“Human Computer Interaction on the Web” Notes

Prof. Emanuele Panizzi. Appunti scritti da Alessio Lucciola durante l'a.a. 2022/2023. Sei libero di:

- Condividere: Copiare e distribuire il materiale in qualsiasi mezzo o formato.
- Adattare: Trasformare o modificare il materiale.

Rispettando i seguenti termini:

- Attribuzione: E’ necessario fornire i crediti appropriati, un collegamento alla licenza e indicare se sono state apportate modifiche.
- Fine non commerciale: Non è possibile utilizzare il materiale per scopi commerciali.

Le note possono contenere errori di battitura. Se ne vedi uno, puoi contattarmi utilizzando i link nella pagina Github. Se questi appunti ti sono utili, potresti prendere in considerazione l'idea di offrirmi un caffè 😄.

1. Wearable Devices

Un **wearable device** è un dispositivo **portatile**, **indossabile** e **impiantabile** nel corpo umano. L’uomo e la macchina risultano intrecciati l’uno all’altro, il che dà la possibilità all’essere umano di essere usato come feedback per la macchina e, viceversa, la macchina può rispondere con informazioni utili all’uomo e questo risulta in una sorta di **loop**. Per questo motivo, alcune proprietà dei wearable device sono:

- **Consistenza**: La macchina è sempre accesa, non c’è bisogno di accendere il dispositivo per poterlo utilizzare (es. il telefono non viene mai spento, neanche durante la notte, per fare in modo che sia sempre attivo e pronto ad essere utilizzato);
- **Abilità al multitask**: Non c’è bisogno di interrompere l’utente che può utilizzare il dispositivo qualsiasi cosa stia facendo (es. smartwatch che è attivo mentre si corre o cammina e tiene conto dei valori vitali).

In generale, è un **oggetto programmabile** con algoritmi complessi, un’interfaccia e la gestione dei dati.
Si può utilizzare in vari modi come la realtà aumentata, il tracciamento delle attività, nel campo medico (salute), fitness, sport e realtà virtuale.
Hanno vari sensori che permettono di catturare alcuni input come il microfono, l’accelerometro, il GPS, il cardiofrequenzimetro, bottoni o il touchscreen. Gli **attuatori** con cui l’utente può interagire sono lo schermo, le casse audio, la vibrazione, il proiettore (laser) e così via. La comunicazione può essere stabilita tipicamente via bluetooth, wifi e NFC.

Molto spesso **non c’è un’interazione diretta** tra noi e questi dispositivi. Ad esempio con l’Apple Watch mentre si cammina, molto spesso nota che si sta facendo un’attività fisica e mostra una notifica per avviare la modalità di attività sportiva. Questo avviene senza che noi gli diciamo cosa stiamo facendo, lo capisce da solo. In ogni caso, l’**interazione può essere complessa** per via dello schermo estremamente piccolo e inoltre, il **contesto in cui ci troviamo può rendere difficile effettuare determinate gesture** (es. siamo in aula e il dispositivo richiede di alzare la mano per attivarlo). Potrebbero risultare scomodi da indossare, talvolta creare stress o raccogliere informazioni specifiche non sempre richieste. Possiamo prestare una **differente attenzione** al dispositivo che tipicamente è bassa nel caso di sensori (captano informazioni sull’utente senza bisogno di un suo feedback) o alta nel caso ci sia bisogno di una manipolazione diretta.

Nel caso di dispositivi a realtà aumentata, si possono usare **metafore spaziali**. Ad esempio si può mostrare all’utente, immagini come cartelle per immagazzinare file o cestini per eliminare un file. Si può anche simulare il suono 3D mostrando facce di persone che stanno parlando in diverse posizioni in maniera tale da migliorare l’interazione.

Bisogna inoltre tenere conto delle **interruzioni** che tali dispositivi possono causare all’utente che, in generale, distolgono l’utente da quello che sta facendo, sono innescate esternamente e molto spesso risultano in perdite di memoria. Ci sono vari tipi di interruzioni:

- **Immediate**: Forzano l’utente a fare qualcosa (es. chiamata in arrivo, batteria scarica, aumento della frequenza cardiaca mentre si corre, ..);
- **Negoziate**: Annunciano l’interruzione in maniera tale da permettere all’utente di assumere il controllo del dispositivo quando desidera e non distrarlo così dall’attività corrente (es. notifica di un messaggio, la si può guardare successivamente);
- **Programmate**: Notificano l’utente periodicamente (es. sveglia quotidiana);

- **Mediate**: Scelgono autonomamente quando notificare all'utente in base al contesto in cui si trova (es. sistemi di monitoraggio del sonno che non disturbano l'utente quando sta dormendo oppure il navigatore ti notifica quando devi cambiare strada).

In generale, le interruzioni possono essere effettuate sulla base di sensori che ci forniscono informazioni sulla persona (es. quando si corre o si dorme si evita di disturbare la persona), stress o si possono anche configurare in base alle preferenze dell'utente.

Si può **interagire** in vari modi, tipicamente **eye-free** (senza l'uso degli occhi), ad esempio muovendo varie parti del corpo come il polso (es. si può girare il polso per attivare il dispositivo) oppure la testa (es. visore ottico).

2. HCI for IoT

2.1. Interface and Interaction for IoT

Internet of Things è un neologismo riferito all'estensione di Internet al mondo degli oggetti e dei luoghi concreti. Con dispositivi IoT si intendono tutti quei dispositivi, diversi da computer o smartphone, che hanno una certa potenza computazionale e accesso ad Internet, come ad esempio i sistemi embedded. Le **interfacce punta-e-clicca e touchscreen sono risultate inadeguate** in questi sistemi a causa dello **spazio** (lo spazio è tipicamente molto limitato, molto spesso si tratta di semplici sensori), del **contesto** (il contesto potrebbe essere non adatto, si pensi i sensori il cui unico obiettivo potrebbe essere quello di ricavare informazioni ed elaborarle) e del **costo** (il costo dei prodotti potrebbe diventare troppo elevato). Esempi di dispositivi IoT sono veri oggetti che abbiamo in casa come luci smart (controllabili con bottoni e programmabili attraverso un'interfaccia a parte, es. applicazione sul telefono), termostato, frigoriferi, smart tv oppure sensori come un rivelatore di perdite d'acqua o di gas e così via.
Ci sono vari modi per **interagire con dispositivo IoT** (vari **input**):

- Tocco, pressione: controlli fisici (bottoni, switch), touch screen;
- Movimento & Manipolazione (Tangible UIs): Si tratta della manipolazione fisica di oggetti, come ad esempio muovere o posizionare oggetti fisici. Il movimento è un input per l'oggetto (es. si gira il telefono, si ruota lo schermo (si attiva lo schermo panoramico));
- Voce: riconoscitore vocale;
- Tutto il corpo: gesture recognition, sensori di prossimità (es. si pensi a strumenti con il Kinect di Xbox);
- Pensieri: Brain-computer interface (es. un dispositivo sulla testa che permette di vedere vari parametri come il livello di concentrazione);
- Frequenza cardiaca: determinata mediante stress e battito cardiaco (es. i wearable devices).

