di un pattern ad ogni parola;
- *Corsivo, neretto e sottolineato:* il corsivo va in linea di massima evitato, in quanto può causare aliasing su monitor a bassa risoluzione. Il neretto e il sottolineato possono essere utilizzati invece per richiamare l'attenzione. È consigliabile però limitarne l'uso ai casi di reale necessità. È convenzione diffusa riservare le sottolineature a link testuali;
- *Dimensione dei caratteri:* è uno degli attributi che più influenza la legibility di un testo, è infatti consigliato utilizzare dimensioni non inferiori al 12;
- *Allineamenti:* è opinione corrente che il testo allineato a sinistra migliori la legibility in quanto fornisce un'ancora visiva per i movimenti di ritorni a capo dello sguardo. Tuttavia tale opinione non sembra essere supportata da conferme sperimentali, ma è difficile negare che rispetto all'allineamento a bandiera destra, risulti meno confusa;
- *Tinta:* non vi sono ancora molte conferme scientifiche sull'influenza che ha la tinta sulla legibility, però è dimostrato che luminosità e contrasto influenzino significativamente la stessa. Si può però ricordare che caratteri di colori diversi, vengano percepiti su piani diversi, soprattutto quelli con colori tra loro lontani nello spettro (es. rosso e blu), rendendo difficoltosa la lettura. È opportuno ricordare che la percentuale di persone con problemi nella visione del colore è significativa, dunque è importante non associare al colore del testo informazioni che non siano veicolate anche con altri mezzi;
- *Polarità:* i confronti sulla polarità non sembrano portare a risultati coerenti ma alcuni esperimenti suggeriscono che la polarità negativa (caratteri scuri - fondo chiaro) sia preferibile a quella positiva (caratteri chiari - fondo scuro).

leggibile

A parità di tutte le altre condizioni, possiamo dire che un testo è più readable quando è costruito da frasi e parole brevi. È ovviamente una semplificazione drastica ma ci permette di definire degli indici di leggibilità attraverso formule matematiche. L'indice più noto è l'indice Gulpease che a differenza degli altri considera la lunghezza delle parole e delle frasi in caratteri e non in sillabe. Il suo valore è un numero compreso tra 0 e 100 (0 = leggibilità minima, 100 = leggibilità massima) e si calcola mediante la seguente formula:

$$89 + (300 * (numero\ delle\ frasi) - 10 * (numero\ delle\ lettere)) / numero\ delle\ parole$$

Le costanti sono state scelte in modo che un valore minore di 80 venga restituito per un testo difficile da leggere per chi ha licenza elementare (< 60 per licenza media, < 40 per licenza superiore). Va precisato che l'indice di attendibilità è utile solo per valutare la complessità lessicale e sintattica ma non quella semantica. Infatti se prendiamo un testo con un dato indice e ne mescoliamo le parole esso conserverà lo stesso indice pur perdendo di significato. È da considerare per la complessità semantica il vocabolario utilizzato. Infatti, un testo composto da parole di uso frequente è più comprensibile di un testo composto da parole tecniche o insolite.

Le parole del vocabolario possono suddividersi in:

- *Vocabolario comune:* si tratta dell'insieme di vocaboli registrato nei dizionari generici della lingua (60k - 140k circa);
- *Vocabolario di base:* si tratta di quei termini del vocabolario comune certamente conosciuti a chi ha frequentato la scuola di base (7k circa);

- *Vocabolario fondamentale:* sono i vocaboli che chi parla una lingua ed è uscito dall'infanzia, conosce, capisce ed usa (2k circa).

La disponibilità dei vocabolari è importante in quanto ci permette di costruire strumenti informatici che ci aiutano a semplificare un testo segnalando le parole "difficili" e proponendo sinonimi di più larga diffusione. Vi sono dei criteri chiamati per appunto *criteri di scrittura controllata*, rivolti a persone che hanno bisogno di testi informativi molto leggibili. Questi criteri sono: brevità dei testi, semplicità delle frasi, utilizzo del vocabolario di base (le parole che non vi appartengono vengono sempre spiegate).

