Interazione Uomo-Macchina

Davide Pucci

2016-2017

Indice

Capitolo 1

Needs

1.1 Need finding

Si osservano gli utenti nel loro ambiente, mentre svolgono attività quotidiane, così da invididuarne le necessità, obiettivi e problemi. Pertanto è importante osservare:

- senza avere chiaro cosa cercare;
- senza coinvolgere direttamente l'utente;
- senza condizionare l'utente;
- imparando i vari passi del processo;
- le emozioni/paure/frustrazioni dell'utente.

I *need* non sono *soluzioni*. Queste ultime limitano l'innovazione, mentre i *need* aprono nuove possibilità.

1.1.1 Tecniche di *need finding*

1. interviste:
   Informali ed economiche, permettono di far scoprire informazioni soggettive inaspettate, ma richiedono parecchio tempo. Vengono fatte a persone rappresentative degli utenti o a utenti di sistemi simili. Si preparano prima le domande, si cerca di registrare l'intervista, si incoraggia l'intervistato ad approfondire le risposte. Evitare domande ovvie, o che contengono la risposta, o quelle troppo generiche; andare più sul concreto e chiedere il perchè di risposte date;
2. interviste specifiche:

- ai *lead users*: utenti particolarmente competenti e sofisticati, che hanno dei *need* nuovi per i quali sono in grado di trovare soluzioni; sono in grado di prevedere i *need* che molti utenti avranno in un futuro prossimo;
- agli *extreme users*: utenti che spingono un sistema all'estremo, trovando problemi altrimenti difficili da individuare;
- a esperti: in queste interviste si sfruttano le conoscenze degli esperti, che possono quindi discutere di problemi più difficili o astratti;
- *history interviews*: interviste utili a comprendere sequenze di eventi che spiegano il comportamento degli utenti;
- *process mapping*: interviste in cui si chiede all'intervistato di descrivere tutto il processo;
- *laddering*: ricorrere al frequente utilizzo della domanda *perchè?*;
- *cultural context*: interviste non strutturate, per conoscere il contesto;
- *intercepts*: intervista a domanda singola.

3. questionari:
   Vengono utilizzate delle domande uguali per tutti e permettono di raccogliere molte informazioni velocemente sulle quali è possibile fare un'analisi più appronfodita. Vengono usati molti tipi di domande (aperte, chiuse, risposta singola, risposta multipla, scala di valori, ranking, ...). Sono d'altra parte meno flessibili delle interviste. Si possono utilizzare diversi strumenti online per realizzarli;

4. diario: utilizzato per raccogliere informazioni su attività sporadiche, per studi longitudinali;

5. *pager studies*: si chiede all'intervistato di fermarsi e prendere nota in tempo reale, in un particolare momento/posto, in modo completamente asincrono, utilizzando note libere o form da riempire;

6. *camera studies*: riprendere l'attività svolta dall'utente per vederla con i *suoi* occhi;

Capitolo 2

Storyboards

2.1 Tasks

Una volta individuati i *need* occorre capire quali di essi la nostra interfaccia si propone di soddisfare, cercando di capire i motivi che spingerebbero un eventuale utente ad utilizzare l'interfaccia e cosa questa gli permetterebbe di fare.
Per questo si individuano i *task*: delle sub-attività che l'utente deve svolgere nel corso dell'attività generale per arrivare a soddisfare i *need* selezionati.
Pertanto ci interessa il ruolo della *UI*, ma non le sue funzioni, gli utenti coinvolti e il contesto.

2.1.1 Storyboards

Per definire i *task* si utilizza disegnarli come *storyboard*, dei semplici fumetti disegnati a mano libera, con pochi (da 2 a 4, massimo) riquadri e poco testo.
In questi fumetti non si disegnano schermate dell'applicativo, perchè ancora non progettate ed altrimenti ci vincolerebbero.
I vantaggi di questo sistema sono:

1. mostra un sistema e il suo contesto d'uso;
2. permette di condividerlo con gli altri;
3. non lega subito ad un'interfaccia particolare;
4. può essere disegnato (e corretto) in pochi minuti;
5. permette la discussione e il miglioramento delle scelte.

Esistono strumenti che permettono di creare gli *storyboard* mantenendo lo stile-fumetto.

In sintesi Si disegna uno *storyboard* per ciascun task, concentrandosi sui *task* principali, che in linea di massima devono essere compresi tra 2 e 4.

Capitolo 3

Prototyping

3.1 Realizzazione del prototipo

Per progettare la *UI* occorre immaginare l'interazione dell'utente con l'applicativo ai fini della risoluzione di un *task* specifico, considerando come strumenti disponibili dispositivi hardware, widget e i vari stili di interazione.
Per capire se la *UI* progettata è completa occorre analizzarne la comprensibilità e la facilità di utilizzo.
Il prototipo è un modello di un sistema interattivo che simula e/o anima alcuni aspetti/caratteristiche/funzioni dell'applicativo finale, utilizzato per valutare l'impatto sugli utenti e dove le condizioni di valutazione devono essere simili a quelle previste per l'interfaccia finale.
I prototipi vengono utilizzati per meglio definire l'applicativo e per testarne una versione iniziale, intermedia o finale.
Esistono tre approcci:

1. *throw-away*: viene utilizzata tutta la conoscenza appresa del contesto per la realizzazione del prototipo;

2. *incrementale*: le varie funzioni vengono aggiunte una alla volta al prototipo;

3. *evolutivo*: un prototipo realizzato viene utilizzato come base per il successivo, che lo migliora.

I problemi principali della prototipazione riguardano i tempi, perchè richiede tempo, e se persistono problemi, il tempo impegnato sulla realizzazione può sembrare buttato. Inoltre, è difficile pianificare un processo di design con la prototipazione, è difficile valutarne i costi e prescinde completamente dalla sicurezza, l'affidabilità e i tempi di risposta dell'applicativo.

3.1.1 Paper prototyping

Si realizzano i prototipi su carta, molto rapidamente, in modo da essere pronti per essere buttati e rifatti. Si disegnano tutte le componenti di un'eventuale *UI* (*label*, bottoni, ...), ma senza dar troppo peso all'estetica, così da non perder troppo tempo perchè l'utente dovrà essere interessato e concentrato solo sui task.
Si mostra il prototipo di carta ad un potenziale utente, ed un *human computer* si occupa di cambiare i foglietti all'interazione dell'utente con l'interfaccia.
I vantaggi riguardano la loro facilità di creazione, la possibilità di fare test subito con utente e che questo non è interessato alla grafica ma ai contenuti; permettono di testare la navigazione, il *workflow*, la terminologia e la funzionalità.

3.1.2 Ausilio dei computer

Esistono strumenti che permettono di lavorare in digitale sui *paper prototypes*: si fotografano i disegni e si importano, si marcano le zone cliccabili e vi si associano altri disegni fotografati. In questo modo si sostituisce lo *human computer* e si possono loggare i test.

Capitolo 4

Valutazione

4.1 Valutazione

Per fare una valutazione occorre fare delle simulazioni, realizzare dei prototipi ed avere una implementazione completa, così da permettere l'esecuzione di test di utilizzabilità e funzionalità del sistema. Viene effettuata in laboratorio, sul campo o in collaborazione con utenti, e si valuta sia il design che l'implementazione.
Il fine ultimo della valutazione è sì la possibilità di individuare problemi specifici, ma anche di stimare l'effettiva funzionalià e l'impatto dell'interfaccia sull'utente.

4.1.1 Valutazione mediante la partecipazione degli utenti

I partecipanti sono rappresentativi degli utenti, e devono aver un livello di esperienza simile ed omogeneo nel campo proposto dall'applicativo. *J. Nielsen* suggerisce un numero di partecipanti compreso tra 3 e 5.
Lo scenario descrive una situazione potenzialmente reale nella quale si immedesimano i partecipanti alla valutazione, così da individuare utenti, azioni, strumenti da usare e contesti. I task sono i singoli compiti svolti dai partecipanti, oppure è possibile svolgere valutazioni di uso libero del sistema, senza task.

4.1.2 Valutazioni in laboratorio

Situazione in cui viene ricostruito un contesto non interrompibile. Il laboratorio sarà diviso in tre parti:

1. *sala utente*: dove l'utente effettuerà i test, ripreso da diverse telecamere;
2. *sala team*: contiene i monitor raffiguranti l'oggetto di ripresa delle telecamere della *sala utente* e la *logging station*;
3. *sala esecutivi*: dove gli esecutivi osservano l'operazione attraverso le finestre delle altre due sale.

4.1.3 Valutazioni sul campo

Si lavora in un contesto naturale e che rappresenti quello richiesto dall'applicativo, col costo di eccessive distrazioni e rumori.
In questo caso viene richiesto all'utente di ragionare ad alta voce, di eseguire valutazioni cooperative e di fare considerazioni a seguito dell'esecuzione dei task. Chi richiede la valutazione prende nota, registra o riprende l'utente, così da permettere la raccolta di informazioni sensibili.

4.2 Valutazione sperimentale

Utilizzata per analizzare aspetti specifici del comportamento contestualizzato all'interazione.
Vengono scelti dei fattori: il soggetto, le variabli (gli elementi da modificare e misurare), l'ipotesi (un'idea provvisoria il cui valore deve essere accertato) e un *experimental design* (la modalità attraverso cui stai facendo questa valutazione).
Le variabli possono essere:

- *indipendenti* (IV): caratteristiche che vengono cambiate per produrre diverse condizioni (cambiamento dello stile delle interfacce, il numero degli elementi di un menu, ...);
- *dipendenti* (DV): caratteristiche misurate nell'esperimento (tempo impiegato, numero di errori, ...).

L'ipotesi rappresenta la predizione che una variazione delle variabili indipendenti causerà una differenza nelle variabli dipendenti. Se non viene fornita ipotesi, si richiede indirettamente di verificare che non ci sarà mai differenza alcuna tra le variabili dipendenti raccolte.
Con l'*experimental design* viene scelta un'ipotesti, le variabili dipendenti e indipendenti, i partecipanti ed una modalità tra la *between subjects* e la *within subjects*. Le condizioni sono, appunto, condizioni di controllo, dove la condizione sperimentale prevede venga modificata una IV rispetto alla condizione di controllo (utile per assicurarsi che è il cambiamento della IV il responsabile del cambiamento misurato nelle DV).

- *between subjects*: ad ogni partecipante viene assegnata una sola delle condizioni, e quindi svolgerà il test una sola volta. Ovviamente richiede la presenza di più partecipanti;
- *within subjects*: ad ogni partecipante vengono assegnate tutte le condizioni, e quindi esegue il test 2 o più volte. Richiede meno partecipanti e risulta quindi meno costoso.

I dati (le DV) vengono poi raccolti su grafici e ne viene fatta un'analisi statistica.

Outline È quindi di fondamentale importanza valutare frequentemente durante lo sviluppo, così da ridurre i costi di correzione. Per ridurre i costi di valutazione con utenti è possibile anche farle senza, focalizzandosi su aspetti specifici che possono causare difficoltà (sono valutazioni limitate, perchè non valutano il vero uso del sistema, ma solo la sua aderenza a principi conosciuti).

4.2.1 Valutazione esperta

Esistono quattro tipi:

1. *cognitive walthrough*: proposta da *Polson et al.* nel 1992, valuta il design circa il suo supporto all'utente nell'imparare i task e quindi quanto è facile apprendere.

2. *heuristic evaluation*: proposta da *J. Nielsen* e *R. Molich* nel 1994, guida le decisioni attraverso la risposta a 10 euristiche: *visibilità dello stato di sistema*, *corrispondenza tra il mondo reale e il sistema*, *libertà di controllo da parte degli utenti*, *coerenza e standard*, *prevenzione degli errori*, *riconoscere piuttosto che ricordare*, *flessibilità ed efficienza d'uso*, *design minimalista ed estetico*, *aiutare gli utenti a riconoscere, diagnosticare e correggere gli errori* e *guida e documentazione*.

3. *model-based evaluation*: vengono utilizzati modelli esistenti per valutare l'interazione: *GOMS* (modello di previsione delle performance dell'utente con un'interfaccia), *KLM* (modello di previsione dei tempi necessari per compiti fisici di basso livello, come uso tastiera e mouse), *dialog models* (per valutare le sequenze dei dialoghi, stati irraggiungibili, dialoghi circolari, ...).

4. *review-based evaluation*: vengono raccolte le informazioni dagli studi precedenti ed incrociate per supportare o rifiutare parti di design.

Capitolo 5

Design e Designer

5.1 Goals, vincoli e trade-offs

I *goals* sono gli obiettivi che descrivono lo scopo per cui si sta progettando il sistema, per chi lo si sta facendo e il perchè eventuali utenti dovrebbero volere il nostro design.
I *constraints* (o vincoli) sono i materiali e gli standard utilizzati dalla nostra applicazione, o i costi, o i framework e le piattaforme, o il tempo necessario.
I *trade-offs* (o compromessi) sono il costo da dover pagare per poter soddisfare determinati *goal* o *constraint*, sono i *goal* e *vincoli* sacrificati per altri *goal* e *vincoli*.

5.2 Design

È di fondamentale importanza prevedere un design per la massima utilizzabilità dell'applicativo, pertanto il lavoro del designer è un lavoro di gruppo, che include la raccolta e il coordinamento di un insieme multidisciplinare di compentenze, dove il vero designer non impone il suo stile, ma sceglie materiali e tecniche e sperimenta, tenendo conto della componente psicologica e dei costi.
Pertanto, il *designer è un progettista dotato di senso estetico che lavora per la comunità* (*B. Munari*, 1971), compie scelte razionali, oggettive, mentre l'artista fa scelte soggettive.

5.3 Modello *Waterfall*

Il modello di sviluppo *a cascata* (o *waterfall*) prevede le seguenti fasi:

1. *specifiche tecniche*: si raccolgono le informazioni necessarie ad individuare cosa il sistema dovrà essere in grado di offrire;

2. *design architetturale*: descrizione di alto livello su come il sistema sarà in grado di offrire i servizi richiesti;
3. *design dettagliato*: affinamento delle componenti architetturali e delle interrelazioni per identificare i moduli da realizzare separatamente;
4. *sviluppo e testing*: costruzione del sistema;
5. *integrazione e testing* integrazione del sistema;
6. *manutenzione*.

Spesso questo modello fallisce perchè alcune delle fasi variano durante lo sviluppo ma non rappresenta un modello ingrado di tornare indietro facilmente, perchè è un modello linerare e sequenziale.

5.4 Modello Agile

Molto più adattivo rispetto al modello a cascata, perchè utilizza metodi orientati alle persone, piuttosto che al processo. È un modello circolare, iterativo, che prevede *pre-release* e *release* cicliche e rapide, è collaborativo con team multidisciplinari e prevede anche l'apporto dell'utente finale.
Prevede due *loop* principali:

- *user-research*/design
- *design/development*

Capitolo 6

Luogo dell'attenzione

6.1 Luogo dell'attenzione

Il luogo dell'attenzione è un oggetto fisico o un'idea alla quale stiamo pensando attivamente ed intenzionalmente.

- *focus*: volonta di focalizzare l'attenzione su qualcosa, mediante un azione;
- *locus*: non si può controllare completamente, perchè l'attenzione viene attratta da qualcosa, che non coincide con il *focus* dell'interfaccia.

Quando si è *assorti* ci si trova in una situazione in cui abbiamo un solo luogo dell'attenzione, ma un evento può spostarlo.
Quando delle azioni (o compiti) vengono effettuate ripetutamente, creano un'abitudine, che permette di esseguirle senza pensarci.
I compiti che eseguiamo senza pensiero cosciente sono detti *automatici*: tutti i compiti che eseguiamo contemporaneamente - tranne uno - sono automatici, quello non automatico riguarda il luogo dell'attenzione.
L'*interferenza* è il tentativo di eseguire contemporaneamente due compiti non automatici, che ci fa peggiorare il rendimento in entrambi.
Le *sequenze di azioni* eseguite ripetutamente diventano dei task automatici, cioè un abitudine. Interrompere una sequenza automatica richiede lo spostamento dell'attenzione su di essa.

Quindi, l'uso sistematico di un'interfaccia crea delle abitudini, e questo può essere sfruttato dal designer, perchè l'utente tende a prendere delle abitudini e l'interfaccia può aiutarlo a farlo.
L'interfaccia dovrebbe permettere la concentrazione sul task, ma se l'esecuzione del task è difficile o delicata, l'utente tende a concentrarsi solo su alcuni aspetti essenziali, ignorando warning e help forniti dall'interfaccia. Ma se sappiamo dove si trova il luogo dell'attenzione possiamo apportare cambiamenti in altre parti del sistema, senza distrarre l'utente.

Generalmente un cambio di contesto richiede 10 secondi, se però diventa abituale, può richiedere molto meno. Pertanto, è importante che l'applicati, alla terminazione di alcuni task, sia in grado di tornare al contesto precedente.

6.2 Modi

Il *content* (contenuto) è l'insieme di informazioni che risiedono in un sistema, utili all'utente.
Il *GID* (*graphical input device*) è il meccanismo per comunicare al sistema una particolare locazione o la scelta di un oggetto (tipicamente la posizione del cursore).
Il *GID button* è il bottone principale del *GID*.
Il *tap* è l'azione di premere e rilasciare un tasto (che torna al suo stato originale).
Il *to click* è il posizionamentod del *GID* e il successivo *tap* del *GID button*.
Il *to drag* è la pressione del *GID button* e, senza rilasciarlo, lo spostamento del *GID*, per poi rilasciarlo in altra locazione.
Una *gesture* (gesto) è una sequenza di azioni umane completata automaticamente una volta avviata (scrivere una parola comune, tipo *che*).
Dato un gesto, un'interfaccia è in un *modo* se l'interpretazione di quel gesto è sempre la stessa. Quando il gesto viene interpretato in maniera diversa, l'interfaccia è in un altro *modo*. Per esempio, il *CAPS LOCK* crea un *modo*.
I *quasimodes* sono *modi temporanei* (*modo* che svanisce dopo l'uso, come il pennello di *word*), o *quasi-modi* (*modo* ottenuto attivando e mantenendo fisicamente un controllo, come il tasto *CTRL*).
I *noun-verb* e i *verb-noun* rappresentano l'esecuzione di azioni (*verb*) su oggetti (*noun*):

- *noun-verb*: si seleziona prima l'oggetto: si seleziona l'oggetto mentre esso stesso è nel luogo dell'attenzione; il luogo dell'attenzione si sposta sull'azione da compiere e a quel punto si esegue il gesto che attiva l'azione. Interrompere l'azione non richiede un'altra azione;

- *verb-noun*: si seleziona prima l'azione (crea un modo).

Capitolo 7

Applicazioni mobili

7.1 Progetto

Occorre sempre tener conto del *contesto d'uso* dell'utente destinatario dell'applicativo.
Per esempio, l'utente di un dispositivo mobile sarà in movimento, non concentrato a lungo, esegue i compiti nei ritagli di tempo ed è frequentemente interrotto. Esiste la possibilità che venga richiesta l'attenzione contemporaneamente da più attività.

Differenze rispetto al desktop

- schermo piccolo;
- memoria limitata;
- una schermata alla volta;
- un'applicazione alla volta;
- help minimale.

7.2 Stili

Esistono tre tipi di applicazione mobile:

1. *productivity application*: permette di svolgere compiti basati sulla organizzazione e manipolazione di informazioni dettagliate e la *user experience* è basata sul task. I dati vengono organizzati gerarchicamente, spesso in lista, dove è possibile aggiungere e rimuovere elementi e/o scendere a livelli di dettaglio successivi. Esempio: applicazione *email*.

2. *utility application*: esegue un task semplice che richiede poco input dall'utente, legge informazioni essenziali su qualche argomento o verifica lo stato di qualcosa. Dà poche informazioni senza struttura gerarchica. Utilizza una o più liste, ma non legate gerarchicamente. Dà la possibilità di cambiare configurazione frequentemente. Esempio: applicazione *meteo*.
3. *immersive application*: utilizzata per compiti che presentano ambienti particolari, che non mostrano grandi quantità di testo e richiedono l'attenzione continuata dell'utente. Hanno spesso interfacce molto ricche, sono a tutto schermo e focalizzate sul contenuto e sull'esperienza di quel contenuto. Non usano controlli standard, ma ne crea di propri. Usa spesso grandi quantità di dati, ma li mostra in modo particolare nel contesto. Esempio: *giochi*.

7.3 Android

- *gesture* standard:
  - *pinch open*
  - *pinch closed*
- *metaphores* (switch);
- non nascondere controllli, utillizare *swipe*;
- *back* button;
- *cancel* nei modal;
- *progress bars*:
  - *determinate*
  - *indeterminate*
  - *buffer*
  - *query indeterminate and determinate*
- suggerimenti;
- customizzazione;
- temi;
- *material design*:
  - *app bar*
  - *floating action button*
  - *card*
  - *lists*
  - *alerts*
  - *bottom sheets*

7.4 iOS

- posticipare il *sign-in* il più possibile
- minimizzare il *data entry*
- non associare azioni non standard a gesti standard
- *branding* con logo, non invasivo e coerente in tutta l'app
- *3D touch* (pressione sul touch screen): mostrare un menu, del contenuto aggiuntivo, una preview, un'animazione, ...
  - pressione leggera: preview;
  - pressione forte: *pop* (accesso all'item) + vibrazione;
  - swipe verso l'alto: menu (*action buttons*).
- notifiche (locali o *push*), arricchite con titolo, testo, suono. Con badge e payload;
- *widget*, per mostrare informazioni o funzionalità essenziali dell'app;
- *Siri*, definire i tastk supportati, validare l'input, eseguire i task e restituire un feedback.
- *status bar*: mostra lo stato corrente, con progress indeterminato per le attività di rete;
- *navigation bar*: in alto. Mostra titolo, pulsante back e alcuni *segmented controls*;
- *toolbar*: contiene bottoni o label per azioni rilevanti nella lista corrente. Max 5 icone o max 3 label di testo;
- *tab bar*: per navigare nelle diverse sezioni. Massimo 5 tab, poi *more*, con icona custom e un label;
- *tabelle*: mostrano contenuto, utili per la navigazione. Ogni riga ha un *title* e un *subtitle* (facoltativo) e possono essere formattate in diversi modi. Possono essere *plain* o *grouped*;
- *alerts*: asincroni, con messaggi brevi da parte del sistema, con 1+ bottoni e di fronte ai quali l'utente deve prendere una decisione;
- *action sheets*: sollecitati dall'utente, dal basso una lista di azioni disponibili e *annulla*;
- *activity views*: per eseguire un task utile nel contesto attuale, come la *condivisione*;
- *controls*: tipi di input, come il seleziona data, il control di *cut/paste/copy*, i tipi di tastiera (numerica, o alfanumerica, ...).