

# Appunti di Interazione Uomo Macchina

Corso di Laurea Triennale in Informatica

Nicolas Pinto

Anno Accademico: 2022-2023

# Indice

<b>1</b>	<b>Usabilità</b>	<b>3</b>
1.1	Introduzione . . . . .	3
1.2	Un modello dell'interazione . . . . .	3
1.2.1	Affordance e feedback . . . . .	4
1.3	Nozione di Usabilità . . . . .	4
1.3.1	Dalla Usabilità alla User Experience (UX) . . . . .	6
1.4	Human-Centred Design . . . . .	7
1.4.1	Modelli iterativi . . . . .	7
1.4.2	Il modello ISO 9241-210 . . . . .	8
1.5	Valutare l'usabilità . . . . .	10
1.5.1	Valutazioni euristiche . . . . .	10
1.5.2	Test di usabilità . . . . .	11
1.5.3	Test formativi e test sommativi . . . . .	12
1.5.4	Test di compito e test di scenario . . . . .	13
1.5.5	Misure . . . . .	13
1.5.6	Come condurre un test di usabilità . . . . .	14
<b>2</b>	<b>I requisiti</b>	<b>17</b>
2.1	Processo di definizione dei requisiti . . . . .	17
2.1.1	La fase di raccolta . . . . .	18
2.1.2	Importanza dello studio degli utenti . . . . .	19
2.1.3	Interviste . . . . .	19
2.1.4	Focus group . . . . .	21
2.1.5	Gli scenari d'uso . . . . .	22
2.1.6	Questionari . . . . .	23
2.1.7	Altre tecniche di raccolta dei requisiti . . . . .	26
<b>3</b>	<b>Leggi della Gestalt</b>	<b>27</b>
<b>4</b>	<b>WIMP</b>	<b>30</b>
4.1	Interfacce visuali . . . . .	30
4.2	Metafore . . . . .	30
4.3	Interfacce WIMP . . . . .	31
4.4	Metafora della scrivania . . . . .	31

<b>5</b>	<b>Modello di Interazione di Norman</b>	<b>32</b>
5.1	Un modello dell'interazione . . . . .	32
<b>6</b>	<b>Cognitive Walkthrough</b>	<b>34</b>
<b>7</b>	<b>Accessibilità</b>	<b>36</b>
7.1	Definizione di accessibilità . . . . .	36
7.2	Tecnologie assistive . . . . .	37
7.3	Web Content Accessibility Guidelines (WCAG) . . . . .	37
<b>8</b>	<b>Programmazione per il Web</b>	<b>39</b>
8.1	HTML . . . . .	39
8.1.1	Vantaggi dell'HTML . . . . .	39
8.1.2	Semantica dell'HTML . . . . .	40
8.1.3	Contenuti della pagina HTML5 . . . . .	40
8.1.4	Web Form in HTML5 . . . . .	41
8.1.5	Video e audio in HTML5 . . . . .	42
8.1.6	Canvas in HTML5 . . . . .	42
8.1.7	Microdata . . . . .	42
8.1.8	Applicazioni Web offline in HTML5 . . . . .	43
8.1.9	Local storage in HTML5 . . . . .	43
8.1.10	Geolocalizzazione in HTML5 . . . . .	44
8.2	CSS 3 . . . . .	44
8.2.1	I selettori . . . . .	44
8.2.2	Impostazionistilistiche . . . . .	45
8.2.3	Box Model . . . . .	45
8.3	Utili per l'esame . . . . .	46

# Capitolo 1

## Usabilità

### 1.1 Introduzione

Questo corso ha l'obiettivo di trasmettere agli studenti una sensibilità alle problematiche della costruzione di sistemi usabili, in particolare prodotti software,. A questo scopo insiste sui concetti di usabilità, progettazione human-centred, e sui metodi di valutazione dell'usabilità che coinvolgono l'utente.

### 1.2 Un modello dell'interazione

Viviamo quotidianamente le difficoltà nel rapporto con gli oggetti che ci circondano, che percepiamo spesso come complicati da usare. Quali sono le radici di queste difficoltà? Se ne possiamo individuare le cause, possiamo studiare il modo per porne rimedio. È quindi utile analizzare il modo con cui interagiamo con gli oggetti per individuare dove nascono le difficoltà nel loro uso e perché.

Il modello più semplice dell'interazione tra un sistema e il suo utilizzatore è rappresentato dal *ciclo di feedback*. L'utente, per raggiungere il proprio scopo, fornisce un input al sistema e riceve da questo una risposta (feedback) che viene interpretata e confrontata con lo scopo iniziale. Il risultato di questo confronto porta alla successiva azione dell'utente innescando così un nuovo ciclo di stimolo-risposta.

Il sistema può essere di natura qualsiasi, un computer, uno smartphone, un prodotto software o anche un semplice oggetto non intelligente, come un interruttore. Ciò che importa è che ricevendo un certo stimolo, esso reagisca producendo un qualche tipo di risposta.

Spesso la facilità d'uso è determinata, o seriamente compromessa, da elementi di modesta entità. probabilmente del tutto irrilevanti sia dal punto di vista tecnico sia da quello dei costi di produzione.

### 1.2.1 Affordance e feedback

Con il termine *affordance*, si denota la proprietà di un oggetto di influenzare, attraverso la sua apparenza visiva, il modo in cui viene usato. Un oggetto che possiede una buona *affordance* invita chi lo guarda a utilizzarlo nel modo corretto, cioè nel modo per cui è stato concepito. Per esempio, l'aspetto di una maniglia ben progettata dovrebbe far intuire immediatamente come la porta va aperta: se tirandola, spingendola o facendola scorrere.

Gli oggetti, oltre ad avere una buona *affordance*, dovranno fornire un *feedback* facilmente interpretabile, cioè un segnale che indichi chiaramente all'utente quali modifiche le sue azioni abbiano prodotto sullo stato del sistema. Se la distanza temporale fra azione e feedback è significativa, i due eventi possono essere interpretati come tra loro indipendenti: a volte bastano pochi secondi di ritardo per disaccoppiarli, nella percezione dell'utente. In alcuni casi è opportuno inserire dei feedback intermedi che segnalino chiaramente il progredire dello stato del sistema. Questi feedback possono essere discreti o continui.

In conclusione il compito del progettista è quello di progettare oggetti con buona *affordance* e con buon feedback.

## 1.3 Nozione di Usabilità

Alla dizione corrente di “facilità d'uso” si preferisce usare il termine più specifico di *usabilità* (in inglese *usability*), proprio per segnalare che intendiamo riferirci a un concetto definito in modo preciso. Ciò che ci interessa è una definizione operativa che permetta di quantificare l'usabilità dandone una misura oggettiva.

La prima definizione di usabilità che riportiamo è ricca di implicazioni pratiche che ci permettono la sua misurazione ed è proposta dallo standard ISO 9241:

L'usabilità di un prodotto, un servizio o un sistema è il grado con cui può essere usato da specificati utenti per raggiungere specificati obiettivi con efficacia, efficienza e soddisfazione in uno specificato contesto d'uso.

Si tratta di una definizione multidimensionale che scompone l'usabilità su tre assi, relativi a tre variabili sostanzialmente indipendenti: efficacia, efficienza e soddisfazione degli utenti, e il cui valore può essere in qualche modo misurato.

- L'*efficacia* viene definita come la accuratezza e completezza con cui gli utenti raggiungono specificati obiettivi. Essa considera pertanto il livello di precisione con cui l'utente riesce a raggiungere i suoi scopi.
- L'*efficienza* è definita come la quantità di risorse spese in relazione all'accuratezza e alla completezza con cui gli utenti raggiungono gli obiettivi. Tali risorse potranno essere di natura differente secondo le situazioni e potranno essere quantificate. Ad esempio il tempo impiegato per ottenere un determinato risultato o il numero di tasti da premere per realizzare una certa funzione.

- La *soddisfazione* è definita come la libertà dal disagio e l'attitudine positiva verso l'uso del prodotto. La quantificazione della soddisfazione dell'utente sarà effettuata chiedendo agli utenti, per mezzo di opportuni questionari, di attribuire delle valutazioni alle componenti del sistema.

Dalla definizione si evince che l'usabilità non è una grandezza assoluta, ma è sempre relativa al compito da svolgere, all'utente che lo svolge e al contesto d'uso.

Una seconda definizione di usabilità è fornita da Jakob Nielsen e definisce l'usabilità come la somma dei cinque attributi seguenti:

- *Apprendibilità*: il sistema dovrebbe essere facile da imparare, in modo che l'utente possa rapidamente iniziare a 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 altro 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, possano facilmente recuperare. Inoltre, non devono avvenire errori catastrofici.
- *Soddisfazione*: il sistema dovrebbe essere piacevole da usare, in modo che gli utenti siano soggettivamente appagati quando lo usano.

È evidente che sebbene siano due definizioni differenti, la seconda costituisce una sorta di approfondimento della prima. Infatti, la definizione data dall'ISO 9241 è comunque in grado di tenere in considerazione anche gli effetti di apprendibilità e memorabilità insoddisfacenti. Se per esempio un certo utente non fosse in grado di ricordare determinati comandi, in una data situazione d'uso, secondo l'ISO 9241 questo comporterebbe necessariamente una riduzione dell'usabilità.

Infine si riporta una terza definizione proposta dallo standard ISO 9126 secondo la quale l'usabilità è la capacità di un prodotto software di essere compreso, appreso, usato e capace di attrarre l'utente, quando è usato in condizioni specificate. L'usabilità è ulteriormente suddivisa in cinque sotto-caratteristiche:

- *Comprensibilità*, la capacità intrinseca del prodotto software di mostrare agli utenti la sua adattabilità ai vari compiti che devono essere svolti nel contesto d'uso;
- *Apprendibilità*, la capacità intrinseca del prodotto software di aiutare gli utenti ad apprendere facilmente le sue funzionalità;
- *Operabilità*, la capacità intrinseca del prodotto software di rendere possibile agli utenti l'esecuzione e il controllo delle sue funzionalità;

- *Attrattività*, la capacità intrinseca del prodotto software di essere gradevole agli utenti;
- *Conformità*, la capacità intrinseca del prodotto software di aderire a standard, convenzioni e linee guida dell'usabilità.

Tutte le definizioni di usabilità enfatizzano il fatto che l'usabilità sia strettamente dipendente dalla particolare circostanza in cui il prodotto viene utilizzato, ad esempio: le tipologie di utenti, i compiti che devono svolgere, le condizioni fisiche e sociali in cui lavorano. Per tanto, i progettisti devono necessariamente analizzare con cura tali circostanze al fine di sviluppare un prodotto con un ottimo grado di usabilità.

### 1.3.1 Dalla Usabilità alla User Experience (UX)

Negli ultimi anni, il concetto di usabilità si è evoluto, insieme al panorama dell'IT. La disciplina dell'HCI si è sempre più interessata all'esperienza utente (UX), includendo attributi soggettivi come, ad esempio, l'estetica, le emozioni e l'interazione sociale in uno spazio di progettazione che in precedenza si è occupato principalmente di facilità d'uso. Il principio di UX è ben espresso da McCarthy e Wright:

"Oggi non ci limitiamo a usare la tecnologia, la viviamo. Molto più profondamente che mai siamo consapevoli che l'interazione con la tecnologia ci coinvolge emotivamente, intellettualmente e sensualmente. Quindi le persone che progettano, usano e valutano sistemi interattivi devono essere in grado di capire e analizzare l'esperienza emotiva delle persone con la tecnologia".