Gli **output** vengono ricevuti con:

- Led: Si può usare un color coding illuminando led in base allo stato corrente (es. led verde acceso quando la webcam è attiva);
- Schermo: Può essere ad esempio un segment display o un LCD display. Ottimale per rendere dinamici gli oggetti, meno ottimale se si vuole mantenere una UX semplice;
- Suono: Si può usare ad esempio un buzzer, si trasmettono in output suoni o text-to-speech. Non bisogna essere vicino al dispositivo o guardarlo per sentirlo. Ottimo per avvisi importanti, meno ottimale dove potrebbe disturbare altre persone (es. suono quando lo schermo del telefono si blocca, batteria scarica);
- Vibrazioni: Oggetto che effettua una certa vibrazione in base al suo stato.
- Scent Messages: Oggetto che emana un certo odore (sintetico) in base al suo stato.
- Temperature output: Oggetto che cambia la temperatura in base al suo stato.

Andiamo nel dettaglio con alcuni dei dispositivi precedentemente descritti:

- **Controlli fisici**: A causa dell'assenza di interfacce standard, si è presentata la necessità di gestire il dispositivo attraverso sistemi di controllo fisici basati su bottoni, switch, slider, rotary knobs, da cui è possibile ricavare anche lo **stato del sistema** (es. l'interruttore della luce ci dà un feedback acceso/spento quando lo tocchiamo). I sistemi di controllo fisici possono fornire all'utente un **feedback aptico** (attraverso il senso del tatto, la posizione della mano sul dispositivo) e **tattile** (solamente attraverso le dita, quando tocchiamo il dispositivo), al momento dell'inserimento dell'input. Quindi:
    - PRO: Ottimi per software che possono essere controllati in modo diretto e veloce; garantiscono l'accessibilità ad utenti non vedenti.

    - CONTRO: Non adatti a software aggiornati frequentemente, o a software che hanno features o impostazioni che possono essere controllate da remoto.
- **Luci**: I dispositivi di output più immediati sono le luci, come per esempio i led, che danno un feedback diretto, attraverso il colore o il blink pattern (intermittenza), sullo stato del sistema (es. led del telefono, lampeggia quando arriva un messaggio, è rossa fissa quando il telefono è scarico, è verde fissa quando è completamente carico). Quindi:
    - PRO: Richiedono poca attenzione, occupano poco spazio e forniscono informazioni a colpo d'occhio.
    - CONTRO: Risulta difficile comunicare informazioni complesse quindi va utilizzato solo se si vogliono comunicare informazioni semplici.
- **Display e schermi**: Esistono molte varietà di dispositivi di output di questo tipo come **Segment Display** (orologio digitale in cui i numeri vengono rappresentati da piccoli segmenti), **Schermi e display mono/poli cromatici** (es. mono: lo schermo del forget-me-not; poli: dispositivi con schermi lcd), **Electronic-ink** (es. lo schermo del kindle). Schermi e display offrono grandi potenzialità ad un prodotto, ma è sempre bene chiedersi se inserire uno di questi dispositivi è una buona idea, il rischio che si corre è cadere nel **feature creep** (quando si riempie il proprio prodotto di funzionalità anche superflue non rendendosi conto che si sta rovinando il prodotto stesso), inoltre, questo genere di dispositivi hanno un **costo elevato** e di conseguenza faranno lievitare il prezzo del prodotto finale. Quindi:

    - PRO: Ottimi quando si vuole dare dinamicità ad un oggetto fisico (mostrare il progresso di un lavoro su una barra di caricamento) e quando il prodotto è flessibile (se ci sono molte update del software).
    - CONTRO: Complicano molto la user experience e sono molto costosi.
- **Audio Output**: I dispositivi di audio output hanno due macro-utilizzi, possono essere usati per leggere un testo scritto riproducendo una voce (lettura automatica dei messaggi, **text-to-speech**) e, essendo **pervasivi**, permettono di **trasmettere emozioni all'utente** (es. audio-book). Permettono di dare un feedback all'utente senza che esso debba guardare il dispositivo (es. con i suoni delle notifiche personalizzati è possibile trasmettere all'utente informazioni ben precise per quel determinato suono, senza che si controlli il dispositivo). Quindi:
    - PRO: Sono ottimi per allarmi urgenti e critici (es. il dispositivo che notifica lo scattare di un allarme), per dare un feedback emotivo all'utente (es. quando il telepass si scarica emette un suono che può sembrare triste), riescono a comunicare informazioni complesse e possono comunicare con l'utente anche quando gli altri canali (visivo, tattile) sono occupati (es. navigatore che ci informa quando e dove girare senza guardare lo schermo).
    - CONTRO: Possono risultare irritanti per l'utente e per le persone intorno.
- **Voice input**: Comandi espressi attraverso la voce, che si basa su un approccio in cui bisogna **ricordare il comando** invece che riconoscerlo tra una serie di comandi proposti. Potrebbero risultare **scomodi e inaffidabili** (quando l'utente non ricorda il comando preciso o quando il comando non viene riconosciuto). Quindi:
    - PRO: Hands-free (si possono usare mentre si hanno le mani occupate), ottimi in ambienti con buona connessione e pochi rumori, comodi quando i dati di input richiesti sono molto complessi (inserimento di sveglie manualmente) e buoni quando gli altri canali sono occupati (l'utente può dare dei comandi mentre ci sono dei processi in corso).
    - CONTRO: Quando i comandi da pronunciare risultano difficili o quando ci sono più localizzazioni quindi si ha bisogno di più dispositivi (aumento del costo).
- **Tangible and Tactile**: Dispositivi **manipolabili e posizionabili** che forniscono all'utente dei feedback sotto forma di vibrazioni. Consistono anche nella pressione di tasti. Sono tipicamente semplici da imparare e capire e hanno sia la capacità di inserire input che di fornire output anche dell'ambiente (temperatura dell'ambiente). Quindi:

    - PRO: Possono essere usati a scopi educazionali, non sembra di interagire con un computer vero e proprio e l'utilizzo richiede poca attenzione.
    - CONTRO: Delle volte non forniscono tante informazioni contemporaneamente ma una per volta (es. sveglia con informazioni in base alla posizione: o orario o temperatura), spesso non si ha il tempo di imparare il loro funzionamento e potrebbero risultare inaffidabili (es. sveglia posizionata nel modo sbagliato).
- **Gestural Input**: Input attraverso gesti: swipe, pinch (pizzico), mid-air gesture (es. movimenti della mano per attivare funzionalità senza toccare l'oggetto) e body movements. Quindi:
    - PRO: Molto usati per video games, wearable devices e brevi interazioni.

    - CONTRO: Non è molto comodo per interazioni lunghe (può essere faticoso se fatto per molto tempo) o di precisione e possono produrre falsi positivi, che non sono accettabili.
- **Context-Sensitive Interaction**: **Interazioni che si basano sul contesto** (in un posto, in una stanza, vicino un prodotto). Queste interazioni **non richiedono molta attenzione dell'utente**, ma hanno bisogno di uno studio dell'interpretazione del contesto da parte dei progettisti. La domotica riesce ad essere un buon esempio di questo tipo di interazioni (es. quando l'utente esce di casa, le luci si spengono). Quindi:
    - PRO: Con le context-sensitive è possibile gestire interazioni complesse con piccole interazioni. Ottimo quando c'è una relazione diretta tra contesto e caratteristiche dei device.
    - CONTRO: Quando può essere percepito come patronizing (quando l'utente dipende dal sistema), ovvero il sistema limita l'utente ad intraprendere azioni in base al contesto.
- **Computer Vision and Barcodes**: Si intendono tecnologie come: facial recognition, biometric interfaces, barcodes, QR codes, OCR (riconoscimento ottico dei caratteri). Quindi:
    - PRO: Ottimi per la sostituzione manuale di input ingombranti.
    - CONTRO: Non comodi quando richiedono altre interazioni più complesse di queste alternative.

