

Appunti HCI (Interazione Persona- Calcolatore)

1. execution-evaluation cycle: modello interazione di Norman:

Norman definisce un modello che suddivide in sette stati il punto di vista dell'utente:

- L'utente stabilisce l'obiettivo;
- formula le intenzioni;
- specifica l'azione attraverso l'interfaccia;
- esegue l'azione;
- percepisce lo stato del sistema (feedback);
- interpreta lo stato del sistema;
- valuta la risorsa del sistema rispetto il suo obiettivo.

2. HTA:

sta per Hierarchy Task Analysis, ovvero Analisi dei task gerarchica. Questo approccio è utilizzato per studiare il comportamento degli utenti, le azioni che compiono per raggiungere gli obiettivi, gli oggetti che vengono coinvolti nelle loro azioni. L'HTA esprime l'interazione tra utente e sistema da un punto di vista esterno (al contrario del GOMS che si mette a livello dell'utente), decompone le azioni in azioni sempre più piccole, fino a quando non arriva alla azione atomica che non può più essere decomposta, ovviamente ci si può fermare anche prima, il livello di dettaglio deve essere soggettivo al progettista. Fatto ciò si specificano i piani di esecuzione dei task, ovvero l'ordine con cui devono essere eseguiti i microtask, se il task non è decomposto non serve il piano. Quindi più precisamente l'HTA prevede:

- produzione lista dei task;
- raggruppamento dei task in task di livello più alto;
- scomposizione dei task in task di livello più basso;
- fermarsi al livello di dettaglio desiderato;
- creare i piani di esecuzione dei task, specificando l'ordine di esecuzione.

3. Esperimenti controllati:

è un metodo empirico per la valutazione con gli utenti. Questo viene fatto in laboratorio e deve essere impostato in modo preciso per non invalidare i risultati dell'analisi. I dati raccolti verranno poi osservati tramite l'uso di tools statistici.

Gli esperimenti devono essere ripetibili, in modo da poter effettuare lo stesso test più volte e confrontare i diversi risultati. Per effettuare un esperimento controllato bisogna scegliere diversi parametri, ovvero:

- soggetti che effettueranno il test;
- scelta delle variabili dipendenti e indipendenti;
- scelta dell'ipotesi da convalidare con il test;

La scelta dei soggetti che dovranno effettuare l'esperimento è caratterizzata da molti fattori, ovvero quanti soggetti dovremo coinvolgere, in base al budget e a quanto preciso deve essere il risultato, inoltre bisognerà rendere omogeneo il gruppo di utenti scelti per non falsare i dati ed infine bisogna scegliere la classe dei soggetti in base al contesto in cui verrà utilizzata l'applicazione.

Le variabili indipendenti sono le variabili del sistema che andremo a cambiare (grandezza del carattere, colore ecc..), mentre quelle dipendenti sono le variabili che andremo ad osservare e misurare e che hanno valori che cambiano in base alle variabili indipendenti (tempi di reazione dell'utente, difficoltà, ecc...), infine l'ipotesi è il risultato che ci attenderemo dall'esperimento e che quindi dovremo validare.

Per validare questa ipotesi quindi dobbiamo confutare l'ipotesi nulla che affermerà che le variabili dipendenti non sono legate a quelle indipendenti in nessun modo. Ovviamente si dovrà validare l'esperimento progettato con un pre-test.

In particolare esistono 2 tipologie di test:

- Within groups: ogni gruppo effettua test differenti in modo da capire qual'è il migliore. Ovviamente i gruppi dovranno essere omogenei tra di loro e c'è il vantaggio che sono necessari pochi soggetti e lo svantaggio che i soggetti dopo il secondo test saranno più veloci nell'effettuare le azioni (effetto learning).
- Between groups: i soggetti effettuano solamente un test con determinate variabili. Il vantaggio è che si elimina l'effetto learning, lo svantaggio che richiede un maggior numero di utenti, oltre che in questo caso i gruppi dovranno essere ASSOLUTAMENTE omogenei.

Solitamente alla fine dell'esperimento si somministra un questionario in modo da misurare anche la soddisfazione. Raccolti i dati si passa all'analisi attraverso metodi statistici quali il t-test e l'ANOVA.

Con il t-test fissiamo un'ipotesi nulla e una non nulla, e tramite i dati inseriti verifichiamo la probabilità che l'ipotesi che abbiamo fatto sia errata. Questo metodo però consente di confrontare solamente 2 raccolte di valori, mentre se l'esperimento restituisce più di due raccolte di dati da confrontare si utilizza il metodo

ANOVA a sostituzione dei t-test incrociati sulle raccolte a due a due, che abbasserebbero il grado di accuratezza. Il metodo ANOVA però restituisce se l'ipotesi nulla è vera o falsa ma non specifica quale dei confronti falsifica la verifica. Per saperlo bisogna usare il t-test sulle varie raccolte.

4. Metafore:

Sono molto usate nel HCI e riportano gli oggetti che vengono coinvolti durante il lavoro dell'utente nell'applicazione in modo da rendere più semplice ed intuitivo il suo uso. La metafora più famosa è il Desktop delle interfacce grafiche, che riporta la scrivania di un impiegato con tutti i suoi oggetti. I primi a ideare questo tipo di metafora furono i programmatori della XEROX che idearono la prima interfaccia WIMP (windows Icons Menu Pointers), purtroppo per loro l'idea venne copiata da Steve Jobs per il suo Macintosh e successivamente da Bill Gates con Windows.

5. Architettura sistema a finestre:

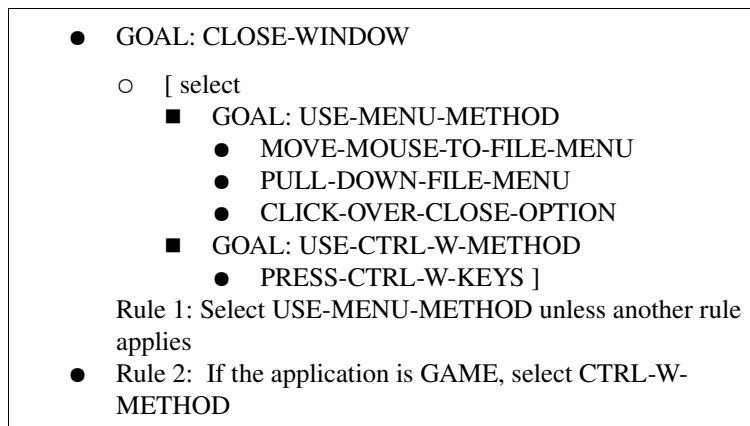
astrae le specifiche hardware dei dispositivi di input e output ai programmatori. Quindi le applicazioni sono separate dai device driver che gestiscono le periferiche. I sistemi a finestre quindi si dividono in client e server, i primi sono le applicazioni, i secondi invece fanno da tramite tra periferiche e applicazioni. Il programmatore crea un terminale astratto e si occupa di far dialogare questo con la propria applicazione. Il terminale astratto è quindi la finestra che viene mostrata all'utente e che interagisce attivamente con esso. Il sistema a finestre per permettere all'utente il dialogo non gestisce solamente i dispositivi di output, ma gestisce anche quelli di input, attraverso particolari segnali. Ovvero nel server è presente un gestore delle finestre a cui il programmatore delega tutti i compiti necessari per ricevere dai dispositivi di input e mostrare su quelli di output. Gli eventi dei dispositivi di input, particolari operazioni (movimento del mouse, click su un pulsante ecc..) che questi dispositivi svolgono, vengono associati dal programmatore a oggetti presenti nell'interfaccia a finestre. Esistono due paradigmi di programmazione per la gestione degli eventi: Read-evaluation Loop e Notification Based. Con il Read-Evaluation Loop il server invia all'applicazione l'evento letto da input e l'applicazione si preoccupa di leggere continuamente se l'evento si è verificato o meno, con il Notification Based invece il codice che aspetta e processa l'evento non risiede nell'applicazione, ma nel server stesso che avvisa l'applicazione dei soli eventi che in precedenza gli ha definito come interessanti.