Fino a poco tempo fa, un obiettivo primario dello sviluppo di prodotti e servizi era quello di fornire funzionalità utili e utilizzabili per consentire alle persone di svolgere i loro compiti. Questi obiettivi sono ancora importanti ma, avendo così tanti beni e servizi ora disponibili, dobbiamo assicurarci che siano anche piacevoli. Il piacere e il divertimento sono componenti importanti della vita: l'apprendimento, l'educazione, il lavoro possono beneficiare del piacere e del divertimento. L'UX è ancora un termine astrattamente definito, che include la soddisfazione e l'acquisizione di sensazioni positive, ma non esiste ancora una definizione universale di UX né una teoria coesiva che possa informare la comunità HCI su come progettare e valutare l'UX.

Ora è riconosciuto che progettare per l'esperienza include, ma è molto più che progettare per l'efficienza e altri attributi tradizionali di usabilità. Mentre l'efficienza è focalizzata su attributi come veloce, facile, funzionale, privo di errori, la UX coinvolge sentimenti e quindi si concentra sul bello, emotivo, stimolante e anche tattile e acustico in caso di interfacce multidimensionali.

Un prodotto, in grado di generare una UX positiva dovrebbe essere utile, usabile e desiderabile. Al fine di creare prodotti desiderabili, la UX mette molta enfasi sul piacere e, quindi, sull'estetica e sul divertimento. Alcuni studi mostrano la correlazione tra estetica e percezione della qualità dell'interfaccia. È stato

dimostrato che le interfacce esteticamente attraenti sono percepite come più utili anche quando sono leggermente meno utili di un'interfaccia con funzionalità simili ma meno attraente. Altri studi sottolineano l'importanza di un buon layout con colori adeguati, stili di font, spazi vuoti, mostrando come i piccoli dettagli abbiano effettivamente un grande impatto sulla percezione dell'interfaccia da parte degli utenti.

## 1.4 Human-Centred Design

Progettare una sistema che gli utenti trovino usabile non è un obiettivo facile da raggiungere. Uno dei motivi per cui molti sistemi computer-based siano difficili da utilizzare è dato dal fatto che durante la progettazione del prodotto, l'enfasi e l'attenzione sono rivolte al sistema e non alle persone che il sistema dovranno utilizzarlo. I progettisti contano sul fatto che gli umani si adattano più facilmente di quanto non possano fare le macchine. La progettazione di sistemi usabili richiede un drastico cambiamento di mentalità rispetto all'approccio di progettazione tradizionale. In quest'ultima, l'oggetto principale dell'attenzione è il sistema da progettare. Il processo di progettazione parte dalla definizione dei suoi requisiti funzionali, descritti in dettaglio in un documento di specifiche funzionali, a partire dal quale il sistema viene progettato e quindi realizzato.

Se l'obiettivo è la progettazione di un sistema usabile, questo approccio non funziona. In questo caso, il progettista dovrà porre la sua attenzione in primo luogo sull'utente e dovrà studiarne le caratteristiche, le abitudini e le necessità in relazione all'uso del sistema. Dovrà preconfigurarne i *contesti* in cui il sistema sarà utilizzato, i suoi diversi *casi d'uso* e i *compiti* che l'utente svolgerà con il sistema.

### 1.4.1 Modelli iterativi

Se il modello a cascata è inadeguato, ci serve un modello che coinvolga gli utenti fin da subito, non solo nella stesura dei requisiti e specifiche, ma anche per sperimentare l'uso di versioni preliminari del sistema e aiutarci, con le loro reazioni e le loro indicazioni, a correggere il tiro, in un processo di prove ed aggiustamenti successivi. 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 e lo si sottopone all'utente che prova ad usarlo. Questo prototipo, per quanto embrionale, permette di verificare alcune assunzioni di partenza ed eventualmente di aggiustare il tiro. Si realizza quindi un nuovo prototipo sempre incompleto, ma un pò più somigliante al sistema finale e lo si sottopone ancora alla prova degli utenti, e così via per approssimazioni successive fino alla conclusione del progetto. In sostanza, le prove d'uso diventano una parte integrante del processo di progettazione. L'avanzamento del progetto non è più scandito dal passaggio da un'attività alla successiva, ma dalla realizzazione dei diversi prototipi.



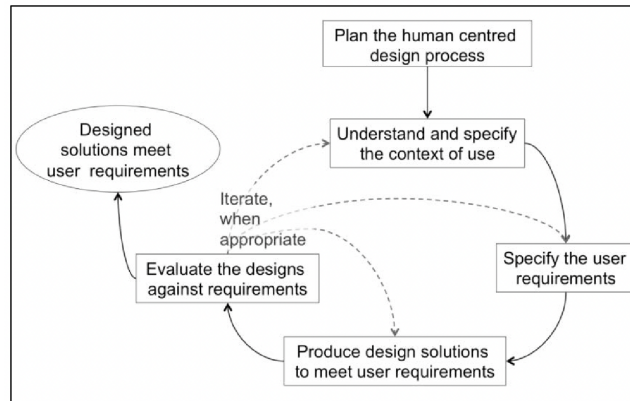


Figura 1.1: Il processo di progettazione human-centred secondo l'ISO 9241-210.

### 1.4.2 Il modello ISO 9241-210

Un modello iterativo divenuto autoritario nell'ambito dell'ingegneria dell'usabilità è quello proposto dallo standard ISO 9241-210, dal titolo "Human-Centred design processes for interactive system", che ha proprio lo scopo di fornire una guida alle attività di progettazione centrata sull'essere umano lungo il ciclo di vita dei sistemi basati su computer. Il modello iterativo corrispondente a tale standard è riportato in Figura 1.1.

Come si vede dalla figura, l'ISO 9241-210 suddivide il ciclo di progettazione nelle seguenti quattro fasi che, dopo il primo ciclo, possono ripetersi in un qualunque ordine.

#### Comprendere e specificare il contesto d'uso

È importante comprendere e definire il contesto in cui il sistema sarà utilizzato per orientare le decisioni sul progetto e fornire una loro successiva convalida. La descrizione del contesto d'uso del sistema dovrebbe comprendere i seguenti argomenti:

- Le caratteristiche degli utenti. Le caratteristiche rilevanti potrebbero comprendere le competenze, le abilità, le esperienze e le preferenze. Spesso sarà necessario classificare gli utenti in diverse categorie (target).
- I compiti che gli utenti dovranno eseguire. Dovrebbero essere analizzati i compiti che possono influenzare l'usabilità del sistema, per esempio indicandone frequenza e durata.
- L'ambiente nel quale gli utenti utilizzeranno il sistema. Si descriveranno le caratteristiche rilevanti dell'ambiente fisico e sociale, includendo ambiente di lavoro, tecnologie utilizzate, eventuali standard adottati, contesto normativo e così via.

### **Specificare i requisiti utente e organizzativi**

Nei processi di progettazione, esiste una consistente attività per la specifica dei requisiti del prodotto o del sistema. Nella progettazione human-centred, quest'attività dovrebbe essere ampliata, per descrivere i requisiti in relazione al contesto d'uso sopra specificato. I requisiti dovrebbero essere organizzati per livelli di priorità e formulati in modo da permettere la loro successiva convalida mediante opportuni test.

### **Produrre soluzioni di progetto**

In questa fase si individuano le possibili soluzioni di progetto basandosi sullo stato dell'arte, sulle conoscenze ed esperienze dei partecipanti e sui risultati dell'analisi del contesto d'uso. Lo standard identifica le seguenti attività.

- Utilizzare le conoscenze disponibili per sviluppare proposte di progetto con un approccio multidisciplinare.
- Rendere le soluzioni di progetto più concrete, utilizzando simulazioni, modelli e prototipo di vario tipo. L'uso di prototipi permette ai progettisti di comunicare più efficacemente con gli utenti. Un prototipo può essere semplice quanto uno schizzo a matita o complesso come una simulazione su computer, quasi indistinguibile dal prodotto finale.
- Presentare le soluzioni di progetto agli utenti, permettendo loro di eseguire i compiti che il sistema è destinato a supportare. Questo serve a raccogliere le loro reazioni, per poi utilizzarle nell'orientare le attività di progettazione successive.
- Modificare il progetto in conseguenza delle reazioni degli utenti, e ripetere questo processo fino a che gli obiettivi della progettazione non siano raggiunti.
- Gestire l'iterazione delle soluzioni di progetto. Per tenere sotto controllo i progressi della progettazione iterativa, si dovrebbero registrare i risultati delle attività precedenti.

### **Valutare il progetto nei confronti dei requisiti**

Lo standard identifica le attività illustrate di seguito.

- Proporre il piano di valutazione, precisando quali parti del sistema devono essere valutate, quali prototipi dovranno essere realizzati e come deve essere eseguita la valutazione.
- Fornire feedback per la progettazione. Per influenzare la progettazione, la valutazione dovrebbe essere condotta in ogni fase del ciclo di vita del sistema.
- Verificare se gli obiettivi sono stati raggiunti.

- Validazione sul campo, ovvero 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, in quanto alcuni effetti dell'utilizzo di un sistema interattivo non sono riconoscibili fino a che il sistema non sia stato utilizzato per 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.

## 1.5 Valutare l'usabilità

Come specifica lo standard ISO 9241-120, la valutazione è un passo essenziale in una progettazione human-centred. All'inizio del progetto, l'obiettivo principale sarà la raccolta di indicazioni per orientare le attività di progettazione successive. nelle fasi più avanzate, con la disponibilità di prototipi più completi, sarà invece possibile quantificare il livello di raggiungimento degli obiettivi dell'utente e dell'organizzazione.

Per compiere attività di valutazione dell'usabilità si possono impiegare diverse tecniche, che rientrano in due categorie:

- Valutazioni effettuate da parte di esperti di usabilità, senza alcun coinvolgimento dell'utente. Queste valutazioni prendono il nome di *ispezioni*. Le più note sono le cosiddette *valutazioni euristiche*.
- Valutazioni effettuate con il coinvolgimento degli utenti, tra queste i *test di usabilità*.

### 1.5.1 Valutazioni euristiche

Nell'ingegneria dell'usabilità, si chiamano euristiche quelle valutazioni di usabilità effettuate da esperti, analizzando il comportamento di un sistema e verificandone la conformità a specifiche regole, chiamate appunto euristiche, derivanti da principi o linee guida generalmente accettati.

In pratica, un ingegnere dell'usabilità si immedesima in un utente che deve svolgere determinati task riflettendo su tutti quegli aspetti che non trova immediatamente chiari e cercando il principio di usabilità violato. L'utilizzo di una tabella opportunamente creata può facilitare e sistematizzare l'esecuzione di tale attività. Con la valutazione euristica è possibile ottenere buoni risultati solo impiegando più valutatori in maniera tale da confrontare i risultati e correlare i problemi riscontrati. In organizzazioni che non prevedono figure dedicate a tale mansione, gli stessi progettisti dovrebbero valutarsi a vicenda i propri moduli.

Le euristiche impiegate sono diverse, in genere si preferisce utilizzare le note euristiche di Nielsen, costituite da dieci regole che, sebbene molto generali,

permettono al valutatore di inquadrare i problemi rilevati in categorie bene individuate.