2.2. User Experience for IoT

L'**HCI** parte dal desktop, si espande per molteplici device e trova il suo culmine nell'uso di un solo servizio in più device differenti. L'obiettivo delle HCI è essere **cross-platform**. I device possono essere di diverso tipo e modello (alcuni senza schermo come la serratura della porta che si può aprire con il riconoscimento facciale, altri con ecc..), ma devono essere interconnessi tra loro anche se non in modo continuo. Le funzionalità più importanti dei dispositivi sono:

- Molteplici device con capacità differenti (per usare, ad esempio, uno stesso servizio);
- Schermo, luci (led), suoni differenti;
- Assenza di I/O (la comunicazione avviene via web o smartphone);
- Servizi **coerenti** per tutti gli utenti (stesso servizio usabile da dispositivi diversi);
- **Inter-usabilità** tra tutti i device (si deve pensare all'usabilità dell'intero sistema e non del singolo dispositivo);

Per quanto riguarda la comunicazione via internet, nonostante i device possano essere innovativi ed originali, c'è sempre la necessità di avere una **connessione affidabile**, caratteristica importante tanto quanto i device stessi (tutto gira intorno alla connessione). L'**IoT si basa quasi interamente sui dati**, per cui i sistemi embedded tipicamente raccolgono informazioni dal mondo reale al fine di fornire servizi sempre migliori sia tramite UI che tramite dispositivi integrati collegati in rete (es. sistema di misurazione e riscaldamento dell'attività elettrica).

Per quanto riguarda la **latenza**, Idealmente ci si aspetta un comportamento lineare in cui i device rispondono in modo affidabile grazie ad una connessione stabile. Realmente accade che i device possono avere ritardi o guasti, con conseguenze inaspettate. È possibile avere un **sistema asincrono** quando, ipotizzando di avere una connessione affidabile su tablet e computer, i dispositivi IoT con batterie, per risparmiare energia, accedono al sistema ad intermittenza (all'occorrenza). Così due parti dello stesso sistema possono non essere contemporaneamente attive nella rete. Tutto questo può avvenire anche a causa di problemi di latenza (es. si imposta una nuova temperatura del termostato via smartphone, ma esso impiega qualche minuto ad aggiornarsi). Anche il **codice** può avere una ripercussione rilevante sul funzionamento dei sistemi IoT. Poiché il sistema può avere codice diverso in ogni suo componente (es. cloud, gateway, app, device...), c'è la possibilità di avere desincronizzazione sul sistema causata da qualche bug sul codice di uno di questi componenti. Pertanto, i progettisti dei sistemi IoT devono rendere il **modello di sistema più chiaro e comprensibile all'utente**, che all'evenienza può decidere di escludere una parte del sistema che non si comporta come dovrebbe (l'**utente deve mantenere il controllo**).

L'intento è creare sistemi, come applicazioni, con interfacce per l'utente in grado di dare la possibilità di **gestire e controllare le funzionalità da remoto**. Per fare questo passo c'è bisogno della **programmazione**.
La **complessità dei sistemi** realizzati spesso richiede:

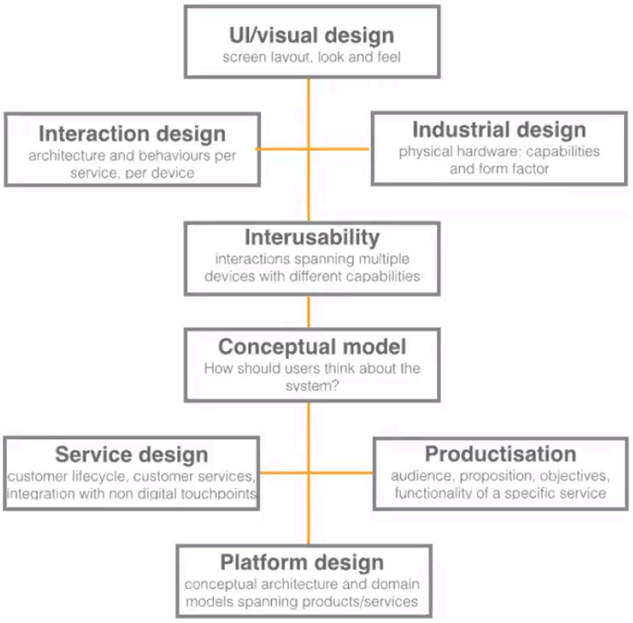
- La creazione di **diverse tipologie di utenti**, ognuno di essi con privilegi differenti, è quindi necessario realizzare UIs specifiche per ogni tipologia.
- **Stabilire regole**, che rendono la comprensione del sistema richiede un'attività lunga ed impegnativa.

La creazione di UIs richiede l'utilizzo di norme tecniche per garantire l'**interoperabilità tra i device**. In particolare, i device presi singolarmente rispondono appieno alle richieste dei produttori, nei sistemi IoT c'è necessità di integrarli tra loro poiché hanno standard diversi. Per garantire il funzionamento di tutti questi

elementi bisogna trovare un compromessovincente delle tecnologie, molto spesso Internet risulta essere il compromesso ricercato: la **connessione avviene su rete anziché tra dispositivi**. La realizzazione di UIs funzionanti e affidabili permette all'utente di avere una **piena padronanza del sistema** e delle funzionalità di esso.

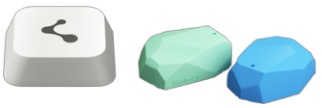
Esiste uno "**stack di progettazione" per IoT**:

- **UI/Visual design**: Layout ed estetica. Non solo l'aspetto visivo ma anche l'esperienza;
- **Interaction design**: Comportamento/sequenza di azioni tra utente e dispositivo;
- **Inter Usability**: Considerazioni che si estendono su più dispositivi. Servizio coerente. Flussi utente cross-device. Progettare più interfacce utente in parallelo;
- **Industrial design**: Forma, materiali e capacità dell'hardware fisico. Bisogna considerare anche i vincoli tecnici dei dispositivi;
- **Service design**: Affronta una visione olistica dell'esperienza utente, tra cui UX di aggiornamenti software e nuove funzionalità, assistenza clienti, esperienza in-store, ecc.;
- **Conceptual model**: Consente agli utenti di capire come interagire con il servizio;
- **Productisation**: Definire una proposta di prodotto convincente. Il prodotto fa qualcosa di valore per gli utenti;
- **Platform design**: Framework software di progettazione/utilizzo che può aiutare gli sviluppatori e gli utenti a scoprire nuovi dispositivi e applicazioni, aggiungerli al sistema, gestire gli utenti e gestire i dati;



Risulta importante vedere i layer d'insieme: considerare l'efficienza dell'unione di questi strati (non si considerano uno ad uno quando si progetta un'interfaccia ma tutti insieme). È fondamentale anche considerare l'**esperienza dell'utente**.

3. Beacon Based Interfaces



Le **interfacce beacon based** sfruttano i **beacon**, dei dispositivi piccoli che sfruttano il BLE (Bluetooth Low Energy) che ha un range d'azione di circa 90 metri e lavorano a 2.4GHz. In questo tipo di interfaccia, vi sono tipicamente due attori:

- **Broadcaster**: Il beacon che manda in broadcast il proprio ID e i suoi servizi da lui offerti;
- **Observer**: Tipicamente lo smartphone che riceve i segnali inviati dal broadcaster.