In commercio esistono diversi manuali di stile che forniscono alcune linee guida per una scrittura di facile comprensione. Le principali linee guida sono 10:

- *Usare parole precise;*
- *Usare parole semplici;*
- *Usare espressioni semplici;*
- *Omettere le parole inutili;*
- *Omettere le precisazioni superflue;*
- *Costruire periodi semplici;*
- *Tenere vicini i termini collegati;*
- *Esprimere le idee analoghe in forma analoga;*
- *Preferire la costruzione positiva a quella negativa;*
- *Usare la forma passiva in forma ponderata.*

Tali linee guida dovrebbero essere ben osservate soprattutto quando lo scopo è quello di comunicare informazioni o istruzioni operative.

Nell'ambito delle pagine Web è comune che l'utente tenda a scorrere la pagine senza leggere tutto l'articolo oppure saltando diverse frasi ritenute di poco interesse. Nielsen introduce il termine *scannable text* per indicare un testo che si può facilmente esaminare in modo rapido e scorrevole. È necessario quindi non operare solo sul testo vero e proprio, ma anche sugli elementi para-testuali. Di seguito alcune linee guida:

- Strutturare il testo in pagine brevi, che preferibilmente non superino le dimensioni di una schermata, per ridurre la necessità di scrolling;
- Fare ampio uso di titoli e sottotitoli brevi e densi di informazione;
- Mettere in evidenza le parole chiave ed i concetti importanti con opportuni artifici tipografici;
- Usare le sottolineature esclusivamente per evidenziare i collegamenti ipertestuali;
- Organizzare i contenuti per livelli successivi di dettaglio, utilizzando le possibilità associative di un ipertesto: titoli brevi, paragrafi brevi, ecc.;
- Inserire rimandi ipertestuali in modo naturale nel testo.

- Capitolo 14: Valutare l'usabilità

Nel modello di progettazione e sviluppo iterativi ad ogni ciclo di iterazione si effettuano dei test di valutazione dell'ultimo prototipo prodotto. All'inizio del progetto, obiettivo principale è raccogliere indicazioni per attività successive. Nelle fasi più avanzate, con la disponibilità di prototipi più completi, sarà possibile, invece, quantificare il livello di raggiungimento degli obiettivi dell'utente e dell'organizzazione.
I termini generici "valutazione" o "test" possono denotare due attività diverse:

- il controllo che il prodotto sia congruente con quanto espresso nei documenti di specifica dei requisiti. Per questo tipo di test si usa il termine *verifica;*
- il controllo che il prodotto soddisfi effettivamente le esigenze per le quali è stato concepito. Per questo tipo di test si usa il termine *convalida;*

La differenza tra verifica e convalida è sostanziale perché non è detto che i documenti di specifica dei requisiti esprimano sempre correttamente le esigenze che il prodotto dovrebbe soddisfare (estensore delle specifiche potrebbe aver male interpretato le richieste degli stakeholder del prodotto).
Le attività di convalida sono molto più difficili delle verifiche. Non si tratta di controllare la congruenza (tracciabilità) tra le caratteristiche del prodotto e le indicazioni contenute nel documento dei requisiti, ma di controllare che il prototipo soddisfi effettivamente le esigenze espresse dal committente. Per questo, la convalida non può essere condotta soltanto dal team di progetto, ma richiede necessariamente il coinvolgimento dell'utente e degli altri stakeholder del prodotto.
Le *attività di valutazione dell'usabilità* rientrano in due grandi categorie:

- Le valutazioni effettuate da parte di esperti di usabilità, senza alcun coinvolgimento dell'utente. (*Ispezioni).* Le più note sono le cosiddette *valutazioni euristiche;*
- Valutazioni effettuate con il coinvolgimento dell'utente. Sono le più importanti e le più utilizzate. (ex. test di usabilità).

