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 14.30-16, 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 Human Centered Desing, spesso si può vedere il termine **Design Thinking**. I termini vengono da due scuole di pensiero molto forti ma con visioni diverse.

Cos'è il Design Thinking Il **Design Thinking** segue il filone Stanford, dove è nato: è un **processo di design** con cui **progettare nuovi prodotti** che verranno **effettivamente adottati dalle persone**. Come processo è più vicino alla disruptive innovation che all'antropocentricità.
Metodo, strumento per sviluppare prodotti innovativi. Per sviluppare qualsiasi modello di business orientato all'essere profittevole.
Si suddivide in 5 fasi iterative.

Empathize **Studiare** il proprio pubblico. Progettare il prodotto in modo che stabilisca un collegamento empatico con l'utente.

Define Delineare meglio le **domande chiave**, cioè quali sono i bisogni a cui assolvere.

Ideate **Brainstorming**, creare soluzioni.

Prototype **Costruire** una o più idee.

Test **Testare** le idee e **ricevere feeback**.

HCD e DT Lo HCD è un mindset che viene sovrapposto al design thinking, il quale è orientato a garantire che le idee siano rilevanti e beneficiali, sul lungo termine, per le persone obiettivo.
Lo HCD quindi viene sovrapposto all design thinking: identificato il modello di business, uso lo HCD per sincerarmi che la famiglia di soluzioni identificate venga "pulita", attraverso un processo che **garantisce l'usabilità da parte di soggetti umani**.

Design Thinking	**Human Centered Design**
Processo iterativo a 5 fasi che porta all'effettivo sviluppo di prodotti/soluzioni che verranno adottate dall'utente finale desiderato	Mentalità e strumento da applicare insieme al Design Thinking che crea un impatto a lungo termine positivo, per gli utenti della soluzione

Quando l'ispirazione (divergente: produrre idee) cala, si passa all'ideazione (convergente: unire le idee simili, scartare idee ridondanti...).

8 Principi Fondamentali dell'Interazione

Life is made of experiences Bravi designer producono **esperienze** piacevoli. L'esperienza è molto importante, perché determina quanto bene gli utenti si ricorderanno l'interazione.

Cognizione ed Emozione Quando la tecnologia si comporta in maniera inaspettata, proviamo confusione, frustrazione e rabbia: **emozioni negative**. Quando invece comprendiamo il comportamento della tecnologia, abbiamo una sensazione di controllo, bravura e persino orgoglio: **emozioni positive**. **Cognizione ed emozione sono profondamente legate**. Se non metto l'utente in un **mood positivo** farà più fatica ad apprendere l'interfaccia. Più mi arrabbio meno sono predisposto a comprendere e riutilizzare il prodotto.

8.1 Sei Fondamenti

La **Discoverability**, cioè il grado di facilità con cui un utente **scopre come funzione l'interfaccia**, è il risultato della corretta applicazione di sei principi psicologici.

8.1.1 Affordance

Il termine **affordance** si riferisce alla **relazione tra un oggetto fisico e una persona**: precisamente la relazione tra **le proprietà di un oggetto e le capacità dell'utente che determinano i possibili utilizzi dell'oggetto**. **Questa proprietà determina il modo con cui l'oggetto può essere usato.**
"Cosa posso fare sull'interfaccia".

Esempi **Un pulsante di una UI**, da premere con uno HID che sia il dito o il mouse, **è un oggetto fisico**. Il pulsante *afforda* (**consente**) l'essere premuto.
Una sedia *afforda* il sostenere, quindi *afforda* di sedercisi.
Un potenziometro *afforda* l'essere ruotato.

Tipi di affordance 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 con un particolare soggetto** (quindi è peculiarità della relazione). Ad esempio, una poltrona *afforda* il sostenere per quasi tutti, ma lo spostamento non è detto sia *affordato* per tutti (per esempio una persona debole non può spostare la poltrona).
Anti-affordance: prevenzione dell'interazione. Ad esempio degli spunzoni per evitare che piccioni si posino su un cornicione, **prevengono l'*affordance* che il cornicione ha verso i piccioni di sedersi**.
Affordance e anti-affordance **devono essere discoverable e percievable**. Questo fatto non è scontato: il vetro *afforda* l'essere attraversato dalla luce e non *afforda* l'essere attraversato dalla materia, ma si può non vedere e **percepire una falsa *affordance*** di passarci attraverso... e ci batto.
Un altro esempio: anche a schermo spento, lo smartphone ha comunque l'*affordance* di essere premuto.