ES. Entriamo in un supermercato, il nostro smartphone riceve un segnale dal beacon che si trova all'ingresso con il suo ID che corrisponde ad un particolare servizio (questo si può fare ad esempio con una tabella con coppie ID-Servizio nello smartphone) e l'utente può, ad esempio, collegarsi al sito o all'applicazione per supermercato per ricevere le ultime offerte.
Esiste una ulteriore classificazione in dispositivi:

- **Periferici** (Peripheral): Dispositivi che forniscono dati;
- **Centrali** (Central): Client che si iscrive ad un certo servizio, riceve i dati e li legge oppure può scrivere nuovi dati (ad esempio nel caso in cui si debba configurare il beacon).

Tipicamente le informazioni condivise non sono molte e la batteria può variare da 3 a 5 anni in base alle impostazioni del beacon. Le impostazioni che si possono cambiare riguardano:

- Frequenza: Ogni quanto si deve trasmettere il segnale. Maggiore è la frequenza, maggiore è il consumo di energia;
- Potenza: Maggiore è la potenza, maggiore è il range del beacon che tipicamente va da 5 a 70 metri. Anche qui, maggiore è la potenza, maggiore è il consumo di energia. Si noti che se si hanno più beacon, si può capire qual è quello più vicino a noi in base alla potenza;
- Dati da pubblicare.

Esistono vari beacon creati da varie aziende con i loro rispettivi protocolli. Un esempio è **iBeacon** creato da Apple in cui vengono usati tre valori che sono lo UUID (un ID associato ad un insieme di beacon, ad esempio

appartenenti ad una stessa applicazione), major (ha lo scopo di identificare e distinguere un gruppo) e minor (ha lo scopo di identificare e distinguere un singolo dispositivo). Le app possono cercare questi valori (iOS può ascoltare fino a 20 beacon ID) su un database e agire di conseguenza.
ES. Voglio installare dei beacon all'università, uso lo UUID come id per l'intera università, il major per identificare le singole facoltà e il minor per identificare i dispositivi singoli in ogni stanza.
Un altro esempio è Eddystone, protocollo sviluppato da Google che usa alcuni valori tra i quali: UID (un ID relativo al beacon) e un URL (che può redirezionare l'utente ad un sito web).

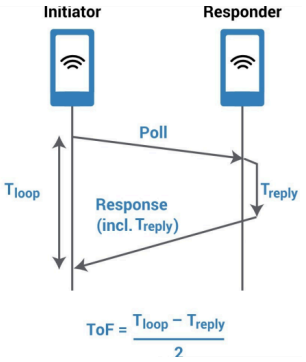
Nell'ottica dei beacon, parliamo di **regione** come l'area in cui viene ricevuto il segnale beacon. L'**operazione di monitoraggio** viene effettuata dal sistema operativo del telefono quando entra o esce dalla regione e le azioni (ad esempio, l'apertura di un'applicazione nell'esempio del supermercato) viene effettuata solo quando si entra o si esce da quell'area (nonostante il beacon continui a inviare segnali, ad esempio ogni secondo). Questa operazione di monitoraggio funziona in ogni contesto (cioè indipendentemente se l'applicazione è in foreground, background o viene chiusa) e consuma poca batteria.
Una altra operazione che si può effettuare è il **ranging** che è un'azione intrapresa in prossimità del beacon ed effettuata in foreground che consente all'applicazione di ottenere informazioni più dettagliate quali la potenza ricevuta e questo permette di stimare, ad esempio, la distanza tra smartphone e beacon (ES. Utile quando si vuole capire quando un oggetto si sta muovendo da noi). Questo consuma una maggiore quantità di batteria.

Quanto consuma?
- Beacon: Dipende dalle impostazioni di frequenza e potenza e dai sensori integrati;
- Smartphone: Dipende dal monitoraggio o dal ranging.

Quali sono alcuni utilizzi?
- Coinvolgimento del cliente;
- Navigazione all'interno di negozi, aziende, hotel;
- Monitoraggio delle persone (con i badge). Ad esempio posso mettere un beacon dentro il mio badge e quando mi avvicino alla porta del lavoro quest'ultima riconosce il mio beacon e mi fa entrare.

4. Ultra Wide Band

Ultra Wide Band (UWB) è un **tecnologia radio** che usa un **livello di energia molto basso** per comunicazioni a corto raggio e ad alta larghezza di banda su un'ampia porzione dello spettro radio. Alcune applicazioni sono raccolta dei dati dei sensori, localizzazione di precisione e applicazioni di tracciamento.
Si tratta quindi di una tecnologia per la trasmissione di informazioni su un'ampia larghezza di banda (>500 MHz) dove la trasmissione avviene senza interferire con la banda stretta convenzionale in maniera da non interferire con altri dispositivi. Un sistema radio UWB può essere utilizzato per determinare il "tempo di volo" (cioè quanto impiega l'onda per percorre una certa distanza) della trasmissione a varie frequenze. Con una tecnica di misurazione bidirezionale simmetrica cooperativa, le **distanze possono essere misurate con alta risoluzione e precisione**.



UWB prevede l'utilizzo di due entità che sono l'iniziatore e il risponditore. L'iniziatore invia un messaggio al risponditore che invierà una risposta. T_{reply} è il tempo impiegato dal risponditore per generare una risposta mentre T_{loop} è il tempo totale che passa dall'inizio del messaggio all'arrivo della risposta. Il tempo di volo (time of flight) si può ottenere sottraendo T_{reply} e T_{loop} e dividendo il risultato per 2 (questo permette di capire qual è la distanza tra i due dispositivi).

Alcuni esempi di utilizzo della tecnologia UWB sono:
- Posizione in tempo reale:
  - Peer-to-peer fine ranging: Consente a molte applicazioni di conoscere la distanza relativa tra due entità (si pensi anche agli AirTag o gli Smart Tag);
  - Le sue capacità di precisione e la bassa potenza lo rendono adatto per ambienti sensibili alle radiofrequenze (dove ci possono essere interferenze con altri dispositivi sensibili), come gli ospedali;
  - Chiavi della macchina;
- Applicazioni radar;
- Segnalazioni nelle applicazioni industriali.

5. Chatbot

Con **chatbot** si intende un programma che interagisce con gli utenti utilizzando il linguaggio naturale. Lo scopo del chatbot è rendere l'interazione uomo-macchina più naturale, permettendo all'essere umano di utilizzare il proprio linguaggio per esprimere interessi, desideri e richieste **simulando una conversazione umana con la macchina**. A livello implementativo, un chatbot è composto da un **modello di linguaggio** necessario per capire le richieste dell'utente e da un algoritmo in grado di formulare la risposta e di fornirla all'utente.
ES. Un esempio di chatbot lo si può trovare nei sistemi di e-commerce che tipicamente li usano per fornire assistenza al cliente. Un utente può avviare la chat tipicamente cliccando su un bottone e si aprirà una finestra nella web application senza installare applicazioni esterne.

Storicamente i chatbot sono stati utilizzati per il "Turing Imitation Game" in cui l'obiettivo era capire se un robot potesse simulare il comportamento umano. Il gioco consisteva nel capire se il proprio interlocutore fosse un essere umano o un robot in base alle risposte che dava alle domande che gli venivano poste.
L'uso dei chatbot oggi si è espanso in tantissimi ambiti come per esempio l'educazione, l'intrattenimento, il recupero di informazioni, business o e-commerce. In ambito commerciale i chatbot vengono utilizzati per fornire assistenza ai clienti attraverso diversi canali di comunicazione come la messaggistica e assistenza vocale mentre in ambito di ricerca si vuole creare un bot in grado di tenere un discorso con un essere umano al fine di costruire una conversazione di qualità sufficiente e che risulti utile per l'interlocutore. I colossi dell'informatica mondiale hanno integrato un chatbot nei loro sistemi di personal assistant per esempio Siri, Google Now, Cortana, Amazon Echo. Tutti sono accessibili tramite interfaccia vocale, ma solo alcuni sono accessibili tramite interfaccia testuale come Siri, Google Now e Cortana.