L'aggettivo *euristico* si usa per denotare un procedimento non rigoroso che consente di prevedere o rendere plausibile un determinato risultato, che in un secondo tempo dovrà essere controllato e convalidato con metodi rigorosi. Nell'ingegneria dell'usabilità sono euristiche quelle valutazioni effettuate da esperti, analizzando sistematicamente il comportamento di un sistema e verificandone la conformità a specifiche "regole d'oro" derivanti da principi o linee guida generalmente accettati. Le euristiche impiegate sono diverse, in letteratura se ne trovano alcune costituite da centinaia di regole dettagliate. SI preferisce quindi, utilizzare euristiche costituite da pochi principi guida generali. Fra queste quelle di Nielsen, costituite da 10 regole che permettono di inquadrare i problemi rilevanti in categorie ben individuate:

1. *Visibilità dello stato del sistema*: Il sistema dovrebbe sempre informare gli utenti su ciò che sta accadendo mediante feedback in un tempo ragionevole;
2. *Corrispondenza tra il mondo reale e il sistema*: Il sistema dovrebbe parlare il linguaggio dell'utente;
3. *Libertà e controllo da parte degli utenti:* Gli utenti spesso selezionano delle funzioni del sistema per errore e hanno bisogno di una "uscita di emergenza" segnalata con chiarezza per uscire da uno stato non desiderato senza dover passare attraverso un lungo dialogo;
4. *Consistenza e standard:* Seguire le convenzioni della piattaforma di calcolo utilizzata;
5. *Prevenzione degli errori:* Eliminare le situazioni che possono provocare errori da parte dell'utente, e chiedergli conferma prima di eseguire le azioni richieste;
6. *Riconoscere piuttosto che ricordare:* Le istruzioni per l'uso del sistema dovrebbero essere visibili o facilmente recuperabili quando servono;
7. *Flessibilità ed efficienza d'uso:* Permettere all'utente di personalizzare le azioni frequenti;
8. *Design minimalista ed estetico:* Ogni informazione aggiuntiva in un dialogo compete con le unità di informazione rilevanti e diminuisce la loro visibilità relativa;
9. *Aiutare gli utenti a riconoscere gli errori, diagnosticarli, e correggerli:* Indicare il problema con precisione e suggerire una soluzione in modo costruttivo:
10. *Guida e documentazione:* Anche se è preferibile che il sistema sia utilizzabile senza documentazione, può essere necessario fornire aiuto e documentazione.

In sintesi, con la valutazione euristica è possibile ottenere buoni risultati solo impiegando più valutatori sullo stesso progetto, che analizzano separatamente il sistema senza comunicare tra loro. In ogni caso, questa tecnica non garantisce che vengano rilevati tutti i problemi di usabilità, e può capitare che vengano segnalati problemi che in realtà non esistono.
E' anche evidente che i risultati saranno tanto più affidabili quanto più i valutatori saranno esperti nel particolare ambito in esame.

Un *test di usabilità* consiste nel far eseguire a un gruppo di utenti dei compiti tipici di utilizzo del sistema in un ambiente controllato. Si sceglie un campione di utenti che sia rappresentativo della categoria di utenti cui il sistema si rivolge, e si chiede a tutti di svolgere, separatamente, gli stessi compiti. Chi conduce il test (osservatori e facilitatori)

osserva e analizza il loro comportamento per comprendere se, dove e perché essi hanno incontrato delle difficoltà.
Facilitatore gestisce la regia della prova.
Osservatore, assiste al test, annota i comportamenti dell'utente che ritiene significativi. Il loro ruolo è critico: essi dovrebbero conoscere bene il sistema, e avere eseguito personalmente i compiti richiesti agli utenti. Solo in questo modo saranno in grado di interpretarne e valutarne correttamente i comportamenti. Un test di usabilità ha lo scopo di ricavare indicazioni concrete per il miglioramento del sistema. E' molto utile la cosiddetta tecnica del "pensare ad alta voce" (think aloud), che consiste nel chiedere all'utente di esprimere a voce alta ciò che pensa mentre compie le varie operazioni.
Fino a qualche anno fa era diffusa la convinzione che per fare un buon test di usabilità fosse indispensabile usare un laboratorio appositamente attrezzato. Esso è costituito da due stanze contigue: una per l'utente che prova e una per gli osservatori. Nella prima, l'utente esegue il test da solo; nella seconda, il facilitatore e gli osservatori lo possono vedere attraverso un finto specchio. (Laboratori di questo tipo sono abbastanza costosi). Tuttavia, per condurre dei buoni test di usabilità non è indispensabile disporre di una struttura di questo tipo, e ci si può organizzare in modo molto semplice. I test di usabilità possono essere classificati, in funzione dei loro obiettivi, in due grandi categorie:

- *Test formativi:* sono utilizzati durante il ciclo iterativo di progettazione, per sottoporre i vari prototipi a prova d'uso con gli utenti, allo scopo di identificarne i difetti e migliorarne l'usabilità. Il loro scopo è individuare il maggior numero possibile di problemi. Utili nelle fasi iniziali della progettazione perché quando il design concept è appena abbozzato, i test mettono in luce rapidamente i difetti macroscopici, che richiedono una parziale o totale riprogettazione dell'interfaccia.
  Inoltre, si verifica spesso un effetto di mascheramento: ogni problema di usabilità nel quale incappiamo monopolizza la nostra attenzione e ci impedisce spesso di vederne altri, soprattutto se di più lieve entità. Quindi si prova in fretta, si modifica rapidamente il prototipo eliminando i difetti più evidenti, e si prova ancora, e così via. Per il primo test di un prototipo iniziale di carta, 2-3 utenti sono in genere sufficienti.
  Nielsen ha introdotto il termine *discount usability* per indicare queste tecniche, rapide, poco costose e non troppo sistematiche per individuare i problemi di usabilità. Questa semplice indicazione pratica (regola di Nielsen) è stata in seguito criticata fino a portare lo stesso Nielsen a modificarla, portando il numero suggerito di utenti a 5.
- *Test sommativi:* indica una valutazione più complessiva del prodotto, al di fuori del processo di progettazione e sviluppo. Sono test più completi di quelli formativi, che non hanno lo scopo di fornire indicazioni ai progettisti, ma di valutare in modo sistematico pregi e difetti del prodotto, o sue particolari caratteristiche. Sono di solito condotti quando il sistema è completamente funzionante e coinvolge un numero maggiore di utenti (per esempio 10-15 o anche di più). Nella scelta degli utenti è consigliabile utenti con caratteristiche diverse. In tal modo, è probabile che essi affronteranno i compiti assegnati con strategie diverse. In ogni caso, tutti gli utenti dovrebbero avere almeno un potenziale interesse nelle funzioni svolte dal sistema (si rischierebbe di incontrare delle difficoltà che non derivano dal sistema, ma dalla poca dimestichezza che l'utente ha con il problema che gli è stato sottoposto).
  Naturalmente, gli utenti per le prove non dovranno mai essere scelti all'interno del gruppo di progetto. I progettisti conoscono troppo bene il sistema per fornirci delle indicazioni significative.

Rispetto alle attività svolte dagli utenti durante le prove, i test di usabilità possono essere classificati in due grandi categorie:

- *Test di compito:* gli utenti svolgono singoli compiti che permettono di esercitare funzioni specifiche del sistema (test che possono essere svolti anche quando il sistema non è completamente sviluppato). Un errore frequente dei conduttori inesperti è quello di suggerire implicitamente agli utenti le operazioni da seguire, dando loro una serie di istruzioni invece che un problema da risolvere. L'utente, invece, deve essere posto in situazioni simili a quelle in cui si troverà nell'uso reale del sistema, quando dovrà decidere da solo che cosa fare. Se l'utente è troppo guidato, i problemi di usabilità, anche se presenti non emergeranno e il test non sarà di alcuna utilità;
- *Test di scenario:* agli utenti viene indicato un obiettivo da raggiungere attraverso una serie di compiti elementari, senza indicarli esplicitamente. L'utente dovrà quindi impostare una propria strategia di azione. Per un test più realistico potrà essere indicato uno scenario complessivo che definisca meglio il contesto in cui dovrà immaginare di muoversi. I test di scenario possono mettere alla prova l'utente e il sistema in modo molto impegnativo dei test di compito. In particolare, permettono agli utenti di utilizzare il sistema in relazione alle proprie specifiche necessità, preferenze e abitudini. Perciò sono molto utili per individuare eventuali carenze nell'impostazione della struttura complessiva dell'interazione, o mancanze di funzionalità utili. Quindi bisogna cercare di anticipare i test di scenario all'inizio del progetto, usando anche prototipi parziali o a bassa fedeltà. I test di compito permettono, invece, una verifica di usabilità più fine, perché localizzata a specifici casi d'uso. Quindi possono essere più utili quando l'architettura funzionale del sistema sia già ben consolidata, per provare l'usabilità di specifiche funzioni.