Assolutamente sbagliato dire che "metto un *affordance*". Posso dire che "metto un significante", ma solo se ho un'*affordance*. I tre pallini per il tasto menu sono un **significante**.

8.1.2 Signifiers

I designer hanno problemi pratici: devono sapere come progettare le cose per renderle understandable. Un significante è un **modo per indicare dove applicare un determinato *affordance* per ottenere un risultato**

Esempi Un box quadrato in una GUI (un pulsante) è un **significante**: se applichi l'*affordance* "tocco" qua ottieni un determinato risultato.
L'*affordance* del touch, lo slide, il pinch... esiste su tutto lo schermo. L'*affordance* dice **cosa** posso fare, il significante dice **dove** fare l'azione.
A volte **i significanti sono indispensabili** perché la maggiorparte delle *affordance* sono invisibili. I significanti servono per fare capire le *affordance* che non si vedono. Per esempio, le porte scorrevoli: se non vedo i cardini, quando vedo la maniglia decido di spingere la porta ma essa non si muove perché è scorrevole. 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.

Convenzioni Come associare l'affordance e il significante ad azioni reali? Nella maggiorparte dei casi tramite **convenzioni**. La comprensione di un'affordance percepita è dovuta alle convenzioni culturali.

Tipi di signifiers I significanti possono essere **voluti** o **accidentali**.

Voluto Ad esempio un'etichetta, una stringa, un'icona.

Accidentale Ad esempio delle persone in fila alla stazione.

8.1.3 Mapping

Il **mapping** è di grande importanza nel progettare le interfacce e stabilire i significanti. La **disposizione** dei significanti, a parità di significanti, può dire di più sull'interfaccia e le funzionalità.
Il **mapping è la relazione tra elementi di due insiemi**. Il modo migliore per fare mapping è **quello naturale**, perché è un'attività in cui il nostro cervello è molto bravo, ed il mapping di forme geometriche è la prima cosa che si impara da bambini.

8.1.4 Feedback

Un altro elemento fondamentale per il design delle interfacce è il **feedback** inteso come **risposta dell'interfaccia verso l'utente**.

Immediato Il feedback **deve essere immediato**. Il sistema sensoriale è parte integrante del sistema cognitivo, e l'uomo usa i propri sensi per guidare i propri ragionamenti. Se progetto un'interfaccia che non abilita i miei sensi a capire cosa sto facendo, inizio a fare più fatica a usare il prodotto o non ci riesco proprio. Un esempio: una pagina web che **non mostra se sta caricando la procedura richiesta**.
Uno dei **problemi principali** del feedback quindi è **il tempo**. Se faccio un'azione, **devo avere un feedback entro un certo lasso di tempo**. Se questo tempo è superato, il mio cervello non è più in grado di associare il feedback all'azione compiuta e ho così due pessimi risultati: **non ho dato feedback** e **ho mandato in confusione l'utente**. **Il feedback deve avvenire entro massimo 100 ms dall'azione, altrimenti non sarà efficace**. Meglio un **buon** feedback che un **bel** feedback.

Informativo Inoltre, il feedback **deve essere informativo**. Questo non significa che deve portare con sé tanta informazione, ma che deve **assolvere al proprio obiettivo**. Un esempio: se premo un pulsante non ho bisogno di fare grandi cose come feedback, posso **semplicemente** farlo diventare grigio. Non servono messaggi del tipo "*ok pulsante premuto*" ecc., sono superflui.
Colorare il pulsante di rosso o di verde **non è più informativo**: creo confusione a causa del mapping naturale tra il colore e il significato (rosso → errore, verde → ok) e l'utente non capirà se ha ottenuto un errore o se la richiesta è stata ricevuta correttamente.
Il feedback deve quindi essere **informativo nell'accezione dell'azione a cui è associato**. *Meglio nessun feedback rispetto ad un feedback errato.*

Semplicità **Non bisogna essere troppo pedanti**. Se il feedback è eccessivo, l'interfaccia utente diventa pesante. Altro problema che può insorgere è un feedback non allineato con il contesto dell'utilizzo del dispositivo. Per esempio, non posso usare lo stesso beep delle cinture per segnalare la riserva. Il beep delle cinture è fastidioso perché *deve esserlo*, ma la riserva, quando viene segnalata, non è in un contesto urgente. Se il beep è fastidioso, o spaventa, posso mettere in pericolo la vita dell'autista se viene spaventato mentre guida. **Non limitarsi alla tecnologia disponibile** "*ho solo quel buzzer, non posso fare altrimenti*". Nell'esempio non sono obbligato a far partire un beep quando si entra in riserva, posso **semplicemente** fare lampeggiare la spia.