Più in dettaglio, un chatbot è un servizio accessibile attraverso un'**interfaccia di comunicazione** che può essere vocale (telefonate) o testuale (chat). La complessità di implementazione di un chatbot dipende da cosa si vuole utilizzare per gestire il modello di linguaggio, infatti può essere **basato su regole**, utilizzando per esempio le **espressioni regolari**, o può utilizzare l'**intelligenza artificiale** (in questo caso la complessità dell'implementazione aumenta). La maggioranza dei chatbot disponibili al pubblico utilizza l'interfaccia testuale, poiché diverse app di messaggistica istantanea come Telegram e Messenger offrono la possibilità di creare il proprio bot per distribuire il proprio servizio e questo ha dato una grande visibilità a questo tipo di servizio. Tipicamente le aziende costruiscono chatbot dove l'utente è attivo (come ad esempio le applicazioni di messaggistica) anziché creare nuovi ecosistemi o piattaforme che l'utente dovrebbe scaricare, imparare ad usare o in cui potrebbe non sentirsi a proprio agio.

Una caratteristica dei bot è quella di poter ricevere i messaggi dall'utente o da gruppi di utenti (es. i poll su Telegram che si possono aprire con il chatbot e a cui possono partecipare più persone) (in maniera anonima), pensare alla risposta e rispondere adeguatamente **in modo asincrono** (ad esempio, il bot può inviare messaggi ad un certo orario per ricordarci di una cosa e non necessariamente in tempo reale quando interagiamo con lui). Questa semplice funzionalità permette ai bot di comunicare simulando una conversazione reale.

Generalmente la conversazione con un chatbot **non è stateful**, cioè non salvano lo stato della conversazione. Ad esempio, se chiediamo al chatbot di inserire un appuntamento alle 15:00 e subito dopo gli chiediamo di inserirne uno un'ora più tardi, il chatbot non sarà in grado di farlo in quanto non avrà salvato l'informazione sull'orario. Per rendere il chatbot più avanzato si può costruire una **conversazione stateful** tenendo traccia delle situazioni o delle risposte dell'utente in variabile e scegliere quindi le risposte in base allo stato o al contesto.
Alcuni accorgimenti quando si costruisce un chatbot sono:

- Fai sapere agli utenti in anticipo che stanno parlando con una macchina;
- Mostra agli utenti comandi (e pulsanti) per continuare la conversazione;
- Guida la conversazione il più possibile per aiutare l'utente;
- Non è detto che il bot riesca a rispondere a tutte le domande e questo va pianificato (ES. ammettere problemi come la domanda non capita, prevedere possibili azioni successive, offrire un contatto con una persona reale).

6. Voice User Interfaces

Voice User Interfaces (VUI) è un'interfaccia che permette all'utente di comunicare con il sistema utilizzando la voce attraverso un **Voice Command Device (VCD)** cioè un device fisico che permette la comunicazione vocale. Una VUI ha bisogno di un VCD per essere attivata, e una stessa VUI può essere attivata su VCD differenti (ES. Si pensi ad Amazon Alexa che può essere su Smartphone, TV, Echo Dot, ecc..).
Alcuni esempi sono Google Home (VCD) e Google Now (VUI), Amazon Echo (VCD) e Alexa (VUI), Apple Home Pod (VCD) e Siri (VUI). Queste interfacce sono anche conosciute come **Intelligent Personal Assistants** (IPAs).

Le interazioni con le VUI possono essere di 3 tipi:

- **Command and Control**: l'utente utilizza la propria voce per dare comandi e controllare il sistema o il device. Questo tipo di interazione è one-way, e trasmette solo input;
- Nel secondo tipo di interazione l'utente trasmette l'input inserendo comandi da tastiera e il sistema risponde all'utente utilizzando una voce sintetizzata quindi l'input è manuale, l'output è vocale;
- **Spoken Dialogue System**: Il terzo tipo di interazione è two-way cioè sia utente che sistema comunicano utilizzando il linguaggio naturale.

L'**interazione con una VUI avviene sotto forma di dialogo**. Esistono diversi tipi di dialogo tra VUI e utente:

1. Si lascia il controllo della conversazione al sistema quindi attraverso coppie di domande-risposte l'utente può richiedere di eseguire un task al sistema (il nostro chatbot);
2. La seconda tipologia di dialoghi consente all'utente di formulare richieste più articolate e di effettuare domande. Spesso dopo un input il sistema risponde con la domanda "come posso aiutarti?" e una volta acquisita la richiesta il sistema calcola la mossa successiva ottimale;
3. La terza tipologia di dialogo permette all'utente di richiedere task complessi al sistema e soprattutto di interromperlo (barge-in) per effettuare una nuova richiesta e cominciare un nuovo task.

Il **flusso di esecuzione** (**pipeline**) di una VUI inizia con l'individuazione da parte del sistema di una richiesta di interazione da parte dell'utente. In alcuni casi il sistema deve anche riconoscere l'utente che sta facendo la richiesta tra una gruppo di possibili utenti (**Speaker Diarization**). Successivamente il sistema, utilizzando l'**Automatic Speech Recognition (ASR)**, deve capire cosa ha detto l'utente e grazie al componente di **Natural Language Understanding (NLU)**, superare l'ambiguità del linguaggio naturale, capire gli intenti dell'utente e tradurlo in un linguaggio comprensibile dalla macchina: questa operazione viene identificata come come **dialog act**. Successivamente il **Dialog Manager (DM)** sceglie l'azione appropriata in base alla richiesta estrapolata dal NLU, tenendo anche conto dei fattori come lo **stato corrente del dialogo** e i **dialoghi recenti**. Il componente **Natural Language Generation (NLG)** trasforma la risposta generata dal DM in una risposta in linguaggio naturale e in conclusione la risposta viene restituita all'utente utilizzando la sintetizzazione **text-to-speech (TTS)**.

Per quanto riguarda la **progettazione (design) di una VUI** è un compito che conta al suo interno più ambiti come l'informatica per (progettare la parte implementativa e rendere il sistema "intelligente"), linguistica (per fare in modo che il sistema comunichi correttamente con l'utente) e psicologia (per comprendere al meglio le richieste dell'utente). La maggior parte delle componenti di una VUI **sono comuni ai componenti delle interfacce visuali**, per esempio in entrambe le interfacce bisogna scegliere quale task può supportare il sistema, scegliere la struttura del dialogo con l'utente, decidere con quali comandi e features implementare il task, lasciare che l'utente specifichi cosa poter fare e non fare e permettere al sistema di dare feedback all'utente.
Gli utilizzatori delle VUI possono essere divisi in due macro-categorie:

- Utenti standard: Necessitano di un sistema facile da utilizzare che fornisca aiuti all'utente;
- Utenti esperti: Necessitano di un sistema basato sulla produttività che dia all'utente la possibilità di inserire diversi pezzi di informazione in una singola interazione con un qualsiasi ordine o combinazione.

