IUM Interazione Uomo Macchina

A.A. 22/23

Who, When and Where

•
• Computer science department, University of Pisa
• Largo Pontecorvo, 2nd floor, room 366
• daniele.mazzei@unipi.it
• Roberto Figliè
• Computer science department, University of Pisa
• roberto.figlie@phd.unipi.it

● Remote lectures streaming via Microsoft Teams, link on esami.unipi.it or HERE
● Teaching material sharing to be defined :)
● Office hours (mainly via remote call): Thursday 9:00-11:00 (30 min slots). Max 2 students per slot via Google Meet (link available in the booked calendar event), book your slot HERE

Microsoft Teams

https://t.ly/7iI1

Books and other material

● La caffettiera del masochista. Il design degli oggetti quotidiani – Donald A. Norman
○ [english] The Design of Everyday Things
● Designing the User Interface: Strategies for Effective Human-Computer Interaction, Global Edition - Ben Shneiderman [english only]
● Human Computer Interaction, L. Gamberini, Pearson
● Web: https://www.usability.gov/ and https://interaction-design.org
● Classroom GDrive for Slides
● Dispense (ITA, non complete) QUI
● Ask ChatGPT :)

Exam

[6 CFU] = Oral exam (admission quiz possible)

[9 CFU] = Oral exam (admission quiz possible) + 3CFU Project

INTRO

UNA NUOVA ERA

Una nuova era
Negli ultimi anni il ruolo degli informatici è decisamente cambiato.
Per anni l'informatico è stato chiuso in uno sgabuzzino ad interagire in solitudine con una tastiera, bevendo bibite gassate mentre qualcuno gli diceva che cosa serviva (o almeno era convinto di sapere cosa servisse) all'azienda per crescere.

Obiettivo del corso
Questo corso è una trattazione adattata per informatici delle teorie di human computer (HCI) e human machine interaction (HMI) [interazione uomo-computer e interazione uomo-macchina].
L'obiettivo di questo corso è quindi quello di fornire agli studenti del corso di laurea in Informatica gli strumenti necessari a comprendere e gestire il processo di sviluppo delle interfacce e dei prodotti interattivi.
Questo corso ambisce a spostare l'informatico dal suo tipico ruolo di sviluppatore elevandolo quindi a progettista (o al minimo contributore) non solo del “codice” ma dell'intero prodotto.

Progettare l’interazione
Questo corso è ispirato alla teoria dell'interazione sviluppata dal Prof. Donald Norman ed in particolare, le dispense sono in parte un adattamento didattico del libro “La caffettiera del masochista” di Donald Norman, pubblicato in Italia da Giunti e disponibile in lingua originale come “The design of everyday things, D. Norman”.

Progettare l’interazione

Nel corso verranno trattati anche aspetti relativi all'Internet delle cose e alle interazioni con robot e altri sistemi “smart”.
Questi aspetti legati all'interazione con oggetti smart sono anch'essi ispirati agli studi di Norman e sono ampiamente trattati nel libro “Il Computer Invisibile, D. Norman”, pubblicato in Italia da Apogeo.

Progettare l’interazione

Per quanto riguarda invece Usabilità e Dispositivi per l’Interazione è consigliato il libro “Human-Computer Interaction, I fondamenti dell'interazione tra persone e tecnologie”, L. Gamberini, Pearson.

Il Prodotto

Nel corso parlerò spesso di prodotto, business, acquisto e altri termini legati al mondo della vendita, dell'economia e del mercato.
Questo perché l'informatico deve sviluppare una consapevolezza fondamentale per il suo lavoro:
Un prodotto che nessuno compra è un prodotto inutile
Non importa quanto geniale sia il codice che avete scritto o rivoluzionario il sistema che avete implementato, se non vi curerete di far si che questo artefatto venga apprezzato e quindi utilizzato dalle persone la vostra creazione, geniale o stupida che sia, morirà dentro il vostro computer.

Progettare l'Interazione fra Uomo e

Macchina

Design

La rivoluzione tecnologica degli ultimi anni ha portato a parlare molto di User Experience Design, User Interface Design, Interaction Design etc.
Cosa si intende con il termine Design?

Da Treccani.it: design s. ingl. [propr. «disegno, progetto», dal fr. dessein, che a sua volta è dall'ital. disegno],
usato in ital. al masch. – Nella produzione industriale, progettazione (detta più precisamente industrial design) che
mira a conciliare i requisiti tecnici, funzionali ed economici degli oggetti prodotti in serie, così che la forma che ne
risulta è la sintesi di tale attività progettuale; quando la forma dell'oggetto viene elaborata indipendentemente dalla
progettazione vera e propria, si parla più propriam. di styling design.

Design
In Italiano tendiamo quindi ad usare il termine design per riferirci sia al processo di progettazione
che al risultato stesso di questo processo
● Processo: "E' stato avviato il design dell'interfaccia grafica"
● Risultato del processo: "Ecco il design dell'interfaccia grafica pronto per essere implementato"
Per design si intende quindi sia il processo di progettazione e pianificazione che l'output stesso di questo
processo

Può un informatico diventare designer?
E' importante notare che nel mondo del design, ed in particolare nel design industriale, si approccia alla risoluzione dei problemi e alla progettazione con una forma mentis molto diversa rispetto a quella "computazionale" tipica del mondo informatico.
Nel 2006 Jeannette Wing, direttrice del Dipartimento di informatica della Carnegie Mellon University, formulò la seguente definizione di Pensiero Computazionale:
il pensiero computazionale è un processo di formulazione di problemi e di soluzioni in una forma che sia eseguibile da un agente che processi informazioni.

Il pensiero computazionale
Il Computational Thinking non consiste semplicemente nel saper programmare, ma nel pensare a diversi livelli di astrazione

Il pensiero computazionale è quindi un processo mentale che consente di risolvere problemi di varia natura seguendo metodi e strumenti specifici, pianificando una strategia; abitua al rigore e quindi rende possibili gli atti creativi. Permette di interagire con persone e strumenti, di fruire delle potenzialità delle macchine quali oggetti capaci di compensare le lentezze o l'imprecisione dell'uomo, se ben programmati.

Pensiero computazionale vs Design

Come tutte le scienze, ha i suoi fondamenti formali nel linguaggio matematico e ha a che fare con oggetti del mondo reale. Il pensiero computazionale è infatti basato sulla suddivisione di un problema in sotto-problemi così da poter giungere ad una formalizzazione del problema sotto forma di algoritmo (serie di passi).

Nel mondo del design invece, la progettazione è tipicamente affrontata in maniera globale.

L'obiettivo principale del design di prodotto non è necessariamente quello di trovare una soluzione al problema specifico ma è piuttosto quello di comprendere il problema nel suo insieme

Pensiero Computazionale vs Design

Nel mondo del design il primo passo è sempre quello di capire perchè il problema esiste e solo dopo aver appurato che l'origine di un problema non può essere eliminata o mitigata ci si adopera per cercare di risolverlo nello specifico.

Viceversa, l'informatico medio non appena si trova davanti ad un problema, apre il proprio editor di testo e inizia a scrivere un algoritmo per cercare di risolverlo senza neanche chiedersi se il problema che si sta affrontando esiste veramente

Il problema esiste veramente?

George Berkeley, un filosofo irlandese del '700 sosteneva che gli oggetti esistono solo in quanto percepiti. Dunque, se un albero cade in una foresta e nessuno lo sente, non fa rumore.

Se un problema non è percepito da un utente allora quel problema non esiste.

Nel design dell'interazione l'obiettivo primo è quindi quello di avere un utente soddisfatto non un software teoricamente perfetto or super-ricco di funzionalità.

L’utente è il vostro nuovo capo

Questo può portare a situazioni che dal punto di vista informatico sono percepite come assurde.

Nel design di prodotto ci si trova infatti spesso costretti a modificare i requirement e le specifiche di prodotto per andare in contro alle esigenze degli utenti e sacrificando funzionalità tecniche e qualità dell'implementazione software.

Trovare il corretto bilanciamento fra esperienza utente, funzionalità e qualità tecnica è la parte più complessa dell'intero processo di sviluppo prodotto.

L’utente è il vostro nuovo capo

Questo processo antropocentrico di adattamento del software alle esigenze dell'utente piuttosto che al virtuosismo tecnico è spesso vissuto dalla maggior parte degli informatici come un'assurda violenza.

Per questo motivo è molto importante che gli informatici studino i principi del design, perché il mondo del design antropocentrico è per i tecnici tipicamente molto complicato da metabolizzare in quanto distante dal pensiero computazionale.
E' importante evidenziare che design dell'interazione e pensiero computazionale non sono mutualmente esclusivi, anzi!
E' nell'unione dei due e nell'integrazione dei due processi di studio e progettazione che nascono prodotti di successo e software di qualità.

“If we want users to like our software, we should design it to behave like a likeable person: respectful, generous and helpful.”
— Alan Cooper, software designer and programmer (Widely recognized as the “Father of Visual Basic")

Interaction Design
Il mondo della progettazione è diventato talmente ampio e variegato che il termine design da solo ormai non ha quasi più significato. Esistono varie sotto discipline del design e con queste numerose professioni, metodi di lavoro, scuole di pensiero e altrettante immancabili faide e lotte fra fazioni.
Un informatico può fare a meno di conoscere al completo il mondo del design ma non può esimersi da possedere i rudimenti base del “design dell'interazione”.
Interaction design, o progettazione dell'interazione, è l'attività di progettazione dell'interazione che avviene tra esseri umani e oggetti in generale.

Interaction Design
L'obiettivo principale dell'interaction design è quello di rendere macchine, servizi e sistemi usabili dagli utenti per cui sono stati pensati e realizzati e non solamente dai propri creatori.
All'interno di un processo di interaction design, si investigano l'uso che verrà fatto dell'artefatto e il target a cui esso si rivolge. Questo significa che le questioni legate agli utenti guidano il processo più di quanto non facciano le questioni tecniche.
Gli sviluppatori devono mettere al centro del processo di sviluppo i bisogni degli utenti, arrivando a realizzare un prodotto più appropriato e maggiormente usabile.
Le forze trainanti lo sviluppo di un prodotto dovrebbero essere quindi gli utenti reali e i loro bisogni e non solo le tecnologie.

Human Machine e Human Computer Interaction
Nel nostro caso ci focalizzeremo sull'interazione tra uomo e sistemi informatici e

quindi andremo a lavorare nel campo dell'interazione uomo-macchina e uomo-computer (in Inglese HMI Human-Machine Interaction e HCI Human Computer Interaction).
L'obiettivo principale del HMI e HCI design è quindi rendere possibile e facilitare al massimo, per un essere umano, l'uso e l'interazione con sistemi del mondo IT (Information Technology) quali, calcolatori, dispositivi mobili, servizi web etc.

Interaction Design
Sotto discipline:
- design di prodotto
- design dell'esperienza utente (UX design)
- design dell'interfaccia (UI Design)

Interaction
Design
Product
Design

UI
Design

UX
Design

Design di Prodotto
E' un processo strategico di risoluzione dei problemi che guida l'innovazione e porta a una migliore qualità della vita attraverso prodotti, sistemi, servizi ed esperienze innovative (Definizione ufficiale di disegno industriale, coniata nel 2015 dalla World Design Organization).
NB: Spesso design di prodotto e design industriale sono utilizzati in modo intercambiabile
Nel design di prodotto si progettano quindi beni e servizi il cui obiettivo principale è quello di essere utilizzati da quanti più utenti possibili migliorandone la vita.
Il designer di prodotto è quindi colui che inventa un nuovo modo o un nuovo oggetto per fare cose che fino a ieri non si potevano fare o si facevano in maniera più complicata.

Il design di prodotto è quindi il punto di partenza dei processi di innovazione e rappresenta quindi il punto di partenza del percorso di design dell'interazione che stiamo esplorando.

L'inventore

L'inventore (designer) del prodotto si focalizza su un problema da risolvere e inventa un prodotto che grazie alle sue caratteristiche tecniche permette di risolvere il problema.

http://cherryhappygirl.blogspot.com/2017/05/
product-designer-vs-ux-designerwhich.html

Progettiamo Spotify
● Nome prodotto: Spotify
● Problema a cui assolve: Ascoltare musica senza possederla;
● Tipologia di prodotto: Servizio basato su Internet;
● Funzionalità Principale: Consentire l'ascolto di qualunque brano, su qualsiasi piattaforma senza richiedere l'acquisto di un supporto fisico e dei diritti d'autore;
● Soluzioni esistenti simili (competitor): Noleggio/prestito CD e altri supporti fisici, download di musica pirata;
● Soluzione: Servizio di streaming basato su business model "freemium" che di fatto rappresenta un jukebox con brani pressoché illimitati;
● Requisiti per l'utilizzo: PC o smartphone o tablet, Connessione internet.

E quindi?.....
E' chiaro però che questo nuovo prodotto potrebbe essere realizzato in tanti modi e che sarà proprio il modo in cui viene costruito a decretarne il successo o il fallimento del prodotto.
Questo perché non basta un'idea per fare un prodotto di successo, ci vuole un lungo e complicato processo di progettazione che vada oltre l'idea e metta al centro l'utente, i suoi bisogni e le sue aspettative.

User Experience Designer (UX Designer)
UX Designing is completely related to how the product feels and completely different from traditional graphic design.
UX design focuses on the logic and structure behind the elements you actually interact with on any web or mobile application or any software.
The UX designers through different approaches to solving the user-specific problem and completely user-centric.
The UX designers use a variety of tools and methods to understand information architecture, human factors and end users by competitive analysis and user interviews by building user persona, wireframe the product for UI design and the flow of the application.

User Experience Designer (UX Design)
Il design dell'esperienza utente è il processo volto ad aumentare la soddisfazione e la fedeltà del cliente migliorando l'usabilità, la facilità d'uso e il piacere fornito nell'interazione tra il cliente e il prodotto.
La progettazione dell'esperienza utente comprende la tradizionale progettazione dell'interazione uomo-macchina e la estende integrandola con tutti gli aspetti di business, marketing e sviluppo prodotto necessari per garantire il successo del prodotto e/o servizio.

UX Designer
Lo UX Designer ha quindi l'obiettivo di far vivere all'utente del suo prodotto la miglior esperienza possibile, evitando quindi che l'oggetto induca nell'utente sensazioni di frustrazione e delusione.
Spesso si tende a dire che si “progetta l'esperienza utente”. In realtà è impossibile progettare l'esperienza utente perchè ogni utente è diverso dall'altro ed è quindi illusorio pensare che chiunque durante l'utilizzo del prodotto si comporti alla stessa maniera e in particolare si comporti esattamente come il progettista ha ipotizzato.

User Experience

“every product that is used by someone has a user experience: newspapers, ketchup bottles, reclining armchairs, cardigan sweaters.” (Garrett, 2003)

You can’t design a user experience, only design for a user experience

UX Design
Come vedremo nei capitoli successivi, lo studio e la progettazione dell'esperienza utente partono sempre dalla analisi e comprensione delle esigenze dell'utente e non dalle specifiche funzionali di prodotto.

User Interface designer (UI design)
Il Design dell'interazione ha come focus il modo in cui le persone interagiscono con la tecnologia, lo scopo è migliorare la loro comprensione di ciò che si può fare, ciò che succede e ciò che è appena successo, basandosi su principi psicologici, tecnici ed estetici.
Dallo studio della UX si crea quindi uno schema di

interazione che poi viene passato allo UI Designer
che ha il compito di progettare l'interfaccia utente al
fine di abilitare l'esperienza progettata.

User Interface designer (UI designer)
Lo UI designer non costruisce quindi l'interfaccia utente, nei team numerosi questa figura è spesso un progettista grafico e non un informatico.
L'obiettivo dello UI designer è quello di progettare l'aspetto estetico e la struttura dell'interfaccia così che questa durante l'utilizzo induca l'utente nel seguire l'esperienza che è stata per lui progettata.
Lo UI designer produce quindi un wireframe, una bozza grafica, dell'interfaccia e una serie di linee guida che poi verranno seguite dagli sviluppatori (UI developer o Front-end developer) per implementare la reale interfaccia del prodotto o servizio

User Interface designer (UI designer)
UI Designer is in charge of giving the art of feeling the project to end-client.
He/she need to think in shapes, surfaces, shading, consistency, visual appearance of the project in view of the UX provided.
He/she need to have the capacity to clear the questions of Front-end (UI) developer on the off chance that he/she incorporates any new components available in the market and ought to have the capacity to furnish the developer with its asset or documentation.
His/her duty is likewise to make design guidelines in view of which encourage developers or visual planners will work.

immagine tratta da
https://blog.prototypr.io/why-you-shouldnt-skip-your-wireframing-1f7a70d5c125

User Interface design vs User Experience design

source: https://nzdigital.co.nz/ui-and-ux/

Front-end Developer (UI Developer)
They ought to have the creation expertise to have the capacity to transform the UI and UX configuration given into the HTML, CSS, JavaScript,... [other languages] that arrangements with the marvels of program compatibilities.
This requires profound information of how modern rendering framework works and ought to have the capacity to execute the UI Design into the functional product.
The real assignment of Front-end engineer is to furnish the last outcome with all connection streams portrayed in the UI Design by UI Designer.

Human Centered Design

Human Centered Design
I progettisti devono produrre cose che soddisfino i bisogni della gente, in termini di funzioni, facilità d'uso e gratificazione emotiva. In altre parole, il design (di prodotto) deve essere pensato come un'esperienza totale (per l'utente).

[Donald Norman — La caffettiera del masochista]

Human Centered Design
Le persone sono sempre più frustrate dalla complessità degli oggetti quotidiani. Dalla complessità sempre maggiore del cruscotto dell'auto, dalla lavatrice piena di incomprensibili funzioni e pulsanti, dalla crescente automazione della casa, e dalla continua proliferazione di funzioni che i progettisti aggiungono con orgoglio ad ogni nuova versione dei loro prodotti.
La vita della maggior parte degli utenti è ormai diventata una battaglia quotidiana per la sopravvivenza alla invadente e iper-funzionale tecnologia.

Human Centered Design
Questo problema origina direttamente dalla modalità con la quale vengono oggigiorno progettati gli oggetti quotidiani ed in particolare quelli tecnologici.
Le macchine (computer) hanno una modalità di funzionamento logica, dovuta all'algoritmo che il progettista ha sviluppato come anima della macchina. Noi umani invece siamo tutt'altro che logici e razionali, siamo intuitivi, flessibili, versatili e curiosi.
E' chiaro quindi che nell'interazione uomo-macchina si va a creare una relazione fra specie diverse che hanno modalità di pensiero e di funzionamento opposte.

Human Centered Design
Gli ingegneri e gli informatici, orgogliosi dei loro progressi tecnologici, hanno preteso da sempre che gli umani si adattassero alle loro macchine.
Le macchine sono viste dai progettisti come un elemento di orgoglio che rappresenta il progresso e chi non è in grado di capirle è retrogrado, vecchio e a volte anche un po' stupido.
Questo approccio tecno-centrico dei progettisti ha in realtà leso lo sviluppo stesso della tecnologia dal momento che ne ha rallentato la sua diffusione e accettazione.
La maggior parte degli utenti oggi è frustrata dall'utilizzo di incomprensibili oggetti

tecnologici di cui non capisce il principio di funzionamento e dove tipicamente si limita ad utilizzare il 10% delle funzionalità disponibili.

Human Centered Design

Le macchine hanno delle loro regole di funzionamento che sono spesso note solo ai progettisti. Quando non si seguono queste regole le cose non vanno come previsto e l'utente si sente stupido ed incapace. La macchina è perfetta, non può sbagliare, quindi se le cose sono andate male è sicuramente colpa dell'umano.
E' vero! ma non è l'umano utente ad aver sbagliato, la colpa è del progettista!
Nel design antropocentrico si inverte il paradigma di progettazione mettendo l'utente al centro del processo.
Le funzionalità del prodotto vengono dopo. Prima ci sono i bisogni dell'utente!

Human Centered Design

Human-centred design is an approach to interactive systems development that aims to make systems usable and useful by focusing on the users, their needs and requirements, and by applying human factors/ergonomics, usability knowledge, and techniques.
This approach enhances effectiveness and efficiency, improves human well-being, user satisfaction, accessibility and sustainability; and counteracts possible adverse effects of use on human health, safety and performance.
ISO 9241-210:2010(E)

Human Centered Design

Questo processo, apparentemente ovvio e banale, risulta in realtà estremamente difficile da applicare per gli informatici.
I tecnici infatti amano le funzioni e le funzionalità, amano le peculiarità tecniche dei sistemi e sono spesso spinti a sviluppare nuove soluzioni non tanto per risolvere il problema ma piuttosto per la soddisfazione personale di aver implementato qualcosa che prima non esisteva.

Esempio di sviluppo tecno-centrico

La blockchain, creata per diletto da degli appassionati di crittografia, ha dato vita alla prima criptomoneta della storia.
Dopo il boom di Bitcoin e delle altre cryptomonete è scoppiata la bolla blockchain dove tutti nel mondo IT hanno iniziato a dichiarare che grazie alla blockchain si sarebbe potuto innovare in maniera radicale tantissimi settori.
Ad oggi in realtà non si è ancora trovata per la blockchain un’applicazione di successo alternativa alle cryptomonete e che non fosse in precedenza comunque realizzabile.

Esempio di sviluppo tecno-centrico

Per dirla in altre parole, nessun utente ci ha chiesto di sviluppare la blockchain in quanto tale, c'era bisogno di scambiarsi denaro in maniera alternativa e quindi sono nate le cryptomonete e la blockchain è nata come tecnologia per abilitarle. Ora la corsa a cercare di applicare la blockchain ad altri settori non sta funzionando perché stiamo cercando un problema per una tecnologia e non una tecnologia per risolvere un problema!

Sviluppo Antropocentrico

La morale di questo ragionamento è molto semplice: se vogliamo progettare tecnologia per le persone dobbiamo capire sia la tecnologia che le persone. Dobbiamo smettere di progettare per le persone come vorremmo che fossero e iniziare a progettare per come realmente sono!

Human Centered Design

Human-centered design is a design philosophy.
It means starting with a good understanding of people and the needs that the design is intended to meet.
This understanding comes about primarily through observation, for people themselves are often unaware of their true needs, even unaware of the difficulties they are encountering.

Human Centered Design

human-centered design (HCD), an approach that puts human needs, capabilities, and behavior first, then designs to accommodate those needs, capabilities, and ways of behaving.
Good design requires good communication, especially from machine to person, indicating what actions are possible, what is happening, and what is about to happen.
Communication is especially important when things go wrong. It is relatively easy to design things that work smoothly and harmoniously as long as things go right. But as soon as there is a problem or a misunderstanding, the problems arise.