Esempi Il **feedback** della luce del pulsante dell'ascensore quando viene premuto.
Un messaggio "*Pagamento eseguito*" a termine di una procedura pagamento.

8.1.5 Conceptual Model

Un **modello concettuale** è **una descrizione estremamente semplificata delle funzionalità del sistema**.
esempio classico i file e le cartelle. Come racconto a qualcuno com'è organizzata memoria archiviaz di un computer? Uso modello concettuale noto, cioè fogli di carta con contenuti, raccolti in raccoglitori e raccoglitori in schedari.
Agli utenti non interessano come funzionano le cose, ma che funzionano. Perché l'hanno comprato.
Modelli concett servono per andare incontro all'utente, per convertire roba di complessità tecnica in comprensibile da chiunque.
I modelli concettuali già in commercio sono difficili da mettere in discussione. Mai far valutare binario 1v1 agli umani, perché c'è vivo o morto. Se invento nuovo sistema streaming musica/film è impossibile. La gente ha spotify e netflix, non importa cosa fanno o come.
Modello concett "film non comprati su dvd" = netflix
Modello concett "musica non comprata su cd" = spotify
modello concettuale è come il designer vuole che l'utente percepisca la piattaforma. Sarebbe l'ambizione di progettare (la comprensione) della UX.

Pensato modello concett e disegnato. Implementata interfaccia per fare si che modello concett venga veicolato. Quando persona s'interfaccia con sistema sviluppa un **modello mentale**. Se modello mentale e concettuale sono allineati, persona è in grado di usare il sistema. Più è grande la differenza tra modello mentalee concettuale più la persona farà fatica.
utente può sviluppare modello mentale diversi per diverse funzionalità dello stesso sistema.
modello concettuale è trasferito all'utente per spiegare come l'interfaccia funziona, non com'è fatta. Pomello comanda tale fornello è mapping, ma che tale fornello spruzzi fuoco o gas è modello mentale dell'utente.

La gente apprende i modelli concettuali direttamente dal device per tentativi, ma il dramma è quando lo apprendi per passaparola. telefono senza filo, da persona a persona cambia l'iterpretazione. Per ogni disallineamento tra concett e mentale la cosa cambia. Per questo necessità concett pressoché univoco col mentale. Più piccolo è il delta, meno rischio di creare comportamenti assurdi col passaparole. Less is more, se feature difficile da veicolare non metterla.
Quando utenti feature la usano con successo per l'85% e 70% utenti dell'app usano feature allora posso inserirne un'altra.
significanti veicolano modello concettuale.

8.1.6 System Image

le persone creano modelli mentali di se stessi, altri, ambiente e delle cose con cui interagiscono. Questi modelli concettuali sono creati attraverso l'esperienza, allenamento e istruzione.
Nell'immagine di sistema troviamo tutti questi componenti. è il modello concettuale che l'utente si crea dell'intero sistema grazie all'esperienza. quasi l'insieme dei modelli mentali.
Racchiude concettuale e mentale e descrive come stanno nel sistema complesso.
utente non può chiedere a svilupp come funziona, ma deve scoprire funzionalità da solo. la teoria del'imm di sis dice che utente sviluppa uno user model (modello utente) grazie a oggetto e elementi a contorno (manuale istr, pubblicità, passaparola...). Progettare modello concettuale è solo parte dell'immagine di sistema. Aiuto utente a sviluppare modello mentale vicino concettuale se lo doto di altre informazioni: blog del software ad esempio.
modello concettuale ciò che voglio utente pensi, mentale ciò che pensa. immagine di sistema è insieme info (ui, materiali di contorno) che consente all'utente di formarsi il modello mentale più consono. essere bravi a farlo in modo che tutti modelli generati siano compatibili con modello concettuale.