E' indispensabile che le istruzioni agli utenti siano date per iscritto, nel modo più chiaro possibile. Solo in questo modo tutti gli utenti partecipanti al test si troveranno nelle medesime condizioni: le spiegazioni date a voce (diverse di volta in volta) potrebbero influenzare i partecipanti in modo differente, rendendo i risultati del test poco confrontabili.

Durante un test di usabilità è utile raccogliere anche delle *misure* oggettive. Quelle più significative sono il tempo impiegato da ogni utente per l'esecuzione di ciascun compito e il tasso di successo (success rate) (cioè la percentuale dei compiti che ciascuno riesce a portare a termine).
Il calcolo del tasso di successo può tener conto anche in parte dei compiti eseguiti.

EX. Considerando 4 utenti che eseguono 6 compiti. Su 24 compiti, 9 sono stati portati a termine, 4 sono stati eseguiti in parte, per i rimanenti gli utenti hanno fallito. In questo caso il tasso di successo potrebbe essere calcolato così:

Tasso di successo = (9 + (4*0.5)) / 24 = 46%

Un test di usabilità viene condotto in quattro fasi successive:

- Pianificazione: L'organizzazione di un test di usabilità dipende in modo sostanziale dagli scopi da raggiungere, che dipendono dalla natura del prodotto e dalla strategia della sua realizzazione;
- *Preparazione del test:* Il team di valutazione deve innanzitutto definire il numero e il profilo degli utenti campione e i compiti che si richiederà loro di svolgere. Da queste decisioni dipenderà in larga misura l'utilità del test. Proseguendo nella preparazione

del test, il team di valutazione deciderà quindi le misure da raccogliere, e predisporrà tutti gli aspetti relativi alla logistica per l'esecuzione delle prove in modo che queste possano avvenire senza troppi disturbi. Preparerà infine i materiali necessari allo svolgimento dei test, e in particolare:

- Un modulo per raccogliere le informazioni sugli utenti;
- Il testo con le istruzioni per lo svolgimento delle prove da consegnare agli utenti;
- La modulistica che gli osservatori utilizzeranno per raccogliere le misure relative all'esecuzione di ciascun compito, e le loro annotazioni durante il test;
- Un questionario per le interviste finali degli utenti.

- *Esecuzione del test:* Se tutto è già ben organizzato e ci si limita a un test con pochi utenti, non dura in genere più di qualche ora complessivamente. Un test più ampio richiederà, al massimo, una o due giornate di lavoro. I test devono essere condotti singolarmente un utente alla volta. E' opportuno prevedere, per ciascuno, un breve periodo di familiarizzazione con il sistema, prima del test vero e proprio. Durante lo svolgimento della prova i valutatori dovranno interferire il meno possibile (solo il facilitatore è autorizzato a parlare con l'utente). Al termine del test di usabilità, è utile intervistare gli utenti sull'esperienza che hanno appena fatto. Intervistatore chiederà a ogni utente quali sono (secondo lui) i punti di forza e di debolezza del sistema, gli aspetti che dovrebbero essere migliorati, e quelli ha gradito maggiormente. Queste possono fornire informazioni ulteriori, ma non devono sostituire le prove d'uso del sistema.
- *Analisi dei risultati e proposte migliorative:* Si analizza il materiale raccolto e si traggono le conclusioni. Ogni gesto, frase, ogni esclamazione dell'utente è un indizio importante, che va considerato e discusso dal team di valutazione, per individuarne cause e implicazioni. Ci sono alcuni errori tipici dei valutatori poco esperti che vanno evitati:
    1. Limitarsi sostanzialmente a riportare i giudizi espressi dagli utenti nelle interviste successive al test. Questi sono utili, ma costituiscono solo una porzione abbastanza marginale dei risultati di un test ben condotto;
    2. Limitarsi all'elencazione di poche difficoltà macroscopiche, senza andare in profondità. Occorre, invece, elencare analiticamente tutti i problemi individuati, grandi e piccoli.

A ciascun problema il team di valutazione assegna un livello di gravità, in base a considerazioni di vario tipo: il numero di volte che tale problema è stato evidenziato nei test, il livello di esperienza degli utenti che hanno sperimentato il problema, l'effetto che il problema ha avuto sul completamento del compito.
L'elenco dei problemi rilevati sarà poi riesaminato per produrre un elenco di interventi proposti per migliorare il prodotto (o prototipo).

L'esito di una valutazione di usabilità dovrebbe essere descritto in modo accurato non solo per test di tipo sommativo ma anche nel caso dei test effettuati durante il processo iterativo di progettazione e sviluppo. Per questo si usa un documento chiamato *rapporto di valutazione* dove sono descritti i risultati dei test effettuati e fornisce evidenza del fatto che essi siano stati condotti con metodi adeguati.
Lo standard ISO 13407 identifica tre tipi fondamentali di rapporti valutazione a seconda dello scopo:

1. fornire feedback per la progettazione;

2. provare la conformità a specifici standard;
3. fornire evidenza del raggiungimento di obiettivi human-centred.

Uno schema generale (più semplice di quello suggerito dall'ISO 13407) per la stesura del rapporto di valutazione può essere:

- Identificazione del documento: Riportare i nomi degli autori, data e versione del documento;
- Sommario: Riportare sintesi dello scopo del documento e delle sue conclusioni;
- Prodotto valutato: Descrivere brevemente prodotto/prototipo sottoposto a test, con ogni informazione che lo identifica con precisione. Indicare le aree funzionali sottoposte al test;
- Obiettivi della valutazione: Descrivere gli obiettivi specifici raggiunti nella valutazione descritta nel documento;
- Metodologia utilizzata: Specificare numero utenti che hanno partecipato al test, loro livello di esperienza e caratteristiche in relazione al prodotto in esame. Specificare i compiti/scenari assegnati, il contesto in cui si è svolto il test e strumentazione utilizzata. Descrivere come è stato condotto il test e da chi, quanto tempo è durato, le misure raccolte, ruolo degli osservatori e come sono stati analizzati i risultati;
- Sintesi delle misure: Fornire una tabella di sintesi delle misure raccolte aggiungendo commenti dove è opportuno;
- Analisi dei risultati: Descrivere analiticamente i problemi incontrati da ciascun utente durante il test allegando ove opportuno screenshot significativi e assegnando ad ogni problema un livello di gravità (il problema sarà numerato per un riferimento più facile). Descrivere in dettaglio (se importanti) reazioni e commenti degli utenti, registrati durante le prove;
- Sintesi delle interviste agli utenti;
- Raccomandazioni: Inserire la descrizione analitica degli interventi migliorativi proposti, raggruppati per livelli di priorità (interventi numerati per una facile tracciabilità);
- Allegati: Allegare i moduli anagrafici compilati dagli utenti, la descrizione dei compiti/scenari data agli utenti prima del test, e tutti i questionari compilati nelle interviste finali. Allegare anche il materiale rilevante prodotto durante il test.

I test di usabilità sono parte necessaria e ineliminabile del processo di progettazione e sviluppo di un sistema interattivo. L'usabilità non è un optional che si possa eliminare per abbassare i costi. Se il prodotto è poco usabile, o non funziona, gli utenti non lo useranno.

- Appendice 1: L'accessibilità

Un sistema si dice accessibile quando si può averne l'accesso (alle sue interfacce d'uso e al suo contenuto) in contesti differenti, da qualunque tipo di utente a prescindere da qualunque hardware, software, lingua, cultura, capacità fisica e mentale. Secondo l'ISO 9241-171 "L'accessibilità è l'usabilità di un prodotto, servizio, ambiente o strumento, per persone con un diverso raggio di capacità". Secondo la "legge Stanca " si garantisce che un sistema di contenga utenza che a causa di disabilità necessita di tecnologie assistive o configurazioni particolari.

La WAI (Web Accessible Initiative) ha proposto un modello costituito su tre componenti:

- *Web Content Accessibility Guidelines (WCAG):* linee guida adottate da molte organizzazioni a livello internazionale che si rivolgono a coloro che progettano i contenuti di siti web accessibili. La prima versione delle WCAG ha una suddivisione in 14 linee guida, ciascuna delle quali comprende numero, obiettivo e un numero di punti di controllo. Ad ogni punto di controllo è assegnata una priorità che va da 1 a 3. Il rispetto delle priorità dei punti di controllo, determina un maggiore o minore livello di conformità allo standard. Il livello di conformità può essere reso pubblico o specificando il titolo delle linee guida con relativo livello raggiunto e ambito di dichiarazione oppure tramite un'icona fornita dal W3C. Una successiva release ha introdotto i criteri di successo che a differenza dei punti di controllo sono verificabili. Per ciascuno di questi sono predisposte delle tecniche di applicazione sufficienti o consigliate;
- *User Agent Accessibility Guidelines (UAAG):* linee guida che si rivolgono agli sviluppatori di browser web, riproduttori multimediali, tecnologie assistive e di altri user agent;
- *Authoring Tools Accessibility Guidelines (ATAG):* linee guida con duplice obiettivo. Assistono gli sviluppatori nella progettazione di strumenti di authoring in grado di generare contenuti accessibili per il web e indirizzarli nella creazione di interfacce accessibili agli utenti.

In Italia la legge stanca viene approvata nel 2003 dal Parlamento garantendo che i sistemi siano sottoposti prima a una verifica tecnica, poi a una verifica soggettiva coinvolgendo

direttamente utenti disabili. Il “bollino blu” è la garanzia di rispondenza ai criteri di accessibilità.
Esempi di soluzioni per utenza disabile è la conversione “equivalente” dell’informazione destinata ad un organo di senso ad un altro (es. sintesi vocale del testo per utenti non vedenti), azionamento diversificato dei dispositivi (es. monitor speciali, emulatori di mouse).
Di seguito le linee guida principali delle WCAG:

- *Alternative testuali (1.1):* Assicurare che tutti i contenuti non testuali siano disponibili anche sotto forma di testo. Il testo ha il vantaggio di poter essere facilmente convertito in formato visuale, audio, tattile ecc;
- *Tipi di media temporizzati (1.2):* garantisce a determinate categorie di utenti di fornire una versione alternativa per tutti i contenuti multimediali sia temporizzati sia sincronizzati
- *Adattabile (1.3):* Assicurare che tutta l’informazione disponibile in un formato possa essere percepita da tutti gli utenti attraverso un formato differente oppure con un layout più semplice;
- *Distinguibile (1.4):* Rendere più semplice agli utenti sia la visione sia l’ascolto del contenuto, separando i contenuti in primo piano dallo sfondo;
- *Accessibile da tastiera (2.1):* garantire che tutte le funzionalità siano gestibili da tastiera, in modo che chi opera da essa o da tecnologie assistive possa interagire con la maggior parte dei contenuti (ex. applicazione che gestisce le immagini. Tale deve permettere tutti i tipi di modifiche);
- *Adeguata disponibilità di tempo (2.2):* Garantire agli utenti la possibilità di leggere ed utilizzare i contenuti senza alcun limite di tempo;
- *Convulsioni (2.3):* Evitare qualsiasi tipo di lampeggiamento che può causare attacchi epilettici;
- *Navigabile (2.4):* Aiutare gli utenti a trovare i contenuti di cui hanno bisogno e consentire loro di tener traccia della loro posizione;
- *Leggibile (3.1):* Far in modo che il contenuto testuale possa essere letto sia da utenti che da tecnologie assistive e garantire la disponibilità delle informazioni necessarie per la sua comprensione;
- *Prevedibile (3.2):* tare gli utenti con disabilità nella navigazione rendendo prevedibile l’ordine in cui vengono presentati i contenuti e il comportamento di componenti funzionali;
- *Assistenza nell’inserimento:* Assistere gli utenti nell’evitare errori, e qualora avvengano, aiutarli a risolverli. I criteri di successo raccomandano di descrivere all’utente in modo esplicito i possibili errori di immissione e fornire adeguate istruzioni per i controlli che richiedono delle azioni da parte dell’utente stesso;
- *Compatibile (4.1):* sostenere la compatibilità di programmi utente attuali e futuri e delle tecnologie assistive;

Anche la validazione dell’accessibilità avviene a livello tecnico, tramite esperti che valuta la conformità agli standard, e a livello reale coinvolgendo l’utenza interessata.

- Appendice 2: Task analysis

La task analysis è una particolare prospettiva basata sul concetto di *compito* (compito che è un obiettivo unito a un insieme ordinato di azioni). Il concetto di compito vede le persone come agenti che interagiscono con una serie di strumenti per raggiungere uno specifico obiettivo in un campo di applicazioni. Persone e strumenti, considerate insieme, costituiscono quello che è definito come un *Work System*. Ogni work system raggiunge i suoi obiettivi definendo un campo di applicazione che consiste in una rappresentazione di un aspetto specifico del mondo reale, al cui interno vi è definito l'obiettivo da raggiungere. L'analisi dei compiti si occupa di comprendere le prestazioni di un sistema di lavoro rispetto a un campo. Il sistema di lavoro nel campo della HCP è costituito da uno o più componenti umane e informatiche e spesso include altri elementi. I compiti sono i mezzi con cui il sistema di lavoro modifica il campo d'azione. Gli obiettivi sono gli stati futuri del campo di applicazione che il sistema di lavoro deve raggiungere mediante i compiti.
Quindi, la task analysis è lo studio di come attraverso i compiti si ottiene questo cambiamento di stato del campo di applicazione.

Un *compito* include spesso dei sotto-tasks con maggior definizione di dettaglio. La struttura di un'attività può includere azioni alternative, ripetizione e la sequenza di alcune azioni. Il compito è descrivibile a livelli crescenti di dettaglio fino a raggiungere il livello delle azioni, definiti come compiti semplici. Un'azione è un compito a cui non è associato alcun processo di soluzione di problemi o strutture di controllo. I metodi possono essere suddivisi tra quelli interessati dalla logica del compito e quelli che si occupano degli aspetti cognitivi. Quest'ultima si occupa di capire quali sono i processi cognitivi che il sistema lavoro deve svolgere per raggiungere un obiettivo. Questo sistema riguarda i processi di pensiero, la soluzione di problemi, l'apprendimento, la memoria e i modelli mentali. Inoltre è necessario studiare la fase di formazione degli obiettivi che riguarda il saper di poter fare una certa cosa.

Ci sono molte prospettive e metodi per realizzare le task analysis e i task design. Di seguito alcune:

- *Obiettivo d'uso della notazione;*

- *Usabilità del metodo a livello di comunicazione;*
- *Usabilità del metodo a livello di modellizzazione;*
- *Adattabilità di una tecnica di task analysis a nuovi tipi di sistema;*

Durante la fase di comprensione, la task analysis si focalizza sulle pratiche di lavoro, considerando la locazione attuale delle funzioni tra le persone e le tecnologie, i problemi esistenti e le opportunità di miglioramento. Durante la progettazione e la valutazione la task analysis si occupa degli sforzi cognitivi richiesti da un particolare design, dalla logica di un possibile design e dalla futura distribuzione dei compiti e delle azioni tra persone e tecnologie. È meglio limitare la task analysis a una o due attività principali in un campo poiché non è nè facile nè economica. Ci sono due tecniche di analisi: la prima si basa sulla task analysis gerarchica e si occupa della logica del compito; la seconda che si fonda su obiettivi, operatori, metodi, regole di selezione, si occupa invece dell'analisi cognitiva dei compiti, concentrandosi sulla conoscenza procedurale necessaria per raggiungere uno scopo.

La task analysis gerarchica è una rappresentazione grafica della struttura di un compito basato su un diagramma strutturale. È altamente iterativa e non raggiunge il suo scopo al primo tentativo. L'analista deve continuamente ritornare all'elenco dei compiti e cercare di ridefinirli in modo da poterli rappresentare gerarchicamente