ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIEINFORMATICHE

Il processo di sviluppo incentrato sull'utente: vantaggi, innovazioni e influenze sul metodo tradizionale

Relazione finale in

Mobile Web Design

Relatore

Dott. Mirko Ravaioli

Presentata da

Dario Giardini

Sessione 3

Anno accademico 2012/2013

Indice

Prefazione

Quando si parte nello sviluppo di una nuova applicazione sono molteplici le strade che è possibile percorrere.

Nell'opinione di chi scrive è imperativo che una grossa fetta di tempo e fatica sia investita nello studio anche psicologico dell'utente finale che andrà ad utilizzare il prodotto una volta ultimato.

Esistono a questo proposito vari tipi di design che mirano proprio a definire dei canoni di lavoro a cui attenersi, con particolare riguardo all'utente e alle reazioni che quest'ultimo mostra mentre utilizza vari prototipi dell'applicazione che gli sono somministrati via via durante lo sviluppo.

Lo scopo che si prefigge questa tesi è quello di dare una panoramica attuale su questo aspetto dello sviluppo e riuscire a definire una serie di canoni deputati al conseguimento di uno sviluppo consapevole incentrato sull'utente.

1-Introduzione

L'argomento che si vuole trattare in questa sede è davvero vasto e complesso.

Arriva a comprendere al suo interno nozioni di analisi, progettazione e implementazione: ognuna di queste discipline si interseca in un affresco nel quale alla fine risulta difficile riuscire ad individuare nuovamente le singole parti.

Come introduzione si cercherà di definire ogni singola parte citata sopra e di capire quale sia l'importanza e in seguito quali elementi di ognuna di esse possano essere migliorati dall'approccio incentrato sull'utente.

1.1 Analisi

In un qualunque ambito di sviluppo la precedenza su ogni altra disciplina è dovuta all'analisi dei requisiti.Spesso sottovalutata con risultati finali disastrosi, è la scienza che si occupa di “capire” esattamente cosa è stato richiesto dal cliente e se è fattibile. Questo fa in modo che si possano superare quasi tutte le ambiguità dovute alla comunicazione tra cliente e sviluppatore.Molte volte infatti il cliente non riesce ad esprimere in maniera coerente le proprie necessità, e sta all'analista discutere più e più volte con il committente per cercare di far emergere un quadro del progetto il più chiaro possibile.
L'ultimo compito che l'analista va a svolgere è capire se il progetto è fattibile a fronte di fattori come disponibilità tecnologica, economica e temporale.

1.2 Progettazione

Una volta che l'analista ha dato un segnale positivo per il progetto si può procedere con il primo passo nello sviluppo di un'applicazione. Prima di procedere una precisazione: il processo di sviluppo di un software va visto come una specie di spirale: non esistono fasi che una volta completate siano a tenuta stagna contro variazioni.

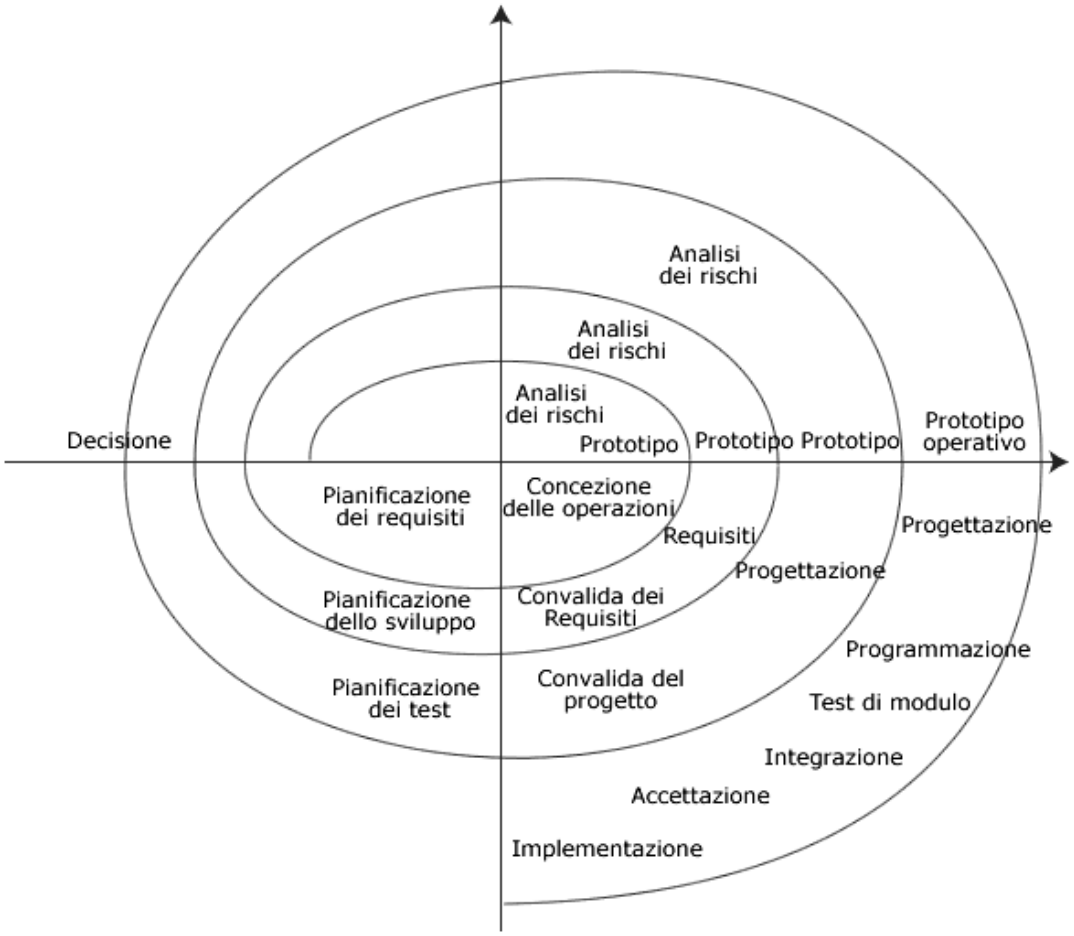
Molte volte si torna indietro ad una fase precedente perchè si è scoperto magari a livello di collaudo interno qualcosa di troppo importante perchè sia ignorato, quindi si procede all'aggiustamento del progetto per poi tornare al passo successivo.

Ma di cosa si occupa esattamente questa disciplina?

La progettazione nasce con lo scopo principale di definire una soluzione ai requisiti emersi in fase di analisi. Si suddivide in molteplici sotto fasi (architetturale, strutturale ecc...) che non andremo a toccare al momento. In linea di massima alla fine della fase di progettazione si arriva ad ottenere un primo schema della struttura dell'applicazione e dei suoi singoli componenti che sono stati individuati (per comprendere questo concetto basti pensare ad un'auto e a quanti componenti interni comprende).

1.3 Implementazione

In questa fase si procede allo sviluppo vero e proprio dell'applicazione. Partendo da tutto ciò che è emerso dalla progettazione i programmatori del team implementano usando uno o più linguaggi di programmazione l'insieme di software che comporranno l'applicazione. Per fare questo esistono vari tipi di ambienti di sviluppo, software creati proprio con lo scopo di rendere sempre più agile e produttiva la fase di implementazione.



Ecco la classica spirale del processo di sviluppo (fonte Wikipedia).

2 - Che cos'è il design incentrato sull'utente?

La prima domanda che vogliamo porci, e sicuramente quella che sarà sorta nella mente del lettore a questo punto, è quella che dà il titolo a questo capitolo.

La risposta paradossalmente si può trovare in un'altra domanda, ben più complessa: come possiamo sviluppare un'applicazione in modo da renderla il più usabile possibile in maniera da sostenere e migliorare l'esperienza e le attività in cui sono coinvolti gli utenti?

> La risposta più ovvia e anche più semplice è: concentrando molti degli sforzi dello sviluppo nella comprensione più ampia e completa dell'utente finale.

Quest'affermazione apre in un certo senso il vaso di Pandora. Chi è l'utente finale? Potremmo rispondere con "chiunque". Ed è vero. Ci sono una moltitudine di diversi tipi di utenti, con un limite che tende a infinito se vogliamo essere davvero precisi. Una casa di sviluppo software può trovarsi davanti al totale smarrimento quando si trova di fronte una richiesta per un'applicazione da destinare ad una fetta di utenza di cui mai si era occupata in precedenza. Un esempio può essere una software house che sviluppa software finanziari per utilizzo interno delle banche. Un giorno una delle banche clienti dell'azienda di sviluppo richiede che sia sviluppato un sistema di home banking di facile utilizzo per gli utenti finali, che non sono esperti di finanza, ma sono persone comuni. La software house ovviamente non rifiuta il lavoro, ma si trova di fronte ad una nuova problematica, che se non venisse considerata avrebbe quasi sicuramente effetti disastrosi sul prodotto finale. Come fare per rendere di facile utilizzo e comprensibile il sistema di home banking?

Ed ecco il ritorno alla domanda complessa posta in fase di apertura del capitolo.

Proviamo a trovare una risposta intuitiva: bisogna cercare di coinvolgere il più possibile l'utente finale in modo da avere continue valutazioni (feedback) sul prodotto, sopratutto in fase di sviluppo.

Possiamo individuare cinque punti fondamentali a questo proposito:

1. Dobbiamo analizzare approfonditamente ciò che gli utenti finali riescono a compiere con più facilità e ciò su cui invece hanno maggiori problemi.
2. Cercare di capire cosa potrebbe aiutare le persone nelle loro attività abituali.
3. Ragionare per capire cosa potrebbe fornire all'utente finale un'esperienza di qualità e di valore.
4. Il punto forse più fondamentale: ascoltare gli utenti finali, coinvolgerli nel processo di progettazione.
5. Utilizzare le tecniche più consolidate per la progettazione incentrata sull'utente (verranno approfondite più avanti).

Prima di partire in un'analisi più approfondita, si fornisce un esempio ancora più lampante che mira a far comprendere al lettore l'importanza dei suddetti punti.

2.1 **Good Design e Bad Design**

Una delle prime discipline che uno sviluppatore deve apprendere prima di poter padroneggiare il design incentrato sull'utente è riuscire a valutare secondo termini di paragone ben definiti un “Buon Design” da un “Cattivo Design”.

Per capire esattamente di cosa si sta parlando partiamo dall'esempio di un'applicazione per smartphone che si occupi di gestire la perdita di peso progressiva tramite il continuo esercizio fisico. Per questo esempio osserviamo una prima versione dell'interfaccia utente che gestisce la registrazione del peso in una certa data e che permette di visualizzare il peso in tutte le date precedenti. Osserviamo:

Che cosa notiamo immediatamente?

L'interfaccia non è per nulla di facile utilizzo. La prima cosa da appuntare a questo design è la scelta di “nascondere” il pulsante *Record* nella linguetta inferiore del riquadro principale. Una scelta che annienta completamente l'intuitività della

schermata, in quanto se fosse apparso come un classico pulsante magari con effetto in rilievo sarebbe stato semplicissimo da individuare per chiunque.

Passando oltre, possiamo vedere come nella parte superiore la scelta della data sia inutilmente macchinosa: abbiamo due "nastri" da fare scorrere per selezionare mese e giorno. Questo tipo di interfaccia non bada assolutamente all'usabilità del prodotto, in quanto è assai prevedibile che l'utente medio avrà un numero spropositato di problemi nel fare scorrere con precisione i due "nastri" nella giusta posizione tramite il touchscreen.

La stessa cosa accade per la parte inferiore dove è possibile inserire il peso: anche qui abbiamo una serie di "rulli" da fare scorrere per selezionare le cifre e l'unità di misura.

Il tutto si traduce in un'operazione che può far perdere qualche minuto all'utente quando dovrebbe protrarsi solo per qualche secondo.

Proviamo invece ora ad analizzare una seconda versione della schermata:

Le differenze con quella precedente sono marcate: innanzitutto la domanda a caratteri grandi "Qual è il tuo peso oggi?" è chiarissima sull'utilità della schermata, dando all'utente la possibilità di inserire subito sotto il peso tramite un classico inserimento a tastiera, molto più utilizzato e veloce dei "nastri" della versione precedente.

Un'altra differenza sostanziale è l'impatto del pulsante verde “Save” che permette di salvare il peso. Questo si impone in maniera molto più vistosa rispetto alla linguetta “record”, stimolando l'utente ad utilizzarlo anche per il fatto che è l'unico pulsante presente nella schermata.

La parte inferiore mostra l'insieme di date e pesi che è possibile scorrere in modo da osservare la progressione dell'utente. Anche qui un semplice scorrimento permette una consultazione estremamente rapida e precisa rispetto alla selezione per “nastri”. In più un piccolo grafico nella parte più bassa permette di avere un'idea intuitiva dei progressi e della costanza di utilizzo dell'applicazione e di perdita del peso.

Dopo queste semplici considerazioni è abbastanza semplice comprendere che, se non si osservano anche solo questi piccoli accorgimenti di sviluppo, la risposta dell’utenza ad un’applicazione può variare enormemente.

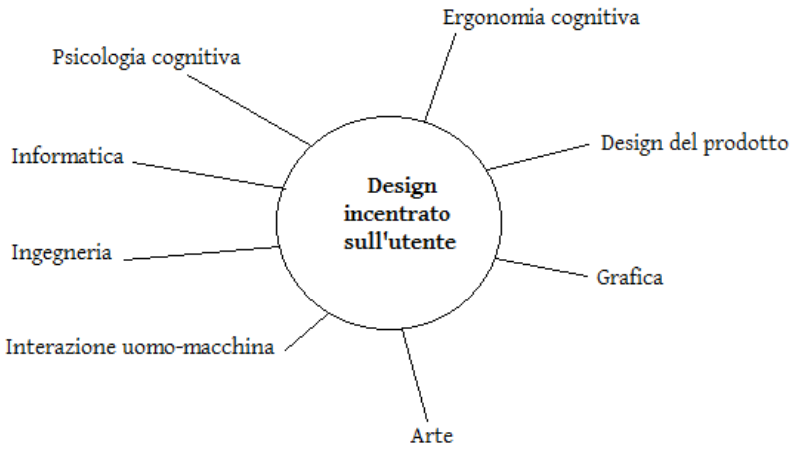
2.2 **La nuova figura lavorativa**

Dopo aver esplicitato tramite l’esempio precedente alcune delle problematiche che sorgono in fase di sviluppo, vogliamo presentare la figura lavorativa che si occupa proprio di studiare la disciplina di questa tesi: il designer delle interazioni.

Questa figura è emersa recentemente a causa della continua rincorsa degli sviluppatori e consulenti web per trovare soluzioni sempre più usabili e migliori, che offrano all’utenza servizi molto più appetibili rispetto a quelli di qualche anno (o addirittura mese) prima.

Assistiamo quindi alla nascita di progettisti e consulenti delle interazioni con l’utenza: le aziende si rivolgono ad essi già dalle prime fasi di sviluppo, in modo da poter ottenere un continuo feedback e progressione nella costruzione di una nuova applicazione.

Osserviamo ora le discipline di cui dev’essere dotato un buon designer delle interazioni:



Spiegheremo con precisione alcune di queste dicipline nei prossimi capitoli, basti sapere al lettore che recentemente alcune aziende software e web-mobile web hanno dovuto chiudere i battenti a causa di applicazioni su cui erano state investite somme davvero considerevoli di denaro e che si sono rivelate poi totalmente fallimentari una volta giunte sul mercato.

Che cos'è andato storto? Il lettore ormai avrà capito a questo punto: nessuna di queste aziende si è rivolta o ha ricevuto scarse consulenze in materia di marketing ma sopratutto in materia di design per gli utenti. Infatti nell'ultima figura possiamo notare una disciplina fondamentale che a sua volta ne comprende davvero molte al suo interno: il design del prodotto. Un designer delle interazioni secondo chi scrive deve assolutamente avere questa diciplina molto generale alla base della sua formazione. Essa infatti comprende al suo interno concetti relativi al marketing e alle mode dell'utenza che possono arrivare a salvare un intero progetto prima del tracollo definitivo.

Far interagire tra loro design e studio del mercato è la combinazione vincente per ogni azienda software che si rispetti o che non desideri fallire in breve tempo. Abbiamo assistito di recente grazie alla nascita dei vari Apple Store, Google Play, ecc... alla formazione di un numero spropositato di software house cosidette "indipendenti". Ebbene, basta sfogliare leggermente le pagine di uno qualunque di questi negozi on line di applicazioni mobile per rendersi conto di quanto, grazie ad uno studio più accurato di mercato e di design, moltissime applicazioni potrebbero rendere almeno il doppio in termini di vendite e di download.

Ovviamente si potrebbe obbiettare che reclutare una figura del genere per una software house indipendente sia un investimento troppo costoso, ma secondo chi scrive è necessario che uno o più membri del team di sviluppo abbiano almeno qualche nozione delle discipline elencate più sopra, pena un totale affidamento alla casualità quando si decide in fase di progettazione cosa/come sviluppare un'applicazione.

Uno degli scopi principali che ci si è posti nella stesura di questo documento è infatti sensibilizzare il lettore che vuole affacciarsi allo sviluppo su questo tipo di problematiche, e sul guadagno che un tale investimento di tempo e/o denaro può portare a lungo termine nello sviluppo di qualunque applicazione.

2.3 **Le fasi di uno sviluppo incentrato sull'utente**

Vediamo ora più nel dettaglio le fasi di un tipico sviluppo incentrato sull'utente, detto anche design incentrato sulle interazioni.

Possiamo individuare quattro fasi fondamentali:

1- Identificare e definire i requisiti (fase di analisi dei requisiti comune a tutti i processi di sviluppo software).
2- Sviluppare una serie di proposte alternative in grado di soddisfare i requisiti individuati nella fase precedente (fase di analisi del problema, anche questa comune a tutti i processi di sviluppo software).
3- Costruire dei prototipi interattivi delle proposte che permettano di comunicarle all'utente e di farle valutare.
4- Una continua valutazione iterativa dei risultati durante il processo di sviluppo, un feedback continuo.

Le prime due fasi sono state esplicitate nel capitolo di introduzione, in quanto comuni a praticamente tutti i processi di sviluppo software esistenti.

Quelle che ci interessano ora sono le ultime due fasi. Partiamo dalla terza e vediamo di capire che cosa significa.

In un processo di sviluppo normale questa fase di solito viene deputata allo sviluppatore che utilizza i cosidetti "piani di collaudo" per verificare il corretto funzionamento dei risultati ottenuti nelle fasi precedenti. In tutto questo si cerca anche di tenere conto di tutti i requisiti individuati progressivamente, e in caso ne emergano di nuovi, tornare iterativamente alle fasi precedenti per integrarli. Quest'ultima attività secondo chi scrive vede un aumento sostanziale di efficacia nel caso la si esegua a stretto contatto con l'utenza.

In questo caso, ad ogni costruzione di un prototipo si deve tenere conto del fatto verrà somministrato in vari modi all'utenza finale. La valutazione del lavoro svolto da parte dell'utenza (quarta fase)rappresenta l'essenza del design incentrato sull'utente.

Come possiamo coinvolgere l'utente in un processo di sviluppo?

Esistono svariate modalità:

Può esservi l'osservazione con partecipazione alle attività degli utenti mentre utilizzano un prototipo, che richiede conoscenze di psicologia cognitiva e capacità di valutazione delle reazioni.

Si possono somministrare interviste per avere valutazioni più precise su certi ambiti del progetto che non sono completamente chiari allo sviluppatore.

E’ molto utile anche somministrare dei test per osservare come gli utenti risolvono un compito eseguibile tramite un prototipo e in quanto tempo. Questo è uno degli aiuti fondamentali che il design incentrato sull’utente fornisce agli sviluppatori. Infatti possiamo ottenere un grandissimo numero di informazioni anche da un solo test. Questo ci aiuta a comprendere meglio come l’utente si pone di fronte a certe attività e quindi come lo sviluppatore può modificare il progetto in modo da renderlo usabile in maniera sempre più efficace.

Un ultimo metodo per poter coinvolgere l’utenza è anche quello più esplicito: si cerca infatti di far entrare l’utente nel processo di design, detto quindi co-design, tramite la somministrazione continua di questionari sulle necessità e modalità in cui un utente preferirebbe vedere le funzioni richieste in fase di analisi di requisiti. Questo tipo di approccio va eseguito anche prima di cominciare la progettazione vera e propria, in quanto permette di indirizzare gli sforzi dello sviluppatore ma soprattutto dell’analista dei requisiti nel cercare di esplicitare ogni tipo di richiesta anche non pervenuta direttamente dal committente dell’applicazione.

La stessa importanza di un coinvolgimento dell’utenza nel processo di sviluppo si trova nella comprensione approfondita delle attività che svolge l’utenza stessa.

Questo andrebbe fatto prima di ogni altra cosa, andandosi a installare come una sottofase dell’analisi dei requisiti. Si cerca infatti di osservare come l’utente interagisce con altri utenti, come interagisce con l’informazione, e come interagisce con la tecnologia.

Tutto ciò dev’essere fatto alla luce delle enormi differenze che si trovano in utenze diverse.

Un bambino ha bisogni totalmente differenti da quelli di un adulto, e durante una fase di analisi dei requisiti anche un esempio semplice come questo porta a considerazioni che mutano completamente il lavoro di esplicitazione delle richieste. Lo si può vedere nei prodotti alimentari come esempio: anche solo il logo sulla scatola di un cibo per bambini è strutturato in maniera appariscente e colorata e magari con dei fumetti per stimolare la psiche dell’infante, dove un adulto invece reputa noiose e inutili certe cose, rivolgendosi ad un altro tipo di prodotti.

Quindi così come si disegnano giochi, vestiti e cibo in maniera differente per adulti e bambini, così il design incentrato sull’utente cerca di adattare ogni progetto di sviluppo alla tipologia di utenza e alle necessità che questa si trova di fronte.

La fase successiva alla comprensione dei bisogni e capacità dell’utenza ci pone di fronte al dover definire degli obiettivi da raggiungere per ottenere il massimo dell’usabilità e coinvolgimento degli utilizzatori finali.

2.4 **Gli obiettivi del design incentrato sull'utente**

Innanzitutto è bene definire due tipi principali di obiettivi che si possono definire nella fase di analisi dei requisiti.

Possiamo infatti individuare due "macro" categorie, ossia:

1- Gli obiettivi di usabilità.
2- Gli obiettivi di esperienza d'uso

Gli obiettivi di usabilità sono tutti quelli che si pongono per cercare come dice il nome stesso di rendere l'applicazione il più usabile e semplice possibile.

Quelli di esperienza d'uso invece puntano invece alla qualità dell'esperienza che l'utente avrà nell'utilizzo dell'applicazione, ad esempio rendere piacevole a livello estetico-grafico l'applicazione per venire incontro ai gusti dell'utente.

2.4.1 **Gli obiettivi di usabilità**

Come già definito sopra gli obiettivi di usabilità cercando di migliorare al massimo le interazioni dell'utente con l'applicazione che stiamo sviluppando. Questo significa studiare come l'applicazione possa aiutare l'utenza nel proprio ambito lavorativo o quotidiano e quindi riuscire a migliorare l'esperienza dell'applicazione tenendo conto di tutto ciò. A questo proposito possiamo individuare questi obiettivi:

1- Efficacia.
2- Efficienza d'uso
3- Sicurezza d'uso
4- Utilità
5- Facilità di apprendimento
6- Facilità di ricordo

Efficacia

Partiamo parlando dell'efficacia: questo è un obiettivo molto generico ma fondamentale in quanto si vuole cercare di rendere l'applicazione in grado di fare ciò per cui è stata pensata e progettata.

Efficienza d'uso

L'efficienza d'uso si pone invece il problema di riuscire a rendere un'applicazione in grado di aiutare l'utente a portare a termine le proprie attività e mansioni, in modo da facilitarne il completamento. Un esempio di questo è l'applicazione di gestione del peso corporeo citata in apertura di capitolo in materia di good design e bad design. La prima versione complica l'utente in maniera esagerata prolungando anche il

tempo di utilizzo a qualche minuto invece di qualche secondo. Al contrario la seconda versione punta a rendere estremamente semplice e veloce l'interazione, e quindi, efficace. Possiamo trovare un esempio lampante dell'efficienza d'uso nei classici siti internet che permettono la memorizzazione dei dati degli utenti nei form di compilazione, ad esempio in un negozio on line. Questo fa risparmiare all'utente abituale un enorme quantitativo di tempo, visto che non è più costretto ad inserire ogni volta tutti i dati personali per fare un ordine. Di recente un esempio ancora migliore è quello di Amazon, che permette addirittura di gestire gli ordini con un unico clic del mouse su un pulsante che va a ricercare da solo tutti i dati necessari dell'utente già memorizzati sul sito.

Sicurezza d'uso

La sicurezza d'uso è un'altra branca fondamentale dell'usabilità: si pone infatti il problema di far percepire all'utente dell'applicazione un ambiente sicuro, in grado di rimediare ai propri errori di inesperienza nell'utilizzo del software. In cosa si traduce tutto questo? Un esempio sono le finestre di dialogo che chiedono conferma per un'azione. Senza di esse un clic sbagliato potrebbe causare una perdita di dati irreparabile che in casi estremi potrebbe portare ad un disastro lavorativo. La sicurezza d'uso però non si ferma a questo, ma porta anche a considerazioni chiave come il fatto di non mettere il tasto "chiudi" esattamente accanto a quello "salva", configurazione che causerebbe palesi fastidi all'utilizzatore. Si tratta quindi di rendere l'ambiente dell'applicazione intuitivo e non spaventoso. Deve stimolare l'utente all'esplorazione delle funzioni e non ad avere paura di un pulsante sconosciuto. Un'ultima considerazione è chiedersi se il sistema permette all'utente di riparare ai propri errori una volta che tutto il resto ha fallito. Ci sono casi in cui aggiungere una funzione del genere è estremamente difficoltoso, ma i vantaggi che porta hanno un peso troppo grande per non essere considerati (si veda appunto il caso di errori lavorativi con esiti disastrosi).

Utilità

L'utilità è un obiettivo abbastanza scontato ma che a volte viene messo in secondo piano senza accorgersene. Ci si pone in questo caso la questione di rendere l'applicazione il più "utile" possibile alle attività dell'utente per cui questa è rivolta. Ciò significa ad esempio agevolare certe attività ricorrenti composte da passi ripetitivi che possono essere "bypassati" in modo da far risparmiare tempo e problemi all'utente. Un esempio può essere un'applicazione bancaria che permette di visualizzare il saldo e i gli ultimi movimenti settimanali di denaro del proprio conto, che dispone anche di una funzione di ricerca dei movimenti stessi, rendendola appunto più utile. Un esempio contrario invece può essere un programma di fotoritocco che permette di compiere ritagli sulle foto solo partendo da forme predefinite e non potendo usare il ritaglio a mano libera col mouse, trasformando l'assenza di questa funzione in un enorme deficit per l'utente che utilizza il programma.

Facilità di apprendimento

La facilità di apprendimento può essere l'aspetto più determinante nel successo o nel fallimento di un'applicazione. Dopotutto, se l'utente medio non è disposto ad imparare e quindi utilizzare investendo il suo tempo sull'applicazione che abbiamo sviluppato tutto il lavoro svolto sino a quel momento sfuma nel nulla.

Quante volte è successo nella nostra vita: proviamo qualcosa di nuovo, non stimola la nostra curiosità, appare difficoltoso da utilizzare e sopratutto ci fa perdere un quantitativo inutile di tempo nel cercare di comprenderne il funzionamento. La conseguenza più ovvia? Lasciamo perdere e ci rivolgiamo ad altro. Nel mondo di oggi abbiamo a disposizione migliaia e migliaia di applicazioni diverse per smartphone, ne possiamo provare a centinaia e scartarne la metà dopo appena qualche minuto di utilizzo, rivolgendoci ad altro, a qualcosa di più semplice da imparare, qualcosa più facile da utilizzare. La regola fondamentale in questo caso è che l'utente deve essere in grado sin da subito di svolgere le sue attività in completa autonomia utilizzando l'applicazione. Un primo canone a cui attenersi per evitare lo spaesamento dell'utente è capire che cosa l'utente *sa* prima di utilizzare il nostro prodotto. Un ingegnere meccanico non ha bisogno che l'applicazione gli insegni la meccanica, ma possiamo venirgli incontro capendo quali siano le sue conoscenze pregresse e utilizzare proprio queste per guidarlo nell'apprendimento progressivo dell'applicazione. Una regola molto semplice che riassume tutti i concetti sin'ora esplicati è "La regola dei dieci minuti". Questa pone come limite massimo di apprendimento per un nuovo prodotto (e nel caso delle applicazioni mobile secondo chi scrive questa regola si applica *sempre*) esattamente dieci minuti. Se l'utente medio impiega più tempo allora possiamo stare certi che qualcosa va modificato. A volte può sorgere il dubbio di aver inserito troppe funzionalità e che molte di queste non forniscano un aiuto realmente considerevole all'utente, andando invece a complicare il suo breve percorso di apprendimento dell'applicazione. In questi casi la rimozione o il loro snellimento è un imperativo, pena il fallimento della regola dei dieci minuti.

Facilità di ricordo

L'ultimo obiettivo pone l'accento sulla facilità di ricordare una funzione usata saltuariamente, in modo infrequente. L'utente non può e non deve imparare ogni volta come si usa una certa funzione dell'applicazione che stiamo sviluppando. Bisogna cercare di rendere le operazioni meno utilizzate facili da ricordare anche a distanza di giorni e mesi dal loro ultimo utilizzo. Come fare? Ci sono svariati modi: uno può essere quello di dare un nome significativo alla funzione. Troppe volte molti utenti rimangono confusi da sigle astruse o illogiche poste sui pulsanti di un'applicazione, rendendo difficile anche solo ricordarsi a cosa servano. Un altro modo può essere quello di disporre le funzioni in un ordine intuitivo o nello stesso spazio. Un esempio può essere raggruppare tutte le icone di ritaglio di un software di fotoritocco in un unico spazio, agevolando la ricerca dell'utente che non ricorda dove trovare una certa funzione.

Un esempio vicino a tutti di questo è la classica disposizione dei pulsanti play-rewind-forward.

Qual è la disposizine naturale ed intuitiva di questi pulsanti che trovate praticamente su ogni dispositivo-programma-applicazione che li sfrutta?

Ovviamente la prima: è strutturata in maniera da rendere intuitivo l'utilizzo dei tre pulsanti, rendendo chiaro subito il significato tramite i simboli posti su di essi, ma esplicitandone completamente le funzioni grazie alla loro disposizione: quindi il tasto posto più a sinistra permette di riavvolgere il file multimediale, e quello più a destra permette di mandarlo avanti. Osserviamo che questo esempio vale anche oltre l'uso saltuario di un'applicazione in quanto la disposizione giusta di questi pulsanti è divenuta una convenzione comune a tutti i prodotti multimediali oramai. L'utente già sa in anticipo come saranno disposti questi pulsanti, e trovarli disposti arbitrariamente come nella seconda figura può portare a casi assurdi come l'utente sovrappensiero che preme il tasto per mandare avanti il file multimediale quando invece voleva premere play, rimanendo ingannato dalla disposizione invertita rispetto a quella convenzionale. Questo è un esempio semplice, ma basta pensarlo applicato al mondo del lavoro per immaginare il disagio che una tale dimenticanza in fase di sviluppo può portare.

2.4.2 **Gli obiettivi di esperienza d'uso**

Quelli che ci apprestiamo a definire sono obiettivi più difficili da individuare, perchè basati più sulla psicologia intrinseca dell'utente, e quindi richiedono un'analisi più ampia e meno tecnica rispetto a quelli di usabilità.

Quasi come l'altra faccia della medaglia dell'usabilità, troviamo tutta una serie di aspetti qualitativi che messi insieme aiutano un'applicazione già di per sè usabile a diventare un must per l'utente medio, che si troverà praticamente invogliato a riutilizzare l'applicazione senza accorgersene, migliorandone in questo modo l'uso e l'apprendimento.

Si possono individuare i seguenti obiettivi per l'esperienza d'uso:

1- Trasmettere soddisfazione nell'uso del prodotto.
2- Rendere il prodotto piacevole da utilizzare.
3- Rendere l'applicazione divertente.
4- Creare l'applicazione in modo che sostenga la motivazione dell'utente durante il suo utilizzo.
5- Rendere il prodotto esteticamente gradevole.
6- L'applicazione dovrebbe essere in grado di stimolare la creatività dell'utente (si veda anche il coinvolgimento nel processo di sviluppo).
7- L'uso dell'applicazione dovrebbe essere gratificante.
8- Il prodotto dovrebbe essere in grado anche di soddisfare dei bisogni legati alla sfera delle emozioni dell'utente che lo utilizza.

Si può intuire la differenza degli obiettivi di esperienza d'uso e di quelli di usabilità in base alla soggettività dei primi e all'oggettività dei secondi.

Abbiamo infatti che gli obiettivi di esperienza d'uso richiedono un'analisi soggettiva e accurata dell'utente in modo da costruire un'applicazione creata su misura per le sue esigenze. Una calcolatrice per bambini delle elementari sarà pensata in modo da stimolare la curiosità del bambino e premiarlo dove riuscisse a fare una scoperta, incentivandolo ad una continua esplorazione dell'applicazione fino al suo completo padroneggiamento. Ci troviamo quindi in questo caso a cercare di analizzare l'*esperienza* dell'utente in modo soggettivo, laddove invece l'usabilità misura la capacità dell'applicazione di essere utile e quindi di rendere l'utente produttivo.

E' quindi molto difficile trattare in maniera tecnica e ben definita tali obiettivi, che verranno trattati approfonditamente nel capitolo riservato alla comprensione e analisi dell'utente.

Un enorme studio sulla soggettività dell'esperienza è stato effettuato negli anni dalle industrie dei videogiochi e dell'intrattenimento interattivo in generale.

Ciò che si è evinto da questi studi è la totale impossibilità di implementare tutti gli obiettivi di usabilità e di esperienza d'uso allo stesso tempo. E' abbastanza intuibile che sviluppare un'applicazione che faccia dell'usabilità il suo cavallo di battaglia non

possa essere studiata per essere un divertimento continuo. Un'applicazione finanziaria potrà essere benissimo resa soddisfacente nell'uso, ma cercare di renderla divertente per l'utenza a cui si rivolge potrebbe causare un crollo catastrofico degli obiettivi di usabilità.

Lo sviluppatore deve comprendere che ogni prodotto va valutato singolarmente o in base al dominio di appartenenza, e una volta fatta un'analisi approfondita e capito il dominio e i requisiti cominciare a porsi gli obiettivi più adatti al caso in cui si trova.

Riuscire a trasmettere piacere nell'uso di un'applicazione è secondo chi scrive una componente fondamentale nello sviluppo di un'applicazione, e questa tesi si concentrerà nei prossimi capitoli anche su questo aspetto.

In chiusura di capitolo si propone uno schema riassuntivo che mostra tutti gli obiettivi di usabilità e di esperienza d'uso citati in precedenza secondo un ordine che diminuisce di priorità andando verso l'esterno.

3 - Comprendere gli utenti

Dopo aver esplicitato i principali obiettivi dello sviluppo incentrato sull'utente vogliamo dedicarci alla prima macro-fase del processo di sviluppo, ossia porre tutti gli sforzi iniziali nell'analisi comportamentale e psicologica dell'utente in relazione all'applicazione che svilupperemo.

Questa fase si interseca in modo quasi inscindibile con la classica fase di analisi dei requisiti, in cui si cerca di esplicitare tutte le richieste del committente dell'applicazione. Nel caso dello sviluppo incentrato sull'utente avremo si dei requisiti da rispettare, ma ci affideremo anche e sopratutto ai risultati di un'analisi psicologica e cognitiva dell'utente medio che andrà ad utilizzare il nostro prodotto.

Secondo chi scrive in un prossimo futuro la figura dell'analista dei requisiti e quella dello psicologo-psicologo cognitivo verranno assimilate in un'unica figura professionale distaccata dal designer delle interazioni, in grado di ritagliarsi un proprio spazio nel mondo del lavoro.

Prima di ogni ulteriore passo vogliamo dare la definizione di ciò che sta alla base di qualunque studio sull'utenza: la cognizione.

3.1 **La cognizione**

La cognizione è ciò che avviene nella nostra mente quando eseguiamo un'attività quotidiana. Include svariati processi cognitivi, come pensare, ricordare, imparare, sognare ad occhi aperti, riuscire a prendere decisioni, vedere, leggere, scrivere e parlare.

Lungo gli anni di studi in materia si è riusciti a suddividere la cognizione generale in due sottocategorie:

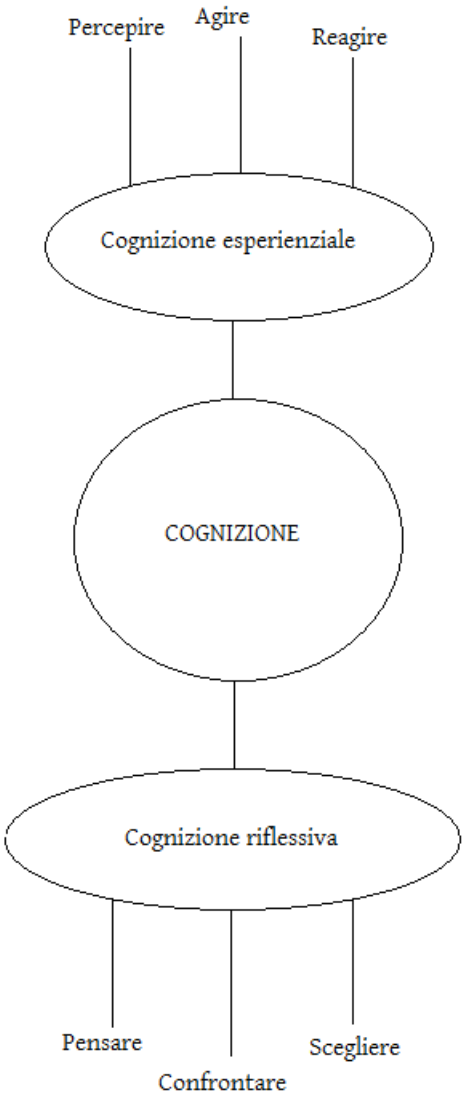
1- La cognizione data dall'esperienza
2- La cognizione riflessiva

La prima categoria si basa sullo stato mentale in cui percepiamo, agiamo, e reagiamo agli eventi che ci circondano in maniera efficace e senza sforzo. Questo include esempi come guidare un automobile, leggere un libro, avere una conversazione, giocare a qualcosa come un videogioco. Ciò che si può dedurre è quindi che la cognizione data dall'esperienza richieda un certo grado di coinvolgimento e ripetizioni di una stessa azione, fino al suo completo padroneggiamento, e quindi, al raggiungimento di risultati senza impiegare energie eccessive ma solo quelle strettamente necessarie all'attività che si sta svolgendo. Questo processo avviene in maniera totalmente automatica: quante volte ci accorgiamo di come agisce il nostro corpo mentre guidiamo? Lo notiamo solo mentre stiamo imparando, mentre ancora spendiamo un enorme quantità di energie nel tentativo di fare nostri quei movimenti, quelle nozioni. Una volta che questi sono stati assimilati, la nostra mente è in grado di recuperarli senza sforzo, permettendoci una coordinazione di cui non siamo neppure consapevoli nella maggior parte dei casi.

Al contrario della prima, la cognizione riflessiva include azioni come pensare, confrontare, e scelte decisionali. A differenza di ciò che otteniamo tramite l'esperienza diretta, questo tipo di cognizione è ciò che porta a nuove idee e in generale accende la creatività che è in noi. Esempi generici includono il design di un prodotto, la scrittura di un libro, l'organizzazione di un nuovo evento. Si potrebbe arrivare a dire che la cognizione riflessiva è ciò che porta all'*ispirazione*, e quindi, in un certo senso, a nuove motivazioni, nuove emozioni, nuovi coinvolgimenti per l'utente.

La psicologia cognitiva afferma che ambedue questi aspetti della cognizione sono indispensabili per la vita quotidiana e fanno parte di noi dall'inizio dei tempi.

Ciò che gli studi successivi hanno invece puntualizzato è che ognuna di queste due categorie di cognizione richiede un differente approccio di design, ed è ciò di cui ci andremo ad occupare in questo capitolo.



3.2 **Il design applicato ai processi cognitivi**

Vogliamo ora concentrarci come annunciato sopra sulle specifiche di design che sono state individuate lungo anni di studi applicati alla cognizione. Per farlo vogliamo cercare di isolare a livello base i principali processi cognitivi di un utente.

Si possono individuare i seguenti processi cognitivi:

1- Attenzione
2- Percezione e riconoscimento
3- Memoria
4- Apprendimento
5- Lettura, conversazione e ascolto
6- Risoluzione di problemi, pianificazione, ragionamento e scelte decisionali

Attenzione

L'attenzione è un processo cognitivo basato sulla selezione di stimoli e informazioni, solitamente visuali o uditive, a seconda della situazione in cui ci si trova. Il classico esempio è l'attesa dal medico o in un ufficio burocratico ponendo attenzione all'altoparlante che deve chiamare il nostro numero o il nostro nome. Possiamo completamente isolare le nostre percezioni quando sentiamo la voce provenire dall'altoparlante, ignorando ogni altro stimolo nel tentativo di comprendere a fondo le parole. Un altro esempio, stavolta visuale, è la ricerca di un numero di telefono sull'elenco telefonico: la nostra vista seleziona immediatamente le zone in cui troviamo la lettera iniziale della persona di cui stiamo cercando il numero, riuscendo via via ad isolare informazioni sempre più precise, ed eventualmente trovando il numero cercato. La facilità o difficoltà dell'esecuzione di questo tipo di processo si trova in due aspetti fondamentali: a seconda che l'utente abbia o meno un chiaro obiettivo e se l'informazione che sta cercando è risulta ben visibile e appariscente nel dominio in cui sta cercando.

Questo ci porta a definire i seguenti punti fondamentali di design relativi a questo processo cognitivo:

1- Rendere l'informazione saliente quando ne è richiesto il suo utilizzo o apprendimento durante una certa fase dell'attività dell'utente che sta utilizzando la nostra applicazione.

2- Per ottenere l'obiettivo definito al punto sopra usare tecniche come grafica animata, colori, sottolineature, ordinamento degli strumenti, sequenziare informazioni differenti e separare gli strumenti adibiti ad usi diversi.

3- Evitare di riempire l'interfaccia con troppe informazioni, ad esempio utilizzando troppi colori, suoni e grafica, facendo risultare l'interfaccia dell'applicazione un ammasso di distrazioni fastidiose che fuorviano l'utente catalizzando la sua attenzione in modo errato, finendo per condurlo continuamente in errore mentre utilizza la nostra applicazione.

4- Le interfacce di natura chiara e semplice sono molto più facili da utilizzare, in quanto nella loro sobrietà permettono all'utente di trovare molto più facilmente le informazioni che cerca, aiutandolo a focalizzare l'attenzione esclusivamente sulle informazioni di cui ha bisogno in quel frangente. L'esempio più lampante è la pagina principale del motore di ricerca Google, che permette di individuare immediatamente lo spazio dove digitare le informazioni che cerchiamo.

Percezione

La percezione di basa sul processo di come l'informazione è acquisita dall'ambiente, attraverso i diversi sensi, come tatto, vista, udito ecc... e trasformata nell'esperienza di eventi, suoni, gusti, oggetti. E' un processo davvero elaborato, che include al suo interno altri come memoria, attenzione e linguaggio. Nel nostro caso, ossia lo sviluppo di un'applicazione, il senso predominante è ovviamente la vista, seguita dal suono. E' importante capire come presentare l'informazione in modo da essere prontamente percepita nella maniera intesa dallo sviluppatore, senza causare quindi incomprensioni o grattacapi all'utente sul significato di una certa icona, scritta, o simbolo. L'esempio classico che abbiamo al giorno d'oggi è il design delle icone per applicazioni mobile. Da quando Apple ha presentato IOS per i suoi smartphone il design delle icone è diventata una parte fondamentale nel riconoscimento veloce di un'applicazione mobile. Il concetto principale in questo caso è rendere ogni icona distinguibile dalle altre, in modo che possa essere riconoscibile direttamente alla vista, senza nemmeno bisogno di leggere, ovviamente deve risultare anche assolutamente significativa per ciò che deve rappresentare. Un'altra parte fondamentale associata al processo di percezione è la sincronizzazione di vari media nella nostra applicazione. Se suoni e immagini vengono usati insieme in maniera logica la potenza percettiva del nostro prodotto aumenta esponenzialmente: possiamo pensare anche semplicemente al classico suono di operazione andata a buon fine quando portiamo a termine un'attività. Il suono potrebbe apparire inutile, ma agisce come ulteriore conferma che l'operazione è andata a termine positivamente, trasmettendo all'utente anche soddisfazione (si vedano gli obiettivi di esperienza d'uso del capitolo precedente) oltre all'associazione di quel suono per tutte le altre operazioni andate a buon fine, agevolando l'apprendimento progressivo dato dall'uso del nostro prodotto.

Anche in questo caso cerchiamo di individuare i punti fondamentali di un buon design che tenga conto del processo di percezione:

1- Le icone e altre rappresentazioni grafiche devono permettere all'utente di individuarne prontamente il significato e utilità, pena una perdita di efficacia d'uso enorme.

2- I suoni devono essere sempre chiari, udibili e distinguibili in modo da permettere all'utente di capire cosa rappresentano e associarli ad una situazione specifica della nostra applicazione. Questo implica anche l'obbligatorietà di utilizzare suono differenti per situazioni differenti se non si vuole causare confusione all'utilizzatore.

3- Nel caso in cui vi siano informazioni trasmesse tramite voce l'utente dovrebbe essere sempre in grado di capire il significato delle singole parole che vengono dette (il classico caso di adattamento ad un linguaggio più user-friendly, quindi senza usare troppi termini specifici ma un registro più vicino alla tipologia di utente che utilizzerà l'applicazione).

4- Il testo dovrebbe essere sempre leggibile e soprattutto distinguibile dallo sfondo dell'applicazione (il classico esempio di combinazioni di colori errate, ad esempio nero su sfondo bianco sarà sempre corretto, mentre un testo giallo su uno sfondo bianco o verde è chiaramente errato).

Memoria

Il processo cognitivo della memoria include il richiamo di vari tipi di conoscenze che ci permettono di agire adattandoci al contesto e alla situazione. E' un processo davvero versatile, che permette ad una persona di svolgere una varietà di cose. Un classico esempio è il caso in cui riconosciamo qualcuno incontrato per strada: la prima fase del processo è il riconoscimento della faccia, poi il ricordo del nome associato a quella persona, il ricordo di quando l'abbiamo vista l'ultima volta e cosa ci siamo detti. In breve, senza memoria l'uomo non potrebbe mai funzionare, è una parte inscindibile del nostro essere, e senza di essa non sarebbe possibile nemmeno comprendere questa tesi. Quello che ci interessa capire in questa sede è che ovviamente non è possibile per il cervello umano ricordare ogni cosa che ci è capitata nella nostra vita, ma come funziona il processo che filtra l'informazione che ci viene trasmessa come "importante" e quindi ulteriormente catalogata e memorizzata, invece di essere scartata. Come sappiamo però, questo processo mentale non è esente da problemi: molte volte non ricodiamo cose di cui necessitiamo nella situazione in cui ci troviamo, o addirittura cose a cui teniamo vengono dimenticate come se la loro importanza fosse totalmente relativa per il subconscio.

Ci si vuole concentrare quindi sul funzionamento di questo sistema di filtraggio della memoria. Come funziona? Come prima cosa avviene la codifica, che determina quale informazione troviamo nell'ambiente e come viene interpretata. Se ci concentriamo maggiormente su una cosa il processo di codifica dell'informazione viene prolungato e diviene più elaborato, portandoci a riflettere e a confrontare ciò che abbiamo appena acquisito con conoscenze pregresse. Tutto questo porta ad una memorizzazione delle informazioni ben più efficace, come il lettore può ben immaginare. In questo caso un esempio ci viene dato dalla psicologia dell'apprendimento, che afferma quanto segue: l'apprendimento attivo per esperienza diretta (quindi la trasmissione di informazioni) risulta molto più efficace dell'apprendimento passivo (testi o video sono considerati mezzi passivi). Ecco quindi che a stimolare l'apprendimento di una nuova materia scolastica vi sono gli esercizi, che aiutano a farci riflettere sui concetti. Anche la discussione attiva con altri dei concetti permette una memorizzazione molto più efficace. Un ultimo fattore che influenza le modalità di memorizzazione dell'utente è il contesto in cui questi si trova quando avviene la fase di codifica dell'informazione. Accade che molti utenti non ricordino il funzionamento di qualcosa finchè questa non gli viene riproposta nella stessa maniera in cui l'hanno codificata la prima volta.

Abbiamo quindi appurato che la fase cruciale per la memorizzazione di qualcosa da parte dell'utente è la codifica dell'informazione, con tutti gli aspetti che ne derivano. Tramite questi è possibile definire i punti cardine del design basato sullo studio della memorizzazione:

1- Non sovraccaricare mai la memoria dell'utente con procedure complicate per eseguire le attività della nostra applicazione. Se inseriamo procedure troppo complesse il processo di codifica delle informazioni diviene confusionario e arriva ad essere controproducente per i futuri utilizza dell'applicazione, portando eventualmente l'utilizzatore ad abbandonarla a causa della troppa complessità.

2- Costruire delle interfacce che aiutano nel *riconoscimento* piuttosto che nel *ricordo* di funzioni della nostra applicazione. Questo può essere ottenuto usando uno stile riconoscibile per ogni menu, icona e oggetto che trova posto all'interno dell'applicazione.

3- Fornire agli utenti modi per poter codificare informazioni elettroniche che li aiutino a ricordare come ripescarle: ad esempio aiutare un utente a ricordare qual è il file che sta cercando tramite la stampa a video delle informazioni sull'orario e data di ultimo utilizzo, oppure tramite l'uso di colori per indicarne le proprietà. Anche l'impiego di icone specifiche a segnalare caratteristiche dell'informazione ricercata permettono di far riconoscere all'utente il contesto in cui l'aveva codificata la prima volta e quindi a trovarlo più velocemente.

Apprendimento

Questo processo cognitivo è stato analizzato in maniera approfondita dalla branca omonima di psicologia. Come citato poco sopra, l'apprendimento può avvenire in due modi differenti: uno è l'apprendimento tramite lo studio teorico, tramite libri o qualunque altro media, e l'apprendimento per esperienza diretta. Quest'ultimo è stato ampiamente confermato come il metodo più efficace di apprendimento umano. E' abbastanza semplice osservare questo concetto applicato al mondo delle applicazioni mobile. Avete mai visto un manuale di istruzioni per un'applicazione? La risposta è praticamente nel 99% dei casi "no". Per capire l'importanza di questa affermazione basta osservare l'utente medio quando posto davanti alla possibilità di apprendere l'uso di un prodotto tramite un manuale oppure tramite l'uso diretto del prodotto stesso, venendo guidato passo passo in tutte le caratteristiche offerte. Come può questo concetto influenzare lo sviluppo di un'applicazione? E' presto detto: inserire un buon tutorial, semplice e chiaro, in grado di guidare l'utente attraverso tutte le funzioni della nostra applicazione, permette di ottenere subito un buon riscontro già durante la somministrazione dei prototipi in fase di test. Esiste anche una possibilità ancora più potente, per quanto restrittiva. Si tratta di porre limitazioni all'uso dell'applicazione fintanto che l'utente non sia stato informato tramite tutorial d'uso delleloro funzioni e significato. Questo può limitare enormemente la possibilità di trarre l'utente in errore a causa di incomprensioni sulle caratteristiche dell'applicazione.

Vediamo di individuare anche in questo caso i punti fondamentali del design tenendo conto della facilità di apprendimento:

1- Costruire interfacce che stimolino l'esplorazione dell'utente. Questo significa "accompagnare con mano" l'utente medio invitandolo ad utilizzare le funzioni della nostra applicazione ma senza forzarlo ad un apprendimento metodico.

2- Disegnare interfacce che "forzino" l'utente a trovare le funzioni di cui ha bisogno, facendogli selezionare le azioni appropriate al contesto di utilizzo. Un buon modo per ottenere quest'obiettivo è costruire un'interfaccia a "settori", mostrando solo macro-aree di competenza all'apertura dell'applicazione e aiutando in questo modo l'utente a filtrare le funzioni fino a trovare l'operazione esatta di cui ha necessità.

3- Collegare dinamicamente rappresentazioni e astrazioni che devono essere apprese. Questo concetto è ciò che sta alla base del web, e dell'ipertesto in generale. Con l'avvento di internet l'utilità del collegamento dinamico di informazioni è salita a livelli vertiginosi. Nel mondo delle applicazioni mobile questo tipo di navigazione delle informazioni è alla base di ogni qualsivoglia esplorazione di contenuti. Nel caso dell'apprendimento di funzioni, il collegamento dinamico può essere utilizzato per stimolare l'utente ad informarsi su una funzione ancora non utilizzata, magari ponendo proprio il collegamento alla spiegazione vicino ad essa.

Lettura, conversazione ed ascolto

Questi tre processi cognitivi vengono associati insieme per via di un unico motivo: quando leggiamo, pronunciamo o ascoltiamo una frase in un certo linguaggio, questa avrà lo stesso significato in tutti e tre i casi. Ciò che quindi li accomuna è il significato, l'informazione che trasmettono. Ovviamente le modalità di trasmissione variano molto quando andiamo ad osservarli nel dettaglio. Un primo aspetto da considerare quando si guarda a questi processi è il fatto che differiscono in efficacia da persona a persona. Un utente potrebbe trovare molto più semplice comprendere qualcosa tramite un ascolto piuttosto che tramite la lettura. Un altro utente potrebbe fare fatica a seguire una voce che spiega, preferendo rivolgersi a contenuti scritti. Esistono svariate differenze che è bene puntualizzare: il linguaggio scritto ad esempio è permanente, dove invece quello parlato è temporaneo. Quando si legge un testo scritto è possibile rileggerlo laddove ci fossero incomprensioni e quindi avere un approccio più approfondito, mentre un'informazione trasmessa in linguaggio parlato non può essere fermata e riascoltata se si trova in una trasmissione diretta. Un altro punto a favore della lettura è che può essere molto più veloce del parlare o ascoltare, in quanto un testo scritto può essere esplorato anche in maniera non sequenziale, ma selezionando solo le informazioni che ci servono in quel dato istante. Ovviamente questo non è possibile quando siamo nel linguaggio parlato, in quanto le parole sono per forza di cose sequenziate e siamo costretti ad aspettare la fine del discorso per averne il significato completo. Un punto a favore dell'ascolto è che richiede uno sforzo cognitivo minore rispetto al leggere o parlare. Ci è possibile assorbire informazioni quasi in modo subliminale ascoltando, arrivando a ricordare cose di cui non avevamo (o credevamo) di aver tenuto conto mentre vivevamo quell'esperienza. Un'altra cosa di cui tenere conto è la grammatica: il linguaggio scritto tende ad essere più formale nella grammatica laddove invece quello parlato è molto più flessibile. Accade spesso che una persona interrompa una frase prima di averla conclusa magari lasciando la parola a qualcun altro.

Andiamo quindi ad individuare i punti fondamentali di un buon design che tenga conto di questi processi cognitivi:

1- Tenere al minimo della lunghezza le spiegazioni audio in un'applicazione se ve ne fosse il bisogno: è stato appurato da vari studi che usare menù contestuali audio è per la maggior parte controproducente per l'utenza, che non riesce a memorizzare troppe informazioni ascoltate in poco tempo. L'ideale sarebbe fornire un massimo di tre-quattro istruzioni o opzioni audio per farle ricordare all'utente medio. Un esempio sono i classici call center telefonici con voce registrata: se vengono aggiunte troppe voci all'elenco di opzioni l'utente sarà costretto ad ascoltare più volte l'elenco prima di trovare quella giusta.

2- Nel caso in cui si debba utilizzare nell'applicazione una voce artificiale che legge testo, dotarla di maggiore intonazione possibile, in quanto le voci generate artificialmente risultano spesso difficili da comprendere rispetto a quelle umane. Negli ultimi anni sono nate molte applicazioni in grado di leggere testo, come quelle in grado di leggere la rubrica di un utente, queste

sono un chiaro esempio di come l'intonazione di una voce artificiale aiuti nell'individuare le giuste opzioni.

3- Cercare di avere sempre l'opportunità di inserire testi grandi e leggibili sullo schermo, senza alterarne la formattazione. Per quanto negli ultimi anni gli schermi degli smartphone si siano ingranditi divenendo simili a quelli dei tablet, le applicazioni dovrebbero essere pensate basandosi sul caso dello schermo più piccolo su cui queste dovrebbero essere in grado di funzionare. Anche un utente che vede benissimo può trovarsi affaticato da un testo troppo piccolo da leggere, per non parlare di coloro che hanno dei deficit visivi: troppo spesso questo aspetto è preso sottogamba dagli sviluppatori, che lo relegano ad una mera ovvietà di cui però in parecchi casi non tengono conto. Osserviamo quali effetti disastrosi e confusionari può portare la non osservanza di questo criterio:

Risoluzione di problemi, pianificazione, ragionamento e scelte decisionali

Ognuno di questi processi cognitivi fa parte della cognizione riflessiva: ciascuno di essi richiede di pensare a cosa fare, quali alternative si hanno, quali conseguenze possono arrivare a seguito di una certa scelta e quindi azione. Coinvolgono molte volte anche la cosiddetta "coscienza" che ci porta ad avere discussioni con altri o anche lunghi monologhi interiori nel cercare di trovare una soluzione adatta al problema che si sta affrontando. Il ragionamento si basa proprio sul cercare di trovare ogni volta la soluzione migliore a un dato problema vagliando tutti gli scenari ed opzioni possibili. Logica e coscienza molte volte si scontrano portando a conseguenze inaspettate e, nella maggior parte dei casi, non desiderate. Lo sviluppatore che vuole tenere conto di tutto questo deve capire anche un altro aspetto che sta dietro a questi processi cognitivi: perchè un ragionamento, una pianificazione, o una scelta dell'utente avvengano un maniera corretta (quindi secondo ciò che lo sviluppatore ha deciso in fase di sviluppo) bisogna che l'utente stesso sia dotato dei mezzi con cui operare tali ragionamenti. Per "mezzi" intendiamo le informazioni di cui l'utente deve essere in possesso per poter avere abbastanza elementi su cui basare i propri processi di ragionamento. Un utente che non ha molta dimestichezza con un nuovo sistema e che è in possesso di poche informazioni riguardo ad esso agirà solitamente facendo assunzioni basate su conoscenze di ambiti similari a quello attuale. Tenderà quindi un approccio di prova ed errore, esplorando e sperimentando via via il sistema. Questo si traduce in una lentezza di apprendimento e soprattutto in una perdita significativa di efficienza. Riuscire a basare la propria applicazione anche sul livello di conoscenze pregresse dell'utente è proprio uno degli obiettivi di usabilità individuati nel capitolo precedente, e in questo caso ora il lettore avrà ben chiare le motivazioni per operare sempre scelte basandosi sul livello di esperienza dell'utente.

In questo caso possiamo individuare i seguenti punti fondamentali di cui tenere conto nel design di un'applicazione:

1- Individuare sempre una media nel livello di esperienza dell'utenza a cui si rivolge il nostro prodotto, fin dove possibile: uniformare al meglio tutte le sezioni dell'applicazione andando a creare un "flusso" di utilizzo è il segreto per ottenere un prodotto di indubbia qualità secondo chi scrive. Questo può essere raggiunto tramite l'osservazione prolungata di svariati campioni di utenza che utilizzano i prototipi della nostra applicazione. Le reazioni dell'utenza sono tutto in questa fase: osservare comportamenti di concentrazione che sfocia nella frustrazione è un chiaro sintomo di incompatibilità dell'applicazione con un certo tipo di livello di esperienza. Nella prossima parte del capitolo ci si concentrerà proprio sullo stato di "flusso" che si può cercare di ottenere nell'utente mentre utilizza la nostra applicazione.

2- Nel caso in cui non ci si possa basare su un unico livello di esperienza nella progettazione di un applicazione a causa di una base di utenza finale troppo vasta, si può operare nel seguente modo: fornire informazioni nascoste

addizionali di facile accesso per gli utenti che vorrebbero comprendere meglio o migliorare il loro uso dell'applicazione, magari proprio per migliorare la propria attività in collaborazione con il nostro prodotto. Un classico esempio di questo sono i pop up delle applicazioni, che per quanto invasivi, molte volte invitano l'utente a migliorare la conoscenza dell'applicazione che sta usando rivelando altre funzioni magari non ancora utilizzate.

3.3 **Lo stato di flusso: oltre la cognizione**

Si vuole dedicare anche una piccola parte di questo capitolo allo stato di "flusso" che un utente può raggiungere nell'uso di un'applicazione. Secondo chi scrive, se lo sviluppatore durante la somministrazione di un prototipo agli utenti ottiene le reazioni descritte qui di seguito, ha praticamente fatto centro: se il prodotto riesce a indurre l'utente in uno stato di flusso abbiamo ottenuto in un colpo solo tutti gli obiettivi di esperienza d'uso. L'esperienza cosidetta di "flusso" è composta da un insieme di parti individuate dallo psicologo Mihály Csíkszentmihályiche guardacaso si intersecano con l'insieme di obiettivi di design che ci siamo posti in apertura di questa tesi. Vediamo quali sono:

Obiettivi chiari: le aspettative e le modalità di raggiungimento sono chiare.

Concentrazione totale sul compito: un alto grado di concentrazione in un limitato campo di attenzione (la persona non ragiona su passato e futuro ma solo sul presente).

Perdita dell'autoconsapevolezza: il soggetto è talmente assorto nell'attività da non preoccuparsi del suo ego.

Distorsione del senso del tempo: si altera la percezione del tempo. Non ci si rende conto del suo scorrere.

Retroazione diretta e inequivocabile: l'effetto dell'azione deve essere percepibile dal soggetto immediatamente ed in modo chiaro.

Bilanciamento tra sfida e capacità: l'attività non è né troppo facile né troppo difficile per il soggetto.

Senso di controllo: la percezione di avere tutto sotto controllo e di poter dominare la situazione.

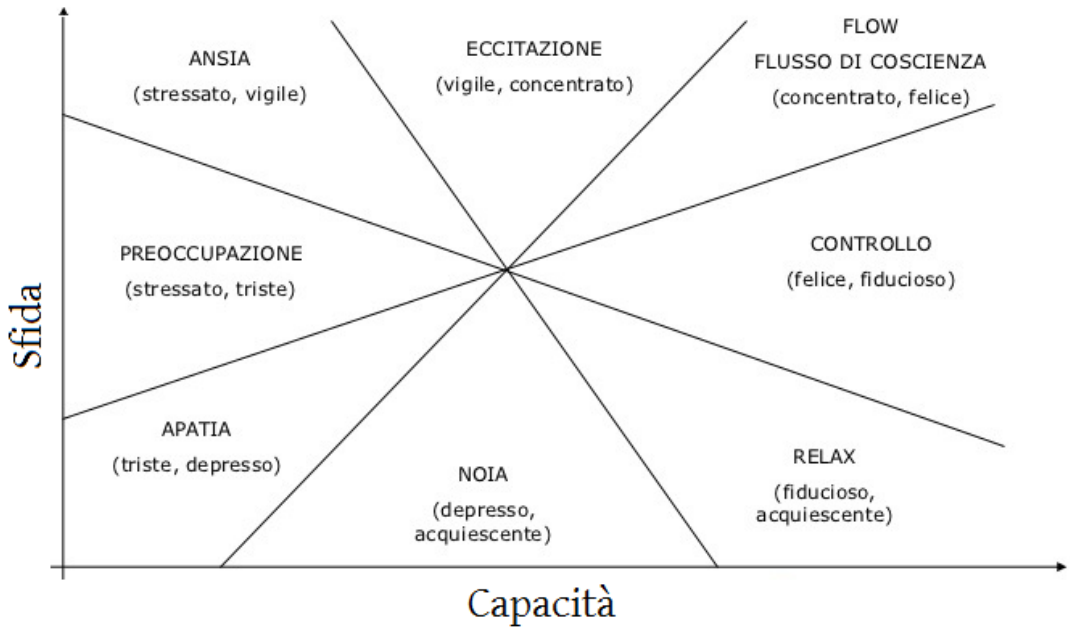
Piacere intrinseco: l'azione dà un piacere intrinseco, fine a se stesso (*esperienza autotelica*).

Integrazione tra azione e consapevolezza: la concentrazione e l'impegno sono massimi. La persona è talmente assorta nell'azione da fare apparire l'azione naturale.

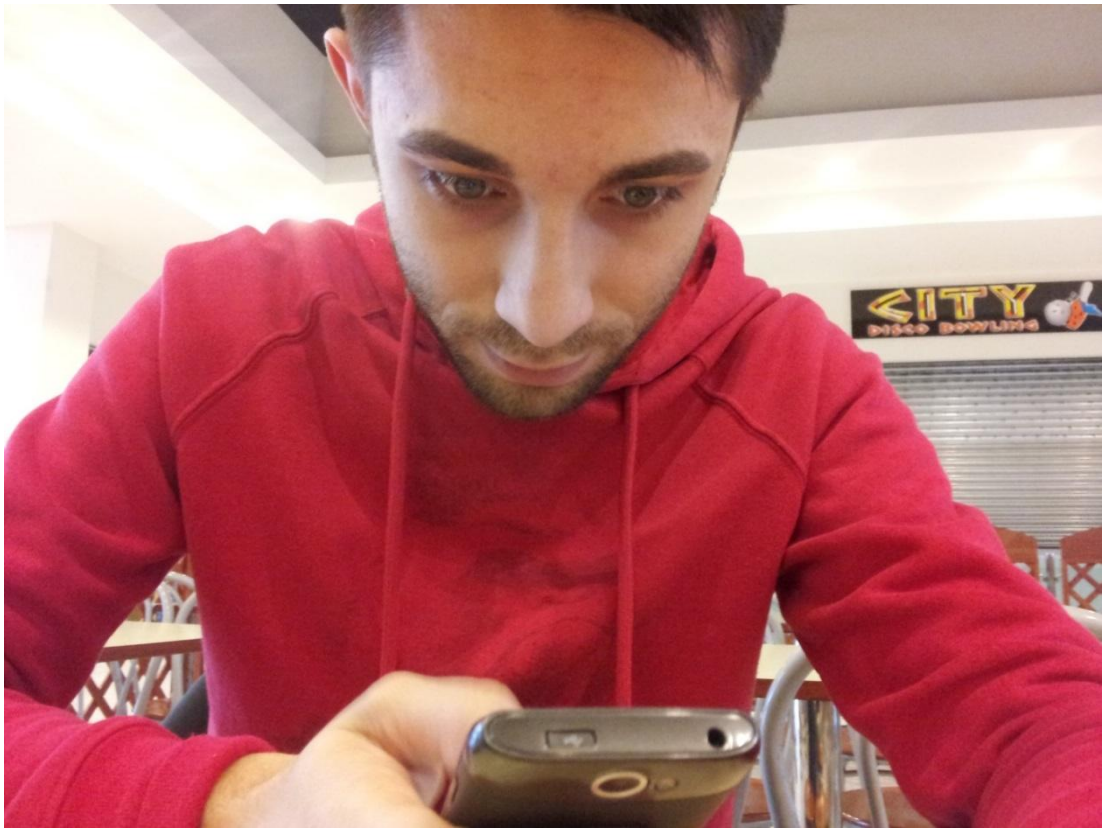
L'insieme di questi punti forma quello che viene comunemente definito "esperienza ottimale". Nella sua iniziale accezione questa veniva impiegata per indicare il grado di "felicità" delle persone su cui erano stati eseguiti gli studi, quindi tendenzialmente venivano testate attività di svago con finalità di "gioco" ed educazione. Recentemente però molti di questi concetti sono subentrati anche in ambito lavorativo e stanno influenzando il concetto che sta alla base della professionalità.

Vogliamo cercare di capire come questo concetto influenzi enormemente anche il processo di sviluppo. Secondo chi scrive i punti elencati poco sopra sono molte volte associati erroneamente solo ai prodotti dedicati allo svago. Mai errore più grande potrebbe essere commesso dall'ignaro sviluppatore. I punti dell'esperienza ottimale possono e devono essere rispettati in qualsivoglia prodotto, sopratutto in quelli con finalità professionali. Un'applicazione che stimoli anche il suo utilizzo da parte dell'utente non solo sarà molto più efficiente ma migliorerà l'efficienza dell'utente stesso in ambito lavorativo.

Osserviamo come lo stato di flusso si ponga come perfetto equilibrio tra sfida nell'uso dell'applicazione e capacità pregresse dell'utente:



In chiusura di capitolo si vogliono proporre delle immagini esemplificative dello stato di flusso e quindi dell'esperienza ottimale nelle espressioni che lo sviluppatore dovrebbe cercare in fase di test quando i prototipi dell'applicazione vengono somministrati agli utenti presi come campione.



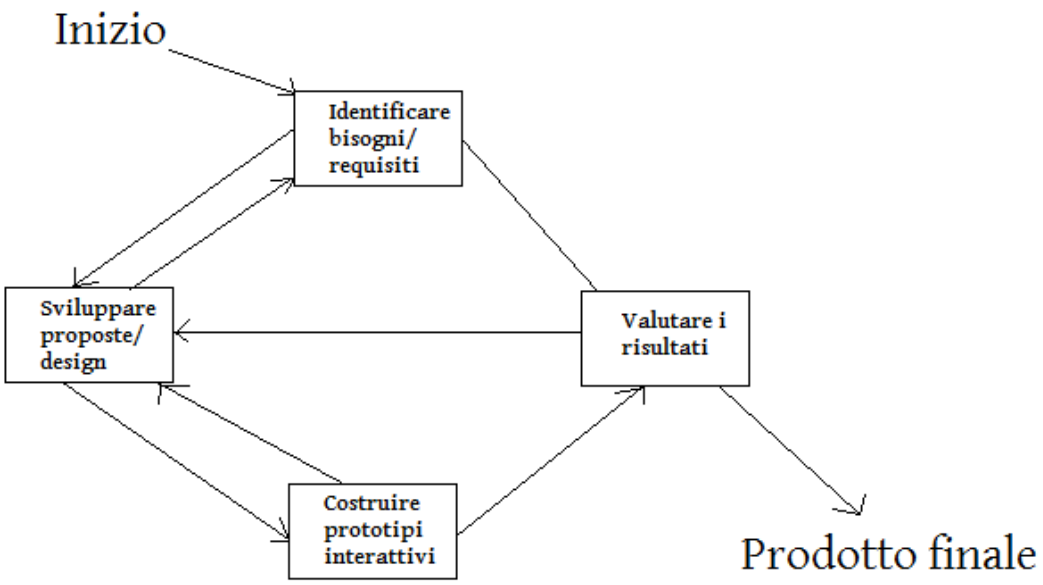
4 – Il processo di sviluppo

A questo punto, dopo aver compreso le basi della comprensione dell'utente, vogliamo concentrarci sul cuore della questione: il processo di sviluppo e design incentrati sull'utente.

Iniziamo cercando di dare una piccola definizione di "design" in senso generale: il design è un piano o uno schema concepito nella mente per essere poi eseguito. Questo significa che il piano o schema generati dovranno tenere conto di una moltitudine di fattori subentranti nel progetto: per sperare di vedere arrivare al successo un'applicazione bisogna tenere conto di analisi di mercato, analisi di fattibilità (ossia disponibilità di mezzi e servizi adatti al nostro progetto) ma sopratutto questioni pratiche e tecniche (ad esempio per quale sistema sviluppare la nostra applicazione). Nel nostro caso lo sviluppo incentrato sull'utente cerca di concentrare gli sforzi più sulla comprensione continua dell'utenza alla quale il nostro prodotto si rivolge, lasciando le questioni tecniche come aspetto secondario durante le fasi inizialidel processo di sviluppo.

In processo di design incentrato sull'utente possiamo individuare quattro attività fondamentali:

1- Identificare i bisogni dell'utente e individuare i requisiti.
2- Sviluppare proposte alternative.
3- Costruire prototipi interattivi dei piani di design.
4- Valutare i risultati e le soluzioni proposte.



Notiamo immediatamente dall'immagine che tutto il processo non è a compartimenti stagni, ma ogni fase è iterativa: abbiamo quindi un ciclo di sviluppo più che una sequenza, in cui vari passi saranno ripetuti mentre il prodotto viene sempre più raffinato ad ogni iterazione. Questo deriva dal fatto che in materia di sviluppo software è (quasi) sempre possibile tornare indietro ad un passo precedente, formando una figura più simile ad una spirale, mentre invece in altri ambiti come la

meccanica questo risulta molto più difficile in quanto si ha a che fare con componenti fisici.

In questo capitolo si cercherà di fare luce su queste quattro attività principali: le prime due sono comuni per larga parte ad un classico processo di sviluppo software. Addirittura il bisogno di individuare requisiti è la prima fase di praticamente ogni processo di sviluppo esistente, così come lo sviluppare un insieme di proposte come soluzione allo sviluppo. Per ulteriori informazioni su di un processo di sviluppo classico, si veda l'introduzione.

4.1 **Identificare i bisogni dell'utente e individuare i requisiti**

Come già spiegato nel capitolo precedente la prima fase si compone di tutti gli sforzi atti alla comprensione più ampia e approfondita possibile dell'utenza per cui stiamo sviluppando la nostra applicazione. Una volta identificati i bisogni dell'utente avremo le basi per poter individuare tutti i requisiti richiesti perchè la nostra applicazione riesca nel suo intento. In questo senso la necessità di identificare bisogni e stabilire requisiti continerà a permanere lungo quasi tutto il processo di sviluppo. Durante un tipico processo incentrato sull'utente infatti i requisiti tendono ad evolversi e si sviluppano man mano che gli utenti presi a campione utilizzano i prototipi proposti in fase di design, decidendo ogni volta quali caratteristiche gli risultano più utili e quali invece scartare. Tutta questa attività ovviamente andrà ripetuta più e più volte fino al raggiungimento di una qualità soddisfacente del prodotto.

Perchè questa fase è così importante? Si tratta in pratica di stendere ed aggiornare costantemente le fondamenta del nostro progetto, ossia i requisiti. Secondo gli ultimi studi in materia i fallimenti più clamorosi di progetti informatici hanno tutti come costante ricorrente "assenza di requisiti chiari e precisi". Questo si traduce in un esito disastroso dell'applicazione una volta immessa sul mercato: se lo sviluppatore non ha chiara esattamente l'utilità e le funzioni che dovrà avere il prodotto che sta sviluppando le conseguenze possono essere disastrose. Il prodotto potrebbe essere totalmente ignorato in favore di un qualcosa di più performante, mentre il caso peggiore può essere addirittura il disprezzo per l'applicazione della base di utenza e un calo di efficienza, quando invece l'obiettivo era esattamente contrario. Chi scrive vorrebbe che il lettore comprendesse quanto gli errori in fase di individuazione dei requisiti possano fare affondare anche il progetto più semplice, e di non sottovalutare mai la questione. Nel nostro caso proprio per evitare degli errori in questa fase subentra lo studio costante e progressivo dei bisogni dell'utente, che riduce drasticamente la possibilità di sbagliare nel definire i requisiti.

Come possiamo individuare i requisiti tramite l'approccio incentrato sull'utente? Ci sono vari metodi, ma prima di tutto bisogna capire che esistono svariati tipi di requisiti, ed ognuno di essi è fondamentale:

1- Requisiti funzionali: sono i requisiti che definiscono ciò che il sistema che stiamo andando a creare dovrebbe fare. Sono comuni ad ogni tipo di processo di sviluppo, in quanto storicamente questa fase era mirata ad individuare solamente questo tipo di requisiti.

2- Requisiti non funzionali: sono requisiti che coprono aspetti non relativi alle funzionalità dell'applicazione, possono indicare la grandezza richiesta per lo spazio di memorizzazione dei dati del nostro prodotto, il tempo di risposta richiesto per l'esecuzione delle operazioni principali (il classico esempio possono essere i sistemi in tempo reale, che fanno della temporizzazione delle operazioni la loro base), e infine anche la deadline entro cui il prodotto dovrebbe essere terminato.

3- Requisiti sui dati: tutto ciò che concerne i dati che andranno inseriti e memorizzati all'interno della nostra applicazione, quindi le loro tipologie (testi, numeri, date, documenti importanti ecc...) e come saranno immagazzinati, capendo se è necessario l'uso di un database ad accesso veloce magari.

4- Requisiti di contesto/dominio: si tratta di esplicitare le circostanze in cui ci aspettiamo che il prodotto venga utilizzato. Si possono suddividere a loro volta in due sotto-tipologie, ossia sociali e organizzativi: quelli sociali individuano le richieste che hanno a che fare con la condivisione di file, sulla privacy, sul lavoro di gruppo e individuale. Quelli organizzativi invece individuano tutto ciò che riguarda la gerarchia dell'ambiente lavorativo, le attitudini di ogni dipartimento (se la base di utenza è suddivisa in base a differenti carattestiche), le strutture di comunicazione a disposizione degli utenti per organizzarsi, il supporto richiesto per il prodotto e così via.

5- Requisiti degli utenti: mirano specificatamente ad individuare le caratteristiche principali della base di utenza utili a comprendere come strutturare la nostra applicazione. Questo include le loro abilità pregresse, il loro background, l'attitudine all'informatica: quest'ultima nel nostro caso dev'essere ulteriormente approfondita cercando di comprendere due fattori fondamentali, che sono la frequenza di utilizzo e il livello di esperienza. Una base di utente composta per la maggior parte di novizi avrà bisogno di un sistema che li guidi passo passo e che gli impedisca di compiere errori, ma sopratutto che abbia sempre informazioni chiare e spiegazioni. Al contrario un utente esperto potrebbe richiedere flessibilità oltre che maggiori funzioni. Quest'ultimo potrebbe anche richiedere delle scorciatoie interne all'applicazione, in quanto ne fa un utilizzo frequente. Al contrario, l'utente novizio non ancora abituato avrà bisogno di istruzioni chiare e di percorsi "forzati" nei menu dell'applicazione.

6- Requisiti di usabilità: questi individuano gli obiettivi di usabilità specifici però del progetto che stiamo andando a sviluppare, esplicitando: facilità di apprendimento, efficienza, flessibilità e le attitudini principali dell'utente medio.

Le tecniche di raccolta dati a nostra disposizione

Abbiamo esplicitato i tipi di requisiti che vogliamo andare ad individuare, ora andremo a vedere le tecniche che si possono utilizzare con la collaborazione dell'utenza per ricavarli. Abbiamo varie tecniche, che sono:

1- I questionari: somministrare questionari alla base di utenza per cui dobbiamo sviluppare può aiutarci in vari modi, permettendoci di estrarre informazioni e requisiti molto specifici. Possono essere strutturati con risposte multiple anche solo SI e NO. Permettono di ricavare sia dati qualitativi che quantitativi vista la loro enorme versatilità e possono essere usati in combinazione con altre tecniche, diventando una scelta obbligata quando ci troviamo di fronte ad una base di utenza troppo vasta.

2- Interviste: possono prendere forma in molti modi, che vanno dal forum on line usato per comunicare con gli utenti, fino ad un'interazione diretta con delle persone scelte dalla base principale. Possono essere strutturate o meno, e sono di solito accorpate ad un prototipo somministrato come test gli utenti, per capire le loro reazioni ed ottenere/aggiornare ulteriori requisiti, ma sopratutto per far emergere eventuali problemi. Ovviamente rispetto ai questionari le interviste possono portare via una grande quantità di tempo utile e sopratutto nei casi di gruppi molto vasti non è possibile intervistare ogni singola persona: bisogna operare una scelta su chi adottare come campione della popolazione di utenza, ottenendo come risultato una stima meno precisa.

3- Gruppi di lavoro: si ottengono organizzando incontri ricorrenti con l'utenza per coinvolgerla maggiormente nello sviluppo del progetto, permettono di eseguire interviste di gruppo e sopratutto di individuare discrepanze nella base di utenza che possono inficiare l'usabilità del prodotto che stiamo sviluppando. Parallelamente permettono di individuare le aree di maggior consenso che la base di utenza riscontra nel nostro prodotto e quindi concentrare i nostri sforzi su di esse per migliorare progressivamente i prototipi, ma non permettono di ottenere un livello di dettaglio molto preciso, vista la natura collettiva.

4- Osservazione dell'utenza: questa tecnica richiede uno sforzo da parte degli analisti e sviluppatori, infatti per metterla in atto questi devono inserirsi nell'ambiente degli utenti, osservandoli nelle loro attività quotidiane. Nel caso di un ambito lavorativo, si tratta di osservare l'utenza al lavoro, proprio mentre questo avviene. Questo tipo di osservazione permette di comprendere al meglio quali attività svolge la nostra tipologia di utente, insieme al loro contesto e alla loro natura. Ovviamente portare avanti una tecnica del genere richiede un grande quantitativo di tempo e può portare ad enormi quantità di dati difficili da gestire. Secondo chi scrive però è proprio questa la tecnica chiave dello sviluppo incentrato sull'utente: per quanto possa essere persino

stressante “mettersi nei panni” dell’utente questo può essere in realtà proprio ciò che serve al designer per fugare ogni tipo di dubbio che si possa essere creato riguardo ai requisiti, evitandogli errori disastrosi che si ripercuoterebbero su tutto il processo, affondandolo inesorabilmente.

5- Studiare la documentazione: si tratta di studiare la documentazione disponibile riguardo alle attività dell’utenza. Questo significa in ambiti lavorativi studiare le procedure e le regole di un certo tipo di attività. Una cosa molto utile e spesso sottovalutata di questa tecnica è la presenza di legislazioni specifiche da rispettare nel caso l’attività degli utenti coinvolga processi sensibili (si vedano ad esempio le applicazioni bancarie) che possono causare problemi imprevedibili dallo sviluppatore non informato. Il rovescio della medaglia di questa tecnica è che non coinvolge direttamente l’utente nello sviluppo, preferendo studiare le caratteristiche dell’attività che questo svolge. Nel nostro caso è una tecnica utilizzata sopratutto a causa di fattori di forza maggiore quali impossibilità di studiare direttamente l’utente a causa della distanza o costi troppo elevati ecc... limitandone l’uso in combinazione con altre tecniche.

Come utilizzare le tecniche di raccolta dati

Vogliamo ora capire quando usare le tecniche spiegate sopra. Ci sono una moltitudine di fattori da considerare riguardo alla base di utenza ma sopratutto l’attività per la quale stiamo sviluppando a supporto la nostra applicazione. Vogliamo capire per esempio se l’attività presa in esame può essere vista come un insieme di passi sequenziali oppure un insieme di sottopassi che si intersecano tra loro, se abbiamo a che fare con attività complesse o semplici da concettualizzare come informazione. Un altro fattore è il livello di abilità dell’utilizzatore della nostra applicazione. E’ un novizio oppure un esperto? Esistono delle linee guida da poter seguire che riportiamo:

- Quando decidiamo di impiegare una tecnica all’inizio dello sviluppo, il nostro focus primario dovrebbe essere quello di identificare i bisogni della base di utenza.

- Dovremmo cercare di coinvolgere tutti i gruppi di utenza dove possibile se questa fosse eterogenea, senza discriminare involontariamente nessun gruppo, in quanto rischiamo di sorvolare su eventuali problematiche di utilizzo.

- Se si dovessero utilizzare delle persone prese a campione, usarne almeno più di una per gruppo di utenza, nel caso in cui si avesse a che fare con un gruppo misto. Questo aiuta a migliorare sensibilmente le stime fatte in analisi e a ricavare requisiti più precisi.

- Usare le teniche di raccolta dati in maniera combinata. Molte volte è utile ad esempio usare sia questionari che interviste per osservare se ci sono discrepanze anche all'interno di un gruppo di utenti molto omogeneo. Usare la documentazione insieme all'osservazione sul campo permette ai designer di arrivare già preparati sull'attività da esaminare e quindi più liberi di concentrarsi sugli aspetti psicologici dell'utenza e di come questa si relaziona con la propria attività.

- Interpretare i dati subito dopo che questi sono stati raccolti. Rinviare la processazione di tutto quello che si è raccolto potrebbe avere l'esito di rallentare o rovinare l'intero processo di sviluppo, per cui è bene tenere anche a mente che i dati raccolti possono perdere il valore iniziale se non considerati in tempi brevi.

- Come conseguenza del punto sopra citato è sempre meglio partire con un'interpretazione iniziale dei dati non troppo approfondita, questo permette di risparmiare un grande quantitativo di tempo utile nel caso in cui i dati appaiano ridondanti rispetto ad altre indagini già effettuate in precedenza. L'analisi approfondita dei risultati dovrebbe avvenire solamente dopo aver effettuato un'interpretazione generale.

- Nel caso si debbano esplicitare concetti complessi ai gruppi di utenza bisognerebbe utilizzare gli strumenti più adatti: quindi usare i diagrammi appropriati come quelli delle classi per sistemi orientati agli oggetti e diagrammi E-R per spiegare l'immagazzinazione dei dati all'interno del sistema. Meglio riusciamo a trasmettere le informazioni agli utenti, meglio questi risponderanno fornendoci requisiti sempre più precisi.

4.2 **Sviluppare proposte e tradurle in prototipi**

Una volta ottenuta una prima versione dei requisiti è giunto il momento di passare alla seconda fase del processo di sviluppo: vogliamo cercare di generare svariate proposte alternative in modo da ottenere in breve tempo i primi prototipi della nostra applicazione e somministrarli agli utenti scelti per i test. Le proposte dovrebbero tenere conto di diverse linee guida quando si utilizza un approccio incentrato sull'utente, e queste sono:

1. Le attività e gli obiettivi degli utenti sono la forza primaria che deve stare alla base dello sviluppo della nostra applicazione.

2. Tutte le proposte dovrebbero tenere conto del comportamento degli utenti e del loro contesto di utilizzo dell'applicazione in modo che questa gli fornisca un valido supporto in quello che fanno (ad esempio in ambito lavorativo).

3. Le caratteristiche degli utenti devono essere catturate e il design dev'essere strutturato usando come metro valutazione proprio l'utilità rispetto ad esse.

4. Consultare costantemente gli utenti durante tutto lo sviluppo, tramite le tecniche spiegate nella sezione precedente. Aggiornare sempre le proposte sia dalle fasi iniziali sino a quelle finali, e prendere ogni volta seriamente le loro risposte/proposte.

5. Tutte le scelte di design devono essere prese tenendo conto in combinazione del contesto dell'utente, della sua attività e dell'ambiente in cui essa si svolge. In questo modo lo sviluppatore "entra" nel contesto dell'utente e può formulare proposte che gli siano vicine e possano essere accettate.

Il design collaborativo/partecipativo con l'utente

Una tipologia di design basata proprio sull'approccio con l'utente è il design collaborativo: si parte dal coinvolgimento della base di utenza partendo con le tecniche di raccolta dati, ma ci si spinge oltre. L'utente viene coinvolto attivamente negli incontro per le scelte di design ed ha quindi un ruolo molto più attivo rispetto al semplice rispondere a test e domande.

Come prima fase nel caso in cui la base di utenza sia molto vasta e sia quindi impossibile coinvolgere tutti bisogna scegliere un rappresentante che funga da tramite tra designer/sviluppatore e utenti. Una volta che si è appurata la comunicabilità con l'utenza si può partire: esistono varie tecniche per poter condividere le proposte di design e discuterne con gli utenti. Una di queste è la rappresentazione condivisa, ossia collaborare insieme nel cercare di schematizzare le

informazioni nel modo più vicino e comprensibile a tutte e due le parti coinvolte. A questo proposito è bene partire usando strumenti semplici come anche solo lavagne o carta e penna fino a video, evitando diagrammi troppo complessi e incomprensibili per l'utente medio. Quest'ultimo potrà collaborare fornendo sue semplici rappresentazioni e discutendo con i designer sulle scelte da operare. La cosa più importante però è secondo chi scrive la fase di valutazione dei prototipi scaturiti dalle proposte: effettuando questa fase in modo cooperativo con l'utenza si possono individuare problemi o funzioni da aggiungere che normalmente il solo punto di vista dello sviluppatore o designer non riesce ad esplicitare.

Quest'ultima affermazione ci porta a voler comprendere meglio cosa siano i prototipi e come questi possano essere costruiti per avere reazioni e risposte dall'utenza.

I prototipi

Molto spesso gli sviluppatori si trovano di fronte a quello che appare come un paradosso: la maggior parte degli utenti e dei committenti di applicazioni non ha ben chiaro quali sono i suoi bisogni finchè non prova con mano un prodotto. Questo ovviamente pone dei grandissimi ostacoli al processo di sviluppo in quanto non è possibile stabilire con precisione i bisogni degli utenti e quindi i requisiti.

I prototipi cercano di risolvere questo problema, e se usati con un approccio mirato all'utente secondo chi scrive arrivano ad avere un'efficacia pari al 90% dei casi possibili. Il loro utilizzo ciclico e continuo in un processo di sviluppo è una delle chiavi dello sviluppo incentrato sull'utente, infatti vogliamo capire adesso che forma possono prendere i tipici prototipi di applicazioni.

Che cos'è un prototipo? Nel nostro ambito possiamo restringere la risposta a questo: solitamente un frammento di software con funzionalità limitate ma con finalità a voler verificare certe questioni. In realtà può assumere anche altre forme che vanno da una serie di schermate, uno storyboard, delle slide presentative, fino ad arrivare ad un video che simula l'uso del sistema che stiamo andando a sviluppare.

Perchè utilizzare un prototipo di questo tipo? Le risposte sono molte:

- La valutazione e il feedback sono fattori centrali nel design incentrato sull'utente, senza di essi perderebbero completamente le fondamenta su cui si poggia.

- Usare un prototipo interattivo di tipo software permette all'utente di vedere, toccare e interagire con mano direttamente col sistema che stiamo sviluppando, in maniera molto più semplice rispetto al dover leggere un documento o comprendere degli schemi-diagrammi.

- Un aspetto da non sottovalutare è anche interno al team di sviluppo, infatti progettare un prototipo aiuta i membri del team a comunicare meglio tra loro le proprie idee, perplessità su alcuni aspetti, e proposte.

- Un altro aspetto da considerare quando si parla di prototipi è il tempo utile che permettono di risparmiare quando si cerca di valutare il prodotto che stiamo sviluppando: essendo progettato in breve tempo questo si traduce in un modo di testare le idee molto più veloce e, in caso di responso negativo, si può passare velocemente alla proposta successiva.

- Possiamo rintracciare anche una caratteristica di psicologia cognitiva che l'utilizzo di un processo può stimolare: la riflessione. Questa è una parte fondamentale dell'attività di design ed è ciò che ci permette di formulare, confrontare e valutare attentamente una proposta o un prototipo.

Quali aspetti di un'applicazione possiamo prototipizzare? Ovviamente un prototipo cercherà di concentrarsi solo su alcuni aspetti utili per la questione che si vuole analizzare tramite la sua valutazione da parte dell'utente. Quindi possiamo trovare prototipi utili per evidenziare problemi tecnici, flusso dell'attività, per capire se un'attività è stata gestita in maniera efficiente; nel nostro caso si vorranno testare anche le disposizioni a schermo degli oggetti della nostra applicazione, l'organizzazione delle informazioni e se queste sono facilmente reperibili, ma sopratutto individuare aree della nostra applicazione che possono risultare difficoltose da utilizzare, addirittura controverse rispetto ad altri aspetti, oppure individuare aree critiche che possono scatenare errori di sistema (quest'ultimo comunemente chiamato alpha test in un primo momento e poi beta una volta che l'applicazione si trova ad uno stato avanzato di sviluppo tale da rispecchiare il prodotto finale).

E' possibile suddividere quindi i prototipi in due macro-tipologie:

- Prototipi a bassa fedeltà: sono prototipi che puntano tutto sulla velocità e facilità di modifica. Solitamente non sono nemmeno un frammento di software ma possono essere una serie di schemi o disegni su carta o lavagna. Possono essere una serie di diapositive da mostrare in sequenze con scelte alternative, in modo da mostrare all'utente il progresso nell'attività che avrà con la nostra applicazione. Questo include anche scelte possibili quali selezionare opzioni da un menu ecc... Questi prototipi hanno una funzionalità molto limitata, ma sono utili come mezzo per rintracciare ulteriori requisiti durante il ciclo iterativo di sviluppo e anche per testare velocemente nuove idee.

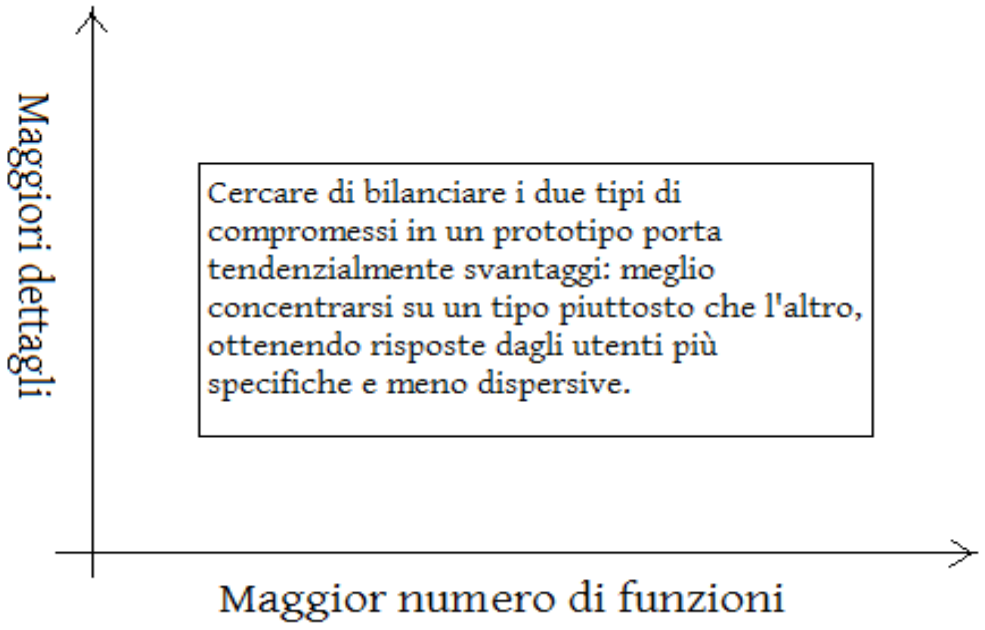
- Prototipi ad altà fedeltà: in contrasto alla prima tipologia troviamo questo tipo di prototipi che trovano il loro punto di forza nel fatto che assomigliano progressivamente sempre più alla versione finale del prodotto. Sono sviluppati utilizzando lo stesso ambiente di sviluppo del prodotto finale risultando quindi in piccole versioni con limitate funzionalità (ossia quelle che vogliamo andare a testare) della nostra applicazione. Tendenzialmente uno sviluppatore dovrebbe iniziare a sviluppare prototipi di questo tipo non appena questo è possibile, la cosa migliore sarebbe subito dopo qualche ciclo di analisi dei requisiti, quando questi sono divenuti più consolidati. Ovviamente il rovescio della medaglia utilizzando questi prototipi è la richiesta più esosa in termini di costi e tempistiche.

Oltre alle due tipologie di prototipi appena spiegate è bene considerare un altro aspetto che li contraddistingue: i compromessi. Quando si costruisce un prototipo bisogna considerare le limitazioni che si stanno ponendo all'utente che andrà a provarlo. Questo vuol dire che fornire un'applicazione con limitazioni troppo evidenti (un esempio: pulsanti che non producono alcun effetto) può essere controproducente in quanto induce uno stato di perplessità nell'utente che lo sta testando. A volte uno stato del genere può produrre alterazioni nelle risposte che stiamo cercando, producendo dei requisiti che non rispecchiano i reali bisogni dell'utente.

Per cercare di non cadere in queste situazioni si deve tener conto di due tipi di compromessi e di come questi influenzino un prototipo:

- Compromessi "orizzontali": il prototipo fornisce un grande numero di funzioni (nelle fasi avanzate di sviluppo anche tutte) ma raggiunge uno scarso livello di dettaglio (ad esempio molti nomi di funzioni e caratteristiche non sono definitivi e quindi rischiano di essere poco significativi).

- Compromessi "verticali": il prototipo è molto specifico, presentando solo poche funzioni ma in maniera molto dettagliata (di solito vicina alla versione finale). L'utilizzo di questo tipo di prototipi è caldamente consigliato nel caso di problematiche tecniche e perplessità da parte dell'utenza in relazione ad alcune funzioni della nostra applicazione.



4.3 **Valutare risultati e soluzioni**

L'ultima fase iterativa di cui ci vogliamo occupare è la valutazione. Questa è un'altra fase cruciale nello sviluppo incentrato sull'utente e permette di capire se il nostro prodotto si sta avvicinando al completamento oppure stiamo deviando da quanto ci eravamo proposti di creare.

Essendo il design incentrato sull'utente un processo iterativo la fase di valutazione dei risultati dovrà essere eseguita un gran numero di volte, sia per valutare le idee iniziali ed i primi modelli concettuali proposti, sia per valutare i prototipi all'inizio molto semplici e poi via via più complessi e somiglianti al prodotto finale.

In più, i designer e sviluppatori hanno bisogno di ottenere un continuo feedback e di valutarlo per capire se stanno seguendo i requisiti giusti o se hanno sbagliato qualcosa in fase di analisi e raccolta dati. Valutare se l'esperienza dell'utente risulta ottimale e sopratutto eseguire delle valutazioni esatte in merito può portare alla quasi certezza del successo di un certo prodotto. Dopotutto è approvato che il ciclo di design e testing con valutazione sia l'unico e valido metodo che riesce a produrre risultati di successo in maniera consistente.

Vediamo però di trovare motivazioni più vicine al caso dello sviluppo di applicazioni, quindi meglio comprensibili per uno sviluppatore che voglia investire nell'approccio con l'utenza e nelle conseguenti valutazioni cicliche:

- Molti problemi delle applicazioni moderne derivano da bug di sistema che non è stato possibile osservare durante lo sviluppo del prodotto, causando però disagi all'utilizzatore finale, che potrebbe essere portato anche a scartare l'applicazione e rivolgersi ad altro. La valutazione costante da parte di utenti in collaborazione con gli sviluppatori e designer permette di ridurre al minimo tali problemi.

- Avere continue valutazioni da parte della base di utenza permette al team di sviluppo dell'applicazione di concentrarsi sulle problematiche reali legate al prodotto; non nel cercare di capire se una funzione possa andare bene o meno agli utenti. Questo fa risparmiare enormi quantità di tempo utile e migliora nel complesso il prodotto.

- Come conseguenza del punto sopra i membri tecnici del team di sviluppo possono concentrarsi nel migliorare il codice dell'applicazione senza preoccuparsi di problemi ingegneristici che sono già stati risolti in una fase precedente: ciò significa che quando il programmatore va a scrivere il codice sa esattamente cosa deve fare, risultando in un certo senso come una compilazione di scheletri di codice che sono stati generati dalle fasi precedenti.

- Un altro fattore da considerare (e che molti sviluppatori tengono come primo aspetto da considerare, sebbene molte volte gli esiti siano negativi) è la velocità di raggiungimento della nostra applicazione sul mercato. Avere

continue valutazioni permette di velocizzare tutto il processo di sviluppo, permettendo anche di stabilire una data di uscita precisa per il prodotto (secondo chi scrive è più un'arma a doppio taglio, quindi dove possibile evitare di utilizzare date).

- L'ultimo motivo è la conseguenza di tutti i primi punti elencati sopra: se la nostra applicazione all'uscita è già pulita da bug e in grado di funzionare perfettamente in ogni suo aspetto la necessità di continui aggiornamenti e supporto dell'applicazione viene meno, risparmiando un grandissimo quantitativo di tempo e lavoro al team di sviluppo, che dovrebbe passare i primi giorni dopo l'uscita a rintracciare tutti i bachi scovati dall'utenza.

Quando dobbiamo eseguire delle valutazioni quindi? In che fasi del processo di sviluppo è richiesto che a conclusione del passo ci sia un giudizio? Ci sono due momenti generici in cui la valutazione è la chiave per la riuscita di un progetto: il primo è quando ci rendiamo conto che la nostra applicazione è qualcosa di totalmente nuovo e alieno per l'utenza (oppure stiamo testando una nuova funzione che gli utenti non hanno mai visto), quindi abbiamo bisogno subito di ottenere diverse valutazioni per capire se può soddisfare i bisogni degli utenti e supportare le loro attività. Il secondo momento è quando dobbiamo confrontare eventuali modifiche e aggiornamenti della nostra applicazione in base alle reazioni dell'utenza, aiutandoci in questo modo a scegliere una proposta piuttosto che un'altra.

Andiamo ora ad esplorare più nel dettaglio la fase di valutazione per capire come funziona. Esistono due tipologie principali di valutazione che è possibile eseguire durante un processo di sviluppo:

- Valutazione formativa: è la tipologia di valutazione più utilizzata generalmente durante lo sviluppo di un prodotto. Viene eseguita alla conclusione di ogni fase di prototipizzazione, di solito in collaborazione stretta tra utenti e sviluppatori, e serve per capire se il prodotto è compatibile con le necessità della base di utenza per cui si sta sviluppando l'applicazione. E' consigliato il suo utilizzo anche nelle primissime fasi di sviluppo, quando ancora non abbiamo prototipi software ma anche solo disegni e schemi, in quanto può aiutare molto nella definizione dei requisiti iniziali.

- Valutazione sommativa: solitamente tra i passi finali di un processo di sviluppo, questa tipologia di valutazione si occupa di affermare se la qualità del prodotto finale incontra quella che ci si era preposti in fase di sviluppo. Un esito positivo può essere l'inizio della conclusione del processo di sviluppo. Un esito negativo al contrario può costringere a tornare indietro di qualche fase per sistemare gli eventuali problemi riscontrati dalla valutazione. Questa valutazione può essere l'ancora di salvataggio per la nostra applicazione. Infatti, se in un caso estremo non fossero emersi problemi dalle valutazioni formative eseguite durante il processo di sviluppo, questa fase funge da ultima segnalazione di problemi del sistema, potendo prevenire un insuccesso disastroso una volta che il nostro prodotto sia stato immesso sul mercato.

Le tecniche di valutazione

Vogliamo ora concentrarci sulle tecniche più utili atte a farci ottenere una valutazione nelle varie fasi del processo di sviluppo. Secondo chi scrive sono tre le tecniche più importanti ed utili, che sono:

- Osservare gli utenti: tecnica usata anche per raccogliere requisiti, ma fornisce talmente tanti dati da fungere anche da valutazione. Questo in quanto può aiutare enormemente nel giudizio dei prototipi che sono somministrati agli utenti nelle fasi di design. Esistono molti modi in cui osservare gli utenti, questi possono essere prendere note mentre si osserva l'utenza alle prese con la nostra applicazione, registrare l'audio dei loro commenti durante i test, registrare video degli utenti (osservare le loro reazioni può far emergere le caratteristiche spiegate nel primo capitolo riguardanti lo stato di flusso). Si pongono anche certe problematiche dovute al fatto di non voler disturbare gli utenti mentre provano i nostri prototipi: a volte le reazioni di persone che sanno di essere osservate sono largamente falsate, differendo in maniera radicale dalle reazioni che avrebbero in uno stato di inconsapevolezza e totale privacy.

- Un'altra tecnica di valutazione molto più diretta rispetto alla precedente è il chiedere opinioni frontalmente agli utenti. Qui possono essere usate anche le tecniche di raccolta dati citate nella sezione precedente, in quanto chiedere singolarmente ad ogni persona nel caso di basi di utenza molto vaste è impossibile. Possiamo quindi utilizzare questionari e interviste per ottenere valutazioni anche molto precise su aspetti specifici dell'applicazione che stiamo sviluppando. In più, se per caso l'utenza fosse composta anche da esperti, il lavoro verrebbe velocizzato in maniera determinante e senza costi, in quanto sarebbe possibile discutere con essi senza mezzi termini o tramiti per aiutare la comprensione.

- Testare le performance degli utenti: osservare l'efficienza degli utenti mentre utilizzano due o più versioni della nostra applicazione può aiutarci a scegliere effettivamente una proposta di design rispetto ad un'altra. A volte due design possono apparire ugualmente validi agli occhi dello sviluppatore, che può utilizzare questa tecnica per ottenere ulteriori dati ma sopratutto una valutazione dell'efficienza di utilizzo reale dell'applicazione. I risultati di una valutazione di questo tipo possono aiutarci anche nello strutturare meglio le interfacce di utilizzo del nostro prodotto, arrivando a predirne l'efficienza e i possibili problemi durante l'utilizzo da parte dell'utente.

Si vuole fornire ora una serie di esempi di metodi di valutazione. Il primo che presentiamo prende il nome di "DECIDE". Si tratta di un insieme di canoni che possono aiutarci a strutturare meglio le tecniche di valutazione che intendiamo usare durante il processo di sviluppo, e a migliorare quelle già consolidate con consigli pratici.

DECIDE: un canone per la valutazione

Ogni lettere che compone l'acronimo "DECIDE" (decidere) fornisce un canone diverso e una serie di linee guida che lo sviluppatore dovrebbe seguire nello strutturare le valutazioni.

Vediamo quali sono:

- Determinare gli obiettivi che la valutazione deve raggiungere e le questioni ad essi relative: cosa, chi, perchè ecc...

- Esplorare le questioni specifiche che devono essere analizzate, quindi suddividendole in problemi più piccoli (divide et impera) per avere una visione più chiara e precisa. Le questioni analizzate possono includere attitudini dell'utenza, sicurezza del sistema, l'interfaccia dell'applicazione, l'affidabilità, l'accessibilità (si vedano gli obiettivi di usabilità ad esempio).

- sCegliere le tecniche con cui rispondere alle domande scaturite dalle questioni in analisi. Quindi adottare la tecnica più utile in base ai problemi che possono essere pratici, etici, e in caso raggiungere dei compromessi dove possibile.

- Identificare i problemi di natura pratica: ad esempio selezionare le persone da prendere come campione dalla base di utenza se questa fosse troppo vasta, oppure, cosa molto importante, studiare come restare nel budget messo a disposizione per lo sviluppo dell'applicazione, come rispettare la tabella di marcia del processo, selezionare gli strumenti tecnici ecc...

- Decidere come comportarsi con problemi di natura etica. Possono sorgere infatti questioni relative alla privacy, a dati sensibili ecc... Un altro consiglio è di mantenere sempre la trasparenza con gli utenti: questi devono sempre conoscere gli scopi delle nostre valutazioni e a cosa serviranno i dati rilevati dalla loro prova.

- inErpretare, valutare e presentare i dati. Decidere che tipo di dati si andranno a raccogliere, come analizzarli e come presentarli in base alle tecniche scelte. Bisogna considerare la loro affidabilità e validità, in quanto spendere tempo ed energie per poi ottenere dati inutili ed errati nel peggiore casi può portare ad un crollo di tutto quello che è stato costruito fino a quel momento nel processo di sviluppo.

Vogliamo anche dare un altro esempio di metodo di valutazione, in modo da sensibilizzare il lettore all'uso di questi utilissimi mezzi per relazionarsi con l'utenza.

La valutazione euristica di Nielsen

Una valutazione euristica è solitamente chiamata una valutazione "al risparmio" in quanto gli studi e le statistiche relative a questa tipologia di valutazioni dicono che anche solo cinque utenti scelti per valutare possono rilevare circa il 75% dei problemi di usabilità. Vediamo quali sono i punti su cui si fonda la valutazione euristica di Nielsen in particolare:

- La visibilità dello stato del sistema dev'essere sempre chiara e comunicabile all'utente. L'applicazione quindi dev'essere sempre in grado di trasmettere lo stato in cui si trova in quel momento (ad esempio usare barre di caricamento progressive invece di semplici schermate di caricamento che non fanno presumere se il sistema stia ancora funzionando o meno).

- Chiarire sempre i collegamenti tra il sistema e il mondo reale: nel nostro caso vogliamo avere sempre chiari i metodi di interazione che l'utente sfrutta per utilizzare la nostra applicazione: touch screen, voce, gestures ecc...

- Valutare sempre i controlli dell'applicazione per l'utente e il grado di libertà che questi gli forniscono. A volte troppa libertà nei controlli può portare ad errori nell'utilizzo dell'applicazione e a perdite di dati, causando disagio all'utente. Controllare sempre che il flusso di esplorazione dei menu sia coeso e semplice.

- Imporre degli standard durante lo sviluppo: questo aiuta a mantenere la consistenza dell'intero progetto evitando che alcuni membri del team prendano derive individuali andando a penalizzare il lavoro degli altri. Gli standard possono coincidere anche con gli obiettivi di usabilità e di esperienza, per indicare la qualità complessiva del progetto e fare previsioni di mercato.

- Valutare quanto il sistema sia in grado di riconoscere, diagnosticare e riparare agli errori degli utenti, riuscendo anche ad aiutarli nel non commettere più gli stessi sbagli ripetutamente. Forse una degli aspetti più difficili da catturare in quanto richiede una continua analisi delle performance durante le somministrazioni dei prototipi agli utenti coinvolti nel processo di sviluppo.

- Valutare quanto il sistema sia in grado di prevenire gli errori: una specie di dogma dell'informatica. Riuscire a prevenire gli errori degli utenti è uno degli obiettivi principali di usabilità e la valutazione dovrebbe misurare quanto questo aspetto sia considerato all'interno del processo di sviluppo e come sia stato affrontato.

- Valutare quanto l'utente sia in grado di ritrovarsi nell'ambiente dell'applicazione anche a distanza di tempo dall'ultimo utilizzo. Un'altra questione chiave che definisce l'usabilità dell'intero sistema e ne può decidere il successo.

- Osservare sempre come l'efficienza e la flessibilità d'uso varino con l'avanzare delle proposte e delle versioni della nostra applicazione. A volte una funzione ritenuta utile osservando i requisiti poi si rivela controproducente in quanto complessa per la base di utenza che utilizzerà la nostra applicazione. In questi casi o la si rielabora in modo più semplice, o è meglio rimuoverla definitivamente se non indispensabile alle specifiche di sistema.

- Valutare anche l'aspetto estetico dell'applicazione, preferendo generalmente design minimalisti e poco faticosi per l'occhio. Coincide con alcuni obiettivi di esperienza d'uso posti nei capitoli precedenti: in questo caso per avere un'analisi esaustiva dei dati ottenuti bisogna avere competenze di arte, grafica ecc... Quindi potrebbe rendersi necessario acquisire anche solo temporaneamente membri nel team che siano competenti in materia.

- Valutare quanto l'applicazione fornisca supporto e documentazione all'utente per aiutarlo nel suo uso. Nel caso delle applicazioni mobile è indispensabile che l'applicazione stessa fornisca un supporto continuo ma non invasivo mentre l'utente ne esplora tutte le funzionalità. Nessuno continuerebbe ad utilizzare un prodotto che non comprende appieno, infatti l'utente avrebbe sempre paura ad utilizzare nuove funzioni di cui non ha capito l'utilità.

5 – Sviluppi futuri

Il processo di sviluppo incentrato sull'utente è in continua evoluzione: anche mentre si procedeva alla stesura della tesi chi scrive ha continuato a scoprire altri testi e documenti che svelavano sempre nuove metodologie nello studio dell'utenza e del processo di design.

Gli sviluppi futuri che si potrebbero volere riguardano la completa disamina di tutto ciò che riguarda il processo di sviluppo, inserendo anche tutta la parte tecnica che è stata volutamente esclusa in questa sede per concentrarsi più sugli elementi innovativi che un approccio basato sullo studio della base di utenza può portare.

Riuscire a formare un manuale di utilizzo che possa fornire consiglio ai team di sviluppo era lo scopo iniziale di questa tesi, che è stato ridimensionato una volta che ci si è resi conto della vastità dell'argomento e quindi dell'impossibilità di potersi concentrare su ogni singolo aspetto. Se ci fosse la possibilità in futuro si vorrebbe estendere questo piccolo documento con tutti gli approfondimenti necessari ad una maggiore comprensione di tutto il processo di sviluppo in generale e di come questo possa essere avvantaggiato dallo studio dell'utenza.

Osservando la questione da un altro punto di vista, si potrebbe estendere il documento anche con gli aspetti più relativi all'usabilitàdi persone con deficit visivi o motori e come questi possano influenzare ulteriormente gli obiettivi e in generale tutto il processo di sviluppo.

Un'altra area che sarebbe possibile estendere ulteriormente è quella relativa alla grafica e allo stato dell'arte per quanto riguarda le applicazioni mobile, che sta diventando un vero e proprio lavoro separato dal design più classico ma non per questo meno importante.

6 – Conclusioni

Quando ho cominciato a raccogliere materiale per questa tesi mi sono trovato di fronte ad un dilemma. Cosa tenere e cosa scartare dall'enorme mole di materiale? Avevo già studiato il processo di sviluppo software lungo il mio percorso di studi, ed era un argomento già esplorato in lungo e in largo. Per questo ho deciso di concentrarmi sull'approccio basato sullo studio dell'utente. Purtroppo anche in questo caso per quanto ci sia un'assenza quasi totale di materiale in lingua italiana, una volta che mi sono rivolto alle pubblicazioni internazionali sono stato sommerso di conoscenze.
Alla fine ho deciso di ridurre l'obiettivo della tesi da un manuale completo ad un manifesto atto a sensibilizzare coloro che vogliono intraprendere o già hanno cominciato il lavoro di sviluppatori in un team ad utilizzare le tecniche e le metodologie incentrate sull'utenza, visto il loro potenziale e la percentuale di riuscita positiva.
Ecco quindi che ho iniziato ad evidenziare tutti gli aspetti che reputo imprescindibili di questo ambito.
La speranza è che qualcuno possa interessarsi come è stato per me a questo aspetto dello sviluppo molte volte sottovalutato, e che la curiosità scatenata da questo documento lo spinga ad approfondire degnamente il tutto.

7 – Bibliografia

Interaction Design: Beyond Human -Computer Interaction - Yvonne Rogers, Helen Sharp, Jennifer Preece

Costruire Sistemi Software: dai Modelli al Codice – Antonio Natali, Ambra Molesini

Flow: The Psychology of Optimal Experience - MihalyCsikszentmihalyi

The art of game design – Jesse Schell

L’enciclopedia dell’umanità Wikipedia, sempre utile a fugare piccoli dubbi.

Ringraziamenti

Vorrei partire ringraziando i miei genitori, che mi hanno sempre supportato nelle scelte che ho fatto sia in materia di studi che nella vita.

Ringrazio i miei due colleghi e ormai fratelli Beto e Gambero, con cui ho formato un bellissimo gruppo di studio e di amici.

Ringrazio il relatore che mi ha subito supportato nella scelta dell'argomento di tesi ed è stato sempre disponibile.

Ringrazio tutti gli amici che mi hanno sempre supportato e sopportato.

Infine ringrazio Christian Zoli, che grazie alle chiaccherate al Quintet ha fatto si che mi interessassi all'argomento consigliandomi i primi testi che ho utilizzato.