Il **vantaggio** principale delle VUI è che **lascia liberi i canali standard dell'utente** (occhi e mani) e questo le rende appropriate per le azioni di routine dell'utente. Mentre gli **svantaggi** comprendono le comunicazioni all'utente di informazioni sul sistema stesso: quando l'utente può parlare e cosa può fare l'interfaccia; la percezione che l'utente ha del sistema, l'utente infatti non sa se l'interfaccia ha capito la richiesta finché l'output non viene fornito, questo porta il risolvere gli errori di interpretazione una parte centrale della progettazione.

Alcune linee guida per la progettazione delle VUI sono:

- **Mapping tra sistema e mondo reale**: Il sistema dovrebbe fornire un'interazione simile al **modello mentale dell'utente** quindi non deve richiedere log o dare informazioni tecniche che confonderebbero

l'utente. Quindi un sistema utilizzato nella vita quotidiana non dovrebbe avere uno stile di comunicazione completamente alieno all'utente.

- **Libertà e controllo dell'utente**: Utilizzando una VUI l'utente sente di **avere meno controllo della situazione rispetto all'utilizzo di un'interfaccia grafica**. Per garantire il senso di controllo all'utente, la VUI dovrebbe rendere chiaro quando l'utente può fornire le informazioni al sistema e avere delle parole di escape in grado di riportare il sistema ad uno stato di default (**interrupt dialog**).
- **Riconoscimento piuttosto che richiamo (recall)**: Gli utenti spesso non sono a conoscenza di come strutturare il proprio discorso su un'interfaccia utente vocale. Una VUI può essere implementata tramite due tipi di riconoscimento dei comandi:
    - **Recognition**: Il sistema riceve l'input dall'utente, cerca in memoria un match con l'input e risponde;
    - **Recall**: L'utente non deve dare un input preciso ma può descrivere un task e il sistema deve capire cosa sta chiedendo l'utente e in seguito rispondere.

  Utilizzare la recognition è più conveniente per una VUI in quanto l'utente spesso non sa come descrivere in modo preciso un task ed avere dei comandi precisi da dare al sistema lo aiuta a ricordare i task possibili ma chiaramente, sistemi differenti possono avere comandi differenti e l'utente li dovrebbe ricordare.
- **Minimalismo nella progettazione e nel dialogo**: In generale una VUI dovrebbe seguire una progettazione minimalista, non avere tanti comandi in modo da permettere all'utente di mantenerli in memoria facilmente, e le conferme che l'utente dovrebbe dare al sistema dovrebbero essere implicite (non c'è bisogno che il sistema chieda esplicitamente conferma per i prossimi passi da eseguire).
- **Consentire agli utenti di riconoscere e recuperare dagli errori**: L'utente dovrebbe essere sempre in grado di recuperare da un errore effettuato. Si noti che per gli utenti non esperti è difficile recuperare da errori commessi senza creare altri errori e, a causa dell'inesperienza, risulta complesso anche tornare ai menù precedenti. Colui che progetta il sistema dovrebbe tener conto che l'interfaccia dovrà ricevere informazioni da un essere umano e, solitamente un uomo dopo aver fatto una richiesta poco chiara, se non riceve velocemente un feedback è portato a riformulare la richiesta o a ripeterla a voce più alta. Per questo motivo il sistema non dovrebbe permettere la modifica di query effettuate.
- **Fornire aiuto e documentazione**: Una VUI dovrebbe contenere la documentazione adatta per gli utenti che la utilizzano, questa dovrebbe aiutare gli utenti contestualmente durante l'interazione in modo da **diminuire il peso cognitivo** che grava sull'utente stesso fornendogli aiuti chiari.
- **Influenza del contesto sull'interazione vocale**: Una VUI dovrebbe poter attivarsi con un tono di voce basso e confortevole per l'utente, questo perché una persona spesso non trova confortevole parlare ad alta voce in pubblico.

7. Haptic Interaction

Le **interazioni di tipo aptico** coinvolgono i sensi di vista e udito ed avvengono grazie alla combinazione di movimenti e tocchi. L'obiettivo è di **manipolare oggetti fisici e/o virtuali** attraverso **azioni controllate dal tatto**. L'interazione visiva (es. guardare i tasti del telecomando quando si clicchiamo) molto spesso è un'operazione poco pratica o addirittura impossibile in quanto ci sono delle **necessità di tipo straordinario** (che riguardano pochi utenti) e **di tipo ordinario** (che riguardano la maggior parte degli utenti): tra le necessità straordinarie possiamo notare la cecità, un impedimento che rende difficile per l'utente interagire, mentre tra le necessità ordinarie possiamo notare il bisogno di poter interagire senza guardare il telefono (es. mentre si guida) oppure la possibilità che ci si trovi in ambienti con poca luce (es. bisogna toccare l'ambiente come muri o mobili per capire dove ci si trova). Nelle interazioni haptic, si hanno quindi due momenti che sono **effettuare una certa azione** e **avere un feedback**.

ES. Alcuni esempi sono i rilievi posizionati tra i tasti fisici dei vecchi telefoni cellulari che permettono di identificare quale sia il tasto numerato con il numero 5 (e di conseguenza la posizione di tutti gli altri tasti) e i rilievi posizionati sulle strade per direzionare le persone cieche e permettere loro di rimanere al sicuro sulla carreggiata.

Le informazioni possono essere ricevute tramite due livelli:

- **Fisico**: Ricavare informazioni dei diversi tipi di stimoli grazie al sistema nervoso periferico;
- **Percettivo**: L'informazione viene portata al cervello e viene interpretata;

Ci sono diversi tipi di sensi:

- **Senso cutaneo**: Essere al corrente della stimolazione dei recettori tattili (es. carpire informazioni su un certo oggetto che stiamo toccando come ad esempio un muro);

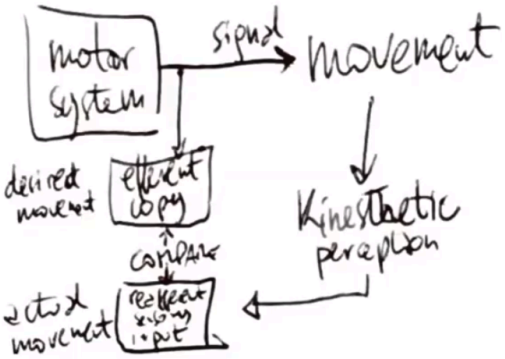
- **Senso cinestetico**: Essere al corrente della relativa posizione del corpo.

La **percezione tattile** (tactual perception) coinvolge uno o entrambi questi sensi, ma dipende solamente dal senso cutaneo (il corpo non si muove). La **percezione cinestetica** dipende solamente dal senso cinestetico (diciamo che è puramente teorica e un esempio lo si ha quando si fanno uso di anestetici in cui non si ha il senso del tatto ma comunque sappiamo dove il nostro corpo di trova). La **percezione aptica** rappresenta l'insieme dell'informazione derivante da entrambi i sensi.

