Alcune visioni personali

Fabio Vitali

Come, scusi?

Introduzione

Oggi esaminiamo in breve due libri di stimati e originali pensatori:

- ***About Face*** di Alan Cooper
- ***The Future Information*** di Theodore Nelson

About Face

The Essentials of User Interface Design

IDG Books (1995)

Alan Cooper

- Consulente di interfacce utente e software design
- Autore di software famosi (SuperProject, MicroPhone II)
- Ideatore e realizzatore di Visual Basic
- Vincitore del premio Windows Pioneer (1994)

About Face

- Il Macintosh ha delle linee guida dal 1986. I programmatori le rispettano, perché sanno che applicazioni che le seguono hanno successo, e le ammirano, perché forniscono risposte chiare e comprensibili ed allo stesso tempo profonde.
- Microsoft ha pubblicato libri analoghi per Windows per ottenere lo stesso effetto di costrizione e entusiasmo, senza riuscirci.
- ***About Face*** ha avuto, nella comunità di programmatori per Microsoft, lo stesso effetto delle linee guida Apple per i programmatori Macintosh.
- Noi non ci occupiamo delle parti più direttamente orientate a Windows, ma delle parti più generali e meno specifiche.

Argomenti di *About Face*

Come organizzare la progettazione
- Goal-oriented design
- I modelli d'uso

I concetti alla base della progettazione
- Idiomi e vocabolario canonico
- Orchestrazione e flusso
- Postura, overhead, idiozia e memoria

Meccanismi dell'interazione
- Mouse, selezione, manipolazione diretta, drag-n-drop
- Menu, dialog box e toolbar
- Gizmos: bottoni, checkboxes, text boxes, ecc.

L'uso delle interfacce
- Errori ed undo
- Classi di utenti

Goal-oriented design (1)

Il software solitamente non è progettato, ma emerge come casualmente dal team di sviluppo come uno zombie da una tomba

Quando è progettato, è progettato sul punto di vista del programmatore, o del settore marketing, o al massimo dal punto di vista dell'utente. Mai dal punto di vista dei goal dell'utente.

Anche nel caso migliore, gli utenti non sono consci dei loro goal. Software che semplicemente permettano loro di completare task non sono soddisfacenti.

Il compito del progettista è di guardare oltre il task vero i goal degli utenti.

Goal-oriented design (2)

Questi non sono veri goal dell'utente:

- Evadere efficientemente una pratica
- Formattare velocemente un'intero libro
- Verificare la correttezza di un ipotesi commerciale

Questi sono veri goal dell'utente:

- Non sembrare stupido
- Non fare grossi errori
- Portare a termine una quantità ragionevole di lavoro
- Divertirsi (o almeno non annoiarsi troppo)

Anche se si riferiscono al lavoro, i goal sono sempre personali, mai lavorativi.

Un software pensato per i goal lavorativi fallirà, un software pensato per i goal personali riuscirà e riuscirà anche a soddisfare i goal lavorativi.

Tre paradigmi

Esistono tre criteri di fondo per progettare un'interazione

- **Il paradigma tecnologico**: come nello stile architettonico dei Metabolisti (anni 60), lo scheletro interno della macchina è espresso in evidenza, quasi orgogliosamente: conoscere il programma equivale a conoscerne il funzionamento interno.
- **Il paradigma metaforico**: Si prende una rappresentazione del mondo reale con qualche attinenza all'ambito dell'applicazione, e si imita il modello reale con il modello del computer, attuando una metafora. La metafora globale forza ogni sua sottoparte ad adattarsi per non rovinare il feeling, ed impedisce all'interfaccia di evolversi verso strade non realistiche.
- **Il paradigma idiomatico**: l'interfaccia usa un vocabolario di base di comandi, anche arbitrario, ma identificabile e ricordabile, che costituisce un livello "automatico" su cui viene formata l'esperienza d'uso del sistema.

Il paradigma idiomatico

Le interfacce grafiche hanno avuto successo non perché grafiche o perché adottavano la metafora della scrivania, ma perché avevano imposto un idioma superiore a quello precedente

Nelle frasi (ad esempio) “Menare il can per l’aia” o anche “figo” il significato letterale, e probabilmente anche quello metaforico, sono irrilevanti e probabilmente persi da molto tempo. Quel che conta è che hanno caratteristica di identificabilità e memorizzabilità peculiare che richiedono poco o nessuno sforzo per apprenderlo e per ricordarlo.

Analogamente, i movimenti base del mouse, il concetto di selezione, la forma dei pulsanti e dei dialoghi, la forma delle finestre sono idiomi sufficientemente semplici da apprendere e da ricordare da superare di gran lunga le caratteristiche delle interfacce a prompt.

Il paradigma idiomatico si basa sulla creazione di un vocabolario specifico, peculiare e **ristretto**. Il prompt non dà nessuna restrizione di vocabolario, e il 99% delle stringhe sono ignote e quindi errori.

Nelle interfacce grafiche ho a disposizione solo comandi significativi come composizione naturale di idiomi di base a formare un vocabolario canonico.

Flusso ed orchestrazione (1)

Nelle barche a vela, c'è un momento magico in cui la velocità è sufficiente a sollevare la barca sopra la sua stessa scia, toccando appena il pelo dell'acqua, così da riuscire a toccare velocità elevatissime. Avviene in maniera improvvisa ed è (dicono) una sensazione esilarante. Tuttavia è anche un momento fragilissimo, poiché basta una manovra goffa per tornare giù nell'acqua e fermarsi come contro un muro.

Gli esseri umani hanno similmente uno stato psicologico chiamato "flusso", che si attiva improvvisamente quando ci concentriamo su un task. Il "flusso" è definito come un "coinvolgimento profondo e quasi meditativo" sul task da realizzare, e induce spesso una "gentile sensazione di euforia", ed una sostanziale perdita di senso del tempo. In stato di flusso, le persone sono molto produttive, specialmente per attività creative o progettuali.

Flusso ed orchestrazione (2)

Come per le barche a vela, anche per gli esseri umani il flusso è uno stato magico e fragilissimo. E' necessario evitare che la goffaggine dell'interazione possa interromperlo. Una volta fuori dal flusso, è difficile e lento riportarcisi.

Per evitare di uscire dal flusso, queste sono tecniche utili:

- **Seguire i modelli mentali**
  uno strumento che organizza le proprie procedure intorno ai modelli mentali dell'utente non disturba.
- **Dirigere, non discutere**
  un volante non discute: limita a due direzioni la scelta dell'utente, ma non inizia una conversazione con l'utente per la scelta ottimale della prossima direzione

Flusso ed orchestrazione (3)

- **Tenere gli strumenti a portata di mano**
  molti programmi sono troppo complessi per permettere in maniera facile di accedere a tutti i comandi. In questo caso le modalità sono indispensabili. Le toolbar permettono all'utente di avvicinare a sé gli attrezzi che gli sono necessari, e cambiare di modalità in maniera veloce e facile.
- **Fornire feedback non modale**
  Il modo più semplice di fornire all'utente un valore di feedback è mostrare una finestra modale, che richiede che l'utente esplicitamente la licenzi. Informazioni non modali sono un modo molto migliore, anche se questo può popolare l'interfaccia di numeri e valori ovunque.

Per ottenere un'interazione migliore, l'orchestrazione delle varie parti dell'interfaccia in un unico coerente ed efficace è il meccanismo fondamentale. L'obiettivo finale è l'invisibilità dell'interfaccia.

Possibilità contro probabilità (1)

Se una proposizione è vera 999.999 volte e falsa 1 volta, per un matematico la proposizione è falsa. Se, di fronte ad una scelta, 999.999 utenti scelgono la prima risposta, ed 1 sceglie la seconda, l'informatico riterrà comunque importante fare la domanda.

Per un informatico o un matematico, possibilità e probabilità si confondono: per esempio, se sono sei ore che sto lavorando su un documento di testo, il programma di dà la possibilità di salvare il documento su disco oppure di buttarlo via.

La cameriera non mi propone di versare la birra sulla camicia ogni volta che mi serve. Perché il software mi propone di NON salvare il mio file? Perché il programmatore ha confuso possibilità con probabilità.

Comandi e funzioni usati centinaia di volte al giorno stanno fianco a fianco con comandi e opzioni che useremo una volta nella nostra vita. Ha senso?

Possibilità contro probabilità (2)

La soluzione non è impedire la scelta, ma far richiedere esplicitamente la scelta improbabile. Invece di fornire il comando “Salva”, fornire “Abbandona i cambiamenti”, da attivare nel raro caso in cui voglio buttare via il lavoro.

In generale, i programmi si debbono porre nel modo di minimizzare il dialogo non necessario, al fine di non ostacolare lo stato di flusso dell’utente.

La regola proposta è: “dirigi, segui o stai fuori dai piedi”. L’utente preferisce un risultato immediato da modificare, piuttosto che una complessa dialog box prima di vedere qualcosa. Ad esempio, un comando di creazione tabelle dovrebbe prima creare una tabella, e poi permettere all’utente di manipolarla e cambiarla a suo gradimento.

Il programma deve chiedere scusa, non permesso. Caratterizzare il feedback è importante. Un’azione che va a buon fine, la normalità, non vanno riportate con un dialogo modale: il feedback non modale serve proprio a questo.

Postura e stato

Come le persone, anche i programmi hanno un atteggiamento, che influenza il modo in cui funzionano e sono percepiti dagli utenti. Questo atteggiamento è anche un modo di porsi all'utente, una postura.

- **Postura sovrana**: alcuni programmi si pongono come padroni dell'interazione, il programma principale usato dall'utente per lungo tempo. Word processor, spreadsheet, ecc. sono di questo tipo.
- **Postura transiente**: esistono programmi che fanno saltuariamente un compito importante, come dirigere uno scanner per importare un grafico. Non essendo un'applicazione sovrana, deve esibire un comportamento diverso. Deve essere più avara di spazio, ma può essere più brillante ed estroversa.
- **Postura demonica**: applicazioni che normalmente non richiedono l'intervento dell'utente: driver di stampa, programmi di fax, ecc.
- **Postura parassitica**: silenziosi testimoni di un processo in corso: l'orologio, lo stato di download di un file, ecc.

Overhead

Per compiere un task, dobbiamo sempre eseguire un certo numero di azioni non direttamente connesse con il task. Questo è l'overhead.

Esistono quindi ***azioni guadagno*** (che ci portano più vicini al completamento del task) ed ***azioni tassa*** (che non contribuiscono direttamente al completamento del task).

Un azione è tassa se serve al computer, e non a me: salvataggio di file, recupero, spostamento e riordinamento di finestre, operazioni di backup, installazione di programmi, configurazione di driver, ...

Eliminare le azioni tassa rende l'utente più efficace.

Un'importante classe di azioni tassa sono le meta-domande: chiedere di poter chiedere. Ad esempio, sullo schermo è mostrata un'informazione che io posso cambiare. Debbo però attivare un comando per poter cambiarlo, invece di cambiarla direttamente: sto chiedendo di poter chiedere l'aggiornamento di un valore!

Permettere l'input in ogni situazione in cui ci sia output.

Idiozia

Mi compare questo messaggio quando il disegno è orientato in maniera diversa dal formato di stampa.

- Perché il programma mi avverte di questa situazione?
- Perché debbono esistere due orientazioni diverse?
- Perché non le sistema da solo?
- Cosa significano i pulsanti Annulla ed Ok?
- Perché almeno non ci sono pulsanti tipo “Converti il disegno in Landscape” e “Converti il formato di stampa in Portrait”?

Memoria

Un trucco semplice per migliorare overhead e idiozia è fornire di memoria il programma: apprendendo dal comportamento dell'utente la volta precedente, il programma potrebbe evitare domande sciocche e ripetute.

Le domande non sono scelte: la dialog box pone l'utente in una situazione di inferiorità, dovendo avere una risposta pronta per tutte le domande che gli vengono poste

Viceversa, una toolbar offre scelte, lasciando l'utente in situazione di superiorità e controllo.

Se vale la pena chiederlo, vale la pena ricordarlo.

Messaggi di errore (1)

I messaggi di errore debbono essere rari. Adesso vengono usati come le aspirine, mentre dovrebbero essere usati come la chirurgia.

E’ necessario vaccinare i programmi in modo che non si possano presentare le condizioni di errore.

Errori come inserimento di caratteri spuri in un formato fisso (es. una data), o fuori ordine, ecc., sono sintomi dell’inabilità del programma di gestire l’interazione, non dell’umano di capire qualcosa.

Non sono errori dell’utente, ma limiti del software.

Messaggi di errore (2)

Gli utenti percepiscono i messaggi di errore come accuse nei loro confronti. Un messaggio come quello sopra verrà percepito come questo sotto.

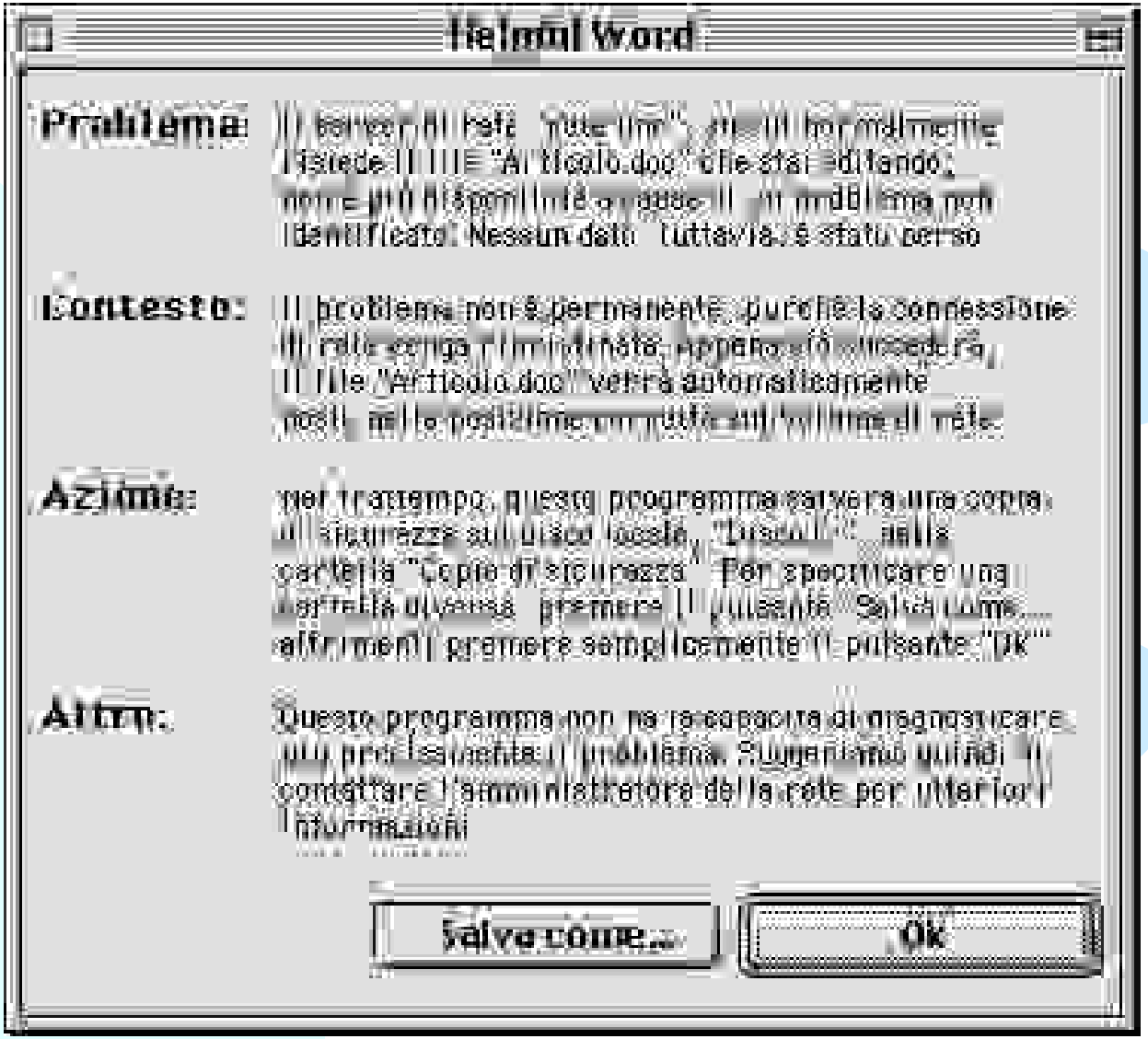
Messaggi di errore (3)

Inoltre, **il messaggio di errore non serve**. Tutto quello che riesce a fare è di impedire di inserire una lettera in un campo che vuole numeri, ma non mi impedisce di inserire il numero sbagliato. Questo è molto più difficile da realizzare.

A volte il messaggio d'errore è inevitabile. In questo caso, il messaggio deve essere:

- Gentile: prendersi le colpe e ammettere i propri limiti
- Illuminante: fornire informazioni lucide e complete su quello che è successo
- D'aiuto: fornire una via d'uscita ed assicurarsi che l'utente possa riprendere la procedura normale il più facilmente possibile.