Le dieci euristiche di Nielsen sono:

1. *Visibilità dello stato del sistema.* Il sistema dovrebbe sempre informare gli utenti su ciò che sta accadendo mediante feedback appropriati in tempo ragionevole.
2. *Corrispondenza fra il mondo reale e il sistema.* Il sistema deve utilizzare termini familiari all'utente ed evitare termini orientati al sistema. Deve far apparire le informazioni secondo un ordine logico e naturale.
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. Fornire all'utente le funzioni di undo e redo.
4. *Coerenza e standard.* Gli utenti non dovrebbero aver bisogno di chiedersi se parole, situazioni o azioni differenti hanno lo stesso significato.
5. *Prevenzione degli errori.* Ancora meglio di buoni messaggi di errore è un'attenta progettazione che eviti l'insorgere del problema. Eliminare le situazioni che possono provocare errori da parte dell'utente, e chiedergli conferma prima di eseguire le azioni richieste.
6. *Riconoscere piuttosto che memorizzare.* Minimizzare il ricorso alla memoria dell'utente rendendo visibili azioni e opzioni. L'utente non dovrebbe aver bisogno di ricordare delle informazioni, quindi le istruzioni per l'uso del sistema dovrebbero essere visibili o facilmente recuperabili.
7. *Flessibilità ed efficienza d'uso.* Acceleratori (short-cut) possono rendere veloce l'interazione dell'utente esperto. Importante è permettere all'utente di personalizzare le azioni frequenti.
8. *Design minimalista ed estetico.* I dialoghi non dovrebbero contenere informazioni irrilevanti o necessarie solo di rado.
9. *Aiutare gli utenti a riconoscere gli errori, diagnosticarli e correggerli.* I messaggi di errore dovrebbero essere espressi in linguaggio semplice (senza codici) indicare il problema con precisione e suggerire una soluzione.
10. *Guida e documentazione.* Anche se è preferibile che il sistema sia utilizzabile senza documentazione, essa può essere necessaria per fornire aiuto.

### 1.5.2 Test di usabilità

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 di rivolge, e si chiede a tutti di svolgere separatamente gli stessi compiti. Chi conduce il test osserva e analizza il loro comportamento per comprendere se, dove e perché hanno incontrato delle difficoltà. Il test coinvolge, oltre all'utente che prova il sistema, almeno altre due persone: un *osservatore* che assiste al test annotando le considerazioni e i comportamenti dell'utente che ritiene significativi e il *facilitatore*, che ha il compito di gestire la “regia” della prova e fornisce supporto all'utente durante il test evitando di influenzarne il corso.

Un test di usabilità ha lo scopo di ricavare indicazioni concrete per il miglioramento del sistema. Chi lo conduce dovrà esaminare in dettaglio le operazioni svolte dagli utenti per capire da dove nascono le difficoltà, da cosa sono causate e in quale modo possano essere rimosse. Per questo è molto utile applicare la 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. È una tecnica molto utile perché permette agli osservatori di raccogliere informazioni sulle strategie messe in atto dall'utente nell'esecuzione dei compiti e sulle difficoltà che egli incontra durante il test.

L'osservatore, per tutta la durata del test, dovrà quindi prendere appunti, registrando le situazioni in cui l'utente manifesta incertezza o viene condotto a commettere errori dal sistema. Questi appunti verranno riesaminati in seguito, per individuare le cause del problema e studiare le correzioni più opportune. se possibile è preferibile eseguire una registrazione (audio e video) della sessione di test, per rivedere in seguito tutto ciò che è avvenuto durante la prova.

Fino a qualche anno fa era diffusa la convinzione che per fare un buon test di usabilità fosse indispensabile usare un laboratorio appositamente attrezzato (*usability lab*). esso è costituito da due stanze contigue separate da uno specchio: una per l'utente che prova e una per gli osservatori. Laboratori di questo tipo sono costosi e non sono indispensabili per eseguire dei buoni test di usabilità.

### 1.5.3 Test formativi e test sommativi

I test di usabilità possono essere classificati, in funzione dei loro obiettivi, in due grandi categorie: test *formativi* e test *sommativi*.

I primi sono utilizzati durante il ciclo di progettazione, per sottoporre i vari prototipi a prove d'uso con gli utenti, allo scopo di identificarne i difetti e migliorarne l'usabilità. Si chiamano formativi perché contribuiscono a “dare forma” al prodotto. È spesso conveniente eseguire questi test in modo rapido e approssimativo in quanto nelle fasi iniziali del progetto i test mettono alla luce rapidamente i difetti macroscopici, rendendo inutile scendere nel dettaglio. Infine spesso si verifica un *effetto di mascheramento*: ogni problema di usabilità nel quale incappiamo monopolizza la nostra attenzione e ci impedisce di vederne altri, soprattutto se di più lieve entità.

In merito al numero di utenti ai quali sottoporre il test, dice Nielsen, i primi cinque utenti metteranno in evidenza la maggior parte dei problemi di usabilità più significativi, gli utenti successivi non farebbero altro che confermare gli stessi problemi.

I test della seconda categoria si chiamano sommativi in quanto forniscono una valutazione sommaria, complessiva, del prodotto. 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.

#### 1.5.4 Test di compito e test di scenario

Rispetto alle attività condotte dagli utenti durante le prove, i test di usabilità possono essere classificati in due grandi categorie: *test di compito* e *test di scenario*. Nei test di compito gli utenti svolgono compiti che permettono di esercitare funzioni specifiche del sistema. Questi test possono essere eseguiti anche quando il sistema non è completamente sviluppato. Un errore da evitare, da parte dei conduttori, è quello di suggerire implicitamente le operazioni da eseguire. L'utente deve, invece, essere posto in situazioni simili a quelle in cui si troverà nell'uso reale del sistema, quando dovrà decidere in autonomia cosa fare.

La seconda categoria è costituita dai test di scenario. In questo caso, agli utenti viene indicato un obiettivo da raggiungere attraverso una serie di compiti elementari, senza indicarli esplicitamente. Per un test più realistico, all'utente 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 in modo più impegnativo dei test di compito. essi permettono agli utenti di utilizzare il sistema in relazione alle proprie specifiche necessità, preferenze e abitudini. Per questo motivo, i test di scenario possono essere molto utili per individuare eventuali carenze nell'impostazione della struttura complessiva dell'interazione o mancanze di funzionalità utili. È indispensabile che le istruzioni agli utenti siano date per iscritto, nel modo più chiaro possibile.

#### 1.5.5 Misure

Oltre all'osservazione qualitativa dei comportamenti degli utenti, 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 di compiti che ciascuno riesce a portare a termine. Il calcolo del tasso di successo può tener conto anche dei compiti eseguiti solo in parte. Per esempio, il test descritto in Tabella 1.1 riporta che 4 utenti eseguono 6 compiti. Su 24 compiti, 9 sono stati portati a termine con successo (S) e 4 sono stati eseguiti solo in parte (P). Per i rimanenti, gli utenti hanno fallito (F).

In questo caso, il tasso di successo potrebbe essere calcolato così:

$$\text{Tasso di successo} = (9 + (4 * 0,5)) / 24 = 46\%$$

In questa formula, ogni successo parziale è stato conteggiato, convenzionalmente, come la metà di un successo pieno. Questo dato ci dà indicazione piuttosto significativa sull'usabilità del sistema.

	Compito 1	Compito 2	Compito 3	Compito 4	Compito 5	Compito 6
Utente 1	F	F	S	F	F	S
Utente 2	F	F	P	F	P	F
Utente 3	S	F	S	S	P	S
Utente 4	S	F	S	F	P	S

Tabella 1.1: Sintesi risultati di un test di usabilità.

### 1.5.6 Come condurre un test di usabilità

Un test di usabilità viene condotto in quattro fasi successive, come indicato nella Figura 1.2.

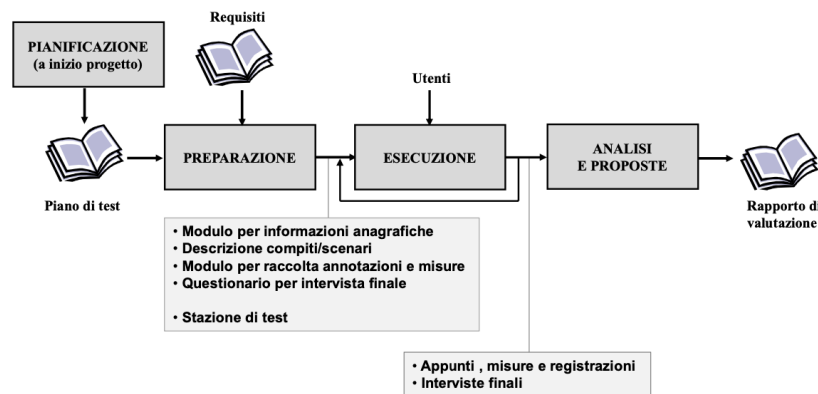


Figura 1.2: Le fasi di un test di usabilità.

#### Pianificazione

I prototipi prodotti in un ciclo di sviluppo iterativo dovrebbero essere realizzati in accordo a un piano di lavoro definito all'inizio del progetto. In accordo con lo standard ISO 9241-120, si raccomanda che l'intero processo di valutazione sia pianificato in anticipo, precisando quali parti del sistema devono essere valutate e come; quali prototipi dovranno essere realizzati, come deve essere eseguita la valutazione; quali dovranno essere le interazioni con gli utenti e come dovrà essere condotta l'analisi dei risultati.

#### Preparazione del test

Nella fase di 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. Il team dovrà decidere le misure da raccogliere e predisporrà la logistica per l'esecuzione della prova. deve, inoltre, preparare i materiali necessari allo svolgimento dei test e in particolare:

- Un modulo per raccogliere le informazioni sugli utenti
- Un modulo di consenso al trattamento dei dati personali
- 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**

La fase di esecuzione del test vera e propria non dura in genere più di qualche ora complessivamente. È molto importante che sia chiarito molto bene che l'obiettivo della prova è valutare il sistema e non la capacità dell'utente a volgere i task nel minor tempo possibile. È indispensabile che il facilitatore metta ogni utente a suo agio per ridurre lo stress da esame. Bisogna spiegare bene che se una persona trova difficile usare il sistema, questo non avviene perché è incapace ma perché il sistema è progettato male.

I test devono essere condotti singolarmente, un utente alla volta. Durante lo svolgimento della prova i valutatori dovranno interferire il meno possibile, ma il facilitatore dovrà, invece, ricordare il think aloud.

Al termine del test di usabilità è utile intervistare gli utenti sull'esperienza che hanno appena fatto. Utilizzando un apposito questionario l'intervistatore chiederà a ogni utente quali sono, a suo parere, i punti di forza e di debolezza del sistema, gli aspetti che andrebbero migliorati e quelli che ha gradito maggiormente.

### **Analisi dei risultati e proposte migliorative**

L'ultima fase del test è quella in cui si analizza il materiale raccolto e si traggono le conclusioni. È la più delicata, e richiede tempo e grande cura.

Il valutatore non deve mai accontentarsi dei commenti degli utenti, ma deve sempre compiere un'analisi diretta e dettagliata dei loro comportamenti, esaminando il materiale registrato o gli appunti presi durante le sessioni di prova. Egli deve inoltre elencare dettagliatamente tutti i problemi individuati, grandi e piccoli. Il risultato di quest'analisi è un elenco dei problemi incontrati nello svolgimento di ciascun compito ed a ciascun problema il team di valutazione assegna un livello di gravità. L'elenco dei problemi sarà dunque riesaminato per produrre un elenco di interventi proposti per migliorare il prodotto. Essi dovranno essere organizzati in diversi livelli di gravità.



## **Il rapporto di valutazione**

L'esito di una valutazione di usabilità dovrebbe essere descritto in modo accurato. A tale scopo si utilizza un documento chiamato *rapporto di valutazione* che non solo descrive i risultati dei test effettuati, ma fornisce anche evidenza del fatto che essi siano stati condotti con metodi adeguati.

## Capitolo 2

# I requisiti

### 2.1 Processo di definizione dei requisiti

Tutti i processi dovrebbero iniziare con la stesura di un documento di specifica dei requisiti.