Ci sono varie modalità tattili:

1. Percezione tattile: Solamente informazione cutanea;
2. Percezione passiva cinestetica: Afferente alla cinestetica;
3. Percezione aptica passiva: Informazione cutanea e afferente alla cinestetica (otteniamo informazioni dal mondo esterno, es. toccando oggetti, e li elaboriamo);
4. Percezione cinestetica attiva: Afferente alla cinestetica e alla copia di efferenza (**efference copy**);
5. Percezione haptic attiva: Informazione cutanea, afferente alla cinestetica e alla copia di efferenza;

La **efference copy** è una copia interna di un segnale uscente che produce un movimento: essa permette al cervello di prevedere gli effetti di un'azione e protegge la percezione di particolari effetti autoindotti.
Quindi, voglio effettuare un certo movimento (il desiderio di effettuare qual movimento è la efference copy), lo effettuo e vedo i risultati del movimento (grazie alla percezione cinestetica) e, una volta che ho l'informazione sul movimento realmente effettuato e quello che volevo effettuare li posso comparare per eventualmente correggere il movimento.
La differenza tra le varie modalità è che le prime 3 sono totalmente passive (si ottengono informazioni dal mondo esterno senza reagire, senza un obiettivo) mentre le ultime 2 sono attive (voglio effettuare un movimento che è il mio obiettivo quindi ricevo un feedback per perseguirlo ed eventualmente correggerlo).



Quando avviene un **conflitto tra i sensi**, né tatto, né vista sono predominanti uno sull'altro, bensì si crea un compromesso tra i due. La cooperazione tra essi è altamente individuale e dipende molto dal task che bisogna effettuare (ogni persona potrebbe effettuare preferire tatto o vista in base al task da effettuare ma tipicamente c'è un compromesso tra i due).
Per quanto riguarda la differenza di informazioni percepite, le **informazioni strutturali** recepite in modo aptico risultano essere più soggette ad errori perché ricevute più lentamente, mentre arrivano in modo molto veloce con la percezione visiva. Le **quantità di sostanza** (come durezza, texture etc.), al contrario, sono recepite velocemente dalla percezione aptica e risultano affidabili, mentre spesso, con la percezione visiva, non si riesce ad avere alcuna informazione riguardo a queste dimensioni.

	Haptic perception	Visual perception
Structural information	Slow Error prone	Fast
Substance dimensions (e.g. hardness, texture…)	Quick Reliable	–

L'interazione aptica prevede uno standard per quanto riguarda i **simboli**:
Le **linee visuali** vengono sostituite da **linee tattili rialzate** (al di sopra della superficie): ciò comporta problemi per quanto riguarda la **tracciabilità** (quanto facilmente possono essere tracciate), la **distinzione dello spessore**, il tracciare le linee nonostante **le intersezioni** e la **distinzione tra linee morbide e dure** (ad esempio le tratteggiate) per tracciare la performance. Ad esempio, se la linea è più spessa di un dito (entrambi i lati non sono percepiti), la performance diminuisce. Inoltre, se ci sono più di 8-10 simboli lineari, potrebbero avvenire degli errori di somiglianza tra i simboli.
I **simboli a punto** (anche questi rialzati) permettono di esplorare con il minimo movimento della punta delle dita. Non sono solo punti, ma ci sono diverse forme (es. una stella o un cerchio). I problemi di questi simboli riguardano quanto bene vengano percepiti in contrasto con lo sfondo (**figure-ground problem**). Inoltre, è necessario dover decidere tra **punti rialzati o incisi** (quelli rialzati sono più semplici da riconoscere) e tra **leggibilità e significato** (se miglioriamo la leggibilità, si potrebbe diminuire il significato e quindi c'è bisogno di un apprendimento del simbolo).
I **simboli ad area** permettono l'utilizzo di una **texture** o di uno **schema tattile** per fornire informazioni. Qui è necessario analizzare il compromesso tra la grandezza della texture e la spaziatura dell'area, la velocità con la quale l'utente possa effettuare le sue azioni attraverso l'area in questione e la forza da applicare che altera la percezione del simbolo.

Ci sono delle limitazioni nell'utilizzo della percezione tattile:

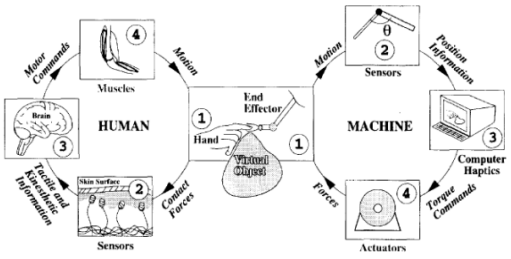
- **Risoluzione spaziale**: La distanza per la quale due punti vengono percepiti come uno solo (dipende da sfocatura, densità, grandezza e sensibilità dei recettori nella punta del dito, numero di neuroni nell'area di proiezione corticale (es. se tocco due punti vicini con la punta del dito li posso percepire entrambi, se li tocco con il palmo della mano potrebbero apparire come un punto unico));
- **Risoluzione temporale**: Abbiamo bisogno di una finestra di tempo (2-40 ms) per ottenere uno stimolo;
- **Interazioni tra stimoli ampiamente distanziati**: Le informazioni potrebbero non essere riconosciute. Ad esempio, se tocchiamo due punti distanziati utilizzando quindi entrambe le mani, potremmo percepire un terzo punto fantasma nel mezzo;
- **Attenzione limitata**: Potremmo non essere capaci di concentrarsi quando c'è troppa informazione.

Se usati insieme, possono essere riconosciuti fino a 8 modelli tattili. L'altezza può essere utilizzata come metodo di filtraggio.

Alcuni metodi per la stimolazione tattile sono la deformazione della pelle, la vibrazione, stimolazione elettrica, allungamento della pelle (con la corrente elettrica si può riuscire a generare allungamenti della pelle che l'utente può percepire), attrito (micro allungamenti della pelle quando si tocca qualcosa) e temperatura.

È possibile **forzare le interfacce aptiche** misurando le posizioni e la forza imposta della mano dell'utente e/o delle altre parti del corpo e mostrando forza e posizione all'utente. Per mostrare questo tipo di dati si devono computare e generare le forze innescate dalle interazioni che l'utente ha con i vari oggetti virtuali, basandosi sulla posizione del dispositivo.
ES. Immaginiamo che vogliamo riprodurre la sensazione che si ha toccando la superficie di un oggetto come una scrivania. Toccandola con una stilo abbiamo un certo tipo di sensazione che non si ha toccando un tablet. In qualche modo la stessa sensazione la possiamo riprodurre anche sul tablet utilizzando un certo tipo di forze che agiscono nello stilo e che permettono di simulare la vibrazione.

L'**interazione aptica** funziona in questo modo: L'umano può ricevere delle sensazioni quando tocca qualcosa e il cervello può elaborare queste sensazioni e mandare informazioni ai muscoli in maniera tale da generare un certo movimento in risposta. Allo stesso modo, il sistema ha dei sensori che possono influenzare lo stato del dispositivo. Queste informazioni vengono elaborate e possono essere utilizzare per produrre delle forze (tramite gli attuatori) per reagire alla forza generata dall'umano.
ES. Pensa ad un volante per giocare ad un videogioco di corsa. L'utente può interagire con il volante ruotandolo a destra e sinistra. Quando l'utente genera troppa forza (gira troppo a destra o a sinistra), il volante genera una forza (si blocca) e questa è un'interazione aptica.



Le norme da seguire per costruire una buona interfaccia aptica sono:

- L'interfaccia deve operare all'interno delle capacità e dei limiti umani;
- Le approssimazioni delle interazioni tattili nel mondo reale sono determinate dai limiti delle prestazioni umane;
- Il movimento dovrebbe essere libero quindi l'inerzia e attrito dovrebbero essere ridotti e non ci dovrebbe essere nessun vincolo di movimento (es. se si indossa un guanto per la realtà virtuale, l'utente non lo deve percepire e non ci deve essere alcuna costrizione);
- Bisogna garantire ergonomia e comfort quindi ridurre la fatica, il dolore, il disagio, etc..;
- Gamma, risoluzione e larghezza di banda adeguate (es. l'utente non dovrebbe essere in grado di passare attraverso oggetti rigidi superando il raggio di forza, si pensi ad un esoscheletro)
- Gli oggetti solidi devono essere rigidi: Ad esempio, gli utenti possono giudicare in modo coerente la rigidità relativa di diversi muri virtuali anche se non sono mai rigidi come i muri reali a causa delle limitazioni dell'hardware.

Il **rendering aptico** è il processo di calcolo e generazione di forze in risposta alle interazioni con oggetti virtuali, in base alla posizione del dispositivo. Il rendering di un oggetto può essere visto come lo spingere un certo oggetto: più si spinge un certo oggetto, maggiore è la forza che ti spinge via. Ciò permette alle superfici di assumere l'**effetto di solidità**. Tipicamente, quando tocchiamo un oggetto inviamo informazioni sulla posizione e dell'orientazione e avviene quindi una rilevazione della collisione. Il sistema comunica quindi con il database per

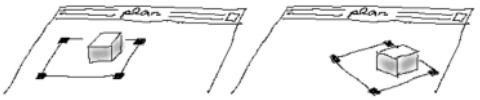
capire qual è la geometria o il materiale dell'oggetto e in base a questo il sistema restituisce una risposta alla collisione, quindi la forza o il movimento torcente (torque) che vengono restituiti all'utente tramite l'attuatore (es. tocchiamo un bottone e questo genera una certa forza in base a quanto lo spingiamo). Il senso del tatto umano è abbastanza sensibile da richiedere una velocità di elaborazione di almeno 1000 Hz per quanto riguarda il rendering aptico.

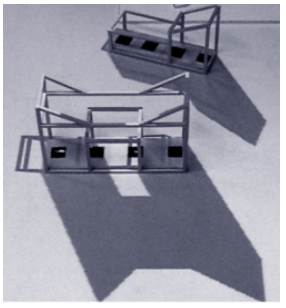
Le applicazioni delle interazioni aptiche sono molteplici: in particolare, in campo medico vengono utilizzate per la visualizzazione e la modellizzazione dei tessuti, per allenare, per effettuare interventi chirurgici a distanza, per la riabilitazione. Per quanto riguarda la modellizzazione tridimensionale, troviamo usi nella prototipazione virtuale, nella scultura virtuale di oggetti 3D (la superficie dell'oggetto può essere percepita già durante la modellizzazione).

8. Tangible User Interfaces

Prima di tutto, ricordiamo che quando parliamo di interfaccia tattile intendiamo ricavare informazioni con il senso del tocco (es. quando tocchiamo il telefono con le dita) mentre con un'interfaccia aptica non riceviamo solo un feedback sulla nostra pelle ma, tale feedback è collegato anche alla posizione del nostro corpo. L'interazione tangible permette di interagire con un oggetto toccando e muovendo l'intero oggetto (quindi non si tratta tanto della sensazione che proviamo quando tocchiamo un oggetto ma di come lo muoviamo e interagiamo con esso). Si dicono **interfacce utente tangibili**, in inglese tangible user interface (TUI), quelle interfacce uomo-macchina che consentono di interagire con un sistema informatico manipolando degli oggetti fisici tangibili, quindi nel tangible interaction non interagisco con il computer ma con gli oggetti. È un altro modo di interagire, la quale sfrutta la nostra capacità naturale di comprendere e manipolare forme fisiche sfruttando allo stesso tempo la potenza della simulazione computazionale. L'obiettivo è **arricchire le interazioni comuni dell'uomo** aggiungendo nuovi effetti attraverso l'uso di dispositivi adeguati. I dispositivi che permettono questo arricchimento hanno un sistema embedded che permette loro di svolgere il lavoro del computer per estendere le interazioni dell'uomo con la macchina.

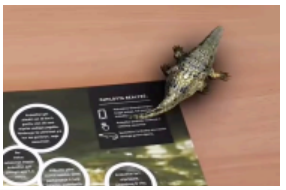
Un primo prototipo di tangible UI è la **Marble Answering Machine**, un tool che avrebbe dovuto sostituire la segreteria telefonica, esso consisteva nell'associare i nuovi messaggi in segreteria e delle biglie: quando si voleva riascoltare il messaggio bastava semplicemente mettere la biglia associata in un apposito scomparto mentre rimettendo apposto la biglia il messaggio veniva cancellato.
Altri esempi di tangible UI sono le **Graspable Interface**, che consistono di una superficie tangible, e degli oggetti fisici ai quali vengono associati degli oggetti virtuali, una volta posizionati gli oggetti fisici sulla superficie in essa compariva l'oggetto virtuale e modificando la posizione dell'oggetto fisico anche quella dell'oggetto virtuale veniva modificata. Un esempio pratico di questa interfaccia è la **tangible bits**, utilizzata per simulare le ombre nate da edifici e costruzioni durante il giorno. Quindi le ombre vengono generate in modo digitale su una certa superficie tenendo in considerazione la forma, la posizione e la dimensione dell'edificio, esso rappresentato da un oggetto fisico. La tangible bits può mostrare anche come il vento viene incanalato nelle strade di una città sempre in base agli edifici (quindi proiettare linee che rappresentano la direzione del vento).
Esistono molti progetti di tangible UI come il Shape-Shifting Digital Clay del MIT che attraverso dei blocchi mostrano il movimento di una corrente d'acqua e il sandscape che, utilizzando la sabbia permette di rappresentare le differenze di altitudini delle forme create.



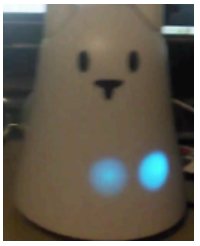


In definitiva le Tangible User Interface sono **sistemi che danno forma fisica a informazioni digitali**, utilizzando oggetti fisici sia per rappresentare gli output sia come controlli gli input dei mezzi computazionali.

L'**Affordance** è l'insieme delle proprietà di un oggetto che suggerisce ad un essere umano le azioni specifiche per manipolarlo: ad esempio, un pomello permette di essere tenuto in mano e girato (non ci verrebbe mai in mente di toccarlo e basta), un bottone permette di essere premuto, etc. Con questa proprietà, emerge la necessità di distinguere le affordances reali e quelle percepite (ad esempio un bottone in una GUI). L'Affordance apre a molte possibilità riguardo l'utilizzo delle TUI in altri campi: ad esempio, si può creare una Tangible augmented reality (combinando TUI e Augmented Reality (AR)) in cui oggetti virtuali sono associati a oggetti fisici, oppure in cui si possono associare dei video a marchi visivi che permettono di effettuare diverse azioni su di essi.



Si pensi ad esempio ad una sorta di libro interattivo che, se inquadrato con una fotocamera, può mostrare un oggetto in rilievo in base alla pagina (es. pagina sui cani, mostra un cane in 3D che si muove sulla relativa pagina). Altri esempi possono trovarsi in:



- Interazioni tangible da tavola: Combina superfici multi touch interattive e TUI attraverso una videocamera, un proiettore o uno schermo traslucido (c'è anche una versione iPad);
- Oggetti d'ambiente: Un fiore che sboccia quando un collega è libero, modificare l'audio e il video di una finestra multimediale a seconda della prossimità dell'utente, il nabaztag bunny che attraverso l'interfaccia tangible fornisce informazioni all'utente (come il meteo, le email in arrivo, l'orario e così via muovendo le orecchie o illuminando dei led), cuscini interattivi;
- Promemoria ed etichette tangible: Piazzare un oggetto particolