Design dell’usabilità

Fabio Vitali

“Il Progetto è il generatore. Senza Progetto, c’è mancanza di ordine e volontà. Il progetto contiene in se stesso l’essenza della sensazione.”

Le Corbusier, 1931

Introduzione

Qui esaminiamo in breve:

- Un’introduzione all’ingegneria dell’usabilità
- User-centered design
- Task-centered design

Il task-centered design sarà poi l’argomento di altri gruppi di lucidi.

Introduzione (1)

La prima generazione di utenti di computer erano i programmatori stessi. Essi avevano esperienza sostanziale e motivazioni forti nell'approccio al computer. Per questo potevano accettare (ed anzi approvare) interfacce complesse e potenti.

La popolazione utente attualmente è composta in minima parte di esperti di computer, ma anzi di persone dotate di diversissime competenze: anni 80 impiegati e professionisti, anni 90 utenti casuali e per "diporto".

Questi utenti non usano il computer per obbligo o per dedizione alla tecnologia, ma perché trovano in esso un supporto al lavoro o all'hobby. Il loro uso è discrezionale, e può essere interrotto e sostituito in qualunque momento.

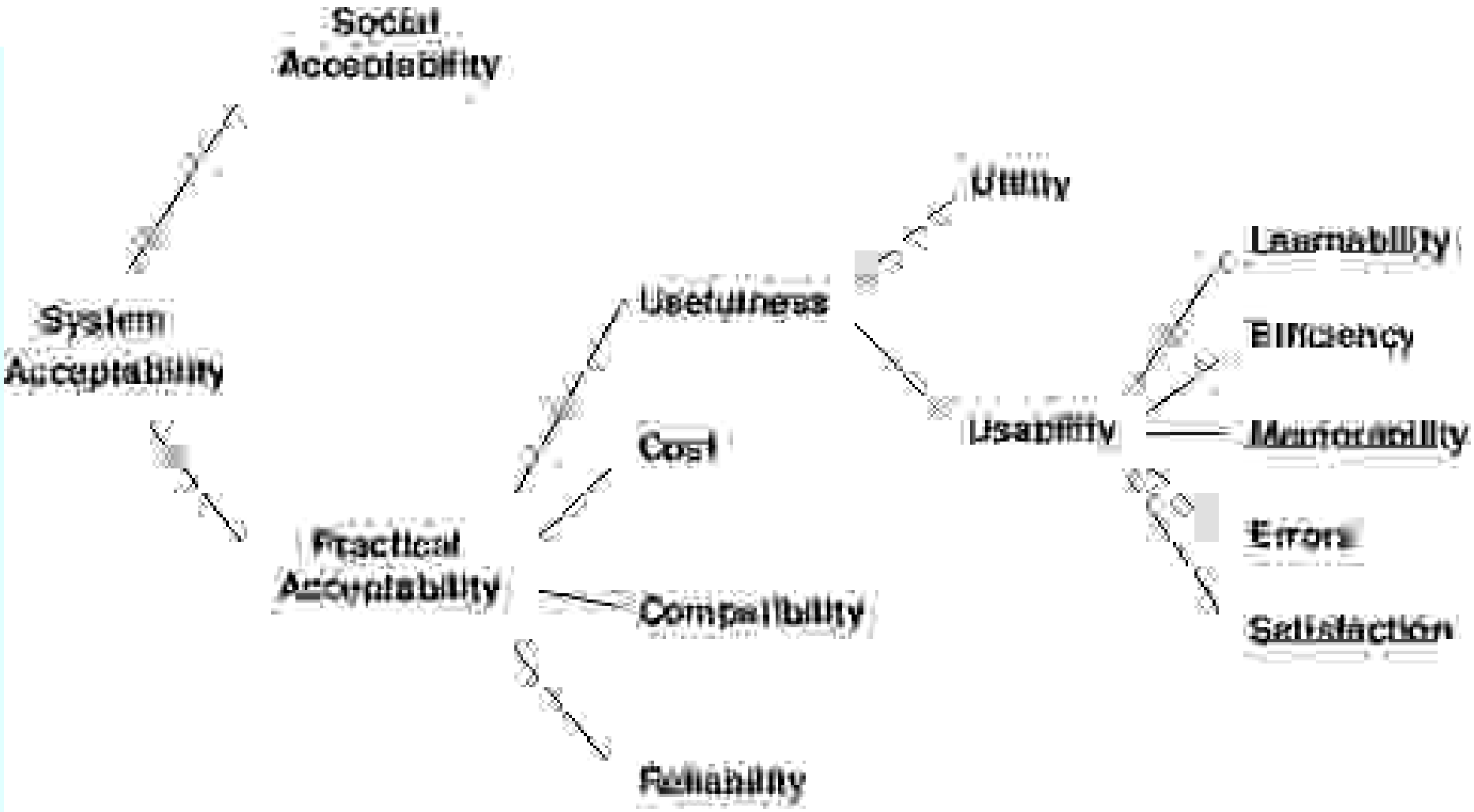
Introduzione (2)

La progettazione dei sistemi deve dunque essere basata fondamentalmente sull'osservazione attenta degli utenti, dei loro task, delle loro preferenze.

I progettisti cerchino il rapporto diretto con gli utenti in tutte le fasi del design, dalla progettazione all'implementazione a tutto il ciclo di vita del software. I metodi di progettazione iterativa permettono test precoci dei prototipi, feedback tempestivo, raffinamenti incrementali.

L'ingegneria dell'usabilità (usability engineering) è una disciplina con pratiche formalizzate e standard riconosciuti, per ottimizzare l'usabilità di un sistema.

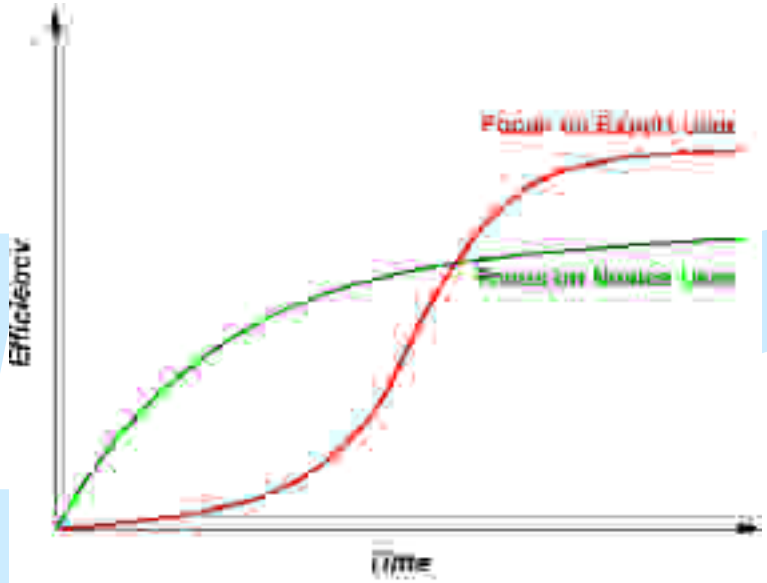
Ingegneria dell’usabilità (3)



I cinque parametri dell'usabilità

- **Apprendibilità**: la facilità di apprendimento per gli ***utenti novizi***.
- **Efficienza**: prestazioni continuative da parte di ***utenti esperti***
- **Memorizzabilità**: facilità nell'uso intermittente del sistema da parte di ***utenti casuali***.
- **Errori**:frequenza di errori catastrofici e/o minori.
- **Soddisfazione soggettiva**: piacevolezza all'uso del sistema

Curve di apprendimento



Un sistema può :

- Focalizzarsi sulla facilità di apprendimento, compiacendo gli utenti novizi.
- Focalizzarsi sulla efficienza di uso, compiacendo gli utenti esperti
- Permettere sia un modo novizio che un modo “esperto” (ad esempio con menù più ricchi e un linguaggio di script)

In questo caso naviga sopra ad entrambe le curve dell’apprendimento

System e User-centered design

System-centered design

Progettazione basata su aspetti di convenienza per il progettista.

- Cosa è facile progettare su questa piattaforma?
- Cosa è facile creare con gli strumenti disponibili?
- Cosa trovo interessante ed appagante progettare?

User-centered design

Progettazione basata su caratteristiche dell'utente a cui il sistema è destinato

- Cosa sa fare l'utente?
- Di cosa ha bisogno l'utente?
- In che contesto opera l'utente?

Sette facili principî per lo user-centered design

1. Sfrutta sia la conoscenza nel mondo che la conoscenza nella testa
2. Semplifica la struttura dei task
3. Rendi visibili le cose
4. Sfrutta correttamente il mapping
5. Sfrutta la potenza dei vincoli
6. Progetta considerando gli errori
7. Quando il resto non basta, standardizza

Task-centered design

La metodologia incentrata sui task è rivolta ad organizzare l'intera progettazione del sistema sull'analisi dei compiti che l'utente destinatario deve svolgere nel sistema.

La analisi tradizionale dei requisiti si occupa di elementi parziali ed astratti dei task (ad es. "Scrivere un libro"), mentre il task-centered design si occupa di task reali, completi, rappresentativi (ad es. "Produrre la prima versione del libro "Sistemi Operativi" di Tanenbaum).

Esso comprende:

A Analisi di task e utenti

B Progettazione

C Verifica

D Test

E Iterazione

Analisi di task ed utenti (1)

L’analisi dei requisiti tradizionale è un mero elenco di funzionalità che ci si aspetta dal sistema.

Una moltitudine di considerazioni importanti si perdono in questo elenco, e possono essere riscoperte solo analizzando le funzionalità nel loro contesto d’uso.

A questo serve l’analisi di task ed utenti.

Un sistema non è utile se non è utile a qualcuno.

Un sistema deve innanzitutto fare “quello che serve”.

Poi il sistema deve integrarsi facilmente nel mondo e nel lavoro dell’utente, senza richiedere ripensamenti globali al suo modo di pensare e lavorare.

Analisi di task ed utenti (2)

Per questo è necessario comprendere la situazione attuale dell'utente, e la sua esperienza d'uso del computer e del suo lavoro.

Ad esempio, un utente Macintosh si aspetta il supporto per Copy & Paste anche dove per l'applicazione i comandi hanno poco senso.

Una buona analisi di task ed utenti richiede uno stretto contatto tra progettisti ed utenti. Questo può essere logisticamente molto difficile, ma è spesso molto importante.

Analisi di task ed utenti (3)

Un processo di progettazione tradizionale, dall'analisi di task ed utenti, deriverebbe una serie di principi assoluti di usabilità e produrrebbe una specifica generale del sistema e della sua interfaccia.

L'approccio task-centered, invece, concretamente identifica alcuni o numerosi task specifici, e ne elabora lo sviluppo.

I task identificati dovrebbero fornire una copertura ragionevolmente completa delle funzionalità del sistema, ed un confronto dovrebbe essere possibile tra l'elenco dei requisiti e le funzionalità descritte nei task presi ad esempio.

Ci dovrebbe essere una giusto equilibrio tra task complessi e task semplici, e i task dovrebbero essere reali, tratti dall'effettiva esperienza degli utenti.

Progettazione

Nel progettare l'interfaccia migliore, è importante cercare tra quelle reali quelle che funzionano e derivare idee da quelle. Questa copia può essere efficace sia per paradigmi di interazione ad alto livello, che per decisioni su controlli e display di basso livello.

- Ad alto livello: quali programmi stanno usando questi utenti, e cosa c'è che li rende utili? Inventare è lento, difficile e pericoloso. Riciclare vuol dire avere già delle decisioni prese (ad esempio, come funziona il copy&paste).
- A basso livello: dove sono messi, in che ordine e con che effetti, i controlli usati nelle altre applicazioni usate dagli utenti?

In generale, tra un'opzione apparentemente più razionale, ed una a cui gli utenti sono abituati, è da preferire quella nota. L'unica eccezione è se il cambio di paradigma è necessario, evidente e sostanziale. Piccole modifiche irritano soltanto.

Valutazione

La fase di valutazione è la fase in cui ogni funzionalità è confrontata con ogni task: per ogni funzionalità per cui non si trova un task corrispondente bisognerà o aggiungere un task, o rimuovere la funzionalità.

E’ necessaria un’analisi della bontà del progetto. Esistono metodi qualitativi e quantitativi per analizzare il progetto

La valutazione del sistema va fatta su qualcosa di visibile ed usabile. Il prodotto può essere una serie di disegni su un foglio di carta, la tecnica del “Mago di Oz” (un collega che esegue le parti di codice che mancano), o un ***mock-up*** usando uno strumento apposito (ad es., ToolBook).

Test del progetto con gli utenti

Per quanto sia dettagliata l'analisi dei task, alcuni problemi salteranno fuori solo con l'interazione con gli utenti.

Esistono tecniche di selezione degli utenti per i test, e tecniche di interazione con loro (ad esempio, "think aloud").

E' opportuno registrare su videocassetta i test, e cronometrarne i tempi, contarne gli errori, e valutare i commenti degli utenti.

Iterazione, implementazione

I problemi evidenziati dagli utenti (ci saranno!) serviranno come guida per il miglioramento. I test non servono per dimostrare, ma per migliorare le scelte progettuali.

La implementazione del sistema con l'interfaccia realizzata dovrebbe essere realizzata con strumenti che permettano rapidi cambiamenti e miglioramenti.

Linguaggi O.O., sistemi di prototipazione rapida, variabili globali e parametri facilmente modificabili permetteranno un rapido fine-tuning senza cambiamenti devastanti al progetto. Ricordarsi che l'interfaccia utente rappresenta più di metà del codice di un prodotto commerciale: queste sono in fondo regole di buona programmazione.

Conclusioni

Nei prossimi lucidi verranno riesaminati con maggiore dettaglio le cinque fasi del task-centered design:

A Analisi di task e utenti

B Progettazione

C Verifica

D Test

E Iterazione

Riferimenti

- K. Andrews, ***Human-Computer Interaction Lecture Notes***, Final Version of 13 July 1999, http://www.iicm.edu/hci/
- C. Lewis, J. Rieman, ***Task-centered user interface design, a practical introduction***, 1994, ftp://ftp.cs.colorado.edu/pub/cs/distribs/clewis/HCI-Design-Book