Un requisito è una proprietà richiesta, o auspicabile, del prodotto. Il documento dei requisiti ha allora lo scopo di raccogliere una descrizione di tutte le proprietà desiderate. Dalla sua formulazione dovrebbe essere chiaro se un requisito esprime una proprietà obbligatoria oppure soltanto suggerita o auspicabile. Segue un esempio di requisito obbligatorio ed un esempio di requisito suggerito:

- Il sito *deve* permettere di inserire nel carrello i prodotti di cui sta valutando l'acquisto.
- L'intero processo di acquisto di un prodotto *dovrebbe* richiedere massimo cinque minuti.

Possiamo, inoltre, distinguere i *requisiti funzionali*, i quali descrivono le funzioni che il sistema deve analizzare, dai *requisiti non funzionali*, i quali descrivono proprietà che il prodotto dovrà possedere. Lo scopo della definizione dei requisiti è individuarli e descriverli nel modo più specifico e meno ambiguo possibile.

È sempre bene tenere a mente che quando specifichiamo i requisiti di un prodotto, non stiamo progettando, ma *stiamo ponendo dei vincoli all'attività di progettazione*, che seguirà. Lo scopo del documento è di indicare *che cosa* deve essere realizzato e *perché*, non *come* deve essere realizzato.

Il processo di definizione dei requisiti può essere suddiviso in tre attività fondamentali, che possiamo chiamare *raccolta*, *organizzazione* e *revisione*, come illustrato in Figura 2.1

Nella fase di *raccolta*, le persone incaricate di produrre il documento dei requisiti raccolgono il maggior numero possibile di informazioni sugli obiettivi e sulle necessità riguardo al sistema da costruire.

Le informazioni vengono raccolte da fonti diverse:



Figura 2.1: Il processo di definizione dei requisiti.

- Dal *committente*, cioè colui che ha avviato il progetto e costituisce il riferimento principale.
- Dalle interviste con gli *stakeholder* del prodotto, cioè tutti coloro che hanno qualche interesse nel prodotto.
- Dall'analisi della *concorrenza*, cioè di quei prodotti con i quali il prodotto in costruzione dovrà confrontarsi e competere.

Durante quest'attività, vengono raccolti appunti e materiale informativo vario che dovranno successivamente essere riesaminati, selezionati e organizzati. Questo è lo scopo della successiva attività di *organizzazione*. L'obiettivo principale di questa fase è costruire una prima bozza del *documento di specifica dei requisiti* in cui vengono descritti in forma completa e chiara le richieste del committente e i vincoli da rispettare nelle fasi successive del progetto. Nella fase di *revisione e approvazione*, la bozza del documento dei requisiti viene presentata al committente per la sua approvazione. Essendo un processo iterativo, il documento dei requisiti non potrà mai considerarsi finale.

### 2.1.1 La fase di raccolta

È il caso di approfondire la lunga e complessa fase di raccolta. Questa fase presenta spesso notevoli difficoltà. I problemi sono di quattro tipi:

- *Problemi di ambito*. Generalmente, i contorni del sistema da progettare non sono ben definiti. Esiste sempre il rischio di ampliare eccessivamente il campo di ricerca oppure di restringerlo a tal punto da tralasciare aspetti che potrebbero rivelarsi importanti nelle fasi successive.
- *Problemi di comprensione*. Gli utenti hanno una comprensione solo parziale dei loro bisogni. Inoltre, chi raccoglie i requisiti ha spesso una conoscenza ridotta del dominio e utilizza un linguaggio diverso da quello dei stakeholders.

- *Problemi di conflitto.* Stakeholder diversi possono avere punti di vista diversi sul sistema che devono essere fatti emergere con chiarezza, e in qualche modo risolti nel documento dei requisiti finale.
- *Problemi di volatilità.* I requisiti evolvono nel tempo.

### 2.1.2 Importanza dello studio degli utenti

Per quanto complicato possa essere, intervistare gli utenti finali del sistema da produrre è fondamentale. Intervistare solo il committente, infatti, non permette di ottenere le informazioni che solo gli utenti possono fornire. Gli utenti sono coloro che decidono se utilizzare un prodotto. Sono persone che hanno preferenze, abitudini, istruzione e formazione professionale che utilizzano ogni qualvolta lavorano con un prodotto. Conoscenze ed esperienze precedenti influenzano come apprendono ed usano i prodotti che sviluppiamo. Non supportare le loro necessità può provocare alti costi in termini di produttività per l'azienda e di frustrazione per l'utente. Seguono le principali tecniche che possono essere utilizzate nella fase di raccolta per la raccolta dei dati relativi agli utenti.

### 2.1.3 Interviste

La tecnica normalmente più usata è quella delle interviste individuali con il committente e i principali stakeholder del prodotto. Per ottenere la massima sincerità, di solito si garantisce agli intervistati che le loro opinioni verranno riportate solo in forma anonima.

Le interviste individuali possono essere più o meno strutturate. Le *interviste non strutturate* o *flessibili* sono di carattere esplorativo e assomigliano a delle conversazioni sugli argomenti di interesse. È utile effettuare queste interviste sulla base di un canovaccio preparato in anticipo, in modo da essere sicuri di non tralasciare alcun aspetto rilevante. L'intervistatore, che dovrà essere un esperto, potrà comunque orientare il colloquio diversamente da quanto pianificato per esplorare eventuali aspetti non previsti che emergessero nella conversazione. Le *interviste strutturate* prevedono invece un insieme di domande predefinite. Essi sono utili soprattutto quando gli obiettivi del colloquio siano stati bene identificati. Queste domande sono poste in forma identica a tutti gli intervistati e non è necessario che il somministratore sia un esperto. Le *interviste semi-strutturate* contengono sia domande libere, con carattere esplorativo, sia domande specifiche.

Durante la conduzione di un'intervista occorre evitare di influenzare l'intervistato, formulando le domande in modo che non contengano implicitamente già la risposta. Inoltre l'intervistatore dovrà evitare di utilizzare termini tecnici.

### Preparazione dell'intervista

Durante la preparazione di un'intervista andrebbero considerati i seguenti aspetti:

- *Quali informazioni si vogliono reperire.* Occorre identificare i bisogni degli utenti per capire quale obiettivo vogliono perseguire attraverso l'applicazione che si vuole sviluppare. Identificare il loro modello di comportamento, capendo quali azioni eseguono per realizzare i loro goal, quali sono le loro priorità e quali difficoltà incontrano. Occorre, inoltre, verificare il grado di soddisfazione degli utenti.
- *Chi intervistare.* Se lo scopo dell'intervista è determinare il profilo o il modello di comportamento dell'utente allora vanno intervistati gli utenti finali del prodotto. Va considerato, però, che all'interno della stessa categoria di utenza vi sono persone diverse per ruolo, area geografica e età. Distinguiamo le interviste singole dalle interviste di gruppo. Le prime consentono di ottenere risposte più dettagliate, di percepire le differenze ed analogie tra intervistati della stessa categoria e si protraggono più a lungo. Le seconde, consentono dialoghi tra gli intervistati, consentono di parlare con più persone impiegando minor tempo, ma sono meno prevedibili e quindi difficili da pianificare.
- *Come strutturare l'intervista.* La struttura di un'intervista può variare. La struttura top-down prevede una fase iniziale di domande di carattere generale per poi passare a domande più dettagliate, riguardanti particolari di interesse. Una intervista con struttura in dettaglio inizia con domande semplici e specifiche ("Cosa?", "Quando?") per passare a domande più finalizzate ("Perché?", "Come?") e all'osservazione dell'utente mentre porta a compimento i suoi compiti. Infine, un'intervista può anche essere effettuata nel contesto in cui l'utente opera. Nella formulazione delle domande è fondamentale evitare che le domande preannuncino o condizionino la risposta attesa. Non vanno chieste valutazioni sull'importanza di un aspetto dell'applicazione, quanto cercare di capire quando tale aspetto è importante e perché. Non si deve predire nella domanda formulata la risposta che si vuole ottenere dall'utente.
- *Dove svolgere l'intervista.* Il luogo più adatto per condurre l'intervista è l'ambiente in cui l'utente opera. Questo permette all'investigatore di osservare direttamente il contesto d'uso e consente agli utenti di mostrare piuttosto che descrivere. Un caso particolare è l'intervista contestuale (contextual inquiry). Se l'ambiente di lavoro non è disponibile, è bene mettere gli utenti a proprio agio in un luogo poco rumoroso e prevedere delle domande per individuare il contesto lavorativo.
- *Come raccogliere e memorizzare i dati.* Per la raccolta dei dati possono essere combinate tecniche diverse. Una registrazione video riproduce l'intervista in tutta la sua interezza, compreso l'aspetto fisico. Il problema è che è costosa e poco efficiente (in termini di tempo) da consultare nella fase di analisi dei dati. Una registrazione audio è meno costosa della registrazione video, ma fornisce anche un minor numero di informazioni.

Inoltre è poco agevole da consultare in fase di analisi dei dati. Le note testuali, invece, sono più facili da consultare in fase di analisi e a posteriori, quando possibile si affiancano ad audio e video.

- *Come analizzare i dati.* Per la fase di analisi dei dati raccolti è opportuno creare tabelle (una per ogni utente) con le osservazioni ed i fatti principali, ricavati dalle registrazioni e dagli appunti scritti. È utile anche raggruppare le tabelle in base alle analogie nei dati dei vari utenti ed etichettare le categorie risultanti. Fare un resoconto finale per ogni categoria dei dati e dedurre solo ciò che si può dimostrare.

Ora è decisivo capire come utilizzare i risultati ottenuti dalle interviste. Nelle varie fasi di design e implementazione, occorre accertarsi che si stia procedendo secondo i bisogni dell'utente individuati tramite l'intervista. Occorre utilizzare gli aspetti evidenziati dall'analisi come spunto per l'ideazione e lo sviluppo di nuove caratteristiche dell'applicazione. Inoltre sono utili per risolvere i problemi eventualmente evidenziati dagli utenti durante l'intervista.

**Le interviste nel ciclo di vita del software** Le interviste sono utilizzate durante le seguenti fasi di sviluppo del software:

- Analisi dei requisiti: individuazione dei requisiti del sistema e determinazione del profilo utente
- Progetto: valutazione delle specifiche dei requisiti e valutazione delle funzionalità del sistema
- Implementazione: individuazione di problemi evidenti
- Prodotto sul campo: valutazione del grado di soddisfazione dell'utente.

#### 2.1.4 Focus group

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, o, a volte, un punto di vista condiviso fra tutti. Sono normalmente condotti da un animatore che guida la discussione e un osservatore che esamina le dinamiche di relazione del gruppo e prende appunti. La conduzione di un focus group non è facile in quanto bisogna evitare che il gruppo sfugga di mano ed occorre fare in modo che tutti possano esprimere le proprie idee.

#### Osservazione sul campo

Non sempre gli utenti sono in grado di spiegare in dettaglio quali sono le modalità di uso desiderate per il prodotto nella loro attività. Uno studio sul campo per apprendere come gli utenti si comportano nella realtà può quindi essere molto istruttivo. Osservare gli utenti sul campo, infatti, permette di individuare quali

fattori ambientali ed organizzativi influenzano il loro lavoro ed in che modo, di vedere cosa accade quando si verificano delle situazioni critiche.

L'osservazione degli utenti sul campo è una tecnica che richiede una attenta pianificazione. È importante, infatti, aver bene in mente:

- cosa si intende apprendere attraverso la visita sul luogo di lavoro
- chi si vuole osservare o intervistare
- dove svolgere l'intervista o l'osservazione
- quando si potrebbe andare sul posto e quanto tempo si prevede che si possa dedicare in ciascuna intervista
- come trovare le persone che si ha bisogno di incontrare e come convincerle a partecipare
- quali tecniche utilizzare per raccogliere i dati, quindi quali media utilizzare: videoregistratori, registratori audio, foto, note a penna, moduli prestampati, ecc.

Affinché l'osservazione sul campo sia ben eseguita si deve:

- chiedere all'utente di pensare ad alta voce (think aloud)
- osservare, parlare e ascoltare l'utente ed annotare le caratteristiche importanti per il progetto
- prender nota dell'ambiente di lavoro di ciascun utente
- capire gli obiettivi degli utenti
- capire i compiti degli utenti
- annotare quali fattori determinano l'esecuzione durante l'osservazione
- fare attenzione alle interazioni con altre persone, altri programmi e documenti
- osservare cosa succede quando l'utente ha terminato il compito.

### 2.1.5 Gli scenari d'uso

Come già detto la progettazione presenta diverse difficoltà date dal fatto che tendiamo a concentrarci sull'oggetto della progettazione, trascurando il contesto d'uso. A tal proposito, una tecnica molto utile per aiutarci a immaginare un nuovo prodotto nel suo contesto di utilizzo è quella di ipotizzare possibili *scenari d'uso*. Uno scenario d'uso è la narrazione di una possibile storia dell'uso del sistema da parte di uno specifico utente, fittizio ma in qualche modo tipico, e descritto in modo molto realistico. Gli scenari d'uso devono mettere in scena situazioni d'uso tipiche, ma non ovvie, non devono contenere dettagli irrilevanti

allo scopo. Inoltre è bene che siano complete, indicando le motivazioni e le conseguenze dell'uso del prodotto nella particolare situazione.

L'impiego degli scenari d'uso è molto utile nella progettazione di un prodotto. L'ideazione di storie d'uso tipiche e concrete è, infatti, un modo molto efficace per collocare il prodotto, ancora da progettare, nei suoi possibili contesti d'uso reali. Scrivere uno scenario vissuto da personaggi dotati di una loro specifica identità ci aiuta a considerare un prodotto in modo più oggettivo. Pertanto, è molto importante che i protagonisti di uno scenario siano persone concrete, anche se fittizie, dotati di una precisa identità. A questi utenti le cui storie raccontiamo negli scenari d'uso si dà spesso il nome di *personae*.

### 2.1.6 Questionari

Un'altra tecnica per lo studio degli utenti consiste nella somministrazione dei questionari, i quali permettono di raccogliere informazioni in forma strutturata, elaborabili con metodi statistici.

I questionari presentano diversi vantaggi ma anche alcuni svantaggi rispetto alle interviste. Essi, infatti, sono meno flessibili delle interviste, in quanto sono costituiti da domande fisse la cui possibilità di approfondimento è ridotta, ma consentono di analizzare un campione di utenti più ampio in una minore quantità di tempo ottenendo un'analisi dei dati più rigorosa. È importante progettare il questionario in modo appropriato tenendo presente quali informazioni si vogliono cercare come si vorranno analizzare i dati.

#### Come organizzare i questionari

Dal momento che la compilazione dei questionari non è sempre gradita dagli utenti è importante creare questionari che siano semplici e brevi, e in caso di questionari più lunghi, si dovrebbero promettere bonus, premi finali o un qualche incentivo.

Le domande all'interno di un questionario possono essere di due tipi: *chiuse* o *aperte*. Le prime invitano l'utente a scegliere una risposta tra le alternative proposte, le seconde chiedono all'utente di fornire informazioni generiche, che non essendo strutturate, non possono essere analizzate formalmente. Per tale motivo in un questionario è preferibile l'utilizzo di domande chiuse, lasciando le domande aperte come fonte di informazioni supplementari. Spesso un questionario sono presenti domande di carattere generico, per l'identificazione della formazione dell'utente e della sua anagrafica.


Successivamente, nella fase post-questionario, i valori delle diverse scale sono convertiti in valori numerici sui quali vengono effettuate analisi statistiche dei dati (media, deviazione standard).

#### Tipologie di domande chiuse


A seconda del tipo di risposta richiesto, le domande a risposta chiusa si distinguono in:



- 🟢 “Che tipo di software ha già usato?”
- |  |  |
|--|--|
| <input type="checkbox"/> Editor di testo | <input type="checkbox"/> Sistemi esperti |
| <input type="checkbox"/> Database        | <input type="checkbox"/> Compilatori     |
| <input type="checkbox"/> Fogli di lavoro | <input type="checkbox"/> Help on-line    |

-  **Ordinare (o Ranked Order):** l'utente è chiamato ad indicare un ordine per gli elementi di una lista, in base alle sue preferenze  
 "Ordinate, in base alla loro utilità, i seguenti metodi di comando"  
 (1="il più utile"; 2="meno utile"; 3="non usato")
- | Selezione da menu        | Linea di comando         | Acceleratori             |
|--------------------------|--------------------------|--------------------------|
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

- Sei in grado di usare i seguenti comandi?"
- |         |                                |                                |                                    |
|---------|--------------------------------|--------------------------------|------------------------------------|
| COPIA   | Sì<br><input type="checkbox"/> | No<br><input type="checkbox"/> | Non so<br><input type="checkbox"/> |
| INCOLLA | Sì<br><input type="checkbox"/> | No<br><input type="checkbox"/> | Non so<br><input type="checkbox"/> |

- ② “Indica il grado di utilità del comando COPIA in base alla scala di seguito riportata”
- Molto utile  Totalmente inutile

◉ “Ritengo che il comando COPIA sia utile”				
Molto d'accordo	parzialmente d'accordo	non ho una opinione precisa	parzialmente in disaccordo	molto in disaccordo

Figura 2.6: Esempio di domanda in scala Likert.

- Scala Likert: richiede che l’utente indichi il suo grado di accordo ad una specifica affermazione
- Differenziale Semantico: scala di giudizio bipolare, con due aggettivi opposti agli estremi. Richiede che l’utente indichi un valore della scala compreso tra i due aggettivi estremi

◉ Usare il sistema è stato:								
	Molto	Abbastanza	Poco	né l'uno né l'altro	Abbastanza	Poco	Molto	
Facile	1	2	3	4	5	6	7	difficile
Piacevole	1	2	3	4	5	6	7	Spiacevole
Inutile	1	2	3	4	5	6	7	Utile

Figura 2.7: Esempio di domanda a differenziale semantico.

Questo tipo di questionari, nel caso in cui ci siano un numero dispari di valori, permette all’utente di non avere un’opinione ben precisa (es. “non saprei” tra le risposte). Al contrario, se presentano un numero pari di valori tra le risposte, gli utenti devono necessariamente prendere una posizione.

## Il questionario SUS

Nell’ingegneria dell’usabilità, la *system usability scale* (SUS) è un questionario su scala Likert con dieci elementi che fornisce una visione globale delle valutazioni soggettive di usabilità. È stato sviluppato da John Brooke presso Digital Equipment Corporation nel Regno Unito nel 1986 come strumento psicométrico da utilizzare nell’ingegneria dell’usabilità dei sistemi computer based.

Il calcolo del punteggio ottenuto dalla compilazione del questionario SUS, per tenere conto delle polarità invertite, ricalcola i punteggi nel modo seguente:

- Per gli item dispari si calcola sottrae un punto dal punteggio assegnato dal partecipante
- Per gli item pari, a cinque si sottrae il punteggio assegnato dal partecipante

Quindi si sommano i punteggi così ottenuti e si moltiplica il valore ottenuto per 2,5, ottenendo un punteggio compreso tra 0 e 100, che rappresenta il valore finale del SUS. Studi in letteratura riportano che il valore 70 è quello per cui si può considerare accettabile l’usabilità complessiva del sistema valutato.

### **2.1.7 Altre tecniche di raccolta dei requisiti**

Altre tecniche da applicare nella fase di elicitazione dei requisiti sono:

- Raccogliere suggerimenti spontanei degli utenti i quali forniscono informazioni utili per una corretta evoluzione del prodotto. Un esempio potrebbe essere un forum di discussione.
- Applicare una scrupolosa analisi della concorrenza con fine di:
  1. Individuare le “pratiche migliori” del settore
  2. individuare i punti di forza e di debolezza dei prodotti concorrenti
  3. Caratterizzare il nostro prodotto in rapporto ad essi enfatizzando che cosa lo contraddistingue e cosa gli dà valore.

## Capitolo 3

# Leggi della Gestalt

La *psicologia della Gestalt*, detta anche psicologia della forma, è una corrente psicologica che si sviluppò nei primi decenni del 1900. In tedesco “gestalt” significa “figura”, “forma”, è qualcosa che percepiamo come unità, come oggetto singolo. L’idea portante della psicologia della Gestalt è che non è corretto dividere l’esperienza umana nelle sue componenti elementari, da analizzare separatamente, perché *un insieme è più della somma delle parti*. In particolare, questo avviene nella percezione visiva: gli elementi che si presentano nel campo visivo interagiscono fra loro in modo complesso, e noi percepiamo qualcosa che è sostanzialmente diverso dalla loro semplice somma. Gli psicologi della Gestalt hanno cercato di individuare le leggi elementari che governano questi fenomeni. A tal proposito, nel 1923, Max Wertheimer descrisse le *leggi dell’organizzazione figurale*, descritte come segue:

1. *Legge della vicinanza*: a parità di tutte le altre condizioni, gli elementi del campo visivo che sono fra loro più vicini tendono a essere raccolti in unità.
2. *Legge della somiglianza*: a parità di tutte le altre condizioni, gli elementi del campo visivo che sono tra loro simili tendono a essere raccolti in unità.
3. *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 (la chiusura è più forte della vicinanza).
4. *Legge della continuità di direzione*: a parità di tutte le altre condizioni, le linee che vanno nella stessa direzione si costituiscono in unità più facilmente delle altre.
5. *Legge della buona forma*: a parità di tutte le altre condizioni, il campo percettivo si segmenta in modo che risultino entità per quanto possibile equilibrate, armoniche e costituite secondo un medesimo principio in tutte le loro parti.

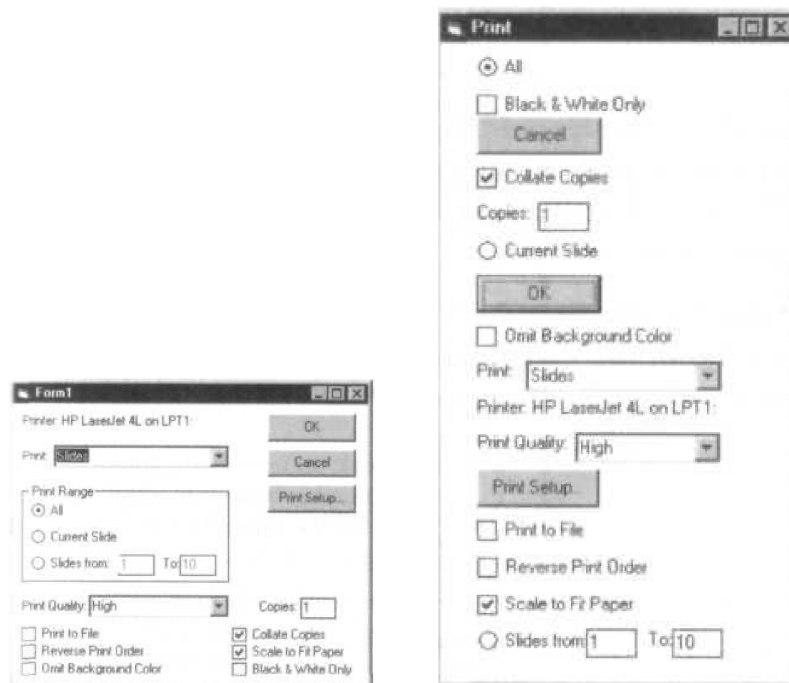


Figura 3.1: Box di dialogo con e senza gestalt.

6. *Legge dell'esperienza passata:* a parità di tutte le altre condizioni, gli elementi del campo visivo che danno origine a una figura familiare o dotata di significato tendono a formare un'unità.

Il principio della vicinanza della Gestalt ci dice che elementi vicini verranno percepiti come appartenenti a uno stesso gruppo. Questo ci suggerisce di porre uno vicino all'altro gli elementi grafici che, dal punto di vista funzionale o semantico, sono fra loro correlati. Ci dice anche di tenere lontani elementi che non hanno alcun rapporto. In questo modo il progettista sfrutta la legge della vicinanza a proprio vantaggio.

Possiamo utilizzare a nostro vantaggio anche la legge della somiglianza dando forma o colore simili a quegli elementi grafici che sono funzionalmente o semanticamente correlati.

In accordo con la legge della chiusura, invece, una tecnica molto efficace per associare visivamente più elementi consiste nel riunirli all'interno di una cornice chiusa.

Un esempio di applicazione delle leggi della Gestalt è fornito dalla Figura 3.1 in cui possiamo notare che il box a sinistra consiste di diverse e ben distinte gestalt formate sulla base delle leggi di chiusura, vicinanza e somiglianza. Il box di dialogo a destra, al contrario, presenta un layout disordinato e poco chiaro.

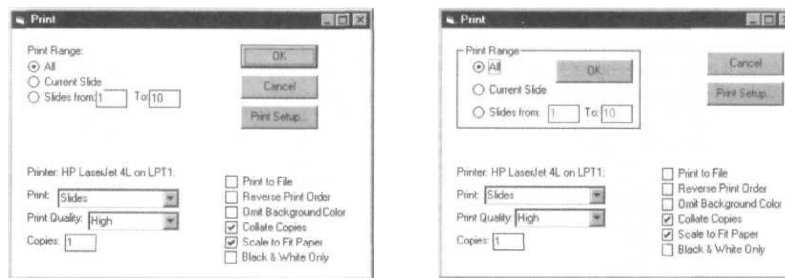


Figura 3.2: Due ulteriori esempi sulla Gestalt

Il box di dialogo a sinistra nella Figura 3.2 presenta quattro blocchi di “controlli” che creano quattro forti gestalt. La legge della vicinanza è la forza maggiore. In ogni blocco vi sono gestalt più piccole che appaiono simili tra loro, e la legge della somiglianza contribuisce a rafforzare ulteriormente le gestalt. Nel box a destra, al contrario, la cornice che racchiude i radio button rafforza la gestalt. La presenza del tasto “OK” all’interno di tale cornice però porta ad una gran confusione. L’utente, infatti, potrebbe pensare che tale tasto “OK” si riferisca ai radio button piuttosto che alla conferma delle impostazioni generali, nonostante sia lo stesso tasto della schermata di sinistra semplicemente spostato. Il motivo di tale confusione risiede nella legge della chiusura secondo la quale l’utente percepisce come una unità tutto ciò che si trova all’interno di una linea chiusa. In altre parole è come se il tasto “OK” funzionasse solo per il print range. Il box a destra, quindi, mostra che le gestalt possono avere un forte influenza sui nostri modelli mentali circa il funzionamento di un sistema.

## Capitolo 4

# WIMP

Lo stile di una interfaccia influenza il dialogo tra utente e sistema. Gli stili di interazione sono molteplici e se ne elencano alcuni: interfaccia a comandi, menu, linguaggio naturale, domanda e risposta, WIMP (Window Icon Menu Pointer), interfaccia multimediale, realtà virtuale, ...

### 4.1 Interfacce visuali

Le interfacce visuali sono definite *formalismi visuali* (VF) in quanto:

- Visuali deriva dal fatto che essi sono generati, compresi e comunicati da umani
- Formali deriva dal fatto che sono manipolati, mantenuti e analizzati da computer.

Il vantaggio dell'applicazione di formalismi visuali è derivato dallo sfruttamento del canale visivo, dall'utilizzo, quindi, dell'abilità umana a percepire relazioni spaziali e ad inferire da esse struttura e significato. I VF mostrano grosse quantità di dati in poco spazio e forniscono informazione semantica non ambigua sulle relazioni tra dati.

I formalismi visuali permettono inoltre una visibilità degli oggetti che consente di effettuare azioni fisiche incrementali e reversibili con risultati immediatamente visibili (click di bottoni).

### 4.2 Metafore

Per sfruttare la conoscenza dell'utente di situazioni tipiche della vita reale e applicarle alle interfacce visuali si utilizzano le *metafore*.

Una metafora è una figura retorica la cui essenza è capire e sperimentare una cosa in termini di un'altra.

L'utilizzo di metafore permette di passare da concetti familiari a concetti sconosciuti, ovvero introdurre un concetto, il *target*, rappresentandolo tramite un concetto *sorgente* ( $S \rightarrow T$ ).

### 4.3 Interfacce WIMP

Una interfaccia WIMP si compone di:

- Window: un'area dello schermo che si comporta come un terminale indipendente.
- Icona: immagine stilizzata e segmentata che rappresenta oggetti del mondo reale, ma anche concetti astratti, azioni, processi
- Menù: presenta una scelta di operazioni o servizi che possono essere effettuate dal sistema a un dato momento
- Puntatore (mouse, joystick, trackball): un cursore sullo schermo controllato da un dispositivo. Le forme del cursore offrono un feedback all'utente.

Nello specifico, un'icona è solitamente una immagine minimale e segmentata (o decontestualizzata), in quanto, prendendo come esempio l'icona di una casa, essa rappresenta una singola casa, non si trova all'interno di una città). Dal momento che un'icona può rappresentare anche concetti astratti che non trovano un riscontro nel mondo reale, trovare un'icona che sia significativa potrebbe essere problematico. A tale scopo si potrebbe ricorrere all'applicazione di convenzioni (simbolo di radiazioni) oppure all'applicazione di didascalie.

### 4.4 Metafora della scrivania

Anche se gli oggetti grafici da manipolare possono essere di qualsiasi tipo, il paradigma della manipolazione diretta è stato principalmente utilizzato per la realizzazione di sistemi basati sulla *metafora della scrivania*. Questo, inventato dai progettisti del PARC di Xerox, apparve per la prima volta sullo Xenon Star, con tutti gli elementi base dei sistemi odierni: finestre (window), menu e icone che rappresentavano documenti e cartelle. Tale metafora ha avuto un successo eccezionale tanto portando aziende come Apple e Microsoft ad adottarla nei propri sistemi. Il fatto che tale metafora sia ancora applicata, sopravvivendo a 40 anni di evoluzione, ne evidenzia la potenza.



## Capitolo 5

# Modello di Interazione di Norman

### 5.1 Un modello dell'interazione

Il modello di Norman descrive l'interazione con un sistema in termini di obiettivi e azioni dell'utente. Nello specifico

- Prima l'utente formula un piano d'azione e lo esegue mediante l'interfaccia
- Dopo aver eseguito un'azione, osserva l'interfaccia per valutare il risultato dell'azione eseguita e determinare le azioni successive.

Come si evince dalla Figura 5.1, queste due fasi possono essere ulteriormente suddivise nelle seguenti sette fasi:

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.

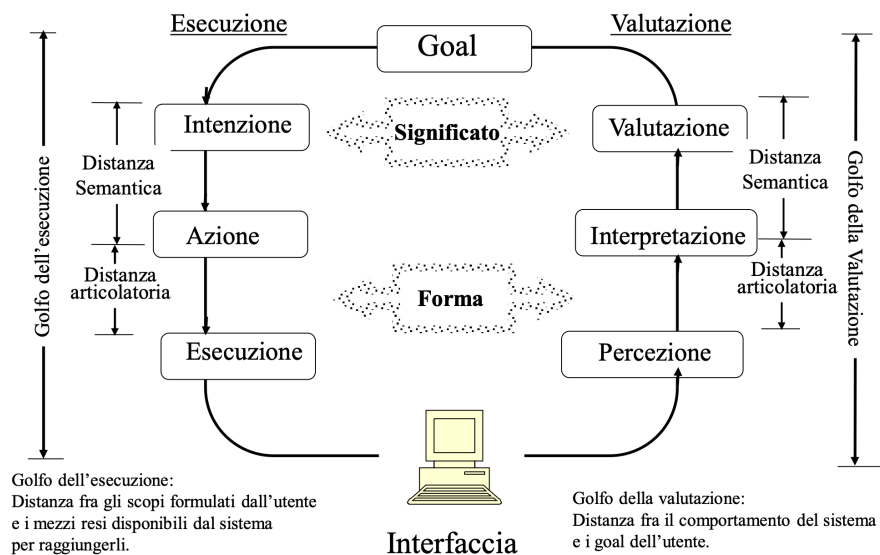


Figura 5.1: Il modello di Norman

È importante osservare che il modello permette di individuare con grande chiarezza i momenti in cui possono presentarsi dei problemi. Nel percorrere i sette stati dell'azione, infatti, è possibile che s'incontrino delle difficoltà nell'attraversare i *golfi* che li separano. In particolare, ci sono due golfi che possono essere particolarmente ardui da superare:

- il golfo dell'esecuzione, che separa lo stadio delle intenzioni da quello delle azioni, e
- il golfo della valutazione, che separa lo stadio della percezione dello stato del mondo da quello della valutazione dei risultati.

In altre parole, il golfo dell'esecuzione è quello che separa le intenzioni dalle azioni che permettono di realizzarle: per superarlo, dovrò identificare fra le azioni che è possibile eseguire con il sistema quelle che mi permetteranno di raggiungere lo scopo.

Il golfo della valutazione, invece, è legato agli ostacoli che l'utente deve superare per interpretare lo stato fisico del sistema dopo le azioni effettuate, e comprendere se ha raggiunto o meno lo scopo prefisso.

## Capitolo 6

# Cognitive Walkthrough

Un *cognitive walkthrough* è un metodo di ispezione di usabilità utilizzato per la valutazione dell'apprendibilità di un sistema da parte di un utente nuovo. Vede le sue origini nel *code walkthrough* dell'ingegneria del software, ovvero una tecnica in cui i revisori esaminano il codice per controllare la presenza di determinate caratteristiche. Questa tecnica a sua volta si basa sulla teoria dell'*exploraty learning* secondo la quale gli utenti, nel processo di soluzione di problemi, non amano consultare i manuali, ma preferiscono esplorare le funzionalità del sistema secondo il pattern seguente:

1. Iniziano con una grossolana descrizione del compito che vogliono effettuare
2. Esplorano l'interfaccia e scelgono le azioni che ritengono possano effettuare il compito (o una parte di esso)
3. Osservano le reazioni dell'interfaccia per vedere se le loro azioni hanno avuto l'effetto desiderato
4. Determinano quali azioni effettuare successivamente.

Alla luce di quanto appena riportato, si può facilmente osservare una forte analogia con lo schema descritto dal modello di Norman, con la differenza che questa volta l'interazione non avviene da parte dell'utente, ma è l'esperto (o gli esperti) che interagisce con il sistema. infatti, diversamente da quanto avviene per i test di usabilità, il CW non coinvolge gli utenti. Questo, così come accade per la valutazione euristica, rende tale procedura più economica da eseguire e si basa sull'esperienza di un insieme di esperti di usabilità che, in maniera strutturata e sistematica, eseguono un determinato task e ne valutano l'esecuzione dal punto di vista di un nuovo utente.

Nello specifico, per effettuare un CW c'è bisogno di:

1. Una descrizione del prototipo del sistema
2. Una descrizione del compito che l'utente deve eseguire sul sistema

3. Una lista scritta completa delle azioni necessarie per completare il task con il prototipo
4. Un'indicazione di chi sono gli utenti e che tipo di esperienza e conoscenze il valutatore può assumere che abbiano.

Al fine di applicare una metodologia sistematica è bene che l'esperto (o gli esperti) tenga a mente alcune domande guida per le quali cercare una risposta ad ogni azione. Le domande, presentano anch'esse una forte analogia con il modello di Norman e sono le seguenti:

1. L'utente, in base alle sue esperienze e conoscenze, è in grado di individuare con facilità l'azione corretta da eseguire (intention-action match)?
2. L'utente è in grado di notare che l'azione corretta è disponibile?
3. L'utente ottiene un feedback nello stesso punto in cui ha eseguito l'azione?
4. Può interpretare correttamente la risposta del sistema, cioè capire di aver scelto l'azione giusta o quella sbagliata?
5. Può valutare i risultati in modo appropriato, cioè determinare se questi soddisfano il suo obiettivo iniziale?

Tutte queste riposte permettono all'esperto di stilare una documentazione risultante ben più verbosa e completa di quanto non accada per la valutazione euristica, ma allo stesso tempo una tale complessità può essere applicata solo ai task principali di un sistema, facendo preferire una più snella valutazione euristica per analisi più sommarie (ovvero che considerino un numero più ampio di task).

## Capitolo 7

# Accessibilità

### 7.1 Definizione di accessibilità

Tim Berners-Lee, creatore di Internet, afferma che il potere del Web sia nella sua universalità. Garantirne l'accesso a chiunque, a prescindere dalla sua disabilità, è un aspetto fondamentale.

Garantire un accesso universale significa rimuovere tutte quelle barriere che impediscono ad un utente diversamente abile di fruire dei contenuti offerti dal sistema. Per farlo è necessario realizzare servizi e siti web che considerino tutte le esigenze di tutti gli utenti indipendentemente da caratteristiche fisiche o cognitive.

Per esempio, un non vedente non è in grado di interagire con un sistema informatico che gli comunichi le informazioni soltanto attraverso il canale visivo.

L'importanza del tema è data anche dal numero di persone affette da qualche tipo di disabilità che si attesta essere circa il 5% dell'intera popolazione italiana. Questo ha portato la politica a focalizzarsi su questo tema e regolamentare l'accessibilità ai sistemi informatici con apposite leggi come la legge Stanca (art.2 legge 4/2004). questa legge si propone di abbattere le barriere che limitano l'accesso dei disabili alla società dell'informazione. Essa definisce l'accessibilità come:

La capacità dei sistemi informatici, ivi inclusi i siti web e le applicazioni informatiche, 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.

Segue la definizione di accessibilità del glossario WAI:

Un contenuto è accessibile quando può essere usato da qualcuno con disabilità.

Infine segue la definizione di accessibilità fornita dal W3C:

L'accessibilità indica la capacità di un sito web di essere acceduto efficacemente da utenti diversi in differenti contesti, qualunque sia il loro hardware, software, lingua, cultura, posizione, o la capacità fisica o mentale.

## 7.2 Tecnologie assistive

Per *tecnologie assistive* la legge intende

gli strumenti e le soluzioni tecniche, hardware e software, che permettono alla persona diversamente abile, superando o riducendo le condizioni di svantaggio, di accedere alle informazioni ai servizi erogati dai sistemi informatici.

Esempi di tecnologie assistive sono per esempio i lettori di schermo, le tastiere braille, sistemi di tracciamento degli occhi, ecc.

A volte si usa anche il termine di *accessibilità universale* per enfatizzare ulteriormente un'accessibilità estesa a tutti i possibili utenti, indipendentemente dalle eventuali ulteriori barriere costituite dalla loro classe sociale, lingua, etnia, cultura, collocazione geografica o altro.

Non bisogna, però, confondere usabilità con accessibilità: sono due concetti diversi, come si comprende facilmente dalla loro definizione. L'accessibilità garantisce la possibilità di accesso al sistema, mentre l'usabilità ne garantisce un uso efficace, efficiente e soddisfacente. Quindi un sistema può essere accessibile, ma non usabile. Per esempio un non vedente potrebbe riuscire a conoscere i contenuti di una pagina web mediante l'uso di un lettore di schermo, anche se questa non fosse strutturata in modo ottimale a tale scopo. Inoltre l'usabilità è un concetto relativo a specifici utenti, compiti e contesti d'uso. Il termine accessibilità ha, invece, un significato assoluto: un sistema accessibile lo è per tutti.

## 7.3 Web Content Accessibility Guidelines (WCAG)

Nell'ottobre del 1997 il W3C, che, a oggi, è l'ente che più si è impegnato nell'affrontare la problematica dell'accessibilità, lancia la Web Accessible Initiative (WAI) il cui scopo è definire il modo in cui rendere i contenuti dei siti web accessibili agli utenti con disabilità. Per raggiungere tale obiettivo, la WAI propone le 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.

I principi alla base delle WCAG sono quattro e sono da considerare i fondamenti dell'accessibilità. Se anche solo uno di questi principi non viene rispettato, gli utenti con disabilità non saranno assolutamente in grado di utilizzare il Web:

1. *Percepibile*: le informazioni e i componenti dell'interfaccia utente devono essere presentati agli utenti in modi in cui essi possano percepirli.

2. *Utilizzabile*: gli elementi dell'interfaccia utente e la navigazione all'interno del sito devono essere facilmente utilizzabili e non devono richiedere agli utenti azioni che qualcuno di loro potrebbe non essere in grado di eseguire.
3. *Comprensibile*: le informazioni e il funzionamento degli elementi dell'interfaccia utente devono essere comprensibili da tutti gli utenti senza alcuna difficoltà.
4. *Robusto*: il contenuto deve essere sufficientemente robusto per essere interpretato in modo affidabile dalla maggior parte dei programmi utente, comprese le tecnologie assistive.

## Capitolo 8

# Programmazione per il Web

### 8.1 Premessa

Il capitolo seguente consiste in una infarinatura molto sintetica sulla programmazione per il web. Ai fini dell'esame, è preferibile fare un minimo di pratica, ad esempio svolgendo gli esempi proposti sulla piattaforma ADA oppure facilmente reperibili sul web.

### 8.2 HTML

L'HTML (acronimo di HyperText Markup Language) è il linguaggio utilizzato per la creazione dei siti Web. Dall'ottobre 2014 è stata rilasciata la versione 5, concepita per definire standard funzionali (es. Video/Audio player) e API. Inoltre HTML5 è progettato per migliorare la SEO rispetto alle versioni precedenti di HTML. È in continua fase di evoluzione.

#### 8.2.1 Vantaggi dell'HTML

1. Interattività senza l'ausilio di plugin esterni (Flash, ...) valida sia per le animazioni grafiche, sia per video streaming e music player
2. Pulizia del codice importante per una facilità di lettura da parte dei motori di ricerca e leggerezza delle pagine
3. Semantica intuitiva grazie ai nuovi TAG di formattazione dove poter etichettare quella determinata porzione di sito in header, nav, ...
4. Form efficaci per la comunicazione con l'utente
5. Compatibilità con tutti i browser
6. Cache non in linea per poter usufruire del sito web anche in modalità offline



7. Abbandono dei cookie sostituiti dal Session Storage e LocalStorage per una velocità di memorizzazione e apertura sito ancora più performante
8. La geolocalizzazione applicabile in ogni parte del sito, dalla compilazione automatica per la provenienza del visitatore al tracciamento per i tablet
9. SEO con i Microdata per inserire nei motori di ricerca molte più informazioni oltre che al classico title e description, che portano grande visibilità
10. Fogli di stile moderni che, grazie al CSS3, non avranno bisogno di un supporto esterno per arrotondare gli angoli dei pulsanti, creare ombreggiature, ...

### 8.2.2 Semantica dell'HTML

La struttura di una pagina HTML è formata da una serie di tag che dividono la pagina in aree logiche. Nei tag `<head></head>` ci sono i tag META e non contiene nulla di visibile su pagina, è solo intestazione. Può contenere il titolo della pagina che compare sulla tab del browser e vi si possono inserire anche i tag *keyword* e *description* per SEO (Search Engine Optimization).

#### Tag principali

Il tag `<article>` specifica un contenuto indipendente dal resto del sito. Il tag `<header>` serve per l'intestazione di un documento e ve essere molteplici all'interno dello stesso documento. Il tag `<body>` definisce il corpo di un documento e può contenere testo, hyperlink, immagini, tabelle, ...

```
<html>
  <head>
    <title>Titolo del documento</title>
  </head>
  <body>The content of the document ...</body>
</html>
```

Il tag `<section>` rappresenta una sezione generica di un documento. Individua un raggruppamento tematico di contenuti solitamente introdotto da un titolo.

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature is ... </p>
</section>
```

Il tag `<aside>` va usato quando si inseriscono info aggiuntive a un elemento. Il tag `<footer>` serve per i piè di pagina e possono essere molteplici in una pagina.

### 8.2.3 Contenuti della pagina HTML5

Il tag `<figure>` serve per inserire immagini, diagrammi, ... mentre il tag `<figcaption>` aggiunge la didascalia all'immagine in `<figure>`. Attributi importanti sono *title* per aggiungere un titolo in modalità mouseover e *width* e *height* per fissare la larghezza e altezza dell'immagine.

Il tag `<hgroup>` raggruppa solo elementi `h1`, `h2`, ..., `h6` e serve per impostare come titolo l'heading con valore maggiore considerando gli altri come sottotitoli a prescindere dal loro ordine. Il tag `<time>` permette di inserire date in formati comprensibili sia agli utenti che alle macchine. L'attributo *datetime* indica data e ora nel formato ISO.

Il tag `<progress>` indica lo stato di completamento di un'attività, mentre il tag `<meter>` rappresenta una grandezza scalare all'interno di un intervallo noto. Il tag `<mark>` evidenzia una porzione di testo all'interno di un documento. `<ul>` permette di creare una lista non ordinata i cui elementi sono indicati con `<li>`, mentre `<ol>` permette di creare una lista ordinata. Il tag `<button>` crea un pulsante cliccabile sul quale può essere mostrato un testo.

### 8.2.4 Web Form in HTML5

Prima di HTML5 l'attributo *type* dell'elemento `<input>` supportava dieci valori (`text`, `button`, `radio`, ...), la nuova specifica ne supporta 23. Vediamo alcuni dei 23 tipi supportati.

L'input type *search* visualizza la barra di ricerca e ha diverse aspetti estetici a seconda del browser e del sistema operativo. Vi si possono inserire dei placeholder, si può attivare l'autocompletamento e imporre dei vincoli sulla lunghezza della query. L'input type *email* visualizza un campo per l'inserimento di un indirizzo email di cui ne controlla la validità eventualmente visualizzando un errore. Si può scegliere se far inserire più indirizzi nella stessa casella. L'input type *URL* visualizza un campo per l'inserimento di un indirizzo Web. Anche in questo caso avviene un controllo sulla validità con eventuale messaggio di errore. Si può decidere di permettere l'inserimento di più url nella stessa casella. L'input type *tel* visualizza un campo per l'inserimento di un numero di telefono, mentre l'input type *number* visualizza un campo per l'inserimento di valori numerici. Visualizza anche due frecce sulla sinistra che permettono di incrementare e decrementare il valore di uno step che si può stabilire con l'attributo *step*. Gli attributi *max* e *min* impostano il massimo ed il minimo valore che è possibile inserire. Per l'inserimento delle date, prima si ricorreva a JS, ora esistono tag appositi quali *datetime*, *date*, *month*, .... L'input type *color* permette la selezione di un colore e il type *datalist* permette la scelta di un'opzione da un elenco preimpostato.

In generale, per gli input type si possono impostare i seguenti attributi:

- *autocomplete*: di tipo booleano di default a on, il browser lo usa per capire se auto compilare o meno un campo in base alle precedenti compilazioni

- autofocus: di tipo booleano, serve ad impostare il focus su un determinato elemento di un form
- pattern: inserimento di espressioni regolari nei campi
- placeholder: testo ingrigito dentro un campo che funge da suggerimento
- required: rende obbligatoria la compilazione di quel campo
- novalidate: salta tutte le validazioni di quel tag

### 8.2.5 Video e audio in HTML5

HTML5 non necessita di player esterni per la gioia dei colossi come Apple, YouTube, Microsoft, ...

Il tag `<video>` è un tag molto semplice che ricorda quello dell'immagine. Supporta formati OGG, MP4 e WebM. Ripetendo l'elemento `<source>` all'interno del tag `<video>` è possibile associare all'elemento più formati dello stesso video così da poter riprodurre un video in tutti i principali browser. L'attributo *controls* è booleano e attiva la toolbar per la riproduzione video e altre opzioni. L'attributo *poster* punta al percorso di un'immagine usata come anteprima finché non parte il video, altrimenti viene scelto un frame casuale. L'attributo *preload* assume i valori:

- none: il buffer si riempie solo alla pressione del tasto play
- metadata: prima dell'esecuzione vengono recuperate solamente le informazioni essenziali come durata, dimensione, ...
- auto: il browser carica preventivamente il video

Continuando, l'attributo *autoplay* è booleano e, se true, fa partire il video in automatico, *loop*, se true, fa reiniziare il video ogni volta che finisce, *muted*, se true, fa partire il video in mute e gli attributi *width* e *height* fissano le dimensioni del player.

Il tag `<audio>` supporta MP3, Wav e OGG. Per l'elemento `<source>` e gli attributi si fa un ragionamento analogo a quanto detto per `<video>`.

### 8.2.6 Canvas in HTML5

`<canvas>` è stata una delle rivoluzioni in HTML5, si tratta di una tela sulla quale si può disegnare con opportune API, funzioni JS che permettono un rendering dinamico di immagini bitmap. Prevede solo due attributi: `<width>` e `<height>`. Prima di cominciare a disegnare è necessario ottenere un riferimento all'oggetto con il metodo JS *getElementById()* e verificare che il browser lo supporti. Con l'utilizzo della libreria Modernizr si può testare il supporto: `if(Modernizr.canvas){...}`

### 8.2.7 Microdata

Per esaltare la semantica dei contenuti, HTML5 oltre ai tag permette la creazione dei microdati, ovvero etichette che aggiungono informazioni utili a crawler, browser e screen reader per comprendere il significato del contenuto. I microdati sono invisibili all'utente.

Per applicare i microdati si possono specificare, per ogni tag HTML, degli attributi che permettono di definire gli oggetti semantici. Prima di tutto dobbiamo applicare ad un elemento radice:

- *itemscope* per specificare che all'interno dell'elemento che lo contiene sono presenti informazioni dell'oggetto che si vuole descrivere
- *itemtype* per definire il vocabolario che specifica il tipo di oggetto che si vuole descrivere
- *itemprop* per definire la proprietà che verrà avvalorata con il testo contenuto nel tag

Grazie all'utilizzo di vocabolari supportati da Google si possono creare i "rich snippet", ovvero risultati di una ricerca in cui compaiono informazioni allegate.

### 8.2.8 Applicazioni Web offline in HTML5

Con le offline API è possibile specificare in un apposito file detto manifest (con estensione .manifest) un elenco di asset dei quali vogliamo che il browser conservi una copia locale. Tali oggetti dopo una prima navigazione online, resteranno accessibili anche in assenza di connessione.

Per prima cosa bisogna aggiungere l'attributo *manifest* all'elemento html. Il valore di questo attributo deve corrispondere al percorso del file contenente l'elenco dei documenti per i quali si richiede la memorizzazione.

Si distinguono quattro casi di funzionamento:

1. Prima visita: quando il browser visita per la prima volta una pagina con riferimento a manifest allora l'utente non conserva una copia del file, quindi il browser lo scarica, lo interpreta e ne richiede le risorse al server
2. visita successiva senza modifiche del manifest: il browser rileva la presenza del file, controlla eventuali modifiche e termina.
3. Visista successiva con modifiche al manifest: il browser rileva le modifiche e sostituisce il file
4. Mancanza di risorse definite dal manifest: viene scatenato un evento per comunicare l'errore durante l'esecuzione

### 8.2.9 Local storage in HTML5

Le WebStorage API permettono il salvataggio locale di informazioni concernenti la navigazione Web. Tutti i browser supportano tali API sebbene trovino resistenza nel sostituire i cookie. Questa tecnologia risolve l'impossibilità di avere due sessioni distinte di un browser relative allo stesso dominio. Il metodo *sessionStorage()* consente di avere un meccanismo di persistenza dei dati distinto per ogni sessione di navigazione in ogni finestra del browser eliminando i dati alla chiusura. Questo permette di gestire due account gmail contemporaneamente sullo stesso browser. Il metodo *localStorage()* permette la persistenza dei dati anche dopo la chiusura del browser, si pensi agli articoli lasciati nel carrello e ritrovati sempre nel carrello a distanza di giorni.

Per entrambi i metodi la memorizzazione è consentita solo per valori testuali. Per i valori numerici si applicano *parse\_Int()* e *parse\_Float()* per passare da una stringa al rispettivo numero.

Per la memorizzazione di oggetti si utilizzano i JSON, dove *JSON.stringify()* permette di ottenere una rappresentazione stringa di un oggetto e (*JSON.parse()*) è l'inversa.

### 8.2.10 Geolocalizzazione in HTML5

Le Geolocalization API non fanno parte della specifica di HTML5, anche se sono associati a quest'ultimo. I metodi più importanti sono *getCurrentPosition* e *watchPosition* per ottenere la posizione corrente. La prima funzione rileva la posizione una sola volta mentre la seconda comunica la posizione ogni qualvolta vi sia una variazione importante della posizione. Queste funzioni accettano tre argomenti:

- *successCallback*: parametro facoltativo che rappresenta la funzione da eseguire nel caso in cui il rilevamento della posizione è andato a buon fine
- *errorCallback*: parametro facoltativo che rappresenta la funzione da eseguire nel caso in cui il rilevamento della posizione sia fallito
- *options* permette di intervenire nel modo in cui avviene il tracciamento

## 8.3 CSS 3

Il *CSS* (Cascading Style Sheets, in italiano fogli di stile) è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML ad esempio in siti web. L'introduzione del CSS ha consentito una efficiente separazione dei contenuti dalla formattazione, permettendo una programmazione più chiara e facile da utilizzare.

I fogli di stile forniscono un supporto al linguaggio HTML per quanto riguarda la visualizzazione delle pagine. La nuova specifica CSS 3, rispetto alle

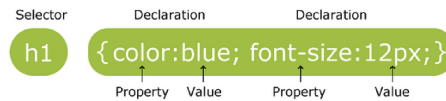


Figura 8.1

precedenti versioni, adotta un approccio modulare. Ciascun modulo quindi copre una determinata area, di conseguenza il ciclo di vita e l'avanzamento è diverso per ognuno di essi.

La sintassi per CSS 3 prevede una *selector* che punta all'elemento html che si vuole stilizzare e uno o più *declaration box*, ognuno dei quali è composto da una coppia proprietà-valore. Ogni coppia è separata da un punto e virgola.

### 8.3.1 I selettori

In CSS 3 si distinguono tre tipi di selettori: di elemento (p), di id (#para1) e di classe (.center) e possono essere utilizzati in combinazione sia per selezionare elementi e classi contemporaneamente (come nel primo esempio), sia per poter raggruppare stili (come nel secondo esempio).

Esempi:

```
p.center
{
  text-align: center;
  color: red;
}
```

oppure

```
h1,h2,p
{
  text-align: center;
  color: red;
}
```

### 8.3.2 Impostazioni stilistiche

Impostando la proprietà font-family, il browser cercherà di utilizzare il primo font. Nel caso in cui non sia disponibile, proverà con il secondo. Esempio: p:font-family: Arial, Verdana;

La proprietà text-shadow permette di ottenere effetti di ombreggiatura nel testo delle pagine web senza dover ricorrere ad immagini apposite. I valori da assegnare a questa proprietà sono quattro:

1. lo scostamento lungo l'asse x del testo ombreggiato
2. lo scostamento lungo l'asse y del testo ombreggiato

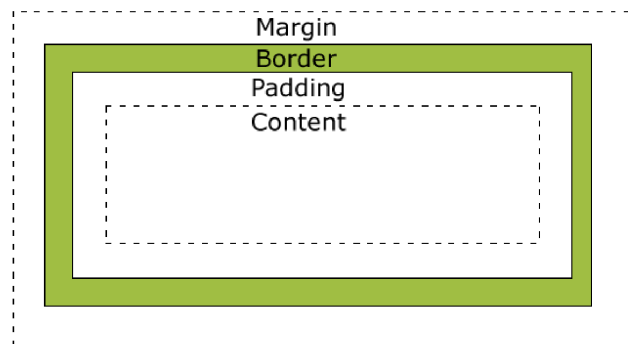


Figura 8.2

3. la sfumatura del testo ombreggiato tramite un raggio di sfumatura espresso in pixel
4. il colore del testo ombreggiato

Nel caso in cui ci sia del testo, posto all'interno di un contenitore non sufficientemente grande per contenerlo, si può applicare la proprietà `text-overflow` con valore `ellipsis` in modo tale da troncare il testo aggiungendo i tre punti di sospensione.

### 8.3.3 Box Model

Tutti gli elementi all'interno dei tag possono essere considerati *box*. Ogni box è caratterizzato da quattro proprietà:

- **Margin:** area di separazione tra il box e l'esterno. È trasparente
- **Border:** bordo attorno al contenuto
- **Padding:** area tra il bordo e il contenuto
- **Content:** box all'interno del quale c'è il testo, l'immagine o altro

Gli elementi per default sono statici seguendo, cioè, il normale flusso della pagina. Si può impostare in maniera fissa una posizione rispetto al browser, alterando il flusso naturale usando l'attributo `position:fixed`. È anche possibile spostare l'elemento relativamente alla sua posizione normale utilizzando l'attributo `position:relative;left:-20px;`. Con la posizione assoluta, invece, si sposta l'elemento relativamente al suo primo genitore che abbia una posizione diversa da static (se non ha genitori, il genitore diventa il blocco html).

Si possono applicare trasformazioni usando i metodi: `translate()`, `rotate()`, `scale()`, `skew()`, `matrix()`.

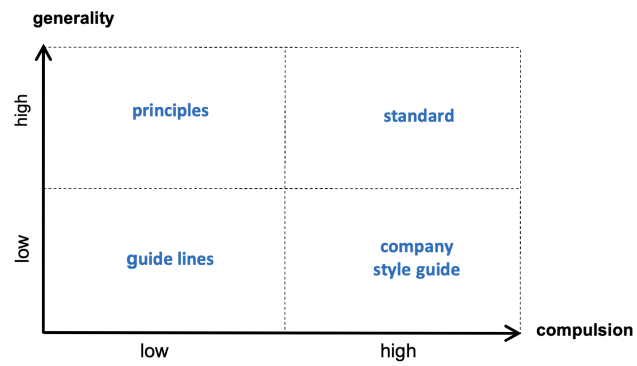


Figura 8.3

## 8.4 Utili per l'esame

C'è sempre una domanda a risposta multipla riguardo questo schema: