Programmazione d'Interfacce

Federico Matteoni

Indice

1 Introduzione

Appunti del corso di **Programmazione d'Interfacce** presi a lezione da **Federico Matteoni**.
More like *design d'interfacce*.

Prof.: **Daniele Mazzei**, mazzei@di.unipi.it
Riferimenti web:

- *?*

Esame: compitini/scritto + orale discorsivo dove si discute un software noto.
Possibile proporre un software personale da presentare all'esame orale, spiegando come si è applicati i rudimenti del corso sul software presentato.

Materiale didattico:

- **Google Classroom**, slide presentate a lezione e altro materiale didattico
  Codice **c14kiy** con le credenziali d'ateneo.
  La suite Google è attivabile a *start.unipi.it/gsuite*
  Non è autorizzata la divulgazione
- La Caffettiera del Masochista, Donald A. Norman
  Eng: The Design of Everyday Things
- Designing the User Interface, Ben Shneiderman
- *www.usability.gov*
- *interaction-design.org*

Ricevimento: Mercoledì 16-18, Stanza 366

2 Il Corso

Interface Development in 2020 diventa **Interface Design in 2020**

Diviso in due parti

- **UX e UI** con introduzione, UI vs UX, HCI, paradigmi, gamification...
- **Strumenti per lo sviluppo dell'interfaccia utente** presentati da vari ospiti: Unity, Zerynth, Ubidots, Angular, Amazon Lex, ...

Interfaccia è qualsiasi metodo utilizzato da una persona per **interagire** con un dispositivo.

3 Design

Cos'è il design Il design è la **pianificazione o la specifica per la costruzione di un oggetto o sistema** o per l'implementazione di un'attività o processo. Diventa l'esatto opposto della decomposizione del problema in sottopassaggi, cioè del pensiero computazionale. Il design parte dalla base del problema e **identifica soluzioni per la causa del problema**. Si può avere anche il design di una strategia di implementazione.
„**Bisognerebbe progettare le applicazioni come se fossero persone che ci piacerebbe frequentare**". Ad esempio Netflix, o il frigorifero.

3.1 XX Designer

Discernere tra Graphic Design, User Experience Design (UX Design) e User Interface Design (UI Design).

UX : come l'utente si sente per interagire e cosa vuole fare. Aspetto più psicologico, guida la UI design in base a statistica fatta su gruppi di utenti. Manda "*l'output*" a chi fa UI e al marketing.

UI : come l'utente interagisce col prodotto (shortcut, sottomenu...)

3.1.1 UX Designer

Si deve porre il problema di quali approcci usare per risolvere problemi evidenziati da analisi di mercato.
Chi paga non è detto che sia chi usa il servizio. Ad esempio Netflix viene pagato da una persona, ma lo stesso account viene usato anche da altre persone (anzi, in particolare **il 90% del tempo** chi usa l'account non è chi paga).
Uno dei metodi usati per fare UX è quello della **definizione delle personas** (cioè un archetipo di utente). Una persona può assumere diverse personas.

User Experience Con User Experience si parla del prodotto e di come si comporta nel mondo reale, che è fatto di *personas*. **Non si può progettare *una user experience*, si può progettare *per la user experience***. La user experience è ciò che fa l'utente, e lo sviluppatore non ha controllo su ciò. L'utente si approccia al software come gli pare.

3.1.2 UI Designer

Dalla UX si crea lo **sketch** dell'interfaccia. Non viene prodotto subito il wireframe ma bisogna partire da altro, ad esempio dai **casi di studio**. Esistono più casi di studio per ogni personas (casalinga voghera che fa bonifico, casalinga voghera che cambia password, ecc.). Ogni caso di studio è **specifico per personas**, poiché personas diverse hanno capacità diverse (non conoscere alcuni concetti, non saper fare determinate operazioni...).
L'**UI design è un procedimento diverso dal front-end developing**, quindi possono essere persone separate. Il designer progetta le guideline che istruiscono il developer.
Si può dire che la UI design è sottoarea di UX design.

3.2 Front-End Developer

Esegue il design della UI convertendolo in funzionalità del prodotto.

4 Interfacce Utente

L'interfaccia utente (UI) L'UI di un sistema è **lo spazio dove avviene l'interazione uomo-macchina**: lo schermo, le casse, il mouse e quant'altro.
L'obiettivo dell'interfaccia è far si che **l'utente possa controllare la macchina**, e **non il contrario**. L'interfaccia può però influenzare il comportamento dell'utente, ad esempio se voglio guidare l'utente in un particolare modo l'interfaccia deve dare un feedback tale da guidare l'utente.
L'altro obiettivo dell'interfaccia è **rendere fruibile *in maniera piacevole* le funzionalità che una macchina eroga** verso l'utente. Il termine **user-friendly** non può essere omesso: tra un'app facile e piacevole da usare e una solo facile da usare, l'utente medio preferirà sempre la prima.
L'interfaccia è strutturata a layer. lo HID (Human Interface Device) è la periferica con cui l'umano interagisce col sistema. Questo server per usare più HID per interagire con diverse applicazioni.
HMI (Human Machine Interface) è più astratta rispetto a HCI (Human Computer Interface), quindi in HMI è più teorica la cosa.

Diversi tipi di interfacce Abbiamo 5 sensi, quindi diverse **categorie d'interfaccia**: le più comuni sono **grafiche** e **tattili** (**GUI**, Graphical User Interface). Se si aggiunge anche il suono diventano **MUI** (Multimedia User Interface). Il concetto di GUI è stato coniato in un tempo in cui l'audio era raro. Adesso **praticamente tutte le interfacce sono MUI**.
Esempio di MUI riprogettata in GUI: Facebook. I video partivano in automatico con l'audio attivo, mentre ora sono mutati. Poi sono stati aggiunti i sottotitoli automatici: questo è un esempio di tecnica ideata per le utentze disabili e riusata per poter far fruire il prodotto a quelle personas che in quel momento non possono usufruire dell'audio. *Meglio un sottotitolo sbagliato che niente.*

Categorizzare interfacce Le CUI (Composite User Interfaces) sono le UI che interagiscono con due o più sensi. Esistono tre diverse macrocategorie di CUI:

> **Standard**, che utilizzano dispositivi standard come tastiere, mouse e monitor
>
> **Virtuale**, che **bloccano il mondo reale e creano un mondo virtuale** e tipicamente utilizzano dei caschi VR
>
> **Aumentata**, che **non blocca il mondo reale e eroga contenuti non completamente digitali**, ma che prendono dalla realtà esterna che circonda l'utente

Le CUI possono anche essere **classficiate per il numero di sensi** con cui esse interagiscono. Per esempio, lo *Smell-O-Vision* è una CUI standard 3S (3 sensi) con un display, suono e odori. Se si aggiungesse la vibrazione della poltrona, diventerebbe 4S poiché si aggiunge il tatto.
Si parla di **Qualia Interfaces** quando si stimolano tutti i sensi.

Mancata evoluzione Le UI **sono le stesse di 10 anni fa**. Bisogna mettere in discussione i paradigmi attuali. L'industria ha convertito l'ambiente fisico della scrivania in ambiente digitale, prendendo ispirazione dall'abitudine dell'utente per rendere più semplice il passaggio. Ora l'utente è abituato, la realtà da cui si prende spunto non esiste più. Bisogna cambiare.

5 Good and Bad Design

Il buon design non esiste, poiché si fa design *per* la user experience **di una determinata personas**. Le due caratteristiche più importanti su cui misurare il buon design sono:

> **Discoverability**: è la **capacità innata di un sistema di veicolare i possibili usi e dire come si usa**. Non è detto che una volta che si è capito cosa si può fare si riesca a farlo.
> Per avere buona discoverability si usa tipicamente la visibilità: un rubinetto con i pomelli bene in vista incrementa la discoverability. Nel software, **questo lavoro lo fanno i pulsanti**.
>
> **Understanding**: è la **capacità di comprendere i possibili usi**. Ad esempio: il fornello, è in cucina quindi so che si usa per scaldare ecc., il problema maggiore però è il mapping pomello → fornello. Si può risolvere con l'icona del fornello corrispondente, ma non risolve effettivamente il problema. Una soluzione efficacie è disporre fornelli e pomelli in modo che sia evidente la correlazione fra essi.
> **Non sottovalutare il costo mentale dell'utente**.


5.0.1 Design of Useful Things

Il paradosso di TripAdvisor „*Quando la gente mangia bene, non recensisce. Quando mangia male, recensisce*".

Sensazioni Quando le cose vanno bene, si dimenticano subito. Questo perché, in qualche modo, l'uomo pensa che **le cose vadano bene per definizione**. Quando qualcosa va storto, invece, **l'amigdala crea un ricordo con un peso molto maggiore**.
Il design deve quindi preoccuparsi di come funzionano le cose, come vengono controllate e della natura delle interazioni. Quando la progettazione è fatta bene, crea prodotti piacevoli e brillanti. Quando è fatta male, i prodotti sono inutilizzabili e ciò porta a notevole frustrazione e irritazione.
Marcatore somatico: ricordo le esperienze in base alle sensazioni che provavo durante esse. **Più forte è la sensazione più si cementifica il ricordo**. Ad esempio se faccio un incidente ad una curva, la ricorderò bene per molto tempo. La strada che faccio per andare in vacanza non la ricordo più già al ritorno.

Con il software si applica lo stesso discorso. Se non riesco ad usare un programma inizio a provare frustrazione. Gli umani non informatici tengono a ritenere le macchine come superintelligenti, quindi associano alla frustrazione l'incapacità personale: **se credo di non essere in grado di usare il software non ci riprovo**.
Confrontando IA e intelligenza umana, l'IA risulta strettamente limitata a computazione e risoluzione di problemi logici. Al contrario, **la mente umana non funziona ad algoritmi ma procede per deduzione**. Per lo più generando ipotesi senza fondamento e autoconvincendosene.
Le macchine seguono regole semplici: gli algoritmi. Essi non hanno la flessibilità (**common sense**) tale da assecondare l'utente. Per esempio, se chiedo telecomando per l'aula D2 ma non esiste o non c'è il proiettore, la signora mi corregge in D1 e dà il telecomando corretto. La macchina dice semplicemente che non esiste l'aula D2 o il proiettore in aula D2.
Le macchine non hanno buonsenso. La maggiorparte delle regole sotto il software sono note solo agli sviluppatori. Potrebbe andare bene, basta renderle discoverable.

Bisogna invertire il paradigma attuale: se qualcosa va storto **è colpa dello sviluppatore** e non dell'utente. **Il dovere della macchina è essere comprensibile** da parte dell'utente.

Bisogna accettare che il comportamento umano è com'è e non come vogliamo che sia.

6 Human Centered Design

Alla fine di ogni passaggio c'è l'utente.
Si tratta di una norma `ISO 9241-210:2010(E)`.

Un approccio Lo HCD è un **approccio di design** specificamente orientato allo sviluppo di sistemi interattivi con l'**obiettivo di fare sistemi utili, altamente usabili e che si focalizzino sull'utente**. Il metodo è orientato **all'efficienza ed all'efficacia**, per aumentare la soddisfazione dell'utente ed evitare il più possibile gli effetti negativi.

Prima l'utente, poi le features Lo HCD mette **i bisogni, comportamenti e capacità umane prima di tutto, e progetta in funzione di esse**.
Significa che se devo risolvere un problema, non mi interessa risolverlo completamente ma raggiungere il miglior risultato che posso far ottenere all'utente che usa il mio software. Se il *70% degli utenti raggiungono il proprio scopo* col nostro software, allora esso ha un'*efficacia del 70%*. Posso puntare ad un'efficienza maggiore magari risolvendo una parte minore del problema.
Less is more. Meglio una feature in meno che una in più. Ogni volta che aggiungi una feature devi dimostrare perché e a cosa serve, perché tale feature va: spiegata, testata, mantenuta oggi e domani (**backward compatibility**).

Il problema principale delle UI è un **problema di comunicazione** in particolare dalla macchina verso la persona. Una buona interfaccia sa comunicare con l'utente.
Progettare interfacce che funzionano egregiamente fintanto che le cose vanno bene è relativamente facile, ma **la comunicazione è ancora più importante quando le cose non vanno bene**: entrano in gioco le **strategie di mitigazione dell'errore**. Si focalizza l'interazione soprattutto nel **comunicare ciò che è andato storto**, in quel momento devo aiutare l'utente frustrato a risolvere il problema perché se lo aiuto a risolvere il problema da solo proverà una sensazione positiva di successo per aver capito cosa non funzionava. Ciò **crea empatia col sistema**.
Quindi bisogna **evitare la frustrazione**, e **aiutare a risolvere** quando insorge un problema.

Capire l'utente Lo HCD è una filosofia di design che parte dalla **comprensione delle persone e dei bisogni** che si intende soddisfare. Spesso gli utenti non si rendono conto dei loro effettivi bisogni e nemmeno delle difficoltà che incontrano.
Per capire l'utente la tecnica più utilizzata è l'osservazione. Non è detto sia sempre possibile. Versioni alpha e beta non servono solo debuggare il software, ma servono anche a capire ciò che fanno gli utenti. Diventa utile avere statistiche sull'utilizzo effettivo del sistema: quanti click su un determinato pulsante, quante volte una determinata procedura finisce e così via.
Le specifiche dello HCD, quindi, **nascono dalle persone** e per questo **non si possono scrivere**. Quindi risulta essere un paradigma che si sposa bene con la computer science perché va avanti per iterazioni: si esegue una specifica ad alto livello, ne implemento una parte, la testo sull'utente reale e tramite il feedback modifico la parte implementata e ri-testo. Quando ritengo buono ciò che ho prodotto lo congelo, e passo ad implementare un'altra parte dell'interfaccia.

Il ruolo dello HCD nel design	
Experience design	Area di focus
Industrial design	Area di focus
Interaction design	Area di focus
Human Centered Design	Il processo che assicura che la progettazione incontra i bisogni e le capacità degli utenti che useranno il sistema

Possiamo progettare per esperienza utente, il design industriale e progettare per l'interazione. lo HCD non è area di focus del processo di design ma è metodo.
Utilizzo l' HCD per progettare tutto il resto.

7 Design Thinking vs HCD

Insieme al termine HCD si può vedere Design Thinking
Vengono da due scuole di pensiero molto forti ma con visioni diverse. Design thinking segue il filone stanford: **processo** di design con cui progettare nuovi prodotti. PIù che antropocentrico è disruptive innovation. Metodo, strumento per sviluppare prodotti innovativi.
Empathize: studiare il proprio pubblico. Disegna prodotto in modo che stabilisca link empatico con l'utente.
Define: delineare meglio le **domande chiave**, cioè quali bisogni a cui assolvere.
Ideate: brainstorming e creare soluzioni
Prototype
Test

Lo HCD nasce su IDEO.org. Mindset che sovrappone il design thinking orientato a come garantire che le ideazioni siano rilevanti e beneficiali, sul lungo termine, per le persone obiettivo.
Quando ispirazione (divergente) cala si passa a ispirazione (convergente) (unire idee simili, scartare idee ridondanti...).

Il design thinking lo si usa per sviluppare qualsiasi modello di business orientato all'essere profittevole.
5-step iterativi processo che porta all'effettivo sviluppo di prodotti/soluzioni che verrà adottato dal beneficiario finale desiderato

Lo HCD si ha sopra il design thinking: identificato il modello di business uso lo HCD per sincerarmi che la famiglia di soluzioni identificate venga pulita attraverso processo che garantisce l'usabilità da parte di soggetti umani.
Mindset e strumento applicato oltre al design thinking che crea un impatto positivo long-term per l'utente della soluzione.

8 fundamentals principles of interaction

Life is made of experiences Bravi designer producono **esperienze** piacevoli.
L'esperienza è critica nel determinare quanto bene si ricorderanno l'interazione.

cognizione ed emozione sono profondamente legate. Se non metto utente in mood positivo farà più fatica ad apprendere l'interfaccia. Più mi arrabbio meno sono predisposto a comprendere e riutilizzare il prodotto.

Sei fondamenti Discoverability

8.1 Affordance

Si riferisce alla relazione tra un oggetto fisico e una persona. (Es. una UI con un bottone da premere con uno HID è un oggetto fisico, che sia con un mouse o col dito).
Relazione tra le proprietà di un oggetto e le capacità della persona che sta interagendo con quell'oggetto. Questa proprietà determina il modo con cui l'oggetto può essere usato. Una sedia "affords" per sostenere, quindi "affords" (consente) di sedercisi. Pulsante afforda per essere premuto
Switch afforda per essere flippato
Potenziometro afforda per essere rotato

ci sono affordance innate nel cervello: **forme** che il sistema visivo e il cervello interpretano automaticamente.
L'affordance è una proprietà scaturita da una relazione (quindi è peculiarità della relazione). Per una poltrona afforda

sostenere per quasi tutti, ma lo spostamento non è detto sia affordato per tutti (un secco non può)
Anti-affordance: prevenzione dell'interazione. Es spunzoni per evitare che piccioni si posino su qualcosa.
Affordance e anti-affordance devono essere discoverable e percievable. Non è immediato: il vetro afforda la luce e non afforda la materia, ma si può non vedere e avere una falsa affordance di passarci attraverso e ci batto.
Schermo spento, smartphone ha comunque affordance di essere premuto.
Cosa posso fare su un interfaccia

8.2 Signifiers

Significanti
Designer hanno problemi pratici
Devono sapere come progettare cose per renderle understandable
Un significante è un modo per dire dove tu puoi applicare quell'affordance per ootenere risultato
Un box quadrato in una GUI è un significante (bottone): se applichi affordance tocco qua ottieni risultato
Non "metto un affordance", è **sbagliato**. Posso dire "metto un significante" ma solo se ho un'affordance. I tre pallini per il tasto menu sono un significante
Affordance del touch, slide, pinch... esiste su tutto lo schermo
Affordance dice **cosa** posso fare, signifier dice **dove** fare l'azione.
Possono essere **voluti** o **accidentali**.
Voluto es. l'etichetta
Accidentale es. persone in fila alla stazione
A volte i significanti sono indispensabili perché maggiorparte affordance sono invisibili. Servono per fare capire le affordance che non si vedono. Es. porte scorrevoli: se non vedo i cardini, vedo la maniglia e spingo la porta ma non si muove perché è da far scorrere. La spinta è un'affordance percepita che non esiste.
Nel design i significanti sono molto più importanti delle affordance, perché comunicano come usare il design. Questo perché viviamo in un mondo in cui le affordance sono state già presentate in genere. Creare nuove affordance è molto molto difficile.
Fare buoni significanti è un mestiere difficile.
Come associare affordance e significante a reali azioni? Nella maggiorparte dei casi tramite convenzioni.
La comprensione di un'affordance percepita è dovuta a convenzioni culturali.

8.3 Mapping

Importante nel fare interfacce e stabilire significanti. Disposizione significanti a parità di significanti può dire di più sull'interfaccia e le funzionalità.
Mapping è la relazione tra elementi di due insieme.