Analisi del progetto

Fabio Vitali

Analisi del progetto

Nella fase di sgrossamento esistono procedure di valutazione della bontà del progetto. Tre sono le tecniche più rilevanti per questo scopo:

- ***Cognitive walkthrough***: una esecuzione fittizia e passo passo di un task, ed una valutazione empirica delle prestazioni
- ***Action analysis***: una analisi quantitativa delle precise azioni che debbono essere eseguite per svolgere un'azione.
- Analisi euristica (o uso di linee guida): la valutazione dell'interfaccia secondo regole di buon senso ricavate dall'esperienza

Cognitive walkthrough

Un cognitive walkthrough è un modo formalizzato di immaginarsi pensieri ed azioni degli utenti quando usano un interfaccia per eseguire per la prima volta un task.

Si prende il sistema, un prototipo o anche una serie di disegni dell'interfaccia da provare. Si seleziona un task da eseguire con quell'interfaccia, e si racconta una storia credibile su ciascuna azione che l'utente deve eseguire per portare a termine il task.

La storia è credibile se si può motivare ciascuna azione dell'utente basandosi sulle conoscenze generali dell'utente ipotizzato e sugli indizi e feedback forniti dall'interfaccia. Se non si riesce a raccontare una storia credibile, c'è un problema di interfaccia.

CW: un esempio (1)

Valutiamo l'interfaccia di una fotocopiatrice. Vengono forniti i disegni di un tastierino numerico, un bottone con scritto "Copy" e un bottone posto dietro per accendere e spegnere. La macchina si spegne dopo cinque minuti di inattività.

Il task è di copiare una pagina singola, e l'utente è una segretaria appena assunta. La storia che raccontiamo è: "La segretaria deve fare una copia, e sa che la fotocopiatrice deve essere accesa, quindi preme il pulsante di accensione. Mette il foglio di carta dentro alla fotocopiatrice e preme il pulsante 'Copy'. "

CW: un esempio (2)

Questa storia non è molto credibile: come fa la segretaria a sapere che la fotocopiatrice è spenta? Come fa a sapere dov'è il pulsante di accensione? Come fa a sapere come inserire il foglio? Siamo sicuri che sappia che il pulsante "Copy" significhi "Copia"?

Aggiungiamo allora un display che indichi se la fotocopiatrice è pronta, un disegno sul fronte che indichi come inserire i fogli, spostiamo l'interruttore in posizione visibile, e cambiamo il pulsante in un pulsante "Copia", e ripetiamo l'esperimento. E così via.

CW: un esempio (3)

Il CW può dunque scoprire vari tipi di problemi:

- Le assunzioni del progettista sui ragionamenti degli utenti (“perché l’utente dovrebbe pensare che la fotocopiatrice vada accesa?”)
- Comandi ovvi al progettista e meno all’utente (“l’utente sa che vuole accendere la macchina, ma sa trovare il pulsante?”)
- Problemi con etichette e prompt (“Come fa a sapere come inserire la carta, e siamo sicuri che capisca ‘Copy’?”)
- Problemi di feedback (“Come fa a sapere se la fotocopiatrice è accesa?”)

CW: un esempio (4)

Un’altra cosa che dice il CW è che i problemi potrebbero essere nelle specifiche più che nell’interfaccia:

- magari i progettisti volevano mettere una label “On/Off” nel pulsante di accensione, e localizzare il pulsante di copia nelle varie versioni nazionali, e si sono dimenticati di inserirli nelle specifiche.

CW: conclusioni (1)

Un CW ha bisogno di quattro elementi:

- Una descrizione o un prototipo dell'interfaccia, piuttosto dettagliata
- La descrizione di un task, possibilmente uno dei task descritti come rappresentativi nel task-based design
- Una lista completa e scritta delle azioni necessarie per completare il task
- Una chiara descrizione dell'utente e delle sue competenze ed aspettative

CW: conclusioni (2)

Due sono gli errori comuni:

- Confondere la lista delle azioni necessarie con il walkthrough medesimo. Il senso del CW è di raccontare in maniera credibile l'utente mentre svolge le azioni ottimali per completare il task, non di descriverlo mentre cerca queste azioni.
- Confondere il CW con lo user test effettivo: il CW identifica una classe di problemi che un test con 5-10 utenti potrebbe non identificare, ma il test reale con gli utenti identifica aspetti di contorno e reali che non possono essere scoperti con il CW.

Action analysis

La action analysis è una procedura di valutazione che esamina da vicino la sequenza di azioni da eseguire per completare un task.

Può essere considerata di due tipi:

- Action analysis formale (***keystroke-level analysis***) è caratterizzata da un dettaglio estremo nella descrizione delle azioni. Può arrivare a predire con uno scarto del 20% il tempo effettivo di completamento dei task, il tempo medio di apprendimento di un'interfaccia, il rapporto azioni/errori. E' tuttavia molto complicata e lunga da eseguire
- Action analysis approssimativa (***back-of-the-envelope analysis***) è invece meno precisa e molto più facile da realizzare. E' tuttavia in grado di evidenziare eccessive complicazioni, tempi eccessivamente lunghi o buchi nell'interfaccia.

Action analysis formale (1)

L’approccio formale alla analisi delle azioni viene usata per fare predizioni accurate del tempo impiegato da un utente esperto a compiere un task.

Per fare questo, vengono stimati i tempi per compiere ogni singolo passo (fisico e mentale) del task, e sommati.

Il passo tipico è la pressione di un tasto (***keystroke***) per cui questa analisi viene anche detta ***keystroke-level analysis***.

Action analysis formale (2)

La stima dei tempi di ogni azione viene ricavata da una tabella ricavata testando centinaia di utenti, migliaia di azioni individuali, in migliaia di situazioni, e poi facendo la media.

Se un controllo non è descritto in questa tabella, o lo si approssima con qualcosa di simile, oppure bisogna eseguire i test sul nuovo controllo.

Per ottenere i tempi necessari per eseguire un task bisogna dunque adottare un approccio top-down nella descrizione del percorso ottimale per eseguire un task, e associare i tempi appropriati ad ogni singola azione.

Average times for interface actions (1)

[da J. Reitman Olson and G. M. Olson, "The growth of cognitive modeling in human- computer interaction since GOMS," ***Human-Computer Interaction***, 5 (1990), pp. 221-265]

VISUAL PERCEPTION

- Respond to a brief light 0.1 s.

  Varies with intensity, from .05 second for a bright light to .2 second for a dim one.
- Recognize a 6-letter word 0.34 s.
- Move eyes to new location on screen0 .23 s.

Average times for interface actions (2)

PHYSICAL MOVEMENTS

- Enter one keystroke on a keyboard: 0.28 s.

  Ranges from .07 second for highly skilled typists doing transcription, to .2 second for an average 60-wpm typist, to over 1 second for a bad typist. Random sequences, formulas, and commands take longer than plain text.

- **Use mouse to point at object on screen 1.5 s.**

  May be slightly lower -- but still at least 1 second -- for a small screen and a menu. Increases with larger screens, smaller objects.

- Move hand to pointing device or function key 0.3 s.

  Ranges from .21 second for cursor keys to .36 second for a mouse.

Average times for interface actions (3)

MENTAL ACTIONS

- Retrieve a simple item from long-term memory 1.2 s.

  A typical item might be a command abbreviation ("dir"). Time is halved if the same item needs to be retrieved again immediately.

- Learn a single "step" in a procedure 25 s.

  May be less under some circumstances, but most research shows 10 to 15 seconds as a minimum. None of these figures include the time needed to get started in a training situation.

- Execute a mental "step" 0.075 s.

  Ranges from .05 to .1 second, depending on what kind of mental step is being performed.

- **Choose among methods** **1.2 s.**

  Ranges from .06 to at least 1.8 seconds, depending on complexity of factors influencing the decision.

Action analysis formale (3)

Un’analisi formale di un’interfaccia complessa è un compito improbo. 10 minuti di un’azione possono richiedere la descrizione di un migliaio di azioni e di tempi collegati.

Inoltre la descrizione del task e delle azioni dell’utente sono a descrizione del valutatore, e ci possono essere discrepanze anche notevoli. Per questo motivo, l’action analysis può essere utile solo in circostanze speciali:

- Per esaminare aspetti molto specifici dell’interfaccia (es.: esaminare le prestazioni di un nuovo widget di un’interfaccia grafica)
- Per esaminare task molto strutturati e controllati (es.: analizzare il carico di lavoro di un operatore telefonico di un servizio di assistenza telefonica)

Action analysis approssimativa (1)

La analisi approssimativa ignora i micro-dettagli, e si concentra sul quadro generale, elencando una serie “naturale” di azioni e valutandole globalmente.

Invece di “togliere la mano dalla tastiera e prendere il mouse”, le azioni qui descritte sono del tipo: “scegliere l’opzione X dal menù Y”.

Action analysis approssimativa (2)

L’enfasi qui non è sul valutare al decimo di secondo le prestazioni di un’interfaccia, ma sul valutare risposte a domande tipo:

- Posso eseguire in maniera semplice un task semplice?
- Posso eseguire velocemente un task frequente?
- Quanti passi e fatti debbo imparare per eseguire un task?
- Ho descritto ogni passo nella documentazione?

L’effetto più evidente dell’action analysis formale è che ci dice che ogni azione richiede almeno due o tre secondi (in giornata buona) e può arrivare anche a dieci (giornata storta, niente caffè, la sera prima gozzoviglie). E si parla di minuti se c’è il minimo errore.

Action analysis approssimativa (3)

Senza essere così precisa, la action analysis approssimativa è molto più robusta alle imprecisioni, e può essere usata:

- Per verificare che l'esecuzione di un task non sia di tempi paragonabili al farlo a mano, con una macchina da scrivere, con schedari cartacei, etc.
- Per decidere se l'aggiunta di una feature possa o meno complicare il resto dell'interfaccia.
- Per decidere se aggiungere modi multipli per realizzare un task (ricordare che la decisione di uno tra più metodi alternativi richiede più di un secondo!).
- Per verificare quali operazioni possano generare un errore, e quanto grave questo errore possa essere (in termini di tempo necessario per ripararvi)

Analisi euristiche

Sia la cognitive walkthrough che la action analysis sono valutazioni task-oriented. Lo user-testing, descritto più avanti, è anch'esso task-oriented.

Le valutazioni task-oriented hanno pregi e difetti:

+ Appropriatezza: Permettono di valutare le caratteristiche di un sistema all'interno di un flusso di lavoro credibile e guidato da motivazioni esterne all'interfaccia.
- Copertura: valutiamo solo alcuni task, ignorando molti degli altri.
- Interazioni inter-task: come si comporta il sistema quando stiamo facendo più cose, o di mantenere la coerenza in gestire operazioni simili in task diversi.

Linee - guida (1)

Le linee guida (o analisi euristiche) sono principi generali che guidano le decisioni di progettazione.Essi sono indipendenti dal task, e permettono di valutare quegli aspetti che nelle valutazioni task-oriented vengono ignorati.

Esistono centinaia di linee guida, da brevi liste di direttive vaghe (“Essere informativi”) a dettagliatissime collezioni di pedantissime raccomandazioni (organizzazione dei menu, posizione dei comandi, fraseazione dei messaggi di errore).

Tipicamente, le liste brevi ignorano molti aspetti importanti dell’interfaccia, mentre le collezioni dettagliate sono spesso troppo voluminose per essere pratiche.

Linee-guida (2)

Tuttavia le linee guida sono importanti quando:

- Esistono obblighi contrattuali o di mercato per seguirle (es. obblighi militari, statali o di impresa)
- E' importante mantenere la coerenza tra sistemi realizzati indipendentemente (es. le line guida proposte dai vari sistemi operativi).

Nella sezione "brevi liste", le Nove Euristiche di Nielsen e Molich sono diventate note per coprire molte aree precedentemente ignorate.

Nielsen e Molich suggeriscono che esse vengano usate indipendentemente da più valutatori, e poi gli errori identificati vengano raccolti in un'unica lista.

Le 9 euristiche di Nielsen e Molich (1)

1. Dialoghi semplici e naturali

   Semplice significa mancanza di informazione non rilevante o usata raramente. Naturale significa un ordine compatibile col task.

2. Uso del linguaggio dell'utente

   Vengono usate parole e concetti appartenenti al mondo e all'esperienza dell'utente, non del progettista e della programmazione.

3. Minimizzazione del carico di memoria

   Non viene richiesto che l'utente ricordi cose da un'azione alla successiva. Le informazioni vengono lasciate sullo schermo fino a quando non servono più.

Le 9 euristiche di Nielsen e Molich (2)

4 Coerenza

L'utente dovrebbe essere in grado di apprendere una sequenza di azioni in una parte del sistema e di riutilizzarla per ottenere risultati simili in altre parti.

5 Feedback

L'utente viene costantemente e correttamente informato dell'effetto delle sue azioni sul sistema.

6 Uscite chiaramente visibili

Se l'utente entra in una parte del sistema che non gli serve, deve sempre essere in grado di uscirne senza compiere danni.

Le 9 euristiche di Nielsen e Molich (3)

7 Scorciatoie

Vengono fornite all'utente esperto delle scorciatoie che gli permettono di evitare dialoghi lunghi e ripetitivi, e messaggi informativi che non lo interessano.

8 Messaggi d'errore positivi ed informativi

I messaggi di errore buoni sono quelli che permettono all'utente di capire qual è il problema e come porvi rimedio.

9 Prevenire gli errori

Ogni volta che viene precisato un messaggio di errore, bisognerebbe chiedersi: era possibile fare in modo che questo errore non potesse mai verificarsi?

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