User testing

Fabio Vitali

Task-centered design (reprise)

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.

Le fasi connesse con il task-centered design sono:

- Analisi di task e utenti
- Progettazione
- Verifica
- **Test - di cui parliamo oggi**
- Implementazione
- Organizzazione dell'interfaccia estesa

Test con gli utenti

Nessuna prova teorico, neanche creata con l'ausilio delle tecniche di analisi più raffinate, troverà tutti i problemi di un software.

L'ausilio degli utenti reali è indispensabile per avere considerazioni realistiche sul sistema, credibili e dimostrabili

Come scegliere gli utenti

Deve esservi somiglianza o approssimazione con l'utente previsto

Problemi etici

- Stress psicologico
- Imbarazzo
- Pressione ambientale
- Gestione della privacy

Soluzione:

- consenso informato,
- monitor costante della rilassatezza del tester,
- in caso pessimo: "rottura" delle apparecchiature

Come scegliere i task

Ce li abbiamo già! Sono i task dell'analisi dei task.

Però a volte questi sono troppo lunghi, o richiedono conoscenze di background che l'utente può non avere

Usare i task già descritti è positivo anche perché i task non sono troppo frammentati. I task della fase di task analysis sono già organizzati per essere complessi.

Un sistema per i test

Trade-off tra costo ed efficacia:

- Disegni su carta
- Mock-Up su computer
- Mago di Oz
- Prototipo
- Sistema funzionanti

Il mago di Oz

Un sistema in cui le parti computazionali non realizzate vengono svolte “dietro le quinte” da un essere umano.

Es.: un sistema di acquisizione di input vocali veniva svolto da un dattilografo professionista a cui arrivava in cuffia la voce dell’utente.

E’ molto comodo per provare alternative di progettazione in assenza delle core functions.

Che dati raccogliere

Process data

- Osservazioni su cosa stanno facendo gli utenti durante il task
- Raccontano il cosa e soprattutto il perché l'utente fa quello che fa

Bottom-line data

- Raccolte di dati su il tempo richiesto per i task, il grado di successo, il numero e qualità degli errori, ecc.
- Riassumono il cosa è successo.

Thinking aloud (1)

Supponiamo di avere un'interfaccia ed un task da completare in un dato tempo. Supponiamo di scoprire l'esistenza di un problema al terzo schermo, dove la gente non ha attivato la funzione di ***radicalizzazione ricorsiva**** che serviva per svolgere il compito.

- Gli utenti non avevano capito la necessità di una radicalizzazione ricorsiva?
- O forse, sapendo benissimo che serviva, non erano riusciti ad identificare correttamente il comando o la schermata in cui attivarlo?

Per dare una risposta a questo tipo di domande, è inevitabile cercare informazioni più dirette su come hanno ragionato gli utenti. A questo serve il think aloud.

* un termine inventato

Thinking aloud (2)

Sostanzialmente: si chiede agli utenti di svolgere un compito, e al tempo stesso di parlare all'osservatore su quello che stanno facendo.

Si deve loro chiedere di raccontare:

- a cosa stanno pensando,
- cosa stanno cercando di fare,
- come pensano che dovrebbero procedere,
- che dubbi gli stanno venendo,
- che cosa leggono e come questo si relaziona al task.

Probabilmente, nel caso in questione, scopriremo il problema a seconda che il tester dica

- "Cosa c'entra la radicalizzazione ricorsiva adesso?"
- "Come cavolo faccio ad attivare la radicalizzazione ricorsiva?"

Thinking aloud (3)

Il vantaggio più evidente del T.A. è scoprire differenze di vocabolario tra utenti e progettisti. Niente di più facile di un utente che dice:

- "Adesso debbo fare la ***estrazione ripetuta****. Come faccio? Qui parla soltanto di radicalizzazione ricorsiva!"

L'esempio classico è un'interfaccia che richiede all'utente dei "parametri", e quelli leggono sistematicamente "perimetri".

* altro termine inventato

T.A.: preparare i tester

- “Dimmi quello che stai pensando durante lo svolgimento di questo compito”
- “Non sono interessato ai tuoi pensieri segreti, ma solo a quello che pensi relativamente allo svolgimento del compito”
- Sottolineare che è il **sistema** sottoposto a test, non il **tester**.
- Spiegare che tipo di registrazione viene fatta, e qual è la politica di gestione della privacy

T.A.: il ruolo dell'osservatore (1)

L'osservatore deve

- Spingere il tester ad agire e a parlare
- Aiutare il tester se si blocca definitivamente

Deve però evitare distorsioni:

- Niente domande/risposte
- Alcune parole mal scelte possono concentrare l'attenzione su aspetti dell'interfaccia che avrebbe ignorato. Vedi il caso dell'esperimento delle corde

T.A.: il ruolo dell'osservatore (2)

Buone domande:

- Dimmi a cosa stai pensando
- Continua a parlare

Cattive domande:

- Secondo te, a cosa servono quei riferimenti alla radicalizzazione ricorsiva? ***(concentra l'attenzione su un aspetto dell'interfaccia su cui l'utente stava sorvolando)***
- Perché hai fatto questa azione? ***(fa venire il dubbio di aver sbagliato o di essere sulla cattiva strada)***

E' opportuno precisare:

- "In questo test è permesso, anzi incoraggiato, fare domande, ma non te ne potrò fornire la risposta."

E' possibile aiutare il tester solo dopo aver verificato che si è effettivamente bloccato, e solo per impedirgli di abbandonare il test

T.A.: la raccolta dei dati

Ci sono vari meccanismi per raccogliere i dati dei test:

- Prendere nota di quello che dice l'utente
- Videoregistrare la scena
- Registrare la traccia delle azioni dell'utente

La disponibilità monetaria gioca un ruolo importante: però prendere note in tempo reale è difficile, e l'osservatore deve anche fare altre cose.

T.A.: valutazione dei dati

A cosa serve il thinking aloud

- Confermare o modificare i risultati del cognitive walkthrough
- Identificare nuovi problemi dell'interfaccia o blocchi temporanei
- Nel caso di confronto tra soluzioni diverse, tenere in conto il peso che ha l'***ultima esperienz***a: il caso del confronto tra collant

Il confronto tra collant

- Un esperimento psicologico famoso (1977) valutò la capacità di razionalizzazione delle scelte in assenza di ragioni oggettive
- Di fronte ad un centro commerciale venne messo un banchetto in cui si spingevano le persone a provare le differenze di tre tipi di collant. In realtà erano tre collant assolutamente identici.
- Ma una percentuale assolutamente ragguardevole identificò senza ombra di dubbio l'ultimo collant provato come più liscio, comodo, ben rifinito.
- C'è una tendenza a scegliere, in match vicini tra loro, l'ultima esperienza fatta, ignorando tuttavia che questo sia il motivo della preferenza

Bottom-line data (1)

E’ opportuno che anche i risultati numerici finali vengano valutati adeguatamente con corrette analisi statistiche.

Ad esempio, il tempo di completamento del task, il numero di errori compiuti, il numero di alternative provate, apprezzamento dell’utente, ecc.

Da notare che questi test debbono essere svolti indipendentemente dai test a thinking aloud, poiché il pensare ad alta voce cambia completamente i tempi di reazione e di azione degli individui.

Bottom-line data (2)

Va considerato che non è possibile fermarsi alla sola media (o al massimo alla mediana) di una serie di valori, perché la scelta del campione è sempre non completamente rappresentativa.

In presenza di una forte deviazione standard, le analisi fatte sono poco predittive. L'errore standard ci dà un valore indicativo della bontà dell'esperimento.

N.B.: la affidabilità di una predizione è proporzionale al quadrato del numero di test!

Confrontare due alternative

Il modo più semplice è l'esperimento con gruppi diversi.

- Ogni alternativa è testata da un gruppo (omogenei per tipo di utenti)
- Con apposita analisi statistica si calcola la differenza tra le medie dei gruppi con l'errore standard combinato

Alternativamente con lo stesso gruppo

- Provoca problemi nel caso di task semplici (alcune persone possono ottenere di fare lo stesso task due volte)
- Richiede attenzione nella distribuzione dei task
- E' opportuno solo per testare piccoli task, ad es. un nuovo widget.

Mettere in piedi un test

Scegliere l'ordine dei task
- Dal più semplice al più complesso o casuale?

Preparare i tester
- A freddo o con un po' di istruzione?

Un test pilota con un tester fuori quota
- Per preparare l'osservazione, le domande, il set up.

Testare poche variabili alla volta
- Fattori psicologici, interazioni tra task e tempo di test si combinano esponenzialmente per creare analisi difficili da interpretare

Interrogare gli utenti
- Chiedere agli utenti **dopo** serve a poco: la gente ricorda le soluzioni, non i problemi.
- Si può però chiedere un parere soggettivo. I valori soggettivi spesso determinano la volontà d'acquisto molto di più dell'usabilità effettiva.

L’interfaccia estesa

L’interfaccia estesa comprende tutto quello che serve per usare un applicazione e che non sono i controlli, comandi e display dell’applicazione:

- Manuali
- Help on-line
- Applicazioni di training
- Customer support

Differenze

Un’applicazione d’uso quotidiano avrà sistemi di assistenza diversi da un’applicazione da aeroporto

Il tempo usato per ottenere informazioni su un sistema è tempo **perso** per il task e per la persona

Inoltre è tipicamente **molto** tempo.

Quindi mai l’interfaccia estesa buona deve essere una scusa per un’interfaccia interna pessima!

Manuali

La prima fonte di informazione degli utenti.

Un buon manuale viene creato come viene creata una buona interfaccia: task centered design

- Analisi di test e utenti
- Cognitive walkthrough
- Test con utenti

Il manuale dovrebbe essere breve, task-oriented, ed usare il linguaggio dell'utente

Manualistica ragionevole

Manuale passo-passo dei task

- Istruzioni passo-passo di come eseguire ogni azione ‘importante’ del sistema

Command reference

- Descrizioni di ogni comando del menu e di ogni dialog box del sistema, più eventualmente dei linguaggi di macro, scripting e programmazione disponibili

Super indice analitico

- Una lista **molto** estesa di nomi di task, esigenze, comandi, ecc. che l’utente può considerare naturali e che il programma non adopera (perché adotta sinonimi o termini più generali)

Il manuale dei task

E’ il manuale più complesso da scrivere, perché deve svolgere compiti diversi per utenti diversi in tempi diversi:

- Utenti novizi che cercano di capire come eseguire i task fondamentali
- Utenti sporadici che si sono dimenticati un passaggio in un task familiare
- Utenti esperti in altri programmi simili che cercano di capire come passare dall’uno all’altro in breve tempo

Ogni task deve essere presentato prima in maniera molto breve, con un livello estremamente alto di dettaglio

Un esempio

Capitolo 3: Mail Merge

Il mail merge serve per creare lettere o moduli identici o simili per molti destinatari diversi.

> *Ad esempio, il mail merge permette di spedire una breve lettera per ogni cliente in ritardo con i pagamenti, con i dettagli dei pagamenti non ricevuti.*

In questo capitolo, ti verrà spiegato come:

- Creare un file contenente la lettera di base e tutti i contenuti comuni a tutte le lettere
- Creare un file contenente nomi, indirizzi e dettagli dei pagamenti per ogni cliente
- Unire i due file per generare tutte le diverse lettere.

Scrivere manuali

- Essere concisi
- Presentare prima una overview, poi la lista dettagliata dei passi
- Parlare il linguaggio dell'utente
- Trovare un correttore che sia anche un utente
- Affidarsi ad una guida di stile interna
- Usare una struttura visiva della pagina chiara ed intuibile

Concisione

Per ottenere concisione, è utile ricordarsi che:

- Introduzioni e conclusioni vengono saltate dagli utenti esperti, e confondono l'utente novizio
- E' inutile dare la prospettiva storica: gli utenti non cercano un posizionamento storico
- E' inutile dare una prospettiva filosofica: gli utenti non badano alle decisioni di progetto
- E' inutile essere tecnici: gli utenti non cercano notizie sui meccanismi interni di funzionamento
- E' inutile il marketing: gli utenti hanno già comprato il prodotto.

On-line help (1)

La manualistica on-line è da sempre insoddisfacente:

- Ci sta meno roba su uno schermo che su carta
- Il testo è meno leggibile
- Su schermo ci si perde più facilmente che su carta
- Lo schermo non si aggiorna con la stessa velocità della carta
- Non si possono sottolineare, evidenziare o appuntare le pagine su schermo
- L'aiuto occupa spazio di schermo dell'interfaccia stessa

On-line help (2)

La manualistica on-line, tuttavia, è utile per tutti gli aspetti puramente mnemonici che l'utente può non ricordare:

- Equivalenti da tastiera
- Aspetti sintattici (es.: linguaggi di scripting, ecc.)
- Opzioni oscure

Mai dare troppe informazioni: una, due righe massimo, o tabelle di riferimenti.

Mai schermate lunghe di testo.

Training

Materiale audiovisivo, istruzioni in classe, istruzioni on line, ecc. costituiscono il materiale di training.

Essi servono specialmente per l'utente novizio e sistematico (la manualistica on-line serve anche per l'esperto e lo sporadico)

Alcuni committenti amano organizzare il training come lezione in classe, più che studio dei manuali.

In questo caso, il manuale, o un documento a parte, deve descrivere i task basilari in maniera diversa.

C'è chi propone l'approccio minimalistico:

- Vengono lasciati volutamente a parte i dettagli
- Viene data molta enfasi sui goal del task
- Si invita esplicitamente l'utente a provare sul sistema l'esecuzione del task.

Come apprendiamo

- La gente impara meglio se usa quello che sta imparando ad usare
- L'abilità migliora con la pratica
- La pratica richiede immediato feedback
- La pratica è più efficace se è spaziata nel tempo, ed intervallata con altri task
- Le cose che non debbono essere imparate debbono variare
- Si possono imparare solo poche cose alla volta
- L'apprendimento non deve essere sistematico, ma progressivo,
  - dal semplice al complesso, non dall'A alla Z
  - Si impara a camminare prima che a correre

Customer support telefonico (1)

Il supporto tecnico è un grande costo per un prodotto di successo, perché il costo è proporzionale al numero di copie vendute, e non al numero di funzioni del sistema.

Nel mettere in piedi il supporto tecnico, tre aspetti sono fondamentali:

- Training: i telefonisti debbono sapere cosa rispondere
- Tracking: utile feedback per la prossima release
- Task-centered support: la creazione di una FAQ serve come analisi dei possibili problemi degli utenti

Customer support telefonico (2)

Ci sono difficoltà nel far parlare un tecnico con un utente non esperto, e ancora di più al telefono.

E’ utile che tutto il team di produzione passi del tempo al supporto telefonico: è l’unica occasione che hanno per parlare direttamente con i loro utenti sul campo.

E’ utile che sia lo stesso tecnico che richiami qualche tempo dopo con la risposta ad un problema spinoso.

Mai e poi mai passarli da un tecnico all’altro, mettendoli in attesa e facendogli rispiegare il problema più e più volte.

Conclusioni

In questo gruppo di lezioni abbiamo parlato del task-centered design.

Il task centered design enfatizza l'uso di descrizioni dettagliate e situate di task realistici come base per la progettazione e la verifica dell'interazione.

Noi abbiamo visto:

- L'analisi di task e utenti (basata sulla descrizione dettagliata)
- La progettazione (basata sull'imitazione dell'esistente)
- La verifica (basata su Cognitive Walkthrough, Action Analysis e linee guida)
- Test con l'utente (incentrato sul Thinking Aloud)
- Organizzazione dell'interfaccia estesa (intesa come manuali, training, help line)

Riferimenti

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