Messaggi di errore (4)



Classi di utenti

I software vengono solitamente pensati per utenti esperti, oppure per utenti completamente ignari e principianti.

- L'utente medio non arriverà mai allo stato di utente esperto, e tante facilitazioni gli saranno sempre negate.
- Tuttavia, tutti gli utenti lasciano presto lo stato di totalmente ignaro. E presto trovano irritante l'atteggiamento paternalistico delle interfacce per principianti.

La maggior parte degli utenti rimangono in uno stato ***perpetuamente intermedio***, con abilità e fluidità che vanno e vengono a seconda della frequenza con cui usano il programma.

Lo scopo del progettista dunque è:

- Trasformare al più presto il principiante in un intermedio
- Non ostacolare quegli intermedi che vogliono diventare esperti
- Tenere gli intermedi perpetui felici e produttivi per sempre

Vettore nella testa e nel mondo

Come dice Don Norman, esistono conoscenze che rimangono nel mondo, e conoscenze che rimangono nella testa.

In maniera del tutto naturale, i comandi usati più frequentemente entreranno nelle nostre conoscenze della testa, mentre quelli non usati no. Il vettore nella testa è l'insieme di comandi che ormai abbiamo imparato.

La maggior parte delle esigenze di una classe di utenti sono simili. E' quindi solo logico che il progettista crei un vettore nel mondo, l'insieme dei comandi più frequenti per l'utilizzatore medio.

Dopo un po', tuttavia, gli utenti inizieranno a sentire la insufficienza del vettore del mondo, e la necessità di un proprio vettore. La personalizzazione dell'interfaccia permette agli utenti intermedi di specializzare il proprio vettore, e proseguire verso lo stato di esperto.

The future information

Ideas, Connections and the Gods of Electronic Literature

http://www.sfc.keio.ac.jp/~ted/INFUTscans/
(1997)

Theodore Holm Nelson

- Padre storico degli ipertesti
- Ideatore del sistema Xanadu
- Autore del libro “Computer Lib”, giustificazione ideologica dei personal computer (1974)
- Premiato con il premio Yuri Rubinski per i contributi alla diffusione della cultura di rete nel 1998.

The future information

Idee ed informazione

- L’illusione dell’ordine
- Una filosofia del design e del software

Inferno, paradiso e strutture

- Religioni informatiche
- L’attuale, orribile mondo dei computer
- Virtualità, e come progettarla

L’illusione dell’ordine (1)

La gente si illude nel pensare che l’informazione sia intrinsecamente organizzata. Tuttavia la forma reale della maggior parte dell’informazione è un agglomerato inestricabile di fatti, fatti parziali, opinioni e credenze, un ***perplex***.

Il modo che usiamo per semplificare un perplex e farlo sembrare ordinato è di presentarlo attraverso un ***punto di vista***.

Ci sono piccoli perplex, e grossi perplex, e perplex enormi. Ogni argomento è un perplex, a meno che qualcuno non lo presenti, assoggettandolo ad un punto di vista.

Biblioteche, scuole, media ed ogni altra fonte di informazione cercano di imporre un ordine sul perplex. A volte si può fare astraendo dall’informazione stessa (l’ordine esisteva nella realtà ad un certo livello), più spesso imponendolo (senza che ci fosse in realtà).

L’illusione dell’ordine (2)

Un sistema di ordine è una finestra che ci permette di osservare un perplex, una semplificazione presentazionale che ci permette di vederlo e capirlo… perché elimina le parti non adatte.

Un sistema di ordine su larga scala è chiamato ***paradigma***, un’idea che dà ordine ad altre idee, e impone una superstruttura sui punti di vista che applichiamo al mondo.

L’informazione non è semplice, ed è un errore cercare di strutturarla o semplificarla arbitrariamente. ***Gerarchie*** e ***categorie*** sono due pessimi modi di strutturare e semplificare una realtà complessa.

Spesso le cose che vengono descritte con gerarchie e categorie sono in realtà sovrapposte, interconnesse, mescolate, ingarbugliate. Gerarchie e categorie perdono queste caratteristiche.

Il peggio è la gerarchia di categorie, o ***tassonomia***. In alcuni casi fortunati (Mendelejev, Linneo, ecc.), queste rappresentano effettivamente descrizioni utili della realtà, ma in generale le tassonomie sono solo scheletri rigidi dentro ai quali forzare a fatica la realtà.

Documenti e parole

I documenti sono oggetti mediatici con un punto di vista. Non c'è modo di NON avere un punto di vista, che lo si voglia o no. I documenti rappresentano dunque una realtà come percepita dagli autori, e formano un perplex a loro volta.

Le biblioteche forzano i libri e gli autori in categorie imposte, che spesso non corrispondono all'idea che l'autore ha di sé. Le biblioteche sono state utili soprattutto grazie al lavoro e alla dedizione di specialisti della catalogazione in spregio alla completa e inevitabile impossibilità del loro successo.

Le parole sono strumenti utili e fondamentali, un incapsulamento di un punto di vista o di un'idea. Tuttavia le parole esistenti bloccano un concetto, incanalano il pensiero, distorcono la percezione ed impediscono lo sviluppo di nuove idee.

Viceversa, i ***neologismi*** permettono lo sviluppo di nuove idee e soluzioni: non discutiamo più sul significato di una parola, perché sono **IO** a dirti cosa significa questa parola.

Una filosofia del design (1)

La progettazione è la attività più importante della mente umana. Progettare significa mettere insieme gli oggetti con un intento: una casa, una forma di governo, il menu di una cena o gli impegni della giornata.

In tutti i casi si considerano molte opzioni contemporaneamente ed in parallelo, che vengono selezionate e scartate per arrivare al progetto. Il problema è trovare le idee giuste, e questo è difficile.

A volte la progettazione porta a combinazioni magiche, strutture di design uniche che hanno proprietà non eguagliabili: quattro ruote in una macchina, per esempio. Ali orizzontali negli aeroplani,la radio supereterodina, ecc.

Tuttavia prima di trovare la combinazione giusta, è necessario esaminare e scartare dozzine di candidati inadeguati: “prima di trovare il tuo principe azzurro, dovrai baciare un sacco di rospi”.

Una filosofia del design (2)

Un principio a cui il mondo dei computer non è ancora arrivato è che creare software è come creare film. Il film è un sistema di eventi su uno schermo che influenza la mente ed il cuore dello spettatore. Il software è un sistema di eventi su uno schermo che influenza la mente ed il cuore dell'utente, e gli permette pure di intervenire. Il film è una sottocategoria del software!

La principale preoccupazione nello sviluppo software, dunque, sono gli effetti sul cuore e sulla mente dell'utente.

La creazione di un film richiede il contributo personale e creativo di molteplici professionalità, anche con alto contenuto di sofisticazione e tecnologia: costumi, luci, musica, recitazione, fotografia, ecc.

Il regista non deve sapere tutto di ciascuno di questi aspetti, anzi spesso può non saperne niente. Il regista deve sapere come portare tutte queste tecnica a combinarsi per creare il risultato globale di influenzare la mente ed il cuore dello spettatore.

Una filosofia del design (3)

Lo stato della progettazione software adesso è paragonabile a quella del cinema dei primi anni, prima della nascita della figura del regista (1893-1904).

Prima il compito di fare il film era dato al cameraman, perché sapeva adoperare l'attrezzatura. Solo con D.W. Griffith si capì la differenza tra regia e ripresa, e nacque una professionalità separata ed indipendente da quella del cameraman.

Il risultato nel software è che ci sono troppe professionalità connesse, ma senza il perno centrale che fornisce una visione coerente e unica dello sforzo collettivo. Ovviamente tutto si sposta in politica. Chiunque lavora nel cinema vuole fare il regista. Hollywood è il meccanismo politico che permette di realizzare i film, identificando le professionalità, i fondi, i responsabili delle decisioni creative

Paradiso, inferno e strutture (1)

Religioni informatiche

I computer non sono rigidi nelle loro operazioni. E' possibile fargli fare quasi tutto , se il programma è fatto in modo sufficientemente intelligente e correttamente.

Il problema è che stiamo usando vecchie parole, vecchi concetti, senza tentare di progredire:

- Ci hanno detto che la gente aveva bisogno di oggetti familiari per apprendere l'uso dei computer, ed ecco che abbiamo icone, finestre, cartelle e cestini della spazzatura in un mondo piatto e bidimensionale.
- Ci hanno detto che il modo giusto e inevitabile di organizzare i nostri dati sulle memorie permanenti è in una gerarchia di contenitori indipendenti e sconnessi, il file system gerarchico.

Il problema è che i designer si abituano a pensare che è così che debba andare il mondo. Certe idee vengono incastonate nei progetti, e certe altre lasciate fuori.

Paradiso, inferno e strutture (2)

Ad esempio, ***connessione*** è lasciata fuori: Non c'è un modo generale per associare un commento ad un file del mio computer, o per creare connessioni persistenti tra due file. No: i file sono nudi e sconnessi. Analogamente per backup, confronto, condivisione.

Purtroppo alla base ci troviamo il meccanismo del file system, con la sua crudezza, rigidità ed inflessibilità. Ci richiedono di spezzare i nostri pensieri in file monotipati, di organizzarli in directory gerarchiche, di forzarle in una struttura che, semplicemente, non è adatta.

Nel 1974, Nelson parlava di come i computer, per come erano stati usati fino ad allora, erano serviti per opprimere la gente. Con i PC, se non stiamo attenti, sarà possibile essere oppressi nel nostro salotto.

Ebbene, si è tutto avverato!

Paradiso, inferno e strutture (3)

L’attuale, orribile mondo dei computer

Tutto quello che ci raccontano sui computer è falso. E’ falso che il file system sia l’unico modo concepibile per organizzare i nostri dati. E’ falso che le applicazioni da ufficio siano Word Processor, Spreadsheet e DBMS. E’ falso, perché ci costringono ad adattare le nostre esigenze ai loro prodotti, e non viceversa.

E’ ridicolo che non ci possa essere un modo per far convivere insieme testi e disegni, dati strutturati e documenti, ipotesi numeriche e commenti.

E’ ridicolo che non ci possa essere un sistema comodo e facile per organizzare in maniere multiple gli stessi documenti, per connetterli tra loro, per farne il backup facile, per gestire copie multiple con verifica automatica della loro correttezza e delle differenze.

Virtualità (1)

Il software design è un arte e non una tecnologia. E' più semplice esprimere le proprie idee in termini di film e di filosofia delle idee, piuttosto che in termini di tecnicismi informatici.

La teoria della virtualità è la generalizzazione della teoria della regia cinematografica applicata al software. La virtualità è l'opposto della realtà: qualcosa che sembra, la sua idea, le sue qualità illusorie e apparenti.

Ogni cosa ha una realtà (la sua natura tecnica), ed una virtualità (il modo in cui la percepiamo). La virtualità ha due parti: la ***struttura concettuale*** e il ***feel***.

- La struttura concettuale di un artefatto è quello a cui pensa la persona quando lo sta usando.
- Il feel sono i suggerimenti sottili, le sensazioni impalpabili, la variazioni con altri oggetti simili.

Per esempio; tutte le automobili hanno la stessa struttura concettuale, ma feel completamente diversi: il suono del motore, la comodità del cruscotto, il rumore della portiera che si chiude.

Virtualità (2)

Nella progettazione, è al feel che dobbiamo l'effetto sul nostro cuore e sulla nostra mente. E' quello che fa sì che due film con trame simili (con la stessa struttura concettuale) abbiamo effetti (feel) completamente diversi. E' il regista che ottiene questo effetto.

E' nel pensare nuove virtualità, strutture concettuali innovative, ma anche nuovi feel, che permetteremo ad idee nuove e potenti di venire alla luce.

- Il caso di orologi e calendari a spirale.
- Il caso della creazione di testo

Nella progettazione della virtualità, si inizia con una lista dei desideri. Dopo di ché si delinea la struttura concettuale, come si costruisce la trama di un film e la sceneggiatura.

Quindi si considerano i possibili feel, cercando quel fattore unificante che permetta a tutti i desideri di diventare veri.

Conclusioni

Oggi abbiamo parlato di alcuni concetti proposti da pensatori originali:

- L’importanza degli idiomi (cioè di un vocabolario ristretto e specifico) nella bontà di un’interazione
- Il concetto di flusso ed i meccanismi per mantenerlo vivo, attraverso l’orchestrazione
- L’importanza dello strumento di scomparire dietro al task, facendosi vivo solo quando strettamente necessario e nella maniera più gentile e informativa possibile
- La futilità dell’ordine e le potenzialità dei neologismi
- La progettazione di una virtualità del software come presupposto per una fase radicalmente innovativa dell’informatica.

Riferimenti

- Alan Cooper, ***About Face - The Essentials of User Interface Design***, IDG Books, 1995
- Theodore H. Nelson, ***The future information - Ideas, Connections and the Gods of Electronic Literature***, http://www.sfc.keio.ac.jp/~ted/INFUTscans/, 1997