Human Centered Design

Designers need to focus their attention on the cases where things go wrong, not just on when things work as planned.
Actually, this is where the most satisfaction can arise: when something goes

wrong but the machine highlights the problems, then the person understands the issue, takes the proper actions, and the problem is solved.
When this happens smoothly, the collaboration of person and device feels wonderful.

Viva i Bug! Viva i crash!
Non bisogna aver paura che l'utente abbia problemi o che il nostro software abbia degli errori o bug, è inevitabile che questo accada.
E' importante quindi progettare perché l'utente venga guidato nella risoluzione e gestione dell'errore senza provare frustrazione. Avremo così un utente soddisfatto.

Le emozioni degli utenti
L'esperienza di utilizzo produce emozioni negli utenti, più emozioni positive (successi) l'utente avrà e migliore sarà la percezione che avrà del nostro prodotto.
E' importante sottolineare inoltre che i ricordi hanno la capacità di far provare emozioni più profonde rispetto al presente.
Un utente che di fronte ad un problema riesce a risolverlo perché ben guidato dalla tecnologia avrà memoria di un suo successo. Questo tipo di sensazioni sono molto forti e se associate al prodotto fanno si che l'utente sviluppi empatia per il prodotto e che quindi lo apprezzi e ne senta il bisogno.
L'obiettivo dello HCD deve essere quindi quello di creare nell'utente empatia verso il sistema.

Human Centered Design
Getting the specification of the thing to be defined is one of the most difficult parts of the design, so much so that the HCD principle is to avoid specifying the problem as long as possible but instead to iterate upon repeated approximations.
This is done through rapid tests of ideas, and after each test modifying the approach and the problem definition.

Human Centered Design
L'HCD è quindi una forma di pensiero ed è quindi compatibile con le varie discipline del design di prodotto che abbiamo precedentemente introdotto
Si può infatti applicare il pensiero HCD sia al design industriale che alla progettazione dell'interazione o dell'esperienza utente, lo HCD non è un'area o un metodo, è una forma di pensiero.

Human Centered Design

Human-centered design is a creative approach to problem solving [...]. It’s a process that starts with the people you’re designing for and ends with new solutions that are tailor made to suit their needs. Human-centered design is all about building a deep empathy with the people you’re designing for.

— fonte: www.designkit.org/human-centered-design

D. Norman HCD principles

https://www.youtube.com/watch?v=rmM0kRf8Dbk&ab_channel=NNgroup

Human Centered Design

A sottolineare quanto l'HCD sia ritenuto oggigiorno fondamentale per la progettazione di sistemi destinati all'utilizzo umano, è importante ricordare che il design antropocentrico è ormai parte della norma ISO EU.

ISO 9241-210:2019 Ergonomics of human-system interaction Part 210: Human-centred design for interactive systems

This document provides requirements and recommendations for human-centred design principles and activities throughout the life cycle of computer-based interactive systems. It is intended to be used by those managing design processes, and is concerned with ways in which both hardware and software components of interactive systems can enhance human-system interaction.

Human Centered Design

Un buon processo HCD parte sempre dall'osservazione dell'utente e dei suoi bisogni.

Tuttavia, tale osservazione non è sempre possibile o facile da attuare.

Analizzeremo varie tecniche di prototipazione rapida e metodi di lavoro finalizzati all'estrazione veloce di bisogni utente e all'esecuzione rapida e a basso costo di test.

Possiamo schematizzare un processo di HCD come un flusso continuo ed iterativo che attraversa le seguenti fasi:

- Specificare il contesto d'uso: identificare gli utenti che utilizzeranno il prodotto, per cosa lo utilizzeranno e sotto quale condizioni e vincoli;
- Specificare i Requirements: Identificare i business requirement e gli obiettivi utente che devono essere

raggiunti grazie all'utilizzo del software;
● Progettare la soluzione: questa fase può essere a sua volta spacchettata in sotto fasi iterative. Si passa tipicamente da delle bozze a dei prototipi e poi alla soluzione;
● Testare e valutare: è fondamentale testare e quindi valutare il sistema così da poter iniziare il ciclo sulla base dei risultati dei test e quindi procedere ad uno sviluppo e miglioramento incrementale.
https://www.usability.gov/what-and-why/user-centered-design.html

Human Centered Design
Per ora, al fine di inquadrare meglio la filosofia HCD nel contesto dello sviluppo software vi basti pensare che le versioni alfa e beta dei nostri software possono diventare potenti strumenti di analisi degli utenti. Le versioni di test non servono quindi solamente a fare debugging del codice e delle funzioni, ma servono anche e soprattutto a capire che cosa fanno e come si comportano gli utenti durante l'utilizzo del nostro software.

Nello sviluppo software diventa quindi indispensabile abilitare dei sistemi di tracking dell'utente finalizzati alla produzione di statistiche di utilizzo.

Progettare l’interazione

Progettare l’interazione

Esistono solo due tipi di design, riuscito e fallito; buono e cattivo (D. Norman).
Il problema è che il buon design non è universale. Un progetto, prodotto o sistema apprezzato da tutti non esiste perché l'esperienza di interazione è soggettiva e quindi dipende più dalla persona che non dell'artefatto e di conseguenza è statisticamente impossibile progettare qualcosa che sia apprezzato da chiunque.
Two of the most important characteristics of good design are discoverability and understanding.

Discoverability
è la capacità di un sistema di veicolare e comunicare i propri possibili usi all'utente.
Un sistema che a prima vista fa capire all'utente a cosa serve e che cosa ci si può fare ha una buona Discoverability
Per avere una buona discoverability si usa

tipicamente la visibilità
Non è detto però che una volta capito cosa si può
fare si riesca a farlo.

Discoverability
è la capacità di un sistema di veicolare e
comunicare i propri possibili usi all'utente.
Un sistema che a prima vista fa capire all'utente a
cosa serve e che cosa ci si può fare ha una buona
Discoverability
Per avere una buona discoverability si usa
tipicamente la visibilità
Non è detto però che una volta capito cosa si può
fare si riesca a farlo.

Discoverability

DISCOVERABILITY

Understanding
è invece la capacità del prodotto di farsi usare correttamente dall'utente.
Se la discoverability è la misura di quanto bene si capisce cosa si può fare con il prodotto, la understanding invece è la proprietà associata a quanto bene un prodotto dice come si usano le funzioni disponibili.
Per capire come si usa un prodotto non basta infatti
aver identificato quali sono i controlli, è necessario
dare con facilità risposta alle seguenti domande:
● Come si usa il prodotto?
● Che funzione ha ciascun controllo?
● Come si combinano i controlli?

Understanding

UNDERSTANDING

Design of useful things
Quando le cose vanno bene, si dimenticano subito, quando vanno male non
si dimenticano mai!

Design of useful things
Design is concerned with how things work, how they are controlled, and the nature of the interaction between people and technology.
When done well, the results are brilliant, pleasurable products. When done badly, the products are unusable, leading to great frustration and irritation.

Design of useful things
Machines, are conceived, designed, and constructed by people.
By human standards, machines are pretty limited. Instead, machines usually follow rather simple, rigid rules of behavior. If we get the rules wrong even slightly, the machine does what it is told, no matter how insensible and illogical.
People are imaginative and creative, filled with common sense; that is, a lot of valuable knowledge built up over years of experience. But instead of capitalizing on these strengths, machines require us to be precise and accurate, things we are not very good at.
Machines have no leeway or common sense. Moreover, many of the rules followed by a machine are known only by the machine and its designers.

Design of useful things
It is time to reverse the situation: to cast the blame upon the machines and their design. It is the machine and its design that are at fault.
It is the duty of machines and those who design them to understand people.
It is not our duty to understand the arbitrary, meaningless dictates of machines.
The reasons for the deficiencies in human-machine interaction are numerous. But most of the problems come from a complete lack of understanding of the design principles necessary for effective human-machine interaction.
Why this deficiency? Because much of the design is done by developers who are experts in technology but limited in their understanding of people.

Design of useful things
“We are people ourselves,” they think, “so we understand people.”
But in fact, we humans are amazingly complex.
Developers typically make the mistake of thinking that logical explanation is sufficient: “If only people would read the instructions, everything would be all right.”
Engineers are trained to think logically. As a result, they come to believe that all

people must think this way, and they design their machines accordingly.

Design of useful things
The problem with the designs of most engineers is that they are too logical.
We have to accept human behavior the way it is, not the way we would wish it to be.

Design of useful things
Donald Norman: “we were designing things for people, so we needed to understand both technology and people. But that’s a difficult step for many engineers: machines are so logical, so orderly. If we didn’t have people, everything would work so much better. Yup, that’s how I used to think.”

Three Mile Island accident
Critical user interface engineering problems were revealed in the investigation of the reactor control system's user interface. Despite the
valve being stuck open, a light on the control panel ostensibly indicated that the valve was closed. In fact the light did not indicate the
position of the valve, only the status of the solenoid being powered or not, thus giving false evidence of a closed valve. As a result, the
operators did not correctly diagnose the problem for several hours

Fundamental principles of interaction

Life is made of experiences
Great designers produce pleasurable experiences. Engineers tend not to like it; it is too subjective. But...
“when I ask them about their favorite automobile or test equipment, they will smile delightedly as they discuss the fit and finish, the sensation of power during acceleration, their ease of control while shifting or steering, or the wonderful feel of the knobs and switches on the instrument.” Those are experiences! (D. Norman)
Experience is critical, for it determines how fondly people remember their interactions. Was the overall experience positive, or was it frustrating and confusing?

Donald Norman video course playlist
https://www.youtube.com/playlist?list=PLAwxTw4SYaPlr4Uq3RoYuwlDADp0WQdGl

Cognition and emotion
When our home technology behaves in an uninterpretable fashion we can become confused, frustrated, and even angry—all strong negative emotions.
When there is understanding it can lead to a feeling of control, of mastery, and of satisfaction or even pride — all strong positive emotions.
Cognition and emotion are tightly intertwined, which means that the designers must design with both in mind.
https://www.youtube.com/watch?v=LTE-v4RzRHs&list=PLJOFJ3Ok_idv_6hQGNt23xVXuQwKFmfxG&index=8&ab_channel=NNgroup

Discoverability
When we interact with a product, we need to figure out how to work it. This means discovering what it does, how it works, and what operations are possible.
Discoverability results from appropriate application of 6 fundamental psychological principles:
● affordances
● signifiers
● constraints
● mappings
● feedback
● conceptual model of the system

AFFORDANCES

Affordances

The term affordance refers to the relationship between a physical object and a person.
An affordance is a relationship between the properties of an object and the capabilities of the agent that determine just how the object could possibly be used.
A chair affords (“is for”) support and, therefore, affords sitting.

Affordances are relationships
We are used to thinking that properties are associated with objects.
But affordance is not a property.
An affordance is a relationship.
Whether an affordance exists depends upon the properties of both the object and the agent.

Affordances are relationships

Most chairs can also be carried by a single person (they afford lifting), but some can only be lifted by a strong person.
If young or relatively weak people cannot lift a chair, then for these people, the chair does not have that affordance, it does not afford lifting.

https://www.youtube.com/watch?v=NK1Zb_5VxuM&ab_channel=Interaction-Design.org

Anti-affordance
anti-affordance -> the prevention of interaction.

Affordances
To be effective, affordances and anti-affordances have to be: discoverable and perceivable.
This poses a difficulty with glass. The reason we like glass is its relative invisibility, but this aspect, so useful in the normal window, also hides its anti-affordance property of blocking passage.
As a result, birds often try to fly through windows.

SIGNIFIERS

Signifiers
Designers have practical problems.
They need to know how to design things to make them understandable.
They soon discovered that when working with the graphical designs for electronic displays, they needed a way to designate which parts could be touched, slided upward, downward, or sideways, or tapped upon.

Signifiers
How could designers describe what they were doing?
There was no word that fit, so they took the closest existing word: affordance.
“I put an affordance there” to describe why they displayed a circle on a screen to indicate where the person should touch, whether by mouse or by finger.

NO -> that is not an affordance! That is a way of communicating where the touch should be.

You are communicating where to do the touching!

Signifiers

The affordance of touching exists on the entire screen:
That's not the same thing as saying what action is possible.

Affordances determine what actions are possible.
Signifiers communicate where the action should take place.

Signifiers

The affordance of touching exists on the entire screen:
That's not the same thing as saying what action is possible.

Affordances determine what actions are possible.
Signifiers communicate where the action should take place.

Affordance

Signifier

Signifiers
Signifiers can be deliberate and intentional, such as the sign push on a door they may also be accidental and unintentional, such as our use of the visible trail made by previous people walking through a field or over a snow-covered terrain to determine the best path.

Signifiers
Affordances represent the possibilities in the world for how an agent can interact with something. Some affordances are perceivable, others are invisible.
Signifiers are signals. Some signifiers are signs, labels, and drawings placed in the world
Some signifiers are simply the perceived affordances, such as the handle of a door

Note that some perceived affordances may not be real

Signifiers
In design, signifiers are more important than affordances, for they communicate how to use the design

Perceiving affordances and signifiers

But how does one go from the perception of an affordance to understanding the potential action?
In many cases, through conventions.
A doorknob has the perceived affordance of graspability. But knowing that it is the doorknob that is used to open and close doors is learned: it is a cultural aspect of the design that knobs, handles, and bars, when placed on doors, are intended to enable the opening and shutting of those doors. The same devices on fixed walls would have a different interpretation: they might offer support, for example, but certainly not the possibility of opening the wall.
The interpretation of a perceived affordance is a cultural convention.

https://www.youtube.com/watch?v=UtuITXJLGOI&ab_channel=LeahGreis

MAPPING

Mapping
Mapping means the relationship between the elements of two sets of things.
Mapping is an important concept in the design and layout of controls and displays.
When the mapping uses spatial correspondence between the layout of the controls and the devices being controlled, it is easy to determine how to use them.
Natural mapping (taking advantage of spatial analogies) leads to immediate understanding.

Mapping
Some natural mappings are cultural or biological, as in the universal standard that moving the hand up signifies more, moving it down signifies less, which is why it is appropriate to use vertical position to represent intensity or amount.
Note that there are many mappings that feel “natural” but in fact are specific to a

particular culture.

FEEDBACK

Feedback

Ever watch people at an elevator repeatedly push the Up button, or repeatedly push the pedestrian button at a street crossing?
What is missing in these cases is feedback: some way of letting you know that the system is working on your request.
Feedback: communicating the results of an action

Feedback
Feedback must be immediate
even a delay of a tenth of a second can be disconcerting. If the delay is too long, people often give up, going off to do other activities.
Feedback must also be informative.
Poor feedback can be worse than no feedback at all, because it is distracting, uninformative, and in many cases irritating and anxiety-provoking.
“My dishwasher likes to beep at three a.m. to tell me that the wash is done, defeating my goal of having it work in the middle of the night so as not to disturb anyone”

Feedback
Too many announcements cause people to ignore all of them, or wherever possible, disable all of them, which means that critical and important ones are apt to be missed.
Feedback is essential, but not when it gets in the way of other things, including a calm and relaxing environment.
Notifications become useless!

CONCEPTUAL MODEL

Conceptual Model

A conceptual model is an explanation, usually highly simplified, of how something works. It doesn't have to be complete or even accurate as long as it is useful.

The files, folders, and icons you see displayed on a computer screen help people create the conceptual model of documents and folders inside the computer, or of apps or applications residing on the screen, waiting to be summoned.

In fact, there are no folders inside the computer, those are effective conceptualizations designed to make
them easier to use.

Conceptual Model

Simplified models are valuable only as long as the assumptions that support them hold true.

Cloud Storage Sync: files appear to be on the device. But in fact, in many cases the actual material is "in the cloud".

The conceptual model is of one coherent storage available on all the user's devices.

This simplified model is helpful for normal usage, but if the network connection to the cloud services is interrupted, the result can be confusing.

Files are still shown on users device screen, but users can no longer open save it

Mental Model

Mental models, are the conceptual models in people's minds that represent their understanding of how things work.

Different people may hold different mental models of the same item.

Indeed, a single person might have multiple models of the same item, each dealing with a different aspect of its operation: the models can even be in conflict.

Conceptual Model

Conceptual models are often inferred from the device itself.

Some models are passed on from person to person. Some come from manuals.

Usually the device itself offers very little assistance, so the model is constructed by experience. Quite often these models are erroneous, and therefore lead to difficulties in using the device.

https://www.slideshare.net/cxpartners
/psychology-for-designers-or-3-predic
tions-from-psychology-for-the-future-o

f-web-design-by-mrjoe/27-mrjoe5101
5_2025mental_model30This_is_the

Conceptual Model

Conceptual Model

SYSTEM IMAGE

System Image
People create mental models of themselves, others, the environment, and the things with which they interact.
These are conceptual models formed through experience, training, and instruction.
These models serve as guides to help achieve our goals and in understanding the world.

System Image

How do we form an appropriate conceptual model for the devices we interact with?

We cannot talk to the designer, so we rely upon whatever information is available to us: what the device looks like, what we know from using similar things in the past, what was told to us in the sales literature, by salespeople and advertisements, by articles we may have read, by the product website and instruction manuals.

Combined information available to us is the system image.

System Image
When the system image is incoherent or inappropriate, then the user cannot easily use the device. If it is incomplete or contradictory, there will be trouble.
The designer’s conceptual model is the designer’s conception of the product, occupying one vertex of the

triangle. After sale the product itself is no longer with the designer, so it is isolated as a second vertex, perhaps on the user's kitchen or wall.
The system image is what can be perceived from the physical structure that has been built

https://www.youtube.com/watch?v=shSCUNxtn18&ab_channel=LeahGreis

Example: System image of a smart Thermostat
Vimar Wifi
https://www.vimar.com/it/it/cronotermostato-touch-screen-wi-fi-da-parete-02911-video-guida-a-11721152.html
NEST https://www.youtube.com/watch?v=dHKD-9uI24I

:) https://www.youtube.com/watch?v=5yJOyKr4GBA

Changing Conventions
People invariably object and complain whenever a new approach is introduced into an existing array of products and systems.
Conventions are violated -> new learning is required.
The merits of the new system are irrelevant: it is the change that is upsetting.
Consistency in design is virtuous.
If a new way of doing things is only slightly better than the old, it is better to be consistent. But if there is to be a change, everybody has to change.
NB: Mixed systems are confusing to everyone!

Rethinking OS
https://uxdesign.cc/introducing-mercury-os-f4de45a04289

CONSTRAINTS,
DISCOVERABILITY

CONSTRAINTS

Constraints
How do we determine how to operate something that we have never seen before?
We have no choice but to combine knowledge in the world with that in the

head.
Knowledge in the world includes perceived affordances and signifiers, the mappings between the parts that appear to be controls or places to manipulate and the resulting actions, and the physical constraints that limit what can be done. Knowledge in the head includes conceptual models; cultural, semantic, and logical constraints on behavior; and analogies between the current situation and previous experiences with other situations.

Constraints
The sizes and shapes of the parts suggested their operation. Physical constraints limited what parts would fit together.
Cultural and semantic constraints provided strong restrictions on what would make sense for all but one of the remaining pieces, and with just one piece left and only one place it could possibly go, simple logic...

Constraints

4 classes of constraints seem to be universal,
appearing in a wide variety of situations:
● physical
● cultural
● semantic
● logical
Constraints are powerful clues, limiting the set of
possible actions.
The thoughtful use of constraints in design lets people readily
determine the proper course of action, even in a novel situation.

Lack of constraints and mapping

The lack of clear communication among the
people and organizations constructing parts
of a system is perhaps the most common
cause of complicated, confusing designs.
A usable design starts with careful
observations of how the tasks being
supported are actually performed, followed by
a design process that results in a good fit to
the actual ways the tasks get performed.

Constraints That Force the Desired

Behavior

FORCING FUNCTIONS
Forcing functions are a form of physical constraint: situations in which the actions are constrained so that failure at one stage prevents the next step from happening.
Forcing functions are the extreme case of strong constraints that can prevent inappropriate behavior.
Not every situation allows such strong constraints to operate, but the general principle can be extended to a wide variety of situations.

FORCING FUNCTIONS: INTERLOCKS

An interlock forces operations to take place in proper sequence.

FORCING FUNCTIONS: LOCK-INS
A lock-in keeps an operation active, preventing someone from prematurely stopping it.
Standard lock-ins exist on many computer applications, where any attempt to exit the application without saving work is prevented by a message prompt asking whether that is what is really wanted
These are so effective that people uses them
deliberately as standard way of exiting.
Rather than saving a file and then exiting the
program

FORCING FUNCTIONS: Lockout
Whereas a lock-in keeps someone in a space or prevents an action until the desired operations have been done, a lockout prevents someone from entering a space that is dangerous, or prevents an event from occurring.

ACTIVITY-CENTERED CONTROLS

Activity-centered design (ACD) is an extension of the Human-centered design paradigm in interaction design.
ACD features heavier emphasis on the activities that a user would perform with a given piece of technology.
ACD has its theoretical underpinnings in activity theory, from which activities can be defined as actions taken by a user to achieve a goal.

It's important to note that ACD is a model, not a process. ACD is just one of many perspectives you can employ when designing.

Activity-Centered Controls
Spatial mapping [of switches] is not always appropriate.
In many cases it is better to have [switches] that control activities:
activity-centered control.
Many auditoriums in schools and companies have computer-based controls, with switches labeled with such phrases as "video," "computer," "full lights," and "lecture."

Activity-Centered Controls

Activity-based controls are excellent in theory, but the practice is difficult to get right. When it is done badly, it creates difficulties.
Activity-Centered Controls must be User-Activity-centered
A related but wrong approach is to be device-centered rather than user-activity-centered.
When they are device-centered the user would need to know the technical model behind the system!

https://www.myharmony.com/en-en/

"To program the Harmony, I simply went to their website, selected the brand and model number of all the equipment I owned (and yes, they had every item), and then connected my remote to the computer via the convenient USB cord. In a matter of minutes, my remote was programmed."

https://jnd.org/activity-centered_design_why_i_like_my_harmony_remote_control/

I still use the specialized remotes, because specialization always beats general purpose devices. But I use the Harmony to set up, to change activities, and at the end, to turn off the equipment. Once in an activity, however, then I usually prefer to

use the specialized controller, with its joystick or wheel, for quite often the physical controls of the specialized remote are superior to the general purpose ones of the harmony. But getting to that point is where the difficulty arises, and this is the problem the Harmony solves.
So, I use the harmony to select the activity and setup all the equipment to the proper state. Then I get the one remote specialized for the device -- TiVo, Satellite receiver, or DVD player. And the remote for the lights. And then I am truly happy. When finished, I pick up the harmony and one button push turns off all the equipment (see note).

Human Error and Mitigation

Strategies

Human Error
Most industrial accidents are caused by human error: estimates range between 75 and 95 percent.
How is it that so many people are so incompetent?
Answer: They aren't. It's a design problem.

Why?
We design equipment that requires people to be fully alert and attentive for hours, or to remember archaic, confusing procedures even if they are only used infrequently, sometimes only once in a lifetime.
We put people in boring environments with nothing to do for hours on end, until suddenly they must respond quickly and accurately.
Or we subject them to complex, high-workload environments, where they are continually interrupted while having to do multiple tasks simultaneously.

Why?
Interruptions are a common reason for error, not helped by designs and procedures that assume full, dedicated attention yet that do not make it easy to resume operations after an interruption.

Human attitude towards errors
"We caught the culprit."
But it doesn't cure the problem: the same error will occur over and over again.
Instead, when an error happens, we should determine why, then redesign the product or the procedures being followed so that it will never occur again or, if it does, so that it will have minimal impact.

ROOT CAUSE ANALYSIS
investigate the accident until the single, underlying cause is found.
What this ought to mean is that when people have indeed made erroneous decisions or actions, we should determine what caused them to err.
This is what root cause analysis ought to be about. Alas, all too often it stops once a person is found to have acted inappropriately.
most accidents do not have a single cause:
there are usually multiple things that went wrong
This is what James Reason has called the
“Swiss cheese model of accidents”

THE FIVE WHYS
originally developed by Sakichi Toyoda and used by the Toyota Motor Company as part of the Toyota Production System for improving quality.
Today it is widely deployed. Basically, it means that when searching for the reason, even after you have found one, do not stop: ask why that was the case.
And then ask why again. Keep asking until you have uncovered the true underlying causes.
Does it take exactly five? No, but calling the procedure “Five Whys” emphasizes the need to keep going even after a reason has been found.

- IUM 22-23 (all rights reserved)

People attitude toward errors
We can’t fix problems unless people admit they exist.
When we blame people, it is then difficult to convince organizations to restructure the design to eliminate these problems.

Why do people err?
Because the designs focus upon the requirements of the system and the machines, and not upon the requirements of people.
Most machines require precise commands and guidance, forcing people to enter numerical information perfectly.
But people aren’t very good at great precision.
People are creative, constructive, exploratory beings. We are particularly good at novelty, at creating new ways of doing things, and at seeing new opportunities.

Dull, repetitive, precise requirements fight against these traits.

DEFINITIONS: ERRORS
Human error is defined as any deviance from “appropriate” behavior.
The word appropriate is in quotes because in many circumstances, the appropriate behavior is not known or is only determined after the fact. B t still, error is defined as deviance from the generally accepted correct or appropriate behavior.
Error is the general term for all wrong actions.

Slips and Mistakes

There are two major classes of error:
slips (lapsus)
mistakes (errori cognitivi)

SLIPS (lapsus)
A slip occurs when a person intends to do one action and ends up doing something else.
With a slip, the action performed is not the same as the action that was intended.
There are two major classes of slips: action-based and memory-lapse.
In action-based slips, the wrong action is performed.
In memory lapses, memory fails, so the intended action is not done or its results not evaluated.

Slips (lapsus)
Example of an action-based slip. I poured some milk into my coffee and then put the coffee cup into the refrigerator. This is the correct action applied to the wrong object.
Example of a memory-lapse slip. I forget to turn off the gas burner on my stove after cooking dinner.

Mistakes (cognitive errors)
A mistake occurs when the wrong goal is established or the wrong plan is formed. From that point on, even if the actions are executed properly they are part of the error, because the actions themselves are inappropriate; they are part of the wrong plan.
With a mistake, the action that is performed

matches the plan: it is the plan that is wrong

Mistakes (cognitive errors)
In a rule-based mistake, the person has appropriately diagnosed the situation, but then decided upon an erroneous course of action: the wrong rule is being followed.
In a knowledge-based mistake, the problem is misdiagnosed because of erroneous or incomplete knowledge.
Memory-lapse mistakes take place when there is forgetting at the stages of goals, plans, or evaluation. (dimenticanza)

Mistakes (cognitive errors)
Example of rule-based mistakes. A mechanic diagnosed a defect on a car battery but decided to do not replace the battery because still working at 50% of performances
Example of knowledge-based mistake. Weight of fuel was computed in pounds instead of kilograms.
Example of memory-lapse mistake. A mechanic failed to complete troubleshooting because of forget a step.

Action Slips

knowledge based and rule
based Mistakes

Mistakes
|
X
|

Memory lapses

|
X
Slips

Slips and Mistakes
● novices are more likely to make mistakes than slips
● experts are more likely to make slips and mistakes-memory lapses (dimenticanza).

Mistakes often arise from ambiguous or unclear information about the current state of a system, the lack of a good conceptual model, and inappropriate procedures.
most mistakes result from erroneous choice of goal or plan or erroneous evaluation and interpretation.
All of these come about through poor information provided by the system about the choice of goals and the means to accomplish them (plans), and poor-quality feedback about what has actually happened.

Interruptions
A major source of error, especially memory-lapse errors, is interruption.
When an activity is interrupted by some other event, the cost of the interruption is far greater than the loss of the time required to deal with the interruption: it is also the cost of resuming the interrupted activity.
To resume, it is necessary to remember precisely the previous state of the activity: what the goal was, where one was in the action cycle, and the relevant state of the system.
Most systems make it difficult to resume after an interruption.

wrong feedbacks
Unnecessary, annoying alarms occur in numerous situations. How do people cope? By disconnecting warning signals, silencing bells etc
The problem comes after such alarms are disabled, either when people forget to restore the warning systems (there are those memory-lapse slips again), or if a different incident happens while the alarms are disconnected.
At that point, nobody notices.
Warnings and safety methods must be used with care and intelligence, taking into account the tradeoffs for the people who are affected.
The design of warning signals is surprisingly complex.

Speech as feedback
More and more of our machines present information through speech. But like all approaches, this has both strengths and weaknesses.
It allows for precise information to be conveyed, especially when the person’s visual attention is directed elsewhere.
But if several speech warnings operate at the same time, or if the environment is noisy, speech warnings may not be understood. Or if conversations among the users or operators are necessary, speech warnings will interfere.
Speech warning signals can be effective, but only if used intelligently.

Error prevention
It should not be possible for one simple error to cause widespread damage.
Here is what should be done:
• Understand the causes of error and design to minimize those causes.
• Do sensibility checks. Does the action pass the “common sense” test?
• Make it possible to reverse actions—to “undo” them—or make it harder to do what cannot be reversed.
• Make it easier for people to discover the errors that do occur, and make them easier to correct.

• Don’t treat the action as an error; rather, try to help the person complete the action properly. Think of the action as an approximation to what is desired.

ADDING CONSTRAINTS TO BLOCK ERRORS
Prevention often involves adding specific constraints to actions. In the physical world, this can be done through clever use of shape and size (bocchettone benzina/diesel).
Electronic systems have a wide range of methods that could be used to reduce error. One is to segregate controls, so that easily confused controls are located far from one another.
Another is to use separate modules, so that any control not directly relevant to the current operation is not visible on the screen, but requires extra effort to get to.

UNDO
Perhaps the most powerful tool to minimize the impact of errors is the Undo command in modern electronic systems, reversing the operations performed by the previous command, wherever possible.
The best systems have multiple levels of undoing, so it is possible to undo an entire sequence of actions.

CONFIRMATION AND ERROR MESSAGES

Many systems try to prevent errors by requiring confirmation before a command will be executed, especially when the action will destroy something of importance. But these requests are usually ill-timed because after requesting an operation, people are usually certain they want it done.

Warning messages are surprisingly ineffective against mistakes

A better check would be a prominent
display of both the action to be taken
and the object, perhaps with the choice

of “cancel” or “do it.”
The important point is making salient
what the implications of the action are.

CONFIRMATION AND ERROR MESSAGES
What can a designer do?
• Make the item being acted upon more prominent. That is, change the appearance of the actual object being acted upon to be more visible: enlarge it, or perhaps change its color.
• Make the operation reversible. If the person saves the content, no harm is done except the annoyance of having to reopen the file. If the person elects Don’t Save, the system could secretly save the contents, and the next time the person opened the file, it could ask whether it should restore it to the latest condition.

SENSIBILITY CHECKS
Electronic systems have another advantage over mechanical ones: they can check to make sure that the requested operation is sensible.
Suppose I wanted to transfer $1,000 into a Korean bank account in won ($1,000 is roughly ₩1,000,000). But suppose I enter the Korean number into the dollar field. Oops—I’m trying to transfer a million dollars. Intelligent systems would take note of the normal size of my transactions, querying if the amount was considerably larger than normal.

MINIMIZING SLIPS
Slips most frequently occur when the conscious mind is distracted, either by some other event or simply because the action being performed is so well learned that it can be done automatically, without conscious attention. As a result, the person does not pay sufficient attention to the action or its consequences.
It might therefore seem that one way to minimize slips is to ensure that people always pay close, conscious attention to the acts being done.
Bad idea!
Skilled behavior is subconscious, which means it is fast, effortless, and usually accurate.

MINIMIZING SLIPS
Many slips can be minimized by ensuring that the actions and their controls are as dissimilar as possible, or at least, as physically far apart as possible.
The best way of mitigating slips is to provide perceptible feedback about the nature of the action being performed, then very perceptible feedback describing the new resulting state, coupled with a mechanism that allows the error to be

undone.
For example, the use of machine-readable codes has led to a dramatic reduction in the delivery of wrong medications to patients.
(These scans do increase the workload, but only slightly. Other kinds of errors are still possible, but these simple steps have already been proven worthwhile.)

MINIMIZING SLIPS
Common engineering and design practices seem as if they are deliberately intended to cause slips.
Rows of identical controls or meters is a sure recipe for description-similarity errors.
Situations with numerous interruptions, yet where the design assumes undivided attention, are a clear enabler of memory lapses—and almost no equipment today is designed to support the numerous interruptions that so many situations entail.
Procedures should be designed so that the initial steps are as dissimilar as possible.

Error mitigation
The Swiss cheese metaphor suggests several ways to reduce accidents:
• Add more slices of cheese.
• Reduce the number of holes (or make the existing holes smaller).
• Alert the human operators when several holes have lined up.
Each of these has operational implications:
More slices of cheese means mores lines of defense (more checks and control steps)
Reducing the number of critical safety points where error can occur is like reducing the number or size of the holes (Reduce distraction and design for error mitigation)

Design Principles for Dealing with Error
People are flexible, versatile, and creative. Machines are rigid, precise, and relatively fixed in their operations. There is a mismatch between the two,
Difficulties arise when we do not think of people and machines as collaborative systems, but assign whatever tasks can be automated to the machines and leave the rest to people. This ends up requiring people to behave in machine like fashion, in ways that differ from human capabilities.
We expect people to monitor machines, which means keeping alert for long periods, something we are bad at.

Design Principles for Dealing with Error

What we call “human error” is often simply a human action that is inappropriate for the needs of technology. As a result, it flags a deficit in our technology. It should not be thought of as error.
We should eliminate the concept of error: instead, we should realize that people can use assistance in translating their goals and plans into the appropriate form for technology.
Therefore, the best designs take that fact as given and seek to minimize the opportunities for errors while also mitigating the consequences. Assume that every possible mishap will happen, so protect against them.

key design principles
• Put the knowledge required to operate the technology in the world. Don’t require that all the knowledge must be in the head. Allow for efficient operation when people have learned all the requirements, when they are experts who can perform without the knowledge in the world, but make it possible for non-experts to use the knowledge in the world. This will also help experts who need to perform a rare, infrequently performed operation or return to the technology after a prolonged absence.
• Use the power of natural and artificial constraints: physical, logical, semantic, and cultural. Exploit the power of forcing functions and natural mappings.

key design principles
● Bridge the two gulfs, the Gulf of Execution and the Gulf of Evaluation. Make things visible, both for execution and evaluation. On the execution side, provide feedforward information: make the options readily available. On the evaluation side, provide feedback: make the results of each action apparent. Make it possible to determine the system’s status readily, easily, accurately, and in a form consistent with the person’s goals, plans, and expectations.

https://uxdesign.cc/how-to-prevent-your-users-from-making-mistakes-d641c6260b29

Personas, Requirements,
user stories, scenarios and

use cases

Personas, scenarios, user stories and use case

source:
https://www.justinmind.com/blog/user-personas-scenarios-user-stories-and-storyboards-whats-the-difference/

4 user research methods to help you create reliable and realistic representations of your target users and design accordingly

USER CENTERED DESIGN!

PERSONAS

PERSONAS

A user persona is a character that represents a potential user of your website or app.

In user centered-design, personas help the design team to target their designs around users.

they are an integral part of the user experience research phase of software development.

In user research, UXers will gather data related to the goals and frustrations of their potential users. Then, they create personas to put that data into context.

PERSONAS

There is usually more than one type of user who will interact with your website or app, and creating personas helps to scope out the range of users.

For UX teams, introducing persona development into the design process helps them learn about the spectrum of goals and needs of their users.

A designer’s checklist if you like.

PERSONAS

User persona development helps us bridge the gap between the company and its users by allowing us to measure true user behavior and figure out what their end goals might be.

It drives design decisions by allowing software teams to get a deeper understanding of the users who will be using the systems they are building.

But while they might be useful, creating user personas isn’t always fun and games.

PERSONAS
Creating a user persona starts with user research.
By observing users, UXers can understand their behavior and motivations, and then design accordingly.
There are plenty of user research techniques that help UXers capture this information, such as:
● task analysis (card sorting, first click testing etc.)
● feedback (contextual interviews and focus groups)
● prototyping (experimenting with ideas prior to developing them)

PERSONAS
Depending on how many data you have, you can define 3 types of personas:
● Proto-personas, a lightweight form of ad-hoc personas created with no new research that designers can define in a few hours workshop.
● Qualitative personas, created by running solid exploratory qualitative research (such as interviewing users) with a small-to-medium sample size (5-30), and then segmenting users.
● Statistical personas, collecting data via a survey sent to a large sample of your user base and then using statistical analysis to find clusters of similar responses. In reality they don't work alone as they come from a previous qualitative research expanded with statistical research.
https://www.nngroup.com/articles/persona-types/

PERSONAS
How many personas should I define?
user personas design should ideally be based around the Pareto Principle. Focus on that 20% of your user-base that will use/buy 80% of your features/products, or that will account for 80% of your revenue.

Pareto principle states that, for many events, roughly 80% of the effects come from 20% of the causes

PERSONAS
Upon observing users, UX designers split up the test data into possible user archetypes, or user personas.
Then all this information is put into context in a user persona template.
Designers can only respond to their users' needs once they know what those needs are.
examples:
https://simplicable.com/new/user-persona

https://www.justinmind.com/blog/user-personas-which-game-of-thrones-character-is-yours/

Types of information in a user persona

● Demographic details, such as age, marital status, or income
● Personal details, such as a short biography, photograph, and name
● Attitudinal and/or cognitive details, such as information about the persona's mental model, pain points, and feelings about the tasks that need to be accomplished
● Goals and motivations for using the product
● Behavioral details about how the persona tends to act when using the product

PERSONAS & ARCHETYPES
In the future you will be tempted to define a persona that is more abstract: this is called an Archetype.
Both personas and archetypes represent the same user type and contain the same core insights about this user type's needs, behaviors, goals, pain points, and even motivations. Both can serve to compare different user priorities and motivations (e.g., the reliability-focused comparison shopper vs. the interior designer that wants to deliver a harmonious look for clients). But archetypes are abstract, while personas wear a human face.

https://www.nngroup.com/articles/personas-archetypes/

PERSONAS & ARCHETYPES

An example of a possible archetype can be:

"The unlikely hero with humble beginnings"

PERSONAS & ARCHETYPES

Instead the personas could be:

Why personas fail?
https://www.youtube.com/watch?v=fal4Wylyt3M

Personas recap
https://www.smaply.com/blog/personas

REQUIREMENTS

Requirements
a requirement is a service, function or feature that a user needs.
Requirements can be functions, constraints, business rules or other elements that must be present to meet the need of the intended users.

Requirements
For example:
In a training company with its own training centre:
● The Course Manager has a requirement to schedule training courses and reserve rooms, in order to make available courses visible and to ensure courses run effectively
● The Training Centre Manager has a requirement to keep track of what training is running, in order to ensure appropriate allocation of trainers to courses
● The Financial Accountant has a requirement to maximise the amount of time that the training rooms are in use, in order to maximise revenue from the rooms

Requirements
Usually you will also find this type of explanation of the possible requirements:

https://usabilitygeek.com/requirements-gathering-user-experience-pt1/
We are focusing on user requirements.

Requirements
However, the attempt to define a full and detailed set of requirements too early in a project often proves to be counterproductive, restrictive and wasteful.
It is not possible to define all of the detailed requirements at the outset of a long project.
The business environment changes as time progresses; new requirements and opportunities present themselves. As the project progresses, the team understand more about the business need.
Defining detailed requirements too early means either needing to change the

specification later, which wastes the original work, or delivering to the originally-specified requirements and subsequently failing to adequately satisfy the business need.

Requirements
The success of any solution is the product of two aspects:
● what it does (functionality, features)
● how well it performs against defined parameters (non-functional attributes, acceptance criteria, service levels)

Categories of Requirements
Functional Requirements (FRs) express function or feature and define what is required, e.g.
● Visit customer site
● Obtain conference venue
The requirements do not state how a solution will be physically achieved.
● Drive to customer site is one possible solution. However, fly to customer site or travel by train to customer site are potential alternative solutions which may be worth consideration
● Build conference centre is one possible solution. Hire a hotel room is an alternative solution

Categories of Requirements
Non-functional Requirements (NFRs) define how well, or to what level a solution needs to behave.
They describe solution attributes such as security, reliability, maintainability, availability (and many other “...ilities”), performance and response time, e.g.
● responding within 2 seconds
● being available 24 hours per day, every day

Categories of Requirements
NFRs may be:
● Solution-wide or impacting a group of functional requirements: e.g.
○ All customer facing functionality must carry the company logo
○ All customer-facing functionality must respond within 2 seconds to requests
● Related to a particular functional requirement, e.g.
○ Hire conference venue might have NFRs of accessibility, security, and availability

USER STORIES

User stories
A user story is a short statement or abstract that identifies the user and their need/goal. It determines who the user is, what they need and why they need it.
There is usually one user story per user persona.
there are often multiple user personas – it's a good thing that user stories are brief!
stories are at the center of the user experience. Why? They put things in context and focus on the 'holistic' rather than the 'artifact'.
A User Story is a requirement expressed from the perspective of an end-user goal.

User stories
For example:
"As a UX Manager, John oversees all the design projects, including assets creation and prototyping efforts, at the design consultancy where he works. He needs easy access to a design tool that allows him to centralize UI libraries so that multiple designers to work simultaneously on a prototype."

Requirements example: The Course Manager has a requirement to schedule training courses and
reserve rooms, in order to make available courses visible and to ensure courses run effectively

User stories
User stories help to document practical information about users, such as the different needs and motivations for accessing a website or app.
They also help the development team estimate a roadmap needed to deliver the end product.

User stories
How to write a user story:
It's super simple to write a user story.
"As a [role], I want [feature] because [reason]."

For example: "As UX Manager, John wants centralized assets management so that his designers are in sync."

this approach helps you to think about who a certain feature is built for and why

Who writes user stories?

Anyone can write user stories.
It's the product owner's responsibility to make sure a product backlog of user stories exists, but that doesn't mean that the product owner is the one who writes them.
Over the course of a good agile project, you should expect to have user story examples written by each team member.
Also, note that who writes a user story is far less important than who is involved in the discussions of it.

User stories
One of the benefits of user stories is that they can be written at varying levels of detail.
We can write a user story to cover large amounts of functionality. These large user stories are generally known as epics.
Here is an epic agile user story example from a desktop backup product:
● As a user, I can backup my entire hard drive.

How is detail added to user stories?
Because an epic is generally too large for a design team to complete in one iteration, it is split into multiple smaller user stories before it is worked on.
The epic in the previous slide could be split into dozens (or possibly hundreds), including these two:
● As a power user, I can specify files or folders to backup based on file size, date created and date modified.
● As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

How is detail added to user stories?
Detail can be added to user stories in two ways:
● By splitting a user story into multiple, smaller user stories.
● By adding “conditions of satisfaction.”
When a relatively large story is split into multiple, smaller agile user stories, it is natural to assume that
detail has been added. After all, more has been written.

The conditions of satisfaction is simply a high-level acceptance test that will be true after the agile user
story is complete.

How is detail added to user stories?

Consider the following as another agile user story example:
As a vice president of marketing, I want to select a holiday season to be used when reviewing the
performance of past advertising campaigns so that I can identify profitable ones.

Detail could be added to that user story example by adding the following conditions of satisfaction:
● Make sure it works with major retail holidays: Christmas, Easter, President's Day, Mother's Day,
Father's Day, Labor Day, New Year's Day.
● Support holidays that span two calendar years (none span three).
● Holiday seasons can be set from one holiday to the next (such as Thanksgiving to Christmas).
● Holiday seasons can be set to be a number of days prior to the holiday.

Using personas and user stories together
To prioritize features:
https://www.youtube.com/watch?v=xamf6hpD5nw

SCENARIOS

Scenarios
A scenario is a situation that captures how users perform tasks on your site or app.
Scenarios describe the user's motivations for being onsite (their task or goal) and/or a question they need answered, and suggest possible ways to accomplish these objectives.
It is essentially a development of the user story, and can relate to multiple target users.
However, scenarios can also be broken down into use cases that describe the flow of tasks that any one user takes in a given functionality or path.

Scenarios
For example, a scenario could outline how John uses a mobile app to buy a ticket to a design workshop whilst on his way to work.
Scenarios help stakeholders envision the ideas of the design team by providing context to the intended user experience – frequently bridging communication gaps between creative and business thinking.
For the design team, scenarios help them imagine the ideal solution for a user's

problem.
“Scenarios are the engine we use to drive our designs.” (UX influencer, Kim Goodwin)

What to Consider When Writing Scenarios
Good scenarios are concise but answer the following key questions:
● Who is the user? Use the personas that have been developed to reflect the real, major user groups coming to your site.
● Why does the user come to the site? Note what motivates the user to come to the site and their expectations upon arrival, if any.
● What goals does he/she have? Through task analysis, you can better understand the what the user wants on your site and therefore what the site must have for them to leave satisfied.
Some scenarios also answer: How can the user achieve their goals on the site? Define how the user can achieve his/ her goal on the site, identifying the various possibilities and any potential barriers.

How to write a scenario
Scenario planning starts with scenario mapping.
The design team, developers and product owner will meet to exchange ideas and create a strategy based on their user personas.
With the primary user defined through persona development, they can now consider the key task that the user hopes to achieve.
The next step is to perform a scenario analysis, put the user’s goals into context and walk through the steps that the user would take.

How to write a scenario
Creating Scenarios requires a special mindset.
It is about focusing on the users’ goals: what will they try to accomplish on a website or inside an app?
Additionally, it is also important to think about their context, their prior knowledge and background.

How to write a scenario
Thanks to Scenarios, we can determine:
● the most important points to focus on during the UX design process
● which steps of the process would require additional help to your users
● the main needs and motivations of your users.
Scenarios are built upon User Story: these short statements describe what a certain User Persona needs, and why. Scenarios take User Stories to the next level by adding the interaction with the product or service to the story.

How to write a scenario

https://uxknowledgebase.com/scenarios-43e05671b07

Type of scenarios
https://www.usability.gov/how-to-and-tools/methods/scenarios.html

Scenarios recap
https://www.interaction-design.org/literature/topics/user-scenarios

USE CASES

Use Cases
A use case is a written description of how users will perform tasks on your app. It outlines, from a user's point of view, a system's behavior as it responds to a request.
Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.

Scenarios vs Use Cases
A Scenario involves a situation that may have single or multiple actors that take a given functionality or path to achieve their goal
A use case involves an actor and the flow that a particular actor takes in a given functionality or path. These often get grouped so you have a "set" of use cases to account for each scenario.
The main difference is "perspective".
The use case is more granular than the scenario.
it usually involves coming up with a scenario and then defines all the use cases that fit into that particular scenario.

Use cases
Use cases add value because they help explain how the system should behave and in the process, they also help brainstorm what could go wrong.
They provide a list of goals and this list can be used to establish the cost and

complexity of the system.
Project teams can then negotiate which functions become requirements and are built.

Use cases

Elements of a Use Case
Depending on how in depth and complex you want or need to get, use cases describe a combination of the following elements:
● Actor – anyone or anything that performs a behavior (who is using the system)
● Stakeholder – someone or something with vested interests in the behavior of the system under discussion (SUD)
● Primary Actor – stakeholder who initiates an interaction with the system to achieve a goal
● Preconditions – what must be true or happen before and after the use case runs.

Elements of a Use Case
● Triggers – this is the event that causes the use case to be initiated.
● Main success scenarios [Basic Flow] – use case in which nothing goes wrong.
● Alternative paths [Alternative Flow] – these paths are a variation on the main theme. These exceptions are what happen when things go wrong at the system level.

Use case?
https://www.youtube.com/watch?v=Ct-lOOUqmyY&ab_channel=NowI%27veSeenEverything

How to write a use case
https://www.usability.gov/how-to-and-tools/methods/use-cases.html
Kenworthy (1997) outlines the following steps:
1. Identify who is going to be using the website.
2. Pick one of those users.
3. Define what that user wants to do on the site. Each thing the use does on the site becomes a use case.
4. For each use case, decide on the normal course of events when that user is

using the site.

How to write a use case
https://www.usability.gov/how-to-and-tools/methods/use-cases.html
5. Describe the basic course in the description for the use case. Describe it in terms of what the user does and what the system does in response that the user should be aware of.
6. When the basic course is described, consider alternate courses of events and add those to "extend" the use case.
7. Look for commonalities among the use cases. Extract these and note them as common course use cases.
8. Repeat the steps 2 through 7 for all other users.

The Takeaway
Engaging in user persona, user story, scenario and/or use case development will help you to identify key information about your users and build products that will delight your users time and time again.
Everything we do to get closer to users is a step in the right direction.

examples
https://uxplanet.org/5-examples-of-brilliant-ux-design-8e847bf0bcc0
https://uxdesign.cc/fitbit-a-usability-case-study-b23e4c539c3c
https://uxplanet.org/foodmix-cooking-app-ux-case-study-d046c1f5896b

Information Architecture

GUI Design
Graphical User Interface (GUI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.
UI brings together concepts from interaction design, visual design, and information architecture.

Interface structures
An organizational structure is how you define the relationships between pieces of content.
Successful structures allow users to predict where they will find information on the site.

It’s important to take into account user expectations and implement consistent methods of organizing and displaying information so that users can extend their knowledge from familiar pages to unfamiliar ones.
The four main organizational structures are Hierarchical, Sequential, Matrix and database model.
source: https://www.usability.gov/how-to-and-tools/methods/organization-structures.html

Hierarchical Structures
In Hierarchical Structures, sometimes referred to as tree structures or hub-and-spoke structures, there is a top down approach or parent/child relationships between pieces of information.
Users start with broader categories of information (parent) and then drill further down into the structure to find narrower, more detailed information (child).
Many users are familiar with structuring information in hierarchies because they see these structures on a daily basis in the way businesses have a formed their lead leadership structure, the way project plans are set-up, and so on.

https://ux.stackexchange.com/questions/90872/display-hierarchical-structure-and-corresponding-reference-data

Sequential Structures
Websites with Sequential Structures require users to go step-by-step, following a specific path through content.
An example of this type of structure is when a user is attempting to purchase something or are taking a course online. Sequential structures assume that there is some optimal ordering of content that is associated with greater effectiveness or success.

Matrix Structures
A Matrix Structure allows users to determine their own path since content is linked in numerous ways. This type of structure takes full advantage of the principles behind hypertext, or HTML. For example, one user could choose to navigate through a set of content based on date while another navigates based on topic.

Database Model

The Database Model takes a bottom-up approach. The content within this structure leans heavily on the linkages created through the content's metadata. This type of model facilitates a more dynamic experience generally allowing for advanced filtering and search capabilities as well as providing links to related information in the system that has been properly tagged.

Creating Sustainable Structures

Site/GUI architecture has a long term impact on the site. It's important to put thought into the structure and ensure that it takes into account content updates in the future. Site managers should keep in mind the following when structuring a site:

Allow room for growth. Creating a site that can accommodate the addition of new content within a section (left image) as well as entire new sections (right image).

Creating Sustainable Structures

Avoid structures that are too shallow or too deep.

Striking a balance is never easy is an important goal of any architecture.

Structures that are too shallow require massive menus.

Users rely on information architects to create logical groupings to facilitate movement throughout the site. In contrast, structures that are too deep bury information beneath too many layers. These structures burden the user to have to navigate through several levels to find the content that they desire.

Information Architecture

Information Architecture

Information architecture (IA) focuses on organizing, structuring, and labeling content in an effective and sustainable way. The goal is to help users find information and complete tasks.

To do this, you need to understand how the pieces fit together to create the larger picture, how items relate to each other within the system.

Why a Well Thought Out IA Matters?

According to Peter Morville Site exit disclaimer, the purpose of your IA is to help users understand where they are, what they've found, what's around, and what to expect. As a result, your IA informs the content strategy through identifying word choice as well as informing user interface design and interaction design through playing a role in the wireframing and prototyping processes.

Information Architecture

To be successful, you need a diverse understanding of industry standards for creating, storing, accessing and presenting information. Lou Rosenfeld and Peter Morville in their book, Information Architecture for the World Wide Web, note that the main components of IA are:

- Organization Schemes and Structures: How you categorize and structure information
- Labeling Systems: How you represent information
- Navigation Systems: How users browse or move through information
- Search Systems: How users look for information

Information Architecture

In order to create these systems of information, you need to understand the interdependent nature of users, content, and context.

Rosenfeld and Morville referred to this as the “information ecology” and visualized it as a venn diagram.

Each circle refers to:

- Context: business goals, funding, politics, culture, technology, resources, constraints
- Content: content objectives, document and data types, volume, existing structure, governance and ownership
- Users: audience, tasks, needs, information-seeking behavior, experience

Organization Schemes

Organization schemes have to do with how you are going to categorize your content and the various ways you'll create relationships between each piece. Most content can be categorized in multiple ways.

Schemes can be broken down into Exact and Subjective.

Depending on the content, it’s conceivable that the site may combine schemes as opposed to treating them independently.

Exact Organization Schemes

Exact organization schemes objectively divide information into mutually exclusive sections.

These systems comparatively are easy for information architects to create and categorize content within. However, they can be a challenge at times for users. It requires that the user understands how what they are looking for fits within the model.

Examples of exact organizational structures include:

Exact Organization Schemes

Alphabetical schemes make use of our 26-letter alphabet for organizing their contents.

For this type of scheme to be successful, it is important that the content labeling matches the words that users are looking for.

Sometimes, alphabetical schemes in the form of an A-Z index serve as secondary navigational components to supplement content findability that is otherwise organized.

Exact Organization Schemes

Chronological schemes organize content by date. For these schemes to be successful there must be agreement about when the subject of the content took place.

Geographical schemes organize content based on place. Unless there are border disputes, this type of scheme is fairly straightforward to design and use.

Often these types of schemes serve as a good supplemental way to navigate a site that that is otherwise organized. For example, you may choose to provide a map to display information or an A-Z index to get to topics grouped primarily by one of the following subjective schemes.

Subjective Organization Schemes

Subjective organization schemes categorize information in a way that may be specific to or defined by the organization or field.

Although they are difficult to design, they are often more useful than exact organization schemes.

When information architects take the time to consider the user's mental models and group the content in meaningful ways, these types of schemes can be quite effective in producing conversions.

This type of categorization can also help facilitate learning by helping users understand and draw connections between pieces of content.

Subjective Organization Schemes

Examples of subjective schemes include the following:

Topic schemes organize content based on the specific subject matter.

Task schemes organize content by considering the needs, actions, questions, or processes that users bring to that specific content.

Subjective Organization Schemes

Audience schemes organize content for separate segments of users. Audience schemes can be closed or open, meaning that users are able to navigate from one audience to another. This type of scheme does present challenges unless the content lends itself to users very easily self-identifying to which audience they belong and perhaps not fitting multiple audience profiles.

Metaphor schemes help users by relating content to familiar concepts. This is used in interface design (folders, trash, etc) but can pose challenges when used as the site's primary organization scheme.

Challenges of Creating Hybrids

Implementing schemes independently has its advantages because it keeps things simple for the user.

They can identify the categorization and form a mental model that can be quickly understood.

Mixing schemes by creating hybrids can cause confusion for users.

This is often proposed as a solution when project teams cannot agree on a single scheme to categorize the content.

Navigation

The study of IA informs the navigation design process.

Navigation is about the user's orientation in the interface, and its main goal is to enable the user to find the information and functions he or she is looking for and, not only that, to encourage him or her in a direction that might be desirable to him or her.

The basic principles for good navigation are findability and discoverability.

Navigation

Findability means being able to get to the information you are looking for, or more precisely, how the user is able to localize the information he or she considers relevant.

In a library, for example, it indicates the ease with which

one can locate a book one is looking for.
Placing elements consistently within the interface and using standards usually promotes findability.

Navigation

Discoverability refers to the possibility that users have of discovering new information or new features that they were not aware of and were not specifically looking for.
In the library example, it concerns the possibility of discovering a new book that catches one's eye while looking at a shelf or while searching for something else.
The interface often indirectly provides the tools that enable the discovery activity because it can only be evaluated in retrospect. However, it is possible to design for discoverability by placing the new feature, for example, near the most viewed or most relevant items.

Findability vs Discoverability

With Findability the users knows or expects that a certain content or feature will be available.

With Discoverability the users don’t know that the content or functionality is there: they have to discover it!

Users behavior patterns

Users behavior patterns
We have already seen how to identify users needs, motivations and goals (personas, stories...).
But when a user starts an interaction with a system, their actions will be influenced by the information available and the interface in which they are present.
During the last 30 years, Information Architecture experts have recognize that users usually perform a certain set of the same behavior patterns: the flow of the

succession of events between user and system.
Specifically, in regard to information-seek, different different patterns of behavior have been identified. In the next slides we will see some of them. (Morville Peter, Jeffery Callender. Search Patterns. Design for Discovery. O'Really Media, 2010.)

Information-seeking behavior patterns

1. Quit: the user performs a search, sees the results and exits.
2. Narrow: the user performs a search, sees the results and refines them using filters, sorting tools or the advanced search.
3. Expand: the user performs a search, sees the results and expands the scope (displaying related results, etc.).

Information-seeking behavior patterns

4. Pearl growing: the user performs a search, opens one of the results and then opens or uses links within the result (mining topics from the result, typical navigation pattern in Wikipedia).
5. Pogosticking: the user performs a search, and then repeats the action of opening a result and returning to the results page.

Information-seeking behavior patterns

6. Thrashing: the user performs a search, sees the results, and then returns to perform a new search by adding details to the query.

7. Berry picking: the user performs a search, opens a result, and from the information received performs a new search refining the query from time to time.

Berry picking

Antipatterns

Some of the patterns we just saw are behaviors that can happens at times: “That’s to be expected, but when it happens a lot, it’s not sampling; it’s a symptom.” (Morville and Callender)

This is the case of Pogosticking, Thrashing and Berry picking.

If this is a repeated behavior within the interface then it must be considered a symptom of antipattern: a search pattern produced by a poor Information Architecture and design!

Design patterns

Design patterns

In parallel and closely related to the behavior patterns, it is possible to identify other possible additional patterns referred, however, directly to the design activity. This design patterns have emerged as repeatable solutions to common problems. They can work like a sort of best practices, or design components that allow you to quickly identify possible features your website or app needs or must provide to the users.

Design patterns

Continuing with the information-seeking activity, a possible set of design patterns could be:

1. Autocomplete: completes the search query at the stage when the user formulates it, thus trying to return the searched query as directly as possible.
2. Autosuggest: similar to autocomplete but tries to help the user by providing him with even distant (but related) ideas from the query.
3. Instant Results: provides results while the query is being typed.

Information-seeking Design patterns

4. Did you mean: after the query is submitted, it provides a hint about the most appropriate result (useful in case of spelling errors for example).
5. Autocorrect: instead of suggesting a

correction in the search query, it automatically applies it and shows the results.
6. Best First: shows as the first results the best ones chosen by an algorithm (by relevance, popularity, date, format, in a customized way for the user specific etc.).

Information-seeking Design patterns
7. Partial matches: shows the results that most closely match the query allowing a page with zero results not to be returned.
8. Related Searches: shows related searches with the query made that can provide inspiration to the user for further research or help clarify an ambiguous query.
9. Federated Search: allows you to search several databases at once. As a result there will be a variety of results that are difficult to refine.

Information-seeking Design patterns
10. Faceted Navigation: provides the user with options to refine results through fields regarding different types of metadata (price, color, tags, etc.).
11. Advanced Search: allows the user to refine the search before performing it through more or less elaborate options.
12. Scoped Search: if content is organized into categories, a dropdown menu can be provided to specify the scope of the search.

Information-seeking Design patterns
13. Personalization: concerns the adaptation of the results shown to the specific user using the system (e.g., recommended results on Amazon or personalized results on Google Maps).
14. Pagination: shows a maximum number of results per page. The standard set by Google is

ten results per page. Crucial here are the snippets provided for each result (what it is about, available links, etc.).
15. Structured Results: each individual result is displayed in the manner most congruent with its content (an address will be represented in a map, a stock market index in a graph, etc.).

Information-seeking Design patterns
16. Actionable Results: depending on its content, each result will provide the ability to interact with it (a phone number that allows a call to be made directly, a video that allows it to be played).
17. Comparing Results: allows the user to compare results with each other (for example, the features and price of different products).
18. Unified Discovery: the search and refinement modes are combined together. On the same page you can find a search bar, exploration of a browsable taxonomy, faceted navigation, and so on.

Interfaces' layout and components

The Document Object Model (DOM)
The Document Object Model, usually referred to as the DOM, is an essential part of making websites interactive.
It is an interface that allows a programming language to manipulate the content, structure, and style of a website.
JavaScript is (can be) the client-side scripting language that connects to the DOM in an internet browser
Almost any time a website performs an action, such as rotating between a slideshow of images, displaying an error when a user attempts to submit an incomplete form, or toggling a navigation menu, it is the result of JavaScript (or another web language) accessing and manipulating the DOM.
more info https://www.taniarascia.com/introduction-to-the-dom/

The Document Object Model (DOM)
The DOM is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part

of the document.
The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects.
DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document.
Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.

The Document Object Model (DOM)
The principal standardization of the DOM was handled by the World Wide Web Consortium, which last developed a recommendation in 2004.
WHATWG took over the development of the standard, publishing it as a living document. The W3C now publishes stable snapshots of the WHATWG standard.

To render a document such as a HTML page, most web browsers use an internal model similar to the DOM.
The nodes of every document are organized in a tree structure, called the DOM tree, with the topmost node named as "Document object".
When an HTML page is rendered in browsers, the browser downloads the HTML into local memory and automatically parses it to display the page on screen.
DOM in Web Browser

DOM in JavaScript
When a JavaScript web page is loaded, the browser creates a Document Object Model of the page, which is an object oriented representation of an HTML document that acts as an interface between JavaScript and the document itself.
This allows the creation of dynamic web pages, because within a page JavaScript can:
● add, change, and remove any of the HTML elements and attributes
● change any of the CSS styles
● react to all the existing events
● create new events

Interfaces' components
When designing your interface, try to be consistent and predictable in your choice of interface elements. Whether they are aware of it or not, users have become familiar with elements acting in a certain way, so choosing to adopt those elements when appropriate will help with task completion, efficiency, and satisfaction.
Interface elements include but are not limited to:

● Input Controls: checkboxes, radio buttons, dropdown lists, list boxes, buttons, toggles, text fields, date field
● Navigational Components: breadcrumb, slider, search field, pagination, slider, tags, icons
● Informational Components: tooltips, icons, progress bar, notifications, message boxes, modal windows
● Containers: accordion
Continue: here

Front-end design

Wireframing

roberto.figlie@phd.unipi.it

Lesson Outline ● Wireframing
● Adaptability
● Responsive design
● Accessibility
● Wireflows

What is a
Wireframe?

Why wireframes?

Suppose you have a wonderful idea for a product (a website, an app, a video game, anything).
If you are not capable to convey that idea to other people, everyone will do what they think it is the best. This means that everyone will go in different directions!
Unless you are working alone, and even in that case you may think to have a precise idea of the design you are thinking of, but in fact it is not.

Why wireframes?

What is the best way to clarify a thought process that humans discovered during the past millennia?

Start with pen and paper!

What is a wireframe?

Coming back to developing a product:
Wireframes are blueprints that help communication between designers and programmers about the structure of the website or app they are developing.
Wireframes represent the beginning of putting information architecture into practice.

What is a wireframe?

The activity of wireframing indicates the process of creating basic sketches of a website or application's interface to demonstrate its structure and layout.

https://www.interaction-design.org/literature/topics/wireframing

What is a wireframe?

What can?
● Give its first visual shape to an information architecture.
● Encourage discussion within the team.
● Determine the functionalities of the UI.

What cannot?
● Give the feeling of the UI final look.
● Provide working functionalities.
● Guarantee you are understood

:)

What is a wireframe?

https://www.usability.gov/how-to-and-tools/methods/wireframing.html

Example:

- No colors
- No images
- Basic fonts

Types of wireframes

Low-fidelity Wireframes:
Simple sketches of the interface,
focusing on the structure and
layout.

High-fidelity Wireframes:
Detailed representation of the
interface with basic typography,
icons and documentation details.

https://www.justinmind.com/wireframe/low-fidelity-vs-high-fidelity-wireframing-is-paper-dead

Adaptability,
Responsive design
& Accessibility

The role of context

Context plays an important role in UX Design.
Examples of how context can affect user behavior and preferences:
- A user browsing a website on a mobile device may have different needs and behaviors than a user on a desktop computer.
- A user with an high degree of knowledge in the domain of a website may have different needs and preferences than a user that does not have it.

Context can include a wide range of factors such as user demographics, device, location, culture, and accessibility needs.
So it is important to conduct user research and testing to understand the different contexts in which users will interact with the design.

Adaptability

Remember: you are not designing for yourself!
This means two things:
1. Others think differently from you (subjective and/or cultural differences, etc.).
2. Others have different needs and capabilities (different skills, physical capabilities, etc.).

Adaptability

Adaptability in UX design is an umbrella term and refers to the ability of a design to adjust and respond to different user needs, contexts, skills, and preferences.
Usually, it covers the notions of:
- Responsive design
- Accessibility

Responsive Design

With the explosion of the device market came a fragmentation of the different display sizes available.
Moreover, each device has different layouts, standards, icons etc.

https://www.interaction-design.org/literature/topics/responsive-design

Responsive Design

With the increasing use of mobile devices to access the internet,

it is crucial to ensure that websites are easily readable and usable on a wide range of devices.
Responsive design comes to help as a method of designing and coding web pages, apps, etc., to adapt to the size and resolution of different devices.
RD allows users to have a consistent experience whether they are accessing the website on a desktop, tablet, or mobile device. It also makes it easier for businesses to maintain and update their website, as they only need to maintain one version of the site.

Responsive design: Techniques

Flexible grid: A flexible grid allows elements to adjust to the size of the screen. This can be achieved in many ways, depending on the framework you are using.

Responsive design: Techniques

Flexible images: Make images scale and adjust to the size of the screen.

Responsive design: Techniques

Media queries & breakpoints: Allows different styles to be applied based on the characteristics of the device.

Responsive design: Mobile first

Mobile-first design is a design philosophy that prioritizes mobile over desktop. This approach ensures that the mobile experience is optimized before designing for larger screens.
By designing for mobile first, designers ensure that the most important content and functionality is easily accessible on small screens, and that the site is easy to navigate with touch-based controls.
This approach also helps to ensure that the site loads quickly on mobile devices with slower internet connections. Additionally, mobile-first design can help improve search engine optimization (SEO) by making sure the site is optimized for smaller screens.

Responsive design: Mobile first

Mobile-first allows for a “progress advancement” design, meaning that the design starts from the minimum functions and basic interactions (usually found in smartphones), then building up adding more interactions and effects (usually found in desktop).
This is opposed to “graceful degradation”, that is starting with a complete design based on advanced devices, then cutting features to accommodate for smaller and less capable devices.
Examples?

Responsive design: other best practices

● Keep it simple: Avoid using complex layouts and keep the design minimalistic.
● Prioritize the content: The content should be the most important element on the page, and it should be easily readable on all devices.
● Design for touch: Make sure buttons and links are large enough to be easily tapped with a finger.
● Optimize for load time: Large images and other heavy elements can slow down a site's load time, especially on mobile devices. Optimizing these elements can improve the user experience.
● Test, test, test: Test the website on a variety of devices to ensure that it looks and functions correctly (possibly outside your “laboratory”).

Adaptive design

Other then responsive design there is another way to make the interface automatically adaptable to the user: adaptive design.
● Responsive design is a type of adaptation to the device in a fluid and flexible way.
● Adaptive design, usually stands for creating different layouts for each device. It is more expensive, but sometimes is better (e.g., big companies that have to make their website better for mobile, without spending time in retrofitting the desktop version)

Adaptive design

Design for adaptability

User Device

Responsive
design

Adaptive
design Accessibility

Adaptive
UX

But there can be
another way in which
the system can be
adaptive to the user.
The difference lays
in the context of
use:
● the context
depends on the
device used
● depends on the
users themselves.

Adaptive UX

How can a system adapt to a user?
Having a basic information on the user (position through GPS, collaborative filtering, or even a user model from analytics) you can leverage them to adapt the the interface to the user.
In the case of collaborative
filtering:
● user A likes X and Y
● and user B likes X
● so user B may be interested
in Y

https://www.amazon.science/the-history-of-amazons
-recommendation-algorithm

Adaptive UX

Together with collaborative filtering, content-based filtering is another possibility, but it is not based on the users, rather on the contents themselves.

Accessibility

Accessibility in UXD is crucial because it ensures that all users can access and interact with the digital products, regardless of their abilities.
With the increasing use of digital products and services, it is essential that designers consider accessibility to make sure that everyone can use them.

https://www.interaction-design.org/literature/topics/accessibility

Accessibility

Remember:
Accessibility helps everyone!
Having a product that provides accessibility, is not only the right thing to do, but is more usable to the whole range of users.

Some examples?

Accessibility

VS

Accessibility

To be aware of the possible accessibility issues, here is a list of a few:
● Visual (e.g., color blindness)

● Motor/mobility (e.g., wheelchair-user concerns)
● Auditory (hearing difficulties)
● Seizures (especially photosensitive epilepsy)
● Learning/cognitive (e.g., dyslexia)

Ability barriers can also arise for any user:
● Incidental (e.g., sleep-deprivation)
● Environmental (e.g., using a mobile device underground)

Accessibility standards: WCAG

Accessibility guidelines and standards provide a framework for creating accessible products.
The most widely recognized accessibility standard is the Web Content Accessibility Guidelines (WCAG) by the W3C, which is a set of guidelines for making web content more accessible to people with disabilities.

https://www.w3.org/WAI/standards-guidelines/wcag/

Accessibility standards: WCAG

Some examples of accessibility best practices are:
● Text alternatives for non-text content (e.g, images, icons, graphs)
● Captions and other alternatives for multimedia (e.g, subtitles on videos)
● Content can be presented in different ways (e.g., provide different layout possibilities, enlarge, read aloud, different colors)
● Content is easier to see and hear (e.g., font size, background color)
● Users can use different input modalities beyond keyboard (consider other input modalities)
● Users can easily navigate, find content, and determine where they are (have a clear information architecture and organize well the content!)
and so on...
You can find a better explanation and others at
https://www.w3.org/WAI/fundamentals/accessibility-principles/

Recap: How to design for adaptability?

You cannot think about adaptability only at the end of the design process.
Designing for adaptability should be integrated throughout the design process, starting from the research phase to the

testing and implementation phase.
Especially when you start to build wireframes, you already should know for whom you are designing and what are their needs.

Recap: How to design for adaptability?

User research: understanding the user needs, behaviors and preferences in different contexts.

Responsive design: creating designs that adapt to different screen sizes and devices.

Accessibility: ensuring that the design is usable for people with disabilities.

Testing: validating the design in different contexts and making adjustments accordingly.

From wireframes to wireflows

User flows

We said that wireframes are used to visually describe the user interface by means of static blueprints. But the real actions that the user will have on that interface are not static!
This is why we can use user flows.
User flows help to describe the possible actions available to the a particular user in our interface. It helps to map out all the possible steps and the movement of the user in our app/website.

https://careerfoundry.com/en/blog/ux-design/what-are-user-flows/#where-do-user-flows-fit-into-the-ux-d
esign-process

User flows: flowchart

When defining the flow the better way is to start from the personas, the requirements, scenarios and (even better) the user journey.
Thinking about the possible touchpoints that the user will have with the system we can define an abstract flowchart.

User flows: wireflows

Wireflows further express a flowchart by using wireframes instead of abstract descriptions.

User flows and wireflows

A wireflow can be also used to describe a particular user flow.
When you take the course of action that a specific persona

can take in your system and
describe it through a
particular flow chart you have
a user flow.
A user flow expressed with the
use of wireframes, as a
wireflow, narrates better the
use of the interface.

Innovations Methods and

Frameworks

Innovation
Innovation is commonly defined as the "carrying
out of new combinations" that include "the
introduction of new goods, ... new methods of
production, ... the opening of new markets, ... the
conquest of new sources of supply ... and the
carrying out of a new organization of any industry"
However, many scholars and governmental
organizations has given their own definition of the
concept.
Some common element in the different definitions
is a focus on newness, improvement and
spread.

Innovation
An innovation is something original and more effective and, as a consequence, new, that "breaks into" the market or society.
Innovation is related to, but not the same as, invention: innovation is more apt to involve the practical implementation of an invention to make a meaningful impact in a market or society, and not all innovations require a new invention.
Technical Innovation often manifests itself via the engineering process when the problem being solved is of a technical or scientific nature.
It is not possible to innovate without a HCD approach! a product or service in order to be innovative must be usable!

Sustaining innovation
Most of the innovation processes are based on
incremental innovation. Minor improvements of
existing products.

● step by step process
● low risk
● low speed
● no changes to company organization required
● no user re-skilling required
● low probability to change/scale-up the business
● target user and market sector stable

Disruptive Innovation
A disruptive innovation is an innovation that creates a new market and value network and eventually disrupts an existing market and value network, displacing established market-leading firms, products, and alliances.
The term was defined and first analyzed by the American scholar Clayton M. Christensen and his collaborators beginning in 1995, and has been called the most influential business idea of the early 21st century.
The term is now generalized to identify disruptive science and technological advances.

Disruptive Innovation
Not all innovations are disruptive, even if they are revolutionary.
For example, the first automobiles in the late 19th century were not a disruptive innovation, because early automobiles were expensive luxury items that did not disrupt the market for horse-drawn vehicles. The market for transportation essentially remained intact until the debut of the lower-priced Ford Model T in 1908. The mass-produced automobile was a disruptive innovation, because it changed the transportation market, whereas the first thirty years of automobiles did not.

https://hcldr.wordpress.com/2017/01/10/disruptive-innovation-in-healthcare/

Disruptive Innovation and Human Centered Design
Disruptive innovation must be user centered!
No users no innovation!

Human Centered Design

Human-centred design is an approach to interactive systems development that aims to make systems usable and useful by focusing on the users, their needs and requirements, and by applying human factors/ergonomics, usability knowledge, and techniques.
This approach enhances effectiveness and efficiency, improves human well-being, user satisfaction, accessibility and sustainability; and counteracts possible adverse effects of use on human health, safety and performance.
ISO 9241-210:2010(E)

Product Management Vs Product Development

product management is about the what, while product development is more concerned with the how. Working alongside each other, product managers and product development teams create the ideal product.
The product development team takes the requirements specified by the product manager and fashions them into a working product that meets the organization's quality standards.
Product development, is the process of getting an idea from concept to market. Product development teams can consist of software developers, designers, engineers, quality assurance testers — anything it takes to bring the ideal product to life.

Methods and Frameworks for Disruptive

Innovation

Human Centered Design Process

Human Centered Design Process
IDEO is one of the most innovative and award-winning design firms in the world. IDEO's main tenet is empathy for the end-user of their products. They believe that the key to figuring out what humans really want lies in doing two things:
● Observing user behavior: Try to understand people by observing them. For example, if you're designing a vacuum cleaner, watch people vacuum.
● Putting yourself in the situation of the end-user: IDEO does this to understand what the user experience is really like; to feel what their users feel.
Then, they use the information they gain to fuel their designs.

Human Centered Design Process
IDEO defines the human centered design process as a creative approach to problem-solving that starts with people and ends with innovative solutions that are tailor-made to suit their needs.

https://blog.movingworlds.org/human-centered-design-vs-design-thinking-how-theyre-different-and-how-to-use-them-together-to-create-lasting-change/

VIDEO
https://vimeo.com/106505300

- PROGRAMMAZIONE INTERFACCE 19-20 (all rights reserved)

image source - PROGRAMMAZIONE INTERFACCE 19-20 (all rights reserved)

image source - PROGRAMMAZIONE INTERFACCE 19-20 (all rights reserved)

Solution Prototype

- PROGRAMMAZIONE INTERFACCE 19-20 (all rights reserved)

Human Centered Design Process
1. Inspiration: The inspiration phase is all about understanding your user's needs and challenges, and dropping any preconceived notions you might have about them. At this stage in the HCD process you need to remove any specific outcomes in mind, and instead open yourself to a wide variety of possible solutions.
2. Ideation: After consolidating your research and findings, this phase is about visualizing, retargeting, brainstorming and discussing all the potential solutions. Penning down your ideas in front of you - regardless of how flawed or impractical - helps you and your end-user hone in on what's going to work and what's not. At this stage, you don't want to start off with expensive prototypes - all you need are some basic sketches, lists or small scale models to tap into your creativity without the pressure to produce a polished final product. What's important about this phase is that once you've gotten feedback early, you can reiterate your best ideas until you've made your way to a well-developed concept that works for everyone, and is aimed at impacting your user in a positive way.

3. Implementation: The first two phases were meant to set the ground for you and your team to find a
concept that feels right, before moving forward and setting aside money to build and run rapid prototypes. In
this phase, this contains the tail end of the pre-production phase, where a high fidelity prototype is put
together for your users to try out, as well as the actual production of the object (or coding, for web and
app-based projects). This is a good time to create a business model around the concept, make necessary
partnerships and prepare your product for real-world use.

Design Thinking

- PROGRAMMAZIONE INTERFACCE 19-20 (all rights reserved)

Design Thinking
Popularized by Stanford's
d.school, is a process that you
go through to create solutions
that will actually be adopted by
people.

https://blog.movingworlds.org/human-centered-design-vs-desi
gn-thinking-how-theyre-different-and-how-to-use-them-togethe
r-to-create-lasting-change/

- PROGRAMMAZIONE INTERFACCE 19-20 (all rights reserved)

Design Thinking
Design Thinking is an iterative process in which we seek to understand the user, challenge assumptions, and redefine problems in an attempt to identify alternative strategies and solutions that might not be instantly apparent with our initial level of understanding.
At the same time, Design Thinking provides a solution-based approach to solving problems.
It is a way of thinking and working as well as a collection of hands-on methods.

- PROGRAMMAZIONE INTERFACCE 19-20 (all rights reserved)

Design Thinking
Design Thinking is extremely useful in tackling problems that are ill-defined or unknown, by re-framing the problem in human-centric ways, creating many ideas in brainstorming sessions, and adopting a hands-on approach in prototyping and testing.
Design Thinking also involves ongoing experimentation: sketching, prototyping, testing, and trying out concepts and ideas.

Design Thinking
5 phases:
1. Empathise: study your users
2. Define your users' needs, their problem,
and your insights
3. Ideate by challenging assumptions and
creating ideas for innovative solutions
4. Prototype
5. Test

HCD process + Design Thinking

Design Thinking vs HCD
To make this more clear, any business can use Design Thinking to build a solution that is capable of making money.
For example, a company may use Design Thinking to create a video game or TV show for kids.
Applying Human-Centered Design on top of this will ensure that the show actually serves the needs of the people watching it.

https://blog.movingworlds.org/human-centered-design-vs-desi
gn-thinking-how-theyre-different-and-how-to-use-them-togethe
r-to-create-lasting-change/

Design Thinking vs HCD Process vs HCD

Human Centered Design

HCD Process Design Thinking

HCD is a mindset.
HCD Process e Design Thinking are design methods

House of Cards by Netflix
Before starting the production of House of Card, by analysing their data sets carefully, Netflix noticed that there was a correlation between fans of the original BBC House of Cards TV show and fans of both

Kevin Spacey and director David Fincher.
Netflix brought together these three elements in one show and, voila, instant cult classic.

https://ideadrop.co/examples-of-data-driven-innovation/

Nappies and beer, Walmart
Beer and baby nappies aren't two things that you'd usually associate with each other. However, these two products have become infamous in data science circles because of their unique relationship.
In 1992, Karen Heath – an analyst at Teradata – discovered that men visiting Walmart were extremely likely to buy beer whenever they stopped in to buy babies nappies. By placing the two items near to each other in the outlet, she was able to increase sales of both items by a significant margin.

https://ideadrop.co/examples-of-data-driven-innovation/

Development methods for innovative

products:

Agile, Scrum and Devops

Waterfall Development
Waterfall methodology is a linear project management approach, where stakeholder and customer requirements are gathered at the beginning of the project, and then a sequential project plan is created to accommodate those requirements.
The waterfall method is so named because each phase of the project cascades into the next, following steadily down like a waterfall.
It's a thorough, structured methodology and one that's been around for a long time, because it works. Some of the industries that regularly use the waterfall method include construction, IT and software development.
However, the term "waterfall" is usually used in a software context.

Agile
Agile is the ability to create and respond to change. It is a way of dealing with,

and ultimately succeeding in, an uncertain and turbulent environment.
It's really about thinking through how you can understand what's going on in the environment that you're in today, identify what uncertainty you're facing, and figure out how you can adapt to that as you go along.
Agile software development is more than frameworks such as Scrum, Extreme Programming or Feature-Driven Development (FDD).
Agile software development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it.

AGILE MANIFESTO and Principles

https://agilemanifesto.org/
https://www.agilealliance.org/agile101/12-prin
ciples-behind-the-agile-manifesto/

Agile
One thing that separates Agile from other approaches to software development is the focus on the people doing the work and how they work together.
Solutions evolve through collaboration between self-organizing cross-functional teams utilizing the appropriate practices for their context.
There's a big focus in the Agile software development community on collaboration and the self-organizing team.
Agile is the best building method for HCD and Design thinking where continuous iterations are required

Risk in product development

Scrum
Scrum is an agile framework for developing, delivering, and sustaining complex products, with an initial emphasis on software development, although it has been used in other fields including research, sales, marketing and advanced technologies.
It is designed for teams of ten or fewer members, who break their work into goals that can be completed within timeboxed iterations, called sprints, no longer than one month and most commonly two weeks.
The Scrum Team track progress in 15-minute time-boxed daily meetings, called daily scrums. At the end of the sprint, the team holds sprint review, to demonstrate the work done, and sprint retrospective to improve continuously.

Scrum
A key principle of Scrum is the dual recognition that customers will change their minds about what they want or need (often called requirements volatility) and that there will be unpredictable challenges for which a predictive or planned approach is not suited.
As such, Scrum adopts an evidence-based empirical approach – accepting that the problem cannot be fully understood or defined up front, and instead focusing on how to maximize the team's ability to deliver quickly, to respond to emerging requirements, and to adapt to evolving technologies and changes in market conditions

Scrum
A sprint is the basic unit of development in Scrum. The sprint is a timeboxed effort where the length is agreed and fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common.
Each sprint starts with a sprint planning event that establishes a sprint goal and the required product backlog items.
Each sprint ends with a sprint review and sprint retrospective, that reviews progress to show to stakeholders and identify lessons and improvements for the next sprints.

Scrum
There are three roles in the Scrum framework.
● The product owner, representing the product's stakeholders and the voice of the customer, is responsible for delivering good business results.The product owner defines the product in customer-centric terms (typically user stories), adds them to the Product Backlog, and prioritizes them based on importance and dependencies.
● The development team
● The scrum master is not a traditional team lead or project manager but acts as a buffer between the team and any distracting influences. The scrum master ensures that the scrum framework is followed.

UX for connected devices

UX for connected devices
When we think of design for connected products, we tend to focus on the most visible and tangible elements:

● the industrial design
● the user interfaces (UIs) found in mobile and web apps and on the devices themselves.
They are important concerns, which have a major impact on the end user's experience of the product.
But they're only part of the picture.
You could create a beautiful UI, and a stunning piece of hardware, and users could still have a poor experience of the product as a whole.

INTERNET

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

The Internet of Things
The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

THE INDUSTRY 4.0

INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

THE INDUSTRY 4.0

●Industry 4.0 describes the organisation of production processes based on technology and devices autonomously communicating with each other along the value chain: a model of the 'smart' factory of the future where computer-driven systems monitor physical processes, create a virtual copy of the physical world and make decentralised decisions based on self-organisation mechanisms. The concept takes account of the increased digitalisation of manufacturing industries where physical objects are seamlessly integrated into the information network, allowing for decentralised production and real-time adaptation in the future.
●Industry 4.0 was initially developed by the German government to create a coherent policy framework to maintain

Germany’s industrial competitiveness.

From: Industry 4.0 Study for the ITRE Committee - www.europarl.europa.eu/RegData/etudes/STUD/.../IPOL_STU(2016)570007_EN.pdf

THE INDUSTRY 4.0

INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

THE INDUSTRY 4.0

INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

INDUSTRY 4.0

https://www.accenture.com/us-en/_acnmedia/Accenture/next-gen/reassembling-industry/pdf/Accenture-Driving-Unconventional-Growth-through-IIoT.pdf

INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

DIGITAL TWIN

A digital twin is a digital replica of a physical entity.

By bridging the physical and the virtual world, data is transmitted seamlessly allowing the virtual entity to exist simultaneously with the physical entity.

Digital twin refers to a digital replica of physical assets, processes, people, places, systems and devices that can be used for various purposes.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

4.0

HOLISTIC

VISION

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)
https://www.forbes.com/sites/louiscolumbus/2018/06/06/10-charts-that-will-challenge-your-perspective-of-iots-growth/#ce3f6623ecce

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)
THE INTERNET OF THINGS

IOT IS JUST AN ENABLING TECHNOLOGY!

INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)
PRODUCTS AND
SERVICES IN THE 4.0
ERA

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

UBIQUITOUS
TECHNOLOGY
AND BIG DATA

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

4.0 SMART PRODUCTS AND SERVICES

INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

A smart product is a physical device with a digital service at its heart

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)
INDUSTRIAL IOT – MASTER 4.0 2019 - - ALL RIGHTS RESERVED

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

FACTORY ON INTERNET

www.zerynth.com

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

WELLS ON INTERNET

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

NEW DESIGN
PRINCIPLES

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

UX for IOT
Designing for the IoT is inherently more complex than web service design.
Some of this is to do with the current state of the technology.
Some of this reflects our as-yet immature understanding of compelling consumer IoT value propositions.
Some of this stems from the fact that there are more aspects of design to consider. Tackling them independently creates an incoherent user experience (UX).

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

UX for IOT
Designing a great connected product requires a holistic approach to user experience.
It spans many layers of design, not all of them immediately visible.

https://aws.amazon.com/solutions/case-studies/irobot-iot/

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Why UX for IOT is different?
Connected products pose design challenges that will be new to designers accustomed to pure software services.
Many of these challenges stem from:
• The specialized nature of IoT devices

• Their ability to bridge the digital and physical worlds
• The fact that many IoT products are distributed systems of multiple devices
• The quirks of networking

Specialized Devices, with Different Capabilities
Many of the “things” in the Internet of Things are specialized, embedded computing devices. Unlike general-purpose computers (smartphones and PCs), their hardware and software is optimized to fulfill specific functions.
Their physical forms must be designed and engineered.
Their UI capabilities may extend from screens and buttons into physical controls, audio, haptics, gestures, tangible interactions, and more.
But user interactions must be designed without the benefit of the style guides and standards that web and mobile designers can rely upon.
Some may have no user input or output capabilities at all. The only way to find out what they are doing or what state they are in may be via a remote UI.

Real-World Context
Connected products exist in the physical world.
Sensors enable us to capture data we did not have before for digital transmission, allowing us to take more informed actions in the real world.
Actuators provide the capability for digital commands to produce real-world effects.
They can be remotely controlled, or automated. But unlike digital commands, real-world actions often cannot be undone.

https://www.oreilly.com/library/view/user-experience-design/9781492048145/

Real-World Context
The physical context of use creates further challenges.
Devices may need to be ruggedized for outdoor use.
An in-car system needs to be designed to minimize distraction while driving.
A remotely controlled oven needs to minimize the risk of fire.
Devices must adhere to regulatory requirements such as radio interference or waste recycling standards.
And the social context of use may be particularly complex, especially in the home.
Techno-centric solutions which are insensitive to the needs of the occupants will fail. For example, an assisted living product needs to balance the need of vulnerable people
for safety and support, while preserving their privacy and autonomy.

Designing for Systems of Devices and Services
Many connected products are systems of diverse devices and services.
Functionality may be distributed across multiple devices with different capabilities.
Designers need to consider how best to distribute functionality across devices.
They need to design UIs and interactions across the system as a whole — not treating devices as standalone UIs — to ensure that the overall UX is coherent.
This is interusability. Much of the information processing for an IoT product will often happen in the Internet service. So the whole system experience is often equally or more important than any single device UX.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Designing for Systems of Devices and Services
Furthermore, they need some understanding of how the system works.
Even quite simple connected products are conceptually more complex than non-connected ones.
Code can run in more places. Parts of the system will inevitably go offline from time to time.
When this happens, basic knowledge of which component does what will help users understand the consequences, and figure out what action may be required.

Designing for Systems of Devices and Services
In this context, the designers and engineers of connected products should do their best to create a unified environment for the IoT system.
In other words, the challenge of a seamless experience is to integrate diverse independent components into a one-stop solution while saving its functionality and reliability.

Direct vs remote configuration
Over the last 30 years, the prevailing trend in UI design has been direct manipulation.
Users control visual representations of objects and immediately see the outcome of their actions (which can be reversed)
But many IoT interactions are displaced in location (remote control) or time (automation). This breaks the link between user actions and visible, reversible consequences!

Design for Networks
Another major factor is the impact of the network on UX.
Designers from web and mobile software backgrounds have the luxury of assuming that devices will be nearly always connected.
In IOT this is not true anymore!

Design for network
Our experience of the physical world is that things respond to us immediately and reliably.
Light switches do not “lose” our instructions or take 30 seconds to produce an effect.
Delays and glitches are inherent properties of physical networks and transmission protocols.
But they may feel strange experienced through “real-world” things. It’s impossible to engineer these issues entirely out of any Internet-connected system.

Design for power saving
In addition, the nature of connected devices is that they often connect only intermittently, in order to conserve power.
Computers promise to provide us with precise, accurate, and timely data about the world around us.
But distributed IoT systems may not always be in sync, and different devices may therefore report different information about the state of the system.

UX for IOT flow and
architeture

UX for IOT flow and architecture
A good overall product requires integrated thinking across all these layers.
A stunning UI means nothing if your product concept makes no sense.
A beautiful industrial design may sell products in the short term, but can’t mask terrible service.

PRETOTYPING

https://www.pretotyping.org/

http://www.pretotyping.org/

Source: www.pretotyping.org

The law of market failure

Source: www.pretotyping.org

The law of market failure
There is only one way to fight the Law of Market Failure: test the market for your ideas objectively, rigorously, and
quickly before you invest to develop them.
Pretotyping provides you with the tools and techniques you need to validate your idea with minimal resources and
in a very short time (as little as a few hours.)

Source: www.pretotyping.org

The law of market failure in innvoation

Source: www.pretotyping.org

8

Source: www.pretotyping.org

9

Source: www.pretotyping.org

THOUGHTLAND

10

Source: www.pretotyping.org

THOUGHTS WITHOUT DATA ARE JUST OPINIONS

11

Source: www.pretotyping.org

The Pretotyping Manifesto

innovators beat ideas

pretotypes beat productypes
building beats talking
simplicity beats features

now beats later

commitment beats committees

data beats opinions

don’t finish what you’ve started

failure is an option
scarcity bring clarity
the more the messier
reinvent the wheel

play with fire

12

13

Pretotype

14

Pretotype

Pretotype

16

The Seven Pillars of PRETOTYPING
1. Obey the Law of Market Failure.
2. Make sure you are building The Right It.
3. Don't get lost in Thoughtland.
4. Trust only in Your Own DAta (YODA).
5. Pretotype It.
6. Say it with numbers.
7. Think global, test local.

17

PRETOTYPING FLOW
Step 1: Isolate the Key Assumption: What is the one assumption about your idea that, if false, means it's
definitely not the right it?
Step 2: Choose a Type of Pretotype: What type of pretotype will let you to isolate and test your key
assumption?
Step 3: Make a Market Engagement Hypothesis: How many (and what kind of) people will do what
with your pretotype? Your hypothesis can be as simple as: X% of Y will do Z -->A solid hypothesis takes the guesswork and
opinion out of testing.
https://www.youtube.com/watch?v=4sZMHAMN0DQ&ab_channel=TheRightIt%E2%80%94VideolessonsbyAlbertoSavoia
Step 4: Test Your Pretotype: Now put your pretotype into the real world, and see how people interact with it.
Start small — one place, one time.
Step 5:Learn, Refine, Hypozoom: Evaluate your results. Refine your pretotype with your new data. If you
hypothesis held, decide what other situations you should test your pretotype in to get a complete picture (what we call
"hypozooming").
https://www.albertosavoia.com/uploads/1/4/0/9/14099067/hypozooming_video.pdf
https://www.youtube.com/watch?v=bKfBbYsJlZc&ab_channel=TheRightIt%E2%80%94VideolessonsbyAlbertoSavoia

Source: www.pretotyping.org 18

FAKE DOOR

19
the Fake Door can be used to advertise a service that is not ready yet and measure interest from users, e.g. a new
process to renew a license, a new expert system to consolidate social services, etc.
https://www.nesta.org.uk/blog/development-and-testing-for-public-labs-fake-it-before-you-make-it/

FAKE DOOR

20

Source: www.pretotyping.org

21

https://hackernoon.com/fake-door-the-mvp-before-the-mvp-61197ed264a3

22
Algorithmic-based solutions are needed to reach scale while keeping the cost low. But before finding the right
"advisor" to build, a Mechanical Turk approach can be used to experiment, where human experts can hide behind an
online form or a SMS-based application.
https://www.nesta.org.uk/blog/development-and-testing-for-public-labs-fake-it-before-you-make-it/

MECHANICAL TURK

23

Source: www.pretotyping.org

24

25
In the context of public labs, this could mean revamping an existing service by providing a different front-end on top of
an existing API.
https://www.nesta.org.uk/blog/development-and-testing-for-public-labs-fake-it-before-you-make-it/
Video “IKEA” Design studio Upwell hacks IKEA... on Vimeo

IMPERSONATOR

26

Source: www.pretotyping.org https://vimeo.com/79313674

27
A Pinocchio pretotype is one in which a fake artifact acts as a proxy for the real thing. The most famous example is the
wood model of the Palm Pilot mentioned above.

Pinocchio pretotypes are the perfect conduit for role play design to encourage people to test your product or service,
but also to explore other ways that the "it" you are building is tailored to their needs.

PINOCCHIO

28

Source: www.pretotyping.org

29
The goal here is to reduce cost or to target a well-known population. In the spirit of failing fast, if the One Night Stand
pretotype applied to a friendly region and receptive population does not succeed, then you know that you are probably
building the wrong "it".
https://www.nesta.org.uk/blog/development-and-testing-for-public-labs-fake-it-before-you-make-it/

ONE NIGHT STAND

30

Source: www.pretotyping.org

31

https://www.youtube.com/watch?v=9ko6oFb-jQY

NEXT STEP... PRODUCTION?

32

MINIMUM VIABLE PRODUCT

33

Source: www.pretotyping.org

34
Source: www.pretotyping.org

Pretotyping tools
● WebsiteWeebly.com: Particularly useful for fake door pretotypes to test using Google/Facebook ads.
● Omnigraffle: Shares across multiple pages and spits out clickable
● Appery - free, cloud-based app platform - http://appery.io
● UX prototype https://www.figma.com/
● https://careerfoundry.com/en/blog/ux-design/prototyping-tools/
● And even more drag and drop app builders:
○ https://codiqa.com/
○ http://www.kinvey.com/
○ http://cloudbase.io/
○ http://mobile.conduit.com/
○ http://mobileroadie.com/
○ http://www.theappbuilder.com/
● Marketplace builder https://www.shopify.com/

35

36

https://blog.adioma.com/how-angry-birds-started-infographic/#:~:text=The%20startup%20behind%20the%20game,are%20launching%20their%2051st%20game.

Alberto Savoia Pretotyping Lecture @ Stanford
https://www.youtube.com/watch?v=3sUozPcH4fY&feature=emb_logo

Prototyping and mockups

roberto.figlie@phd.unipi.it

Lesson outline

- Mockups
- Prototypes
  - Fidelity
  - Scope
  - Techniques
- Mockups vs Wireframes vs Prototypes
- From Wireframes to Prototypes

- Figma basics

Mockups

Mockups

A mockup is a high-fidelity representation of the product that includes more visual detail and design elements than a wireframe: it is a more polished and refined version and it is used to test and refine the look and feel.
A realistic visual design that resembles what the new product or functionality will look like.

Mockups

Creating a mockup involves adding visual details and design elements to a wireframe, such as typography, color schemes, and images. Mockups can be created only digitally using design software, such as Sketch, Adobe XD (or Photoshop, just like the good old times), or Figma.
So, mockups can be used to refine and showcase visual design elements, from the placement of images and icons to the overall style and branding of a product or service. But is essential to note that while mockups may resemble the final product, they lack the functionality required to perform meaningful testing or gather insightful feedback. This is because mockups are static representations of the design and

lack the interactive elements that are necessary for usability testing.

Mockups

When and why?
UX/UI teams commonly rely on mockups during the design process to showcase their ideas to stakeholders and clients. These mockups serve as visual aids that help UX professionals communicate their design concepts and features more effectively. In addition, mockups are helpful for certain types of usability studies where feedback is needed on the overall look and feel of the design (5-seconds test, first-click test).

Prototypes

What is a Prototype?

A prototype is an experimental and early sample of a design that allows users to visualize and interact with it before a final product is developed. Teams build prototypes of varying degrees of fidelity to capture design concepts and test on users.

So, prototyping is the process in which design teams ideate, experiment with, and bring concepts to life. Is the phase in which design teams implement ideas into tangible forms from paper to digital.
What is Prototyping? | IxDF (interaction-design.org)

Why do we need a Prototype?

We never (or at least rarely) get things right the first time!
Prototyping allows designers to test their ideas on users and refine their designs based on feedback. This can help save time and resources by identifying potential issues early on.
With prototypes, you can validate your designs so to release the right product.

Why do we need a Prototype?

Prototyping is an essential part of UX design that usually comes after ideation, where you/your team have created and selected ideas that can solve users' needs.
Some advantages of creating a prototype include:
● Providing a solid foundation for ideation and giving stakeholders a clear picture of potential benefits, risks and costs.
● Changes can be made early in the process to avoid costly oversights and commitment to a single version.
● User feedback on the prototype helps identify what works best and if an overhaul is needed.
● Prototyping allows for experimentation with user needs and problems to gain insights into less obvious areas.
● Stakeholders have a sense of ownership and emotional investment in the product's success.
● Time-to-market is improved by minimizing errors before product release.

Prototypes

"They slow us down to speed us up. By taking the time to prototype our ideas, we avoid costly mistakes such as becoming too complex too early and sticking with a weak idea for too long."

— Tim Brown, CEO & President of IDEO

Prototypes

In summary:
A prototype is a design solution candidate that addresses specific problems/needs.
With it, usability problems can be identified before committing to the development of the final design.

How do we do a Prototype?

The prototyping process involves selecting key features to test, creating a design prototype that exemplifies those features, and then testing it on users. There are many different methods for prototyping, ranging from paper

sketches to digital versions.

prototype fidelity,

scope and
techniques

Prototypes fidelity

How much a prototype resembles the final product (in look, feel and function) is a matter of fidelity.
Generally there are 2 types of fidelities (plus another one).

Low-fidelity prototypes

Pros:
● Usually done with pen & paper (cheap & disposable).
● Connects the low-fidelity wireframes between them.
● Can provide very simple interactions (if not any) and feel.
● Done in less time.
● Easier to modify (more iterations).
● Encourages Design Thinking.

Cons:
● Lack of realism (difficult for users to give feedback).
● Results from early versions may be hard to apply.
● May be too basic to reflect the user experience of the finished product.
● Can oversimplify complex issues.
● Lack of interactivity deprives users of direct control.
● Users must imagine how they would use the product.

https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/
https://www.interaction-design.org/literature/topics/prototyping

Low-fidelity prototypes

https://xd.adobe.com/ideas/process/prototyping/low-fi-and-hi-fi-prototyping/

High-fidelity prototypes

Pros:
● Can be coded or developed in a specific software (Figma, XD, Sketch).
● Gives a final-like feel.
● You can test more elements (interactions, visual, text readability, etc.).
● Engaging for all stakeholders.
● Allows stakeholders to judge how well the prototype matches users' needs and solves their problems
● Testing yields more accurate and applicable results.

Cons:
● Longer and costlier to create.
● Users may focus on superficial details rather than content.
● Designers may dislike making changes after hours of work.
● Users may mistake the prototype for the finished product and form biases.

https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/
https://www.interaction-design.org/literature/topics/prototyping

High-fidelity prototypes

https://www.justinmind.com/blog/low-fidelity-vs-high-fidelity-prototypes/

Finding the right balance: Mid-fidelity prototypes

Another possibility is given by mid-fidelity prototypes where you keep some of the benefits of using Low-fidelity and high-fidelity ones.
An example of a mid-fidelity prototype could be an advanced wireframe with some digital interactivity (i.e., adding interactions to wireflows).
However, you have to remember that the concept of fidelity is relative: you have to choose wisely what is best for you depending on the stage of design and features to test.

Mid-fidelity prototypes

https://medium.com/the-home-team/homeguide-prototype-portfolio-a8746296e958

6 dimensions of prototypes' fidelity

The fidelity of a prototype and the activity of prototyping mainly moves on these main 6 dimensions:
1. Visual/Physical realism: How closely the prototype resembles the final product in terms of visual design and aesthetics.
2. Scope: The breadth and depth of the design represented in the prototype.
3. Functionality: What actually works in the prototype, such as links or buttons in a web app.
4. Data: Whether the prototype operates on real or fake data.
5. Autonomy: Whether the prototype can operate alone or requires input from the designer or user.
6. Platform: Whether the prototype is an interim or final implementation.

Scope of prototypes

Say you have decided to test your design and its functionalities through an mid-fidelity prototype.
This gives you information about how good the prototype will look and feel. But how much the user will be able to see and test?
This is a matter of scope of the prototype. There are two possible scopes:

1. Horizontal
2. Vertical

Scope of prototypes: Horizontal

An horizontal prototype gives an entire overview of the system, although not going too much in detail for each feature: you display a wide range of features without actually implementing them. This means that the prototype is constrained to only one layer.
It is useful in the early stages of design as it allows stakeholders to quickly identify potential issues and make design decisions based on feedback from users.

Scope of prototypes: Vertical

A vertical prototype gives a detailed view of a specific feature: you have only one or few features actually designed. This means that the prototype delivers the system and its subsystems.

In this way you can test the technical feasibility of the product or that a specific requirement is fulfilled by the user (in the interaction with the system).

Prototyping techniques

Paper prototypes

A paper prototype is a low-fidelity prototype created on paper or cardboard that

represents the basic layout and functionality of a product or service. It's a quick and inexpensive way to test and refine user interface designs before investing time and resources in more high-fidelity prototypes.

Paper prototypes

Creating a paper prototype involves sketching out the basic layout and functionality of a website or app on paper or cardboard. Users can then interact with the prototype by physically manipulating the paper or cardboard, simulating how they might interact with a digital interface. Paper prototypes can be used to quickly test a wide range of design elements, from the placement of buttons and menus to the flow of information and user interactions. They are particularly useful in the early stages of the design process, when designers are still exploring different design options and gathering feedback from users.

(Mid-fi) Wireframe prototypes

A wireframe prototype is a digital prototype that represents the basic layout and functionality of a product or service, without much detail or visual design. It's a quick and inexpensive way to test and refine the basic structure of a product or service, without getting bogged down in visual details.

(Mid-fi) Wireframe prototypes

Creating a wireframe prototype involves sketching out the basic layout and functionality of a product or service, using simple shapes and symbols to represent different UI elements.

Wizard of Oz (or Mechanical Turk) prototypes

A Wizard of Oz is a prototype that simulates the behavior of a product or service using human input instead of actual technology. It's used to test the user experience of a product or service before actual technology is developed.

https://en.wikipedia.org/wiki/Mechanical_Turk

Wizard of Oz prototypes

Creating a Wizard of Oz involves using a human operator or team of operators to simulate the behavior and responses of a product or service. This can involve using pre-recorded audio or video, scripted responses, or even live interactions with users. The goal is to simulate the experience of using a product or service without actually building the technology behind it.

IBM voice editor (1984)

Functional prototypes

A functional prototype is a prototype that includes working or partially working features. It is used to test and refine the technical aspects of a product, such as the responsiveness and functionality of different features.

Functional prototypes

Creating a functional prototype involves building a prototype with actual code and programming, or using a prototyping tool that allows for interactive elements and functionality.
They are particularly useful in the later stages of the design process, when designers are looking to fine-tune the technical details of their designs before moving on to full development.

Non-Functional prototypes

A non-functional prototype is a prototype that looks and feels like a real product or system but does not have any functional capabilities. It is typically used to demonstrate the physical design and user interface of a product or system, without the need for any actual working components or systems.
They are typically used to test the design and user experience of a product or system before investing resources into building a functional prototype. A little coding can be still necessary, but without going too much in detail and without real working functionalities or at least just superficially: for example, you have the website and its navigation working but without an actual backend service.
https://www.youtube.com/watch?v=2CTMJxWh5w4

Non-Functional prototypes

There are several ways that non-functional prototypes can be used for testing:
1. Testing User Experience: can be used to test the user experience (UX) of a product or system. This involves testing how users interact with the prototype, how easy it is for them to navigate and use, and whether it meets their needs and expectations.

2. Testing Visual Design: can also be used to test the visual design as an advanced mockup with some interaction. This involves testing the aesthetics, layout, and overall look and feel of the prototype, and whether it is visually appealing and consistent with the brand identity.

Clickable or static prototype?

Wireframes

vs
Mockups
vs
Prototypes

Wireframes vs Mockups vs Prototypes

People often confuse the concept of mockups with that of prototypes. However, prototypes are more advanced versions of mockups that include navigation and functionality, allowing for more meaningful testing and feedback from users.
Designers usually use mockups and wireframes to first create prototypes that simulate the functionality and user experience of the final product.
The prototypes are then used to test and validate the design before moving on to the development phase.

Wireframes vs Mockups vs Prototypes

Has visual
design,
colors,
typography
and
images?

Can respond to
actions in
other ways ?

Is it
clickable and
interactive?

Functional
Prototype

Functional
Prototype Mockup Wireframe

NO NO NO

YES YES YES

From wireframe to mockups to prototypes

Lo-fi wireframe

Paper prototype

Hi-fi wireframe

Mockup

early design

Mid-fi prototype

late design

Hi-fidelity
prototype

Working System

Prototyping tools

Tool Main Features Platforms Price

Figma

Collaborative design tool,
Vector Networks, Auto Layout,
Prototyping, Developer
Handoff, Version History

Web-based

Free for individuals (limited
features), 12€/month per
editor for Professional plan
(free for students!)

Sketch

Vector-based design tool,
Libraries, Prototyping,
Plugins, Handoff

macOS 9€/month per editor

Adobe XD

Vector-based design tool,
Auto-Animate, Prototyping,
Plugins, Developer Handoff,
Voice Prototyping,
Collaboration

Windows, macOS

12.19€/month per editor for
Single App plan

InVision

Design System Manager,
Prototyping, Collaboration,
User Testing, Animations,
Inspect

Web-based

Free for individuals (limited
features), $4.95/month per
user for Starter plan (for
collaborative teams)

and others: Axure, Marvel, Principle, Flinto, ProtoPie,
Proto.io, UX Pin, ...

Figma

Introduction - What is Figma?

Figma is a web app that helps designers build
wireframes, mockups, prototypes and so on.

Some advantages:
● The professional plan is free for students.
● Many collaboration features.
● Plugins and community.

https://help.figma.com/hc/en-us/categories/360002051613-Get-started

Layers on FIgma

Layers work differently on Figma respect Illustrator or other drawing software you may know.
But understanding how to use them is simple:
● Every object in the scene is a layer.
● If you group many objects together they make a layer.
● If you place them in a frame, the frame itself is a layer with many sub-layers.
In addition to this you can use pages to better organize your workplace.

Some basic but useful shortcuts

When moving objects:
● Shift + mouse move: snap to other objects.
● Alt + mouse move: copy and paste object (on release).
● You can use them together.
When resizing objects:
● Shift + drag borders: resize maintaining aspect ratio.
● Alt + drag borders: resize maintaining center position.
● You can use them together.

Human Interface Devices

(HID)

Human Interface Devices
A human interface device or HID is a type of computer device usually used by humans.
HID takes input from humans and gives output to humans.
The term "HID" is used to indicate the physical devices and also the USB-HID specification.
The term was coined by Mike Van Flandern of Microsoft when he proposed that

the USB committee create a Human Input Device class working group. The working group was renamed as the Human Interface Device class at the suggestion of Tom Schmidt of DEC because the proposed standard supported bi-directional communication.

HID software specification

HID (software specification)
The HID standard was adopted primarily to enable innovation in PC input devices and to simplify the process of installing such devices.
Prior to the introduction of the HID concept, devices usually conformed to strictly defined protocols for mouse, keyboards and joysticks; for example, the standard mouse protocol at the time supported relative X- and Y-axis data and binary input for up to two buttons, with no legacy support.
All hardware innovations necessitated either overloading the use of data in an existing protocol or the creation of custom device drivers and the evangelization of a new protocol to developers. By contrast, all HID-defined devices deliver self-describing packages that may contain any number of data types and formats.

HID (software specification)
A single HID driver on a computer parses data and enables dynamic association of data I/O with application functionality, which has enabled rapid innovation and development, and prolific diversification of new human-interface devices.
The HID protocol has its limitations, but all modern mainstream operating systems will recognize standard USB HID devices, such as keyboards and mice, without needing a specialized driver. When installed, a message saying that "A 'HID-compliant device' has been recognized" generally appears on screen.
In comparison, this message does not usually appear for devices connected via the PS/2 6-pin DIN connectors which preceded USB. the PS/2 standard does not support the HID protocol.
The USB human interface device class describes a USB HID.

HID (software specification)
In the HID protocol, there are 2 entities: the "host" and the "device".
The device is the entity that directly interacts with a human, such as a keyboard or mouse.
The host communicates with the device and receives input data from the device on actions performed by the human. Output data flows from the host to the device and then to the human.

HID (software specification)
The HID protocol makes implementation of devices very simple.
Devices define their data packets and then present a "HID descriptor" to the host.
The HID descriptor is a hard coded array of bytes that describes the device's data packets.
This includes:
● how many packets the device supports,
● the size of the packets,
● the purpose of each byte and bit in the packet.
For example, a keyboard with a calculator program button can tell the host that the button's pressed/released state is stored as the 2nd bit in the 6th byte in data packet number 4

HID (software specification)
The device typically stores the HID descriptor in ROM and does not need to intrinsically understand or parse the HID descriptor. Some mouse and keyboard hardware in the market today is implemented using only an 8-bit CPU.
The host is expected to be a more complex entity than the device. The host needs to retrieve the HID descriptor from the device and parse it before it can fully communicate with the device.
Parsing the HID descriptor can be complicated. Multiple operating systems are known to have shipped bugs in the device drivers responsible for parsing the HID descriptors years after the device drivers were originally released to the public.
However, this complexity is the reason why rapid innovation with HID devices is possible.

HID (software specification)
The above mechanism describes what is known as HID "report protocol".
Because it was understood that not all hosts would be capable of parsing HID descriptors, HID also defines "boot protocol". In boot protocol, only specific devices are supported with only specific features because fixed data packet formats are used.
The HID descriptor is not used in this mode so innovation is limited.
However, the benefit is that minimal functionality is still possible on hosts that otherwise would be unable to support HID.
The only devices supported in boot protocol are: Keyboard and Mouse
https://makeymakey.com/

Other protocols using HID
Since HID's original definition over USB, HID is now also used in other computer

communication buses.
This enables HID devices that traditionally were only found on USB to also be used on alternative buses. This is done since existing support for USB HID devices can typically be adapted much faster than having to invent an entirely new protocol to support mouse, keyboards, and the like.
Known buses that use HID are:
● Bluetooth HID – Used for mouse and keyboards that are connected via Bluetooth
● Serial HID – Used in Microsoft's Windows Media Center PC remote control receivers.
● ZigBee input device – ZigBee (RF4CE) supports HID devices through the ZigBee input device profile.
● HID over I2C – Used for embedded devices in Microsoft Windows 8[2]
● HOGP (HID over GATT) – Used for HID devices connected using Bluetooth low energy technology

HID Devices

HID (peripheral)
HID peripherals can be organized in two main categories:
● Input devices
● Output devices

Input devices are based on sensors (a device, module, or subsystem whose purpose is to detect events or changes in the physical world its environment and convert it in analog or digital electronic information)
Output devices are based on actuators (a device, module, or subsystem whose purpose is to convert analog or digital electronic signals in physical events aimed at changing the physical)

HID classification (old fashion)
Input or Output HID are typically divided in classes according to the type of input/output used by the HID:
● Texts and chars
● Positions
● Sound
● Images
● Environmental parameters
● Position
● Health/bio/physiological parameters
Nowadays most of the novel HIDs use mixed technology so it is difficult to still classify them on the basis of what they sense...

Input Devices

Keyboards
The most used text and chars HID is the keyboard
Different types of keyboards are available and each is designed with a focus on specific features that suit particular needs. Today, most full-size keyboards use one of three different mechanical layouts, usually referred to as simply ISO (ISO/IEC 9995-2) ANSI standard.
ANSI standard alphanumeric keyboards have keys that are on three-quarter inch centers (0.75 inches (19 mm)), and have a key travel of at least 0.15 inches (3.8 mm).
Modern keyboard models contain a set number of total keys according to their given standard, described as 101, 104, 105, etc. and sold as "Full-size" keyboards.

Keyboard layouts
A keyboard layout is any specific physical, visual or functional arrangement of the keys, legends, or key-meaning associations (respectively) of a computer keyboard, mobile phone, or other computer-controlled typographic keyboard.
Physical layout is the actual positioning of keys on a keyboard.
Visual layout the arrangement of the legends (labels, markings, engravings) that appear on those keys.
Functional layout is the arrangement of the key-meaning association or keyboard mapping, determined in software, of all the keys of a keyboard: this (rather than the legends) determines the actual response to a key press.

Keyboard layouts
Modern computer keyboards are designed to send a scancode to the operating system (OS) when a key is pressed or released: this code reports only the key's row and column, not the specific character engraved on that key.
The OS converts the scancode into a specific binary character code using a "scancode to character" conversion table, called the keyboard mapping table. This means that a physical keyboard may be dynamically mapped to any layout without switching hardware components – merely by changing the software that interprets the keystrokes.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

QWERTY Layout

The QWERTY layout far the most widespread layout in use, and the only one that is not confined to a particular geographical area.
QWERTY is a keyboard layout design for Latin-script alphabets.
The name comes from the order of the first six keys on the top left letter row of the keyboard (Q W E R T Y)

Multifunctional keyboards
Provide additional function beyond the standard keyboard.
Many are programmable, configurable computer keyboards and some control multiple PCs, workstations and other information sources, usually in multi-screen work environments.
Users have additional key functions as well as the standard functions and can typically use a single
keyboard and mouse to access multiple sources.
Multifunctional keyboards may feature customised keypads, fully programmable function or soft
keys for macros/pre-sets, biometric or smart card readers, trackballs, etc.
New generation multifunctional keyboards feature a touchscreen display to stream video, control
audio visual media etc.
Common environments for multifunctional keyboards are complex, high-performance workplaces
for financial traders and control room operators (emergency services, security, air traffic management; industry, utilities management, etc.).

Barcode readers
A barcode reader (or barcode scanner) is an optical scanner that can read printed barcodes, decode the data contained in the barcode and send the data to a computer.
Like a flatbed scanner, it consists of a light source, a lens and a light sensor translating for optical impulses into electrical signals.
Additionally, nearly all barcode readers contain decoder circuitry that can analyze the barcode's image data provided by the sensor and sending the barcode's content to the scanner's output port.

Barcode Readers

A barcode is a method of representing data in a visual, machine-readable form.
Standard, barcodes represent data by varying the widths and spacings of parallel lines. A barcode encodes a string (typically numbers)
A QR is a bidimensional barcode. It uses four standardized encoding modes (numeric, alphanumeric, byte/binary, and kanji) to store data efficiently;

RFID
An RFID tag consists of a tiny radio transponder; a radio receiver and transmitter. When triggered by an electromagnetic interrogation pulse from a nearby RFID reader device, the tag transmits digital data, usually an identifying inventory number, back to the reader. This number can be used to inventory goods.
There are two types:
● Passive tags are powered by energy from the RFID reader's interrogating radio waves.
● Active tags are powered by a battery and thus can be read at a greater range from the RFID reader; up to hundreds of meters.
Unlike a barcode, the tag doesn't need to be within the line of sight of the reader, so it may be embedded in the tracked object.

NFC
Near-Field-Communication (NFC) is a set of communication protocols for bidirectional communication between two electronic devices over a distance of 4 cm (1 1⁄2 in) or less.
NFC offers a low-speed connection with simple setup that can be used to bootstrap more-capable wireless connections.

Is it a HID or a communication technology?

Pointing Devices
A pointing device is an input interface that allows a user to input spatial (i.e., continuous and multi-dimensional) data to a computer.
CAD systems and graphical user interfaces (GUI) allow the user to control and provide data to the computer using physical gestures by moving a hand-held mouse or similar device across the surface of the physical desktop and activating switches on the mouse.
Movements of the pointing device are echoed on the screen by movements of the pointer (or cursor) and other visual changes.
Common gestures are point and click and drag and drop.

Pointing Devices
While the most common pointing device by far is the mouse, many more devices have been developed. However, the term "mouse" is commonly used as a metaphor for devices that move the cursor.

Fitts's law
Fitts's law is a predictive model of human movement primarily used in human–computer interaction and ergonomics.
This scientific law predicts that the time required to rapidly move to a target area is a function of the ratio between the distance (D) to the target and the width of the target (W).
Fitts's law is used to model the act of pointing, either by physically touching an object with a hand or finger, or virtually, by pointing to an object on a computer monitor using a pointing device.

Fitts's law
Movement Time:
● a = time to start/stop in seconds (empirically measured per device)
● b = inherent speed of the device (empirically measured per device)
● D is the distance from the starting point to the center of the target.
● W is the width of the target measured along the axis of motion.

Notice that because the ID term depends only on the ratio of distance to width, the model implies that a target distance and width combination can be re-scaled arbitrarily without affecting movement time, which is impossible. Despite its flaws, this form of the model does possess remarkable predictive power across a range of computer interface modalities and motor tasks, and has provided many insights into user interface design principles.

Fitts's law
Proven to provide good timings for most age groups
Newer versions taken into account
● Direction (we are faster horizontally than vertically)
● Device weight
● Target shape
● Arm position (resting or midair)
● 2D and 3D (Zhai ’96)
● Zero gravity environment

Pointing Devices classification
direct vs. indirect input

In case of a direct-input pointing device, the on-screen pointer is at the same physical position as the pointing device (e.g., finger on a touch screen, stylus on a tablet computer).
An indirect-input pointing device is not at the same physical position as the pointer but translates its movement onto the screen (e.g., computer mouse, joystick, stylus on a graphics tablet).

Pointing Devices classification
absolute vs. relative movement
An absolute-movement input device (e.g., stylus, finger on touch screen) provides a consistent mapping between a point in the input space (location/state of the input device) and a point in the output space (position of pointer on screen).
A relative-movement input device (e.g., mouse, joystick) maps displacement in the input space to displacement in the output state. It therefore controls the relative position of the cursor compared to its initial position.

Pointing Devices classification
isotonic vs. elastic vs. isometric
An isotonic pointing device is movable and measures its displacement (mouse, pen, human arm) whereas an isometric device is fixed and measures the force which acts on it (trackpoint, force-sensing touch screen). An elastic device increases its force resistance with displacement (joystick).

Pointing Devices classification
position control vs. rate control
A position-control input device (e.g., mouse, finger on touch screen) directly changes the absolute or relative position of the on-screen pointer. A rate-control input device (e.g., trackpoint, joystick) changes the speed and direction of the movement of the on-screen pointer.

What we know
● Direct pointing is Faster but less accurate than indirect (Haller ’84)
● Lots of studies confirm mouse is best for most tasks for speed and accuracy: Trackpoint < Trackballs & Touchpads < Mouse
● For short distances cursor keys are better than pointing devices
● Disabled prefer joysticks and trackballs:
○ If force application is a problem, then touch sensitive is preferred
○ Vision impaired have problems with most pointing devices
● Use multimodal approach or customizable cursors improve usability and performances

● Keep in mind Fits' law:
○ Large targets reduce time and frustration;
○ Designers should smooth out and reduce trajectories

The father of all pointing devices
A computer mouse is a hand-held pointing device that detects two-dimensional motion relative to a surface.
This motion is typically translated into the motion of a pointer on a display, which allows a smooth control of the graphical user interface of a computer.

The mouse evolution
In 2000, Logitech introduced a "tactile mouse" that contained a small actuator to make the mouse vibrate. Such a mouse can augment user-interfaces with haptic feedback (output), such as giving feedback when crossing a window boundary.
Other modern pointing devices have
extended the input dimensions up to 6 DOF

Novel Pointing Devices

Eye Tracking
Eye tracking is the process of measuring either the point of gaze (where one is looking) or the motion of an eye relative to the head.
An eye tracker is a device for measuring eye positions and eye movement.
Eye trackers are used in research on the visual system, in psychology, in psycholinguistics, marketing, as an input device for human-computer interaction, and in product design.
Eye trackers are also being increasingly used for rehabilitative and assistive applications (related for instance to control of wheel chairs, robotic arms and prostheses).
There are a number of methods for measuring eye movement. The most popular variant uses video images from which the eye position is extracted.

Eye Tracking
Light, typically infrared, is reflected from the eye and
sensed by a video camera or some other specially
designed optical sensor.
The information is then analyzed to extract eye rotation
from changes in reflections.
Video-based eye trackers typically use the corneal

reflection and the center of the pupil as features to
track over time.

Eye tracking methods
The most widely used current designs are video-based eye-trackers.
A camera focuses on one or both eyes and records eye movement. Most modern eye-trackers use the center of the pupil and infrared/near-infrared non-collimated light to create corneal reflections (CR).
The vector between the pupil center and the corneal reflections can be used to compute the point of regard on surface or the gaze direction. A simple calibration procedure of the individual is usually needed before using the eye tracker.
Two general types of infrared/near-infrared (also known as active light) eye-tracking techniques are used: bright-pupil (top) and dark-pupil (center). Their difference is based on the location of the illumination source with respect to the optics.
Another, less used, method is known as passive light (bottom). It uses visible light to illuminate

Eye tracking methods
If the illumination is coaxial with the optical path, then the eye acts as a retroreflector as the light reflects off the retina creating a bright pupil effect similar to red eye. If the illumination source is offset from the optical path, then the pupil appears dark because the retroreflection from the retina is directed away from the camera.
Bright-pupil tracking creates greater iris/pupil contrast, allowing more robust eye-tracking with all iris pigmentation, and greatly reduces interference caused by eyelashes and other obscuring features. It also allows tracking in lighting conditions ranging from total darkness to very bright.
Bright-pupil tracking is more reliable but require more complex hardware setup

Eye-tracking vs. gaze-tracking
Eye-trackers necessarily measure the rotation of the eye with respect to some frame of reference. This is usually tied to the measuring system. Thus, if the measuring system is head-mounted, as with video-based system mounted to a helmet, then eye-in-head angles are measured.
To deduce the line of sight in world coordinates, the head must be kept in a constant position or its movements must be tracked as well. In these cases, head direction is added to eye-in-head direction to determine gaze direction.
If the measuring system is table-mounted, as with table-mounted camera (“remote”) systems, then gaze angles are measured directly in world coordinates. Typically, in these situations head movements are prohibited.

Some results are available on human eye movements under natural conditions where head movements are allowed as well.

Eye Tracking mouse
https://www.youtube.com/watch?v=oSo8fbZfHLk&ab_channel=TobiiGaming
https://www.youtube.com/watch?v=4fvdBhPdhIU&ab_channel=TobiiPro

Data Glove
A wired glove (also called a "dataglove") is an input device for human–computer interaction worn like a glove.
Various sensor technologies are used to capture physical data such as bending of fingers. Often a motion tracker, such as a magnetic tracking device or inertial tracking device, is attached to capture the global position/rotation data of the glove.
These movements are then interpreted by the software that accompanies the glove, so any one movement can mean any number of things.
Gestures can then be categorized into useful information, such as to recognize sign language or other symbolic functions.

Haptic Devices
Haptic devices (or haptic interfaces) are mechanical devices that mediate communication between the user and the computer. Haptic devices allow users to touch, feel and manipulate three-dimensional objects in virtual environments and tele-operated systems.
Most common computer interface devices, such as basic mice and joysticks, are input only devices, meaning that they track a user's physical manipulations but provide no manual feedback. As a result, information flows in only one direction, from the peripheral to the computer.
Haptic devices are input-output devices, meaning that they track a user's physical manipulations (input) and provide realistic touch sensations coordinated with on-screen events (output).

Haptic Devices

Smart Paper, Whiteboards and similars
A novel generation of devices aimed at digitizing user

interaction with paper and
whiteboard have been developed
in the last years.
These devices digitize the user
writing by means of tracked
smart pens and/or sensorized
surfaces

This is a cross categories interface where
pointing and images are used as blended
inputs

Speech and Auditory Input Interfaces

Audio and sound acquisition
In physics, sound is a vibration that propagates as an acoustic wave, through a transmission medium such as a gas, liquid or solid.
In human physiology and psychology, sound is the reception of such waves and their perception by the brain.
Only acoustic waves that have frequencies lying between about 20 Hz and 20 kHz, the audio frequency range, elicit an auditory percept in humans.
In air at atmospheric pressure, these represent sound waves with wavelengths of 17 meters to 1.7 centimetres.
Sound waves above 20 kHz are known as ultrasound and are not audible to humans. Sound waves below 20 Hz are known as infrasound. Different animal species have varying hearing ranges.

Audio and sound acquisition
Sound can be acquired using microphones.
A microphone is a device – a sensor– that converts sound into an electrical signal.
Microphones are widely used as HID
Several types of microphone are used today, which employ different methods to convert the air pressure variations of a sound wave to an electrical signal.

Microphone arrays
A microphone array is any number of microphones operating in tandem.
The are used for:
● Systems for extracting voice input from ambient noise (notably telephones, speech recognition systems, hearing aids)
● Surround sound and related technologies

● Binaural recording
● Locating objects by sound: acoustic
source localization
● High fidelity original recordings
● Environmental Noise Monitoring

Microphone Arrays
Typically, an array is made up of omnidirectional microphones, directional microphones, or a mix of omnidirectional and directional microphones distributed about the perimeter of a space
All microphones are linked to a computer that records and interprets the results into a coherent form.
Arrays may also be formed using numbers of very closely spaced microphones. Given a fixed physical relationship in space between the different individual microphone transducer array elements, simultaneous DSP (digital signal processor) processing of the signals from each of the individual microphone array elements can create one or more "virtual" microphones.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Speech and Auditory Interfaces
There’s the dream.... Then there’s reality
Practical apps don’t really allows freeform discussions with a computer :(
Main Design Goals:
● Low cognitive load
● Low error rates
● Natural user experience

Speech and Auditory Interfaces
Problem and limitations:

● Bandwidth is much lower than visual displays
● Ephemeral nature of speech (tone, etc.)
● Difficulty in parsing/searching → higher computational load
https://www.youtube.com/watch?v=IKZToY-V16w&ab_channel=EliyahuShatz

Speech and Auditory Interfaces
Succeed With:
Specialized vocabularies (like medical or legal)
● Dictate reports, notes, letters

● Communication skills practice (virtual patient)
● Automatic retrieval/transcription of audio content (like radio, CC)
● Security/user ID
https://www.youtube.com/watch?v=ZLpuYQ401s8&ab_channel=JonWahrenberger

Image based Input User Interfaces

Image Sensors
An image sensor is a sensor that detects and conveys information used to make an image.
It does so by converting the variable attenuation of light waves (as they pass through or reflect off objects) into signals, small bursts of current that convey the information.
The waves can be light or other electromagnetic radiation.
Image sensors are used in electronic imaging devices of both analog and digital types, which include digital cameras, camera modules, camera phones, optical mouse devices, medical imaging equipment, night vision equipment such as thermal imaging devices, and others.

Image Sensors
The two main types of electronic image sensors are the charge-coupled device (CCD) and the active-pixel sensor (CMOS sensor).
Both CCD and CMOS sensors are based on metal–oxide–semiconductor (MOS) technology, with CCDs based on MOS capacitors and CMOS sensors based on MOSFET (MOS field-effect transistor) amplifiers.
Analog sensors for invisible radiation tend to involve vacuum tubes of various kinds.

3D image capture device
3D scanning is the process of analyzing a real-world object or environment to collect data on its shape and possibly its appearance (e.g. colour).
The collected data can then be used to construct digital 3D models. 3D data is useful for a wide variety of applications.
These devices are used extensively by the entertainment industry in the production of movies and video games, including virtual reality.
Other common applications of this technology include augmented reality,motion capture, gesture recognition, robotic mapping, industrial design, orthotics and prosthetics,reverse engineering and prototyping, quality control/inspection and the digitization of cultural artifacts.

3D image capture device
A 3D scanner can be based on many different technologies, each with its own limitations, advantages and costs.
Many limitations in the kind of objects that can be digitised are still present. For example, optical technology may encounter many difficulties with shiny, reflective or transparent objects.

3D scanning technologies can be divided in two main categories: Passive and Active

Types of 3D scanners
Passive scanning: Passive 3D imaging solutions do not emit any kind of radiation themselves, but instead rely on detecting reflected ambient radiation.
Most solutions of this type detect visible light because it is a readily available ambient radiation.
Other types of radiation, such as infrared could also be used. Passive methods can be very cheap, because in most cases they do not need particular hardware but simple digital cameras.
Stereoscopic cameras are the most common passive 3D scanning systems

3D passive scanning
Stereoscopic systems usually employ two video cameras, slightly apart, looking at the same scene. By analysing the slight differences between the images seen by each camera, it is possible to determine the distance at each point in the images. This method is based on the same principles driving human stereoscopic vision.

3D passive scanning
Photometric systems usually use a single camera, but take multiple images under varying lighting conditions. These techniques attempt to invert the image formation model in order to recover the surface orientation at each pixel.

3D passive scanning
Silhouette techniques use outlines created from a sequence of photographs around a three-dimensional object against a well contrasted background. These silhouettes are extruded and intersected to form the visual hull approximation of the object. With these approaches some concavities of an object (like the interior of a bowl) cannot be detected.

3D Active Scanning
Active scanners emit some kind of radiation or light and detect its reflection or radiation passing through object in order to probe an object or environment. Possible types of emissions used include light, ultrasound or x-ray.
Various typologies: Time-of-flight, Triangulation, Structured light, Modulated Light

3D Active Scanning
The time-of-flight 3D laser scanner is an active scanner that uses laser light to probe the subject. At the heart of this type of scanner is a time-of-flight laser range finder.
The laser range finder finds the distance of a surface by timing the round-trip time of a pulse of light. A laser is used to emit a pulse of light and the amount of time before the reflected light is seen by a detector is measured. 3.3 picoseconds (approx.) is the time taken for light to travel 1 millimetre.
The laser range finder only detects the distance of one point in its direction of view. Thus, the scanner scans its entire field of view one point at a time by changing the range finder's direction

3D Active Scanning
Triangulation based 3D laser scanners are also active scanners that use laser light to probe the environment. With respect to time-of-flight 3D laser scanner the triangulation laser shines a laser on the subject and exploits a camera to look for the location of the laser dot. Depending on how far away the laser strikes a surface, the laser dot appears at different places in the camera's field of view.

This technique is called triangulation because the laser dot, the camera and the laser emitter form a triangle.

3D Active Scanning

Structured-light 3D scanners project a pattern of light on the subject and look at the deformation of the pattern on the subject. The pattern is projected onto the subject using either an LCD projector or other stable light source.
A camera, offset slightly from the pattern projector, looks at the shape of the pattern and calculates the distance of every point in the field of view.
The advantage of structured-light 3D scanners is speed and precision. Instead of scanning one point at a time, structured light scanners scan multiple points or the entire field of view at once. Scanning an entire field of view in a fraction of a second reduces or eliminates the problem of distortion from motion.

3D Active Scanning

Modulated light 3D scanners shine a continually changing light at the subject. Usually the light source simply cycles its amplitude in a sinusoidal pattern.
A camera detects the reflected light and the amount the pattern is shifted by determines the distance the light travelled.
Modulated light also allows the scanner to ignore light from sources other than a laser, so there is no interference.

Microsoft Kinect

Kinect is a line of motion sensing input devices produced by Microsoft and first released in 2010.
The technology includes a set of hardware originally developed by PrimeSense, incorporating RGB cameras, infrared projectors (active - structured light) and detectors that mapped depth through either structured light or time of flight calculations, and a microphone array, along with software and artificial intelligence from Microsoft to allow the device to perform real-time gesture recognition, speech recognition and body skeletal detection for up to four people, among other

capabilities.
This enables Kinect to be used as a hands-free natural user interface device to interact with a computer system.
Product discontinued→ alternatives

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)
Novel UI are blended!

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Wii Remote and similar Motion Sensing devices
The Wii Remote, is the primary game controller for Nintendo's Wii home video game console.
An essential capability of the Wii Remote is its motion sensing capability, which allows the user to interact with and manipulate items on screen via gesture recognition and pointing, using IMU and optical sensor technology.
The Wii Remote was eventually succeeded by the more advanced Joy-Con controllers of the Nintendo Switch.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

IMU Inertial Measurement Unit
An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers.
9-Axis devices combine a 3-axis gyroscope, 3-axis accelerometer and 3-axis compass (magnetometer) in the same chip together with an onboard Digital Motion Processor capable of processing the complex MotionFusion algorithms

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)
Hacking the user interface and experience
https://www.youtube.com/watch?v=6YIAR4WZmes&ab_channel=DanieleMazzei
https://www.youtube.com/watch?v=6F7MzmySx9g&ab_channel=DanieleMazzei
https://www.youtube.com/watch?v=oQ2VCi-6uGI&ab_channel=DanieleMazzei

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)
Wearable Devices and User Interfaces
A wearable computer is a computing device worn on the body. The interface and the computing unit are merged together!
Wearables may be for general use, in which case they are just a particularly small example of mobile computing. Alternatively they may be for specialized purposes

such as fitness trackers.
They may incorporate special sensors such as accelerometers, thermometer and heart rate monitors, or novel user interfaces such as Google Glass, an optical head-mounted display controlled by gestures.
It may be that specialized wearables will evolve into general all-in-one devices, as happened with the convergence of PDAs and mobile phones into smartphones.

Wearable Devices and User Interfaces

Wearables are typically worn on the wrist (e.g. fitness trackers), hung from the neck (like a necklace), strapped to the arm or leg (smartphones when exercising), or on the head (as glasses or a helmet), though some have been located elsewhere (e.g. on a finger or in a shoe).
Devices carried in a pocket or bag – such as smartphones and before them pocket calculators and PDAs, may or may not be regarded as 'worn'.
Wearable computers have various technical issues common to other mobile computing, such as batteries, heat dissipation, software architectures, wireless and personal area networks, and data management.
Many wearable computers are active all the time, e.g. processing or recording data continuously.

Heart Rate Wearable Monitor

A heart rate monitor (HRM) is a personal monitoring device that allows one to measure/display heart rate in real time or record the heart rate for later study.
Consumer heart rate monitors are designed for everyday use and do not use wires to connect and are (typically) not suitable for medical applications
they commonly use electrical and/or optical measurement principles. Both types of signals can provide the same basic heart rate data, using fully automated algorithms to measure heart rate.
ECG (Electrocardiography) sensors measure the bio-potential generated by electrical signals that control the expansion and contraction of heart. VIDEO
PPG (Photoplethysmography) sensors use a light-based technology to measure the blood volume controlled by the heart's pumping action. some devices using this technology are able to measure aslo blood oxygen saturation (SpO2). more info

PPG (Photoplethysmography)

EEG Headset

Electroencephalography (EEG) is a monitoring method to record the electrical activity of the brain.
Wearable EEG headsets position noninvasive electrodes along the scalp. The clinical definition of EEG is the recording of brain activity over a period of time.

EEG electrodes pick up on and record the electrical activity in your brain.
The collected signals are amplified and digitized
then sent to a computer or mobile device for
storage and data processing.
EEG Headset are used as pointing devices for
dibale and impaired subjects

Le Interfacce utente

Le interfacce utente
Un'interfaccia è qualcosa che sta fra due
facce.
E' il punto di contatto fra due sistemi che
tentano di comunicare.
Le interfacce possono far comunicare due
macchine fra loro oppure possono far
comunicare l'uomo con la macchina
Lo strumento è ciò che fa qualcosa,
l'interfaccia è ciò che serve per guidarlo
nell'esecuzione dell'azione.

INTERFACCIA

STRUMENTO

Le interfacce utente
Quando parliamo di Interfaccia Utente (User Interface o UI) intendiamo lo spazio
di un sistema dove avviene l'interazione fra uomo-macchina.
Tipicamente, si parla di UI
in ambito informatico e
tecnologico e quindi le
interfacce utente sono
comunemente identificate
come sistemi atti a mettere
in comunicazione l'uomo
computer, sistemi
informatici e oggetti
intelligenti.

Le interfacce utente
"User interfaces are a mapping from the sensory, cognitive, and social human world to these collections of functions exposed by a computer program." Amy J. Ko, Washington university

https://faculty.washington.edu/ajko/books/uist/theory.html#:~:text=User%20interfaces%20offer%20learnable%20representations,anything%20in%20the%20natural%20world.

Le interfacce utente
L'obiettivo primario dell'interazione fra uomo e macchina è quello di consentire all'utente di controllare e far funzionare la macchina in modo efficace.
L'interfaccia deve quindi essere progettata per semplificare l'interazione fra l'uomo e la macchina rendendo così l'esperienza d'uso piacevole e prolifica.
L'interazione fra uomo e macchina
deve sempre essere facile,
efficiente e divertente così da
massimizzare la User Experience
del prodotto.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Le interfacce utente
Un'interfaccia ben progettata consente all'utente di controllare l'apparato richiedendo uno sforzo fisico e cognitivo minimo.
La buona interfaccia massimizza inoltre
la quantità di informazioni utili trasferite
all'utente durante l'interazione evitando
un sovraccarico informativo che
provocherebbe nell'utente confusione e
quindi frustrazione.

Le interfacce utente
Per questo motivo, la progettazione di un'interfaccia è per definizione un'attività interdisciplinare che va oltre la programmazione grafica e abbraccia la psicologia, le neuroscienze, il design e la fisica.

Le interfacce utente
User interfaces are composed of one or more layers including a human-machine interface (HMI) interfaces machines with physical input hardware such a keyboards, mice, game pads and output hardware such as computer monitors, speakers, and printers.
A device that implements a HMI is called a human interface device (HID).

HMI

HID

Le interfacce utente
Le interfacce utente sono tipicamente organizzate sulla base dei sensi che utilizzano per stabilire l'interazione fra umano e macchina. Gli umani possiedono cinque sensi (Tatto, Vista, Udito, Olfatto e Gusto).
Questo porta ad identificare cinque categorie di interfacce possibili, più una sesta che è legata al cosiddetto senso dell'orientamento (balance in inglese) che però non è considerato un senso vero e proprio nella fisiologia umana.
Possiamo quindi organizzare le interfacce in 6 categorie:
- Tactile UI
- Visual UI
- Auditory UI
- Olfactory UI
- Gustatory UI
- Equilibrial UI

La maggior parte delle interfacce utente utilizza però più di un senso umano per stabilire il collegamento.
Le interfacce che usano più di un senso sono dette CUI o Composite User Interface.
Le più comuni e note CUI sono chiaramente le famose GUI o Graphical User Interface, le quali sono composte da interfacce grafiche (visual) e tattili (tactile).
Se ad una GUI andiamo ad aggiungere anche il suono otteniamo una MUI o Multimedia User Interface.

Estendere le interfacce con più canali (sensi) non è sempre una buona idea

The user interface (UI)
There are three broad categories of CUI: standard, virtual and augmented.
Standard composite user interfaces use standard human interface devices like keyboards, mice, and computer
monitors.
When the CUI blocks out the real world to create a virtual reality, the CUI is virtual and uses a virtual reality
interface.
When the CUI does not block out the real world and creates augmented reality, the CUI is augmented and uses
an augmented reality interface.

Standard Virtual Augmented

Le CUI possono essere anche classificate tramite il numero di sensi che utilizzano.
Ad esempio, lo Smell-O-Vision è una CUI standard 3S, cioè è un'interfaccia di tipo standard che nell'utilizzo coinvolge 3 sensi dell'utente (Visione, Udito e Olfatto).
Se si aggiungesse un quarto senso (per esempio le poltrone mobili dei cinema 4D) diventerebbe 4S.
Quando un'interfaccia utente interagisce con tutti i sensi umani viene chiamata Qualia Interface. il termine Qualia interface deriva dalla teoria filosofica dei Qualia (https://it.wikipedia.org/wiki/Qualia).

Il Microsoft Reactable è un interessante esempio di interfaccia aumentata 3S (MUI)

Natural User Interfaces

NUI Natural User Interfaces
"Until now, we have always had to adapt to the limits of technology and conform the way we work with computers to a set of arbitrary conventions and procedures. With NUI, computing devices will adapt to our needs and preferences for the first time and humans will begin to use technology in whatever way is most comfortable and natural for us."
—Bill Gates, co-founder of the multinational technology company Microsoft

Defining NUIs is difficult, but often when we think about user interfaces that are natural and easy to use, we think of user interfaces where the interaction is direct and consistent with our 'natural' behaviour.

source:
https://www.interaction-design.org/literature/article/natural-user-interfaces-what-are-they-and-how-do-you-design-user-interfaces-that-feel-natural

Natural User Interfaces
A natural user interface, or NUI, is a user interface that is effectively invisible, and remains invisible as the user continuously learns increasingly complex

interactions.
The word natural is used because most computer interfaces use artificial control devices whose operation has to be learned. An NUI relies on a user being able to quickly transition from novice to expert.
While the interface requires learning, that learning is eased through design which gives the user the feeling that they are instantly and continuously successful.
Thus, "natural" refers to a goal in the user experience – that the interaction comes naturally, while interacting with the technology, rather than that the interface itself is natural.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Natural User Interfaces
NUI contrasted with the idea of an intuitive interface, referring to one that can be used without previous learning. Several design strategies have been proposed which have met this goal to varying degrees of success.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Natural User Interfaces
Some iPad gestures come naturally and intuitively—e.g., swiping with one finger to the left or right—which is all part of the ‘magic’ that took the iPad to such prominence.
When you swipe with one finger, you scroll through pages or you move content from one side of the screen to the other. The gesture itself corresponds to the action you are performing.
it shows the power of the principle here—things in the ‘digital world’ behave as they do in the ‘analogue world’.
Some gestures, though, require more learning—e.g., a four-finger swipe to the left or right. The four-finger swipe is not intuitive—it doesn’t come naturally to us. Swiping with four fingers requires you as a user to learn it as a dedicated movement, because you need an understanding of the underlying system

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Natural User Interfaces
“Voice, gesture, touch does not necessarily Natural User Interface make.”
—Bill Buxton, Principal Researcher at Microsoft
Microsoft has done a lot of research into NUIs.
NUIs exploit skills that we have acquired through a lifetime of living in the world, which minimizes the cognitive load and therefore minimizes the distraction.
NUIs should always be designed with the use context in mind. No user interface can be natural in all use contexts and to all users.
While gestures, voice and touch are important components of many NUIs, they will only feel natural to a user if they match her skill level and her use context.

Natural means no learning
Two- and 3-year-old children could not follow any prompting technique and only a minority (27%) could tap the touchscreen
at an intended place. Four- to 6-year-old children could perform simple gestures like a tap and slide (57%) and follow
instructions provided through animation (63%). Seven- and 8-year-old children could perform more sophisticated gestures
like dragging and dropping (30%) and follow instructions provided in audio and video formats (34%). source here

Joshua Blake lists four guidelines for designing NUIs
● Instant expertise
● Progressive learning
● Direct interaction
● Low cognitive load (primarily use innate abilities and simple skills)
more info here
https://www.interaction-design.org/literature/article/natural-user-interfaces-what-are-they-and-how-do-you-design-user-interfaces-that-feel-natural

NUI in a nutshell
An NUI is a user interface that feels natural to use because it fits the skills and context of the user.
● An NUI should take advantage of the users' existing skills and knowledge.
● An NUI should have a clear learning path and allow both novice and expert users to interact in a natural way.
● Interaction with an NUI should be direct and fit the user's context.
● Whenever possible, you should prioritize taking advantage of the user's basic skills.
great design is about satisfying needs, not outsmarting users.

Reality Based Interfaces
One strategy for the design of NUI is the use of a "reality user interface" ("RUI"), also known as "reality-based interfaces" (RBI) methods.
One example of an RUI strategy is to use a wearable device to render real-world objects "clickable", i.e. so that the wearer can click on any everyday object so as to make it function as a hyperlink, thus merging cyberspace and the real world.
Because the term "natural" is evocative of the "natural world", RBI are often confused for NUI, when in fact they are merely one means of achieving it.

https://www.youtube.com/watch?v=LZWnGo_IsDw&ab_channel=RUISVR

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

Natural User Interfaces without RBI
One example of a strategy for designing a NUI not based in RBI is the strict limiting of functionality and customization, so that users have very little to learn in the operation of a device.
Provided that the default capabilities match the user's goals, the interface is effortless to use.
This is an overarching design strategy in Apple's iOS.
https://www.youtube.com/watch?v=qhEt1Pd-twM&ab_channel=phototristan
Because this design is coincident with a direct-touch display, non-designers commonly misattribute the effortlessness of interacting with the device to that multi-touch display, and not to the design of the software where it actually resides.

- PROGRAMMAZIONE INTERFACCE 22-23 (all rights reserved)

CEEDs project
https://www.youtube.com/user/ceedsproject

Usability Testing

roberto.figlie@phd.unipi.it

Lesson outline

- What does it mean and why?
- Types of testing
- How to plan tests?
- How to conduct tests?
- Best practices
- Testing Tools

What does it mean to test and why we should do it?

What does it mean to test and why we should do it?

When you have some sort of a prototype what you should want to do is testing with the users.
This is the process where you can evaluate a product or service by testing it directly with its intended users.

https://www.nngroup.com/articles/usability-testing-101/

What does it mean to test and why we should do it?

In this way you can identify many (but not all) usability issues, opportunities for improvement, and how your users behave what they prefer.

But what is usability?

Usability is a quality attribute that assesses how easy user interfaces are to use.
Usability is defined by 5 quality components:
● Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?
● Efficiency: Once users have learned the design, how quickly can they perform tasks?
● Memorability: When users return to the design after a period of not using it, how easily can they reestablish proficiency?
● Errors: How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
● Satisfaction: How pleasant is it to use the design?
https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Usability and Utility

Other than usability another key concept in UX design is utility, which refers to the design's functionality: Does it do what users need?
Usability and utility are equally important and together determine whether something is useful:

➔ It matters little that something is easy if it's not what you want.
➔ It's also no good if the system can hypothetically do what you want, but you can't make it happen because the user interface is too difficult.
To study a design's utility, you can use the same user research methods that improve usability.

Usability + Utility

Utility = whether it provides the features you need.
Usability = how easy & pleasant these features are to use.
Useful = usability + utility.

Note that usually you will find mention to usability only: it is not conceptually right but it makes it easier to refer to the type of testing.

User and Usability testing

So, testing is essential in creating products that meet user needs and expectations following the principles of Human-Centered Design.
Only discovering usability issues and testing with real users you can create an improved user experience that leads you to higher user satisfaction, and ultimately drive business success.

User and Usability testing

As a user research methodology we define a a usability-testing session when a researcher (called a “facilitator” or a “moderator”) asks a participant to perform tasks, usually using one or more specific user interfaces.
While the participant completes each task, the researcher observes the participant’s behavior and listens for feedback.

https://www.youtube.com/watch?v=v8JJrDvQDF4

Heuristics
evaluation

Heuristic evaluation

Before testing with real users, something else can be done to discover usability issues.
Heuristic evaluation is a method used to evaluate user interfaces based on established usability principles or "heuristics". This evaluation typically involves a small team of evaluators, as the process can be challenging for a single individual to complete. Also, involving multiple evaluators (3-5 are enough) can improve the accuracy of the evaluation.
Although heuristic evaluation does not provide a systematic approach to fixing usability problems or assessing redesigns, it can guide the creation of revised designs based on the violated principles of good interactive systems in a cost-effective way.

Nielsen’s 10 Usability Heuristics

Nielsen’s 10 Usability Heuristics

Nielsen’s 10 Usability Heuristics

Nielsen’s 10 Usability Heuristics

Elements of
Usability Testing

Elements of Usability Testing

The Facilitator

The facilitator guides the participant through the test process.
The facilitator:
● gives instructions,
● answers the participant's questions,
● and asks follow up questions.
The facilitator works to ensure that the test results in high-quality, valid data, without accidentally influencing the participant's behavior. Achieving this balance is difficult and requires training.

The tasks

The tasks in a usability test are realistic activities that the participant might perform in real life. They can be very specific or very open-ended, depending on the research questions and the type of usability testing.
A few examples of tasks:
● Your printer is showing "Error 5200". How can you get rid of the error message?
● Find out articles that uses Cloud technologies to solve a warehouse management problem. Can you find them? How?

The tasks

Task wording is very important in usability testing. Small errors in the phrasing of a task can cause the participant to misunderstand what they're asked to do or can influence how participants perform the task (a psychological phenomenon called priming).
Task instructions can be delivered to the participant verbally (the facilitator might read them) or can be handed to a participant written on task sheets.
If you give them written tasks ask participants to read the task instructions out loud. This helps ensure that the participant reads the instructions completely, and helps the researchers with their notetaking, because they always know which task the user is performing.

The Participant

The participant should be a realistic user of the product or service being studied. That might mean that the user is already using the product or service in real life. Alternatively, in some cases, the participant might just have a

similar background to the target user group, or might have the same needs, even if he isn't already a user of the product.
Ideally you should select them starting from the personas you made.
Participants are often asked to think out loud during usability testing (called the "think-aloud method"). The facilitator might ask the participants to narrate their actions and thoughts as they perform tasks. The goal of this approach is to understand participants' behaviors, goals, thoughts, and motivations.

Elements of Usability Testing

Types of usability

testing

Qualitative vs Quantitative

There are 2 types of usability testing depending on the data you can collect:
Qualitative (qual) data, consisting of observational findings that identify design features easy or hard to use.
Quantitative (quant) data, in form of one or more metrics (such as task completion rates or task times) that reflect whether the tasks were easy to perform.

Qualitative vs Quantitative

Qualitative testing typically involves gathering data through open-ended questions, observations, and user interviews to gain insight into how users interact with a product and what their feelings and opinions are about it.
It is often conducted in the early stages of the design process to help designers understand user needs, goals, and pain points. But it can also be used to validate design decisions and test prototypes before launching a product or service.

Qualitative vs Quantitative

Quantitative testing is a research method that involves collecting and analyzing numerical data (metrics) to measure user behavior and attitudes towards a product or service. It typically involves gathering data through surveys, analytics, A/B testing, and other quantitative research methods.
The goal of quantitative testing is to provide designers with statistically significant data that can be used to make informed design decisions. By collecting quantitative data, designers can measure user behavior and preferences in a more objective and systematic way, making it easier to identify patterns and trends.
Quantitative testing can be used throughout the design process, from the early stages of research and discovery to testing and validating design decisions. It can be especially useful for measuring the effectiveness of design changes and improvements over time.

Qualitative vs Quantitative

Both qualitative and quantitative testing are important methods in UX design, and they complement each other well. Qualitative tests can help designers understand the "why" behind user behavior, while quantitative tests can provide a more comprehensive understanding of user behavior and preferences.

Qualitative Research Quantitative Research

Questions answered Why? How many and how much?
Goals Both formative and summative:
● inform design decisions
● identify usability issues and find solutions for them

Mostly summative:
● evaluate the usability of an existing site
● track usability over time
● compare site with competitors
● compute ROI

When it is used Anytime: during redesign, or when you have a

final working product

When you have a working product (either at the beginning or end of a design cycle)

Outcome Findings based on the researcher’s impressions,

interpretations, and prior knowledge

Statistically meaningful results that are
likely to be replicated in a different study

Methodology ● Few participants

● Flexible study conditions that can be
adjusted according to the team’s needs
● Think-aloud protocol

● Many participants
● Well-defined, strictly controlled study
conditions
● Usually no think-aloud

How to plan and
conduct usability

tests

Planning a test - The goals

First of all you have to define the purpose of your
research. What do you want to test? What is the end goal in
conducting tests?
This will first lead you to understand if you need
qualitative or quantitative research.

Planning a test - The format

In lab or in field: consider the location: should it be in-house or at
the participant's location?
Moderated or unmoderated: Moderated studies provide richer insights and
better access to open-ended comments but cost more and difficult to
organize. Unmoderated studies can be cheaper, quicker and may provide
better access to hard-to-recruit participants.
In-person or remote: In-person studies are recommended whenever
possible as they allow for more personable interaction and the ability

to detect subtle cues, but remote studies can be more accessible and sometimes necessary due to budget or logistical constraints.

Planning a test - number of participants

How many testers?
Jakob Nielsen highlighted that 5 people is a good number for qualitative studies as more will not guarantee you many other insights.
However, this will not apply to quantitative studies! Here the recommended number is 20.

Planning a test - Recruit the Right Participants

To get the most valuable insights from user testing, it's important to use representative participants who match the demographics and behaviors of your target users.
Avoid asking proxy users to pretend or imagine scenarios, as this can lead to invalid results. This is especially important for specialized websites and content-rich or B2B sites, where finding exact matches for your target users is crucial.

Planning a test - Write the right tasks

In usability testing, tasks are written as scenarios and should match the study goals. Tasks can be exploratory or specific:
● exploratory tasks are open-ended and used for learning how people explore information (not suited for quantitative research),
● specific tasks are focused and have a correct answer or end point, and can be used for both qualitative and quantitative testing.
Also, pay attention to the wording of the task! (More on this later)

Planning a test - Rehearse it fist

Before conducting usability testing, run a pilot study to refine task wording, determine task order and quantity, and ensure you are recruiting the right participants.
This is especially important for online unmoderated studies where there is no opportunity for clarification or correction during the session.
Catching problems early can prevent issues during the actual testing session.

Planning a test - Decide the metrics (if any)

Qualitative usability studies prioritize gaining design insights over measuring usability, as metrics are unlikely to be representative for the whole user population with few users.
However, in quantitative studies with well-defined tasks and a large number of users, measuring usability is important and common metrics include time on task, satisfaction ratings, success rate, and error rate.
If collecting subjective measurements, decide when to give questionnaires: after each task, at the end of the session, or both.

Planning a test - Write down the plan

Once you've figured out how you're going to conduct the research, document your approach in a test plan and share it. This document serves as a communication tool among team members and a record for future studies. The document doesn't need to be lengthy, but should contain key information such as:

- Name of the product or site being testing
- Study goals
- Logistics: time, dates, location, and format of study
- Participant profiles
- Tasks
- Metrics, questionnaires
- Description of the system (e.g., mobile, desktop, computer settings)

Planning a test - Don't do it alone

Usability studies can foster collaboration and buy-in by allowing stakeholders to observe how users respond to the

interface design, leading to more efficient communication and design.
Invite stakeholders and team members to observe moderated testing sessions. Traditional or simplified usability labs are ideal, but remote viewing options can also be offered to include team members in different locations.

10 things to avoid
when testing

10 things to avoid when testing

1. Telling Users Where to Go. To avoid priming, remove any words that appear in your interface, so you give yourself a fair chance to see if users can find their way around the site.
2. Telling Users What to Do. Don't warn participants about the steps they need to take, such as registering or downloading, to gather valuable feedback about their experience and potential issues that may arise.
3. Creating Out-of-Date Tasks. If the task is to find a flight leaving February 20, don’t run that test on February 22. A task about the latest news on a site should be updated the day before or the day of testing to include current content.
4. Making Tasks Too Simple. Your goal isn’t to make tasks too simplistic or unnecessarily complex, but to give users a realistic task that requires processing, rather than just locating information.

10 things to avoid when testing

5. Creating an Elaborate Scenario. Using scenarios in user testing tasks can provide context and help participants understand the purpose of the task, but should be used with caution as they may add unnecessary complexity and increase the cognitive load for users. Scenarios should be kept simple and not used to justify unusual or unrealistic activities.
6. Writing an Ad, not a Task. Make sure your tasks don’t include marketing phrases like “exciting new feature,” business phrases like “thinking outside the box,” or mysterious corporate acronyms. Use user-centric language, not maker-centric language.
7. Risking an Emotional Reaction. Part of the responsibility of running a usability test is to ensure the well-being of your participants. Stick to harmless and vague relationships instead – friend, colleague, a friend’s child.

10 things to avoid when testing

8. Trying to Be Funny. Don’t joke, use famous names in tasks, or otherwise try to lighten the mood. Doing so can backfire and make some participants feel awkward or, even worse, as though you are making fun of them.
9. Offending the Participant. Avoid potentially offensive details in tasks. Societal issues, politics, health, religion, age, and money all have the possibility of offending a participant.
10. Asking Rather than Telling. Don’t ask participants “how would you” complete a task — unless you want them to talk you through what they theoretically would do on a site, rather than doing it. The point of usability testing is to see what users do, not to hear what they would do.

For more information and details on usability testing:

https://www.nngroup.com/articles/qual-usability-testing-study-guide/

https://www.nngroup.com/articles/quantitative-research-study-guide/