6. GOMS:

è un altro metodo per studiare come l'utente si muove per raggiungere l'obiettivo, ed esprime il punto di vista utente, quindi si concentra sui processi cognitivi interni che l'utente si forma. GOMS sta per:

- Goals: obiettivi che l'utente intende raggiungere;
- Operators: azioni semplici che l'utente compie per raggiungere gli obiettivi;
- Methods: tipi di decomposizione per ogni goal in sottobiettivo e operazioni;
- Selection: selezione fra i diversi tipi di decomposizione.

Esempio:



Quindi il Goms specifica l'obiettivo primario dell'utente e specifica i diversi modi per arrivare a quell'obiettivo, attraverso diverse azioni e specificando le regole con cui quelle azioni vanno eseguite.

7. STN:

è un sistema grafico per la rappresentazione del dialogo tra utente e sistema. Il dialogo generalmente possiede 3 aspetti: lessico, sintattica e semantica. Il diagramma STN (State Transition Network) è composto da un insieme di stati collegati tra di loro tramite archi, le azioni. Ad ogni azione corrisponde un cambiamento di stato del sistema. Il passaggio da uno stato ad un altro è segnalato all'utente tramite feedback che vengono elencati sull'arco stesso, proprio come le azioni compiute dall'utente. Il problema di questo tipo di

rappresentazione è che a fronte di tante scelte, abbiamo bisogno di molti stati, quindi il diagramma si ingrandirebbe moltissimo. Esistono però le H-STN (Hierarchical STN) che risolvono in parte questo problema, si tratta di STN che incorporano in uno stato un altro STN in questo modo, si possono rappresentare gli stati e le transizioni che rappresentano un blocco più piccolo a parte e inserire nel diagramma generale solamente un rettangolo che rappresenta un collegamento a questo. Questo aumenta quindi il valore espressivo dei diagrammi STN.

8. Metodi di valutazione:

I metodi di valutazione utilizzati in HCI sono l'esperimento controllato (discusso in precedenza) e i field studies (studio sul campo). Il field studies è un metodo di valutazione effettuato direttamente nel contesto di utilizzo del prodotto, quindi può essere usato solamente quando si è in possesso di un sistema completo. Utilizzando questo metodo è difficile avere misure precise, proprio perché l'utente si trova nel suo contesto reale e potrebbe essere disturbato. Infatti vengono esaminati solo dati qualitativi. Un altro svantaggio di questo metodo è che il valutatore è visibile all'utente e quindi potrebbe indurre stress nell'utente stesso. I field studies include vari metodi, quali il thinking aloud, il cooperative evaluation ed il post task walkthrough. Il thinking aloud è un metodo molto semplice e poco dispendioso, infatti è chiesto all'utente di usare il software e commentare durante l'utilizzo oppure annotare le operazioni che svolge. Questo metodo permette quindi di valutare qualitativamente il prodotto, ma i valori raccolti sono valori soggettivi che cambiano da utente ad utente, inoltre si induce del disturbo in quanto l'utente non svolge le proprie azioni in libertà, ma deve anche registrare impressioni e critiche (che oltre a rallentare il lavoro non è poi tanto naturale). Il cooperative evaluation è come il thinking aloud, ma il valutatore è presente durante il test e può interagire con l'utente chiarendo alcuni dubbi e facendolo uscire da una fase di stallo, ovviamente la sua presenza introduce un disturbo.

Infine il post talk walkthrough prevede che l'utente esegua in autonomo il test e solo alla fine di questo commenti al valutatore le azioni e le sue impressioni. Questa tecnica ha il vantaggio di poter essere usata in quei task critici in cui sarebbe difficile per l'utente interagire per descrivere le azioni, ma ha lo svantaggio che l'utente potrebbe soffrire di mancanza di freschezza, ovvero potrebbe dimenticare alcuni aspetti importanti. Per quest'ultimo motivo il post task walkthrough viene effettuato immediatamente dopo la fine del test. Ovviamente per scegliere quale metodo utilizzare bisogna valutare le disponibilità che possediamo, se vogliamo risultati qualitativi o quantitativi, e il livello di intrusione che possiamo sopportare.

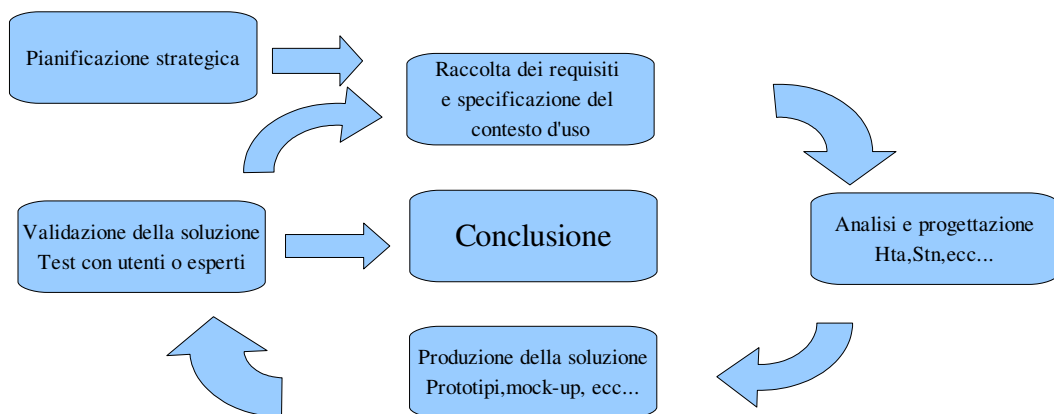
9. Caratteristiche dei link:

I link, parte fondamentale del Web dovrebbero essere chiari e leggibili, quindi si dovrebbe evitare di usare immagini per i link, che spesso nascondono funzionalità più o meno essenziali. I colori dei link sono molto importanti per renderli facilmente individuabili, è preferibile utilizzare colori che risaltano sullo sfondo e che cambiano al passaggio del mouse, o se il link è stato visitato in precedenza fargli assumere un colore diverso da quello originale. Questa funzionalità è spesso ignorata dai Webmaster, ma è una funzionalità che è necessaria in grandi siti che tendono ad avere molta informazione ed in cui l'utente non sa se è già passato per quel link.

10. Slips and mistakes:

sono errori effettuati dall'essere umano. Gli Slips (sviste) sono gli errori meno gravi, ovvero l'utente ha compreso il sistema e i suoi obiettivi, ma sbaglia a metterli in pratica, le cause possono essere la scarsa esperienza, problemi fisici ecc.. I mistakes invece, gli errori veri e propri, sono causati da una errata interpretazione del sistema. L'utente potrebbe incorrere in questo errore in quanto si è creato un modello mentale errato e quindi esiste una distanza tra il modello mentale del progettista e dell'utente. Questo tipo di problema è quindi una conseguenza di una malprogettazione che potrebbe costare tempo e risorse.

11. Ciclo progettazione software secondo i principi dell'HCI:



12. Keystroke-level model:

è una tecnica per studiare e misurare i tempi di reazione dell'utente. Molto spesso è utilizzato nelle applicazioni Time Critical dove l'interazione fra utente e sistema deve essere il più veloce possibile. Il KLM divide la microazione in 3 parti, fase motoria, mentale e del sistema. La fase motoria si divide ulteriormente in microbiobiettivi, come il puntamento, la pressione di un tasto ecc... Il tempo totale impiegato dall'utente per raggiungere l'obiettivo è dato dalla somma di tutti i tempi delle microazioni. Questa tipologia di approccio, non va a sostituire gli altri modelli, ma ad integrarli.

13. Toolbars and Palette:

Le toolbars sono una lunga serie di icone che contengono strumenti o funzionalità. Ovviamente sono dei widgets particolari caratteristici dell'interfaccia a finestre che devono utilizzare icone che rendono chiara la loro funzionalità. Le palette invece sono una tipologia di toolbars, e quindi un insieme di icone che una volta selezionate attivano uno strumento che verrà utilizzato fino a quando non si deselezionerà.

14. Backus-Naur form:

è una notazione molto utilizzata nell'informatica, un linguaggio che descrive le sequenze di azioni ovvero i task. Esprimere un task sotto questa forma aiuta a misurare la complessità dell'azione, cioè quanto è difficile raggiungere l'obiettivo. Questo linguaggio però non tiene conto delle percezioni dell'utente e del modello mentale che esso si forma.

15. Prototipo usa-e-getta, incrementale ed evolutivo:

Dopo aver raccolto i requisiti difficilmente si delineeranno tutti i requisiti necessari agli utenti. L'unico metodo per essere veramente sicuri di aver capito tutti i requisiti o per capirne di altri è far testare ai reali utilizzatori il sistema. Ovviamente in questo momento non avremmo sicuramente un sistema pronto e completo e si dovranno utilizzare i prototipi. Esistono fondamentalmente 3 tipi di prototipi:

- Usa e getta:
il prototipo viene costruito, testato e successivamente abbandonato. Tutte le informazioni che si raccolgono verranno poi usate per la progettazione finale.
- Incrementale:
il prodotto finale è costruito unendo vari componenti diversi. Quindi ogni prototipo viene usato per raffinare il componente che costituirà successivamente il sistema. Le parti comuni ai vari componenti saranno presenti in tutti i prototipi e unite alla fine.
- Evolutivo:
il prototipo non viene gettato, ma bensì viene migliorato e portato avanti. Diventerà infine il progetto finale.

La realizzazione dei prototipi non è semplice ed immediata. Si possono creare prototipi che simulano solamente alcuni aspetti del sistema o prototipi che la includono integralmente. Ma per ogni tipo di approccio, esistono vari fattori da tenere conto:

- Tempo.
- È necessaria l'esperienza per realizzare un corretto piano di sviluppo.
- In alcuni sistemi potrebbero essere necessari requisiti non funzionali (velocità, sicurezza, affidabilità..).
- Lo sviluppo di un sistema è legato ad un contratto e spesso i prototipi non possono formare la base legale per un contratto. Inoltre la velocità essendo presente nel contratto come valore aggiunto andrebbe modificata se si sceglie un approccio di tipo prototipale.

16. Differenze tra Back e Undo:

Per aumentare la usabilità del sistema e tenere conto dei principi o delle linee guida più comuni bisogna inserire nel prodotto finale, sia la verifica degli errori, sia la loro gestione, inoltre è necessario inserire un servizio per l'annullamento dell'operazione appena compiuta in modo da ripristinare il sistema allo stato precedente. Questi strumenti sono proprio il Back e l'Undo, ma tra loro c'è una differenza molto importante, ovvero il back permette di far tornare indietro il sistema, ma le azioni compiute non vengono annullate, così come gli errori commessi. Un esempio attuale di Back si ha nei browser web, quando clicchiamo sul pulsante indietro, questo ci riporta alla pagina precedente, ma non annulla per esempio il form che abbiamo inviato e che ci siamo accorti di aver compilato male. L'undo invece è l'operazione che oltre a riportare il sistema nello stato precedente annulla le operazioni appena compiute, e ci permette di risolvere gli errori o le sviste commesse.

17. Design Rules (Principi, linee guida e standard):

Sono delle regole di progettazione che ci aiutano ad incrementare l'usabilità del prodotto software finale. Esistono diversi tipi di Design Rules, ma principalmente si possono racchiudere in 3 differenti categorie: principi, linee guida e standard. I principi sono regole astratte che non sempre bisogna seguire precisamente ma è comunque consigliato farlo, non dipendono molto dal dominio tecnologico dell'applicazione, gli standard

sono regole di design precise che riguardano precisi campi di applicazione e che dovrebbero essere sempre rispettati, infine le linee guida sono una via di mezzo tra le due, cioè non sempre devono essere rispettati e riguardano una classe più ampia di applicazioni.

I principi si suddividono in 3 macro categorie che a loro volta comprendono altre categorie:

- apprendibilità: facilità di utilizzo del software da parte degli utenti.
 - **Predicibilità:**
indica quanto la conoscenza da parte dell'utente è necessaria per determinare i risultati di future interazioni col sistema.
 - **Sintetizzabilità:**
è la capacità dell'utente di potersi rendere conto degli effetti di azioni passate sullo stato corrente; si parla anche di principio di onestà ovvero della capacità del sistema di mostrare all'utente i cambiamenti di stato che in esso si verificano.
 - **Familiarità:**
misura la relazione che intercorre tra la conoscenza necessaria per utilizzare il sistema e quella posseduta dall'utente.
 - **Generalizzabilità:**
gli utenti spesso cercano di estendere la loro conoscenza a sistemi che non conoscono ma che hanno alcune funzioni simili ad altri.
 - **Consistenza:**
bisogna progettare un sistema consistente, ovvero deve seguire un preciso modello che gli permetta di associare correttamente un output ad un determinato input.
- flessibilità: molteplicità in cui gli utenti scambiano informazione con il sistema;
 - **Iniziativa del dialogo:**
riguarda chi tra l'utente e il sistema prende per primo l'iniziativa ad iniziare un dialogo. Se il dialogo lo inizia il sistema si parla di system pre-emptive, mentre se lo inizia l'utente si parla di user pre-emptive. Se è il sistema che guida il dialogo, l'utente vede il sistema come poco flessibile, mentre se è lui a prendere iniziative e guidare il dialogo sente l'aumento di flessibilità del sistema.
 - **Multi-threading:**
è la capacità del sistema ad eseguire più task contemporaneamente. Questo può farlo in due diversi modi: Concurrent Multi-Threading o Interleaved Multi-Threading. Il primo gestisce in contemporanea diversi flussi informativi in modo da permettere l'esecuzione di più task, con il secondo metodo invece, il sistema porta avanti tutti i task insieme, ma in realtà in un intervallo temporale viene eseguito un singolo task.
 - **Migrabilità del task:**
è la caratteristica che permette di passare il controllo del task dal sistema all'utente, per esempio nei casi di emergenza di sistemi che lavorano in campi critici.
 - **Sostituibilità:**
capacità di rappresentare uno stato o una risposta in diversi modi.
 - **Personalizzazione:**
si intende la capacità del sistema di andare incontro alle esigenze dell'utente. Si può parlare di adattività, ovvero è il sistema che si modifica automaticamente, oppure di adattabilità cioè la personalizzazione è in mano all'utente.
- robustezza: livello di supporto garantito all'utente per raggiungere i propri obiettivi;
 - **osservabilità:**
permette all'utente di osservare lo stato interno del sistema e comprende i seguenti sotto-principi:
 - **navigabilità:**
la possibilità dell'utente di osservare lo stato interno del sistema, attraverso l'uso di strumenti limitati. L'uso di questi strumenti non deve però influire minimamente sul sistema, ovvero dovrebbero essere strumenti passivi.
 - **Defaults:**
potrebbe aiutare l'utente a ricordare, o comunque facilitare l'utilizzo del sistema. Inoltre l'uso di defaults può limitare gli errori.
 - **Raggiungibilità:**
è la possibilità di navigare tra gli stati osservabili del sistema.
 - **Persistenza:**
è la durata dell'effetto di una comunicazione e la possibilità che ha l'utente di utilizzarla.
 - **Recuperabilità:**
dopo aver compiuto un errore gli utenti necessitano di recuperare il sistema, in modo da poter

completare correttamente il loro task e raggiungere così il loro goal.

- Reattività:
misura il tasso di comunicazione tra utente e sistema.
- conformità del task:

Gli standard sono invece principi validi a livello internazionali che cercano di generalizzare la progettazione in modo da rendere il progetto facilmente condivisibile ad una larga comunità di utenti. Gli organi che redigono gli standard sono tanti tra cui troviamo anche l'ISO.

Infine le linee guida sono una raccolta di consigli, note e informazioni utili per lo sviluppo di sistemi interattivi usabili, sono stati redatti molti libri e how-to a questo scopo che riguardano diversi campi di applicazione.

18. Golden Rules:

Sono regole di carattere generale che vanno oltre le linee guida o i principi, scritti da esperti di usabilità. Si possono pensare come una lista da controllare continuamente in fase di progettazione per essere sicuri di non essersi dimenticati di nulla e di aver considerato tutti gli aspetti necessari. Esistono diverse raccolte di golden rules, le più famose sono tuttavia quelle di Shneiderman, Norma e le euristiche di Nielsen.

Partiamo da quelle di Shneiderman:

1. Battersi per la consistenza.
2. Cercare di far usare agli utenti abituali le shortcut.
3. Deve essere sempre presente un feedback informativo.
4. Ogni dialogo deve avere una chiara chiusura.
5. Inserire funzionalità di controllo e prevenzione degli errori, oltre al trattamento semplificato di situazioni di errore.
6. Permettere il facile rovesciamento delle operazioni.
7. Supportare il controllo del sistema.
8. Ridurre il carico cognitivo a breve termine.

Principi di Norman:

1. Usare sia la conoscenza del mondo che quella dell'utente.
2. Rendere semplice la struttura dei task.
3. Rendere le cose facilmente visibili.
4. Creare i giusti mapping tra utente e sistema.
5. Esplorare il potere dei vincoli.
6. Progettare per gli errori.
7. Quando fallisce tutto, standardizzate.

Le euristiche sono sempre linee guida o principi che vengono usate per verificare l'aderenza del sistema a particolari aspetti. Sviluppate da Nielsen, le euristiche possono essere usate in qualsiasi momento dello sviluppo. La dinamica di utilizzo è semplice, bisogna interrogare degli esperti affinché analizzino il prodotto per scoprire vari problemi di usabilità. Quindi i valutatori (devono essere da 3 a 5 persone) esaminano il prodotto utilizzando una o più euristiche e verificando laddove il sistema ha dei problemi il perchè e assegnando un grado di gravità per l'errore commesso. Vengono usati 4 fattori per l'analisi del problema, ovvero:

- Quanto è comune il problema.
- Quanto è facile per l'utente superare il problema.
- È un problema sporadico o permanente.
- Quanto è percepito il problema.

Le 10 euristiche di Nielsen sono le seguenti:

- Visibilità dello stato del sistema.
- Aderenza tra sistema e mondo reale.
- User-control e libertà.
- Consistenza e standard.
- Prevenzione dell'errore.
- Riconoscere piuttosto che ricordare.
- Flessibilità ed efficienza.
- Estetica e design minimale.
- Aiutare l'utente a rilevare, diagnosticare e risolvere gli errori.
- Help e documentazione.

19. Focus Group:

I focus group possono essere usati sia durante la raccolta di prerequisiti che durante la valutazione con gli

utenti. Questo prevede la riunione di più utenti, progettisti e a volte anche il committente, per parlare di come dovrebbe funzionare il sistema, le caratteristiche che dovrebbe possedere e dei problemi che bisognerà tenere conto in fase di progettazione. Utilizzando questo metodo, diversamente dal questionario, si raccoglieranno solamente dati qualitativi, ma è comunque difficoltoso da realizzare in quanto bisogna essere in grado di gestire l'intero gruppo.

20. Cognitive Walkthrough:

Questo metodo consiste nel esaminare passo passo un task eseguendolo sul sistema in modo da verificare problemi di usabilità. Viene ereditato dall'ingegneria del software dove spesso si deve eseguire la verifica passo passo del sorgente, e nasce dalla constatazione che gli utenti spesso imparano ad usare una applicazione esplorando i menù e mai leggendo il manuale.

Per effettuare questa analisi necessitano 4 cose:

- Le specifiche o il prototipo abbastanza dettagliato.
- I task che l'utente deve compiere.
- La lista delle azioni richieste per completare il task.
- Indicazioni su chi sono gli utenti e il loro grado di preparazione.

Per ogni azione che esamina, l'esaminatore deve rispondere alle seguenti domande:

1. L'effetto dell'azione è lo stesso che aveva previsto l'utente?
2. La funzionalità è visibile?
3. Una volta che gli utenti hanno individuato l'azione corretta sono in grado di capire che è l'unica di cui necessitano?
4. Compiuta l'azione il feedback dato dal sistema è abbastanza chiaro?

21. Ergonomia:

è la disciplina che studia le caratteristiche fisiche dell'interazione in modo da migliorarne l'efficienza e la semplicità. Infatti il suo sviluppo si ebbe durante la seconda guerra mondiale, quando si cercava di costruire macchine con un alto grado di interazione con i soldati per aumentare la velocità delle operazioni e la loro efficienza. Successivamente con l'introduzione delle interfacce grafiche questo studio sfociò nell'interazione persona-calcolatore che odiernamente tiene conto di molte discipline, quali psicologia cognitiva, informatica, design, ergonomia, intelligenza artificiale, ecc...

22. Proprietà funzionali:

Le proprietà funzionali di un software, ovvero le caratteristiche che vengono percepite dall'utente durante l'uso del software sono:

1. Correttezza.
2. Affidabilità.
3. Robustezza.
4. Sicurezza.
5. Usabilità.
6. Efficienza.

23. Proprietà non funzionali:

Le proprietà non funzionali sono quelle proprietà più utili per i programmatori che per gli utenti, infatti sono proprietà interne all'applicazione, e sono:

1. Strutturazione.
2. Modularità.
3. Comprensibilità.
4. Verificabilità.
5. Portabilità.
6. Manutenibilità.

24. Modello PIE:

25. Principio apprendibilità:

è la capacità di un sistema interattivo che permette ad un utente, di comprendere il funzionamento di base, e successivamente di acquisire sempre maggiori conoscenze. È importante tenere conto che un utente quando si avvicina ad un sistema che ancora non conosce, tende a crearsi un modello mentale, ovvero si fa una sua idea precisa riguardo il funzionamento dell'applicazione. Questo modello mentale deriva da esperienze precedenti con altri sistemi, o dal mondo reale, o deduzioni momentanee. Inizialmente il modello sarà imperfetto, ma con l'andare avanti si acquisiranno sempre maggiori informazioni e quindi questo verrà raffinato progressivamente.

26. Abduzione e ragionamento abduttivo:

La persona ragiona e arriva a delle conclusioni, combinando diversi tipi di processi. Quindi oltre alla semantica, e ai processi di ragionamento logici, si affiancano la deduzione, l'induzione e l'abduzione. La deduzione permette, data una relazione e un suo stato iniziale, di trovare lo stato finale, l'induzione permette

invece di trovare la relazione che lega due stati, mentre l'abduzione permette di capire la causa, data la conseguenza e la relazione che le lega.

27. Direct Manipulation:

è un'invenzione di Engelbart che teorizzò come gli utenti tramite questo approccio si sentano più vicini ai loro dati. Questo tipo di approccio è molto usato nei sistemi a finestre, dove siamo noi che ci prendiamo le informazioni, invece di chiederle al computer, come nel caso delle shell Unix o Dos. Infatti con gli OS a linea di comando, gli utenti sentono tra loro e i dati uno strato, in quanto per visualizzarli devono chiedere attraverso un preciso comando all'OS di poterli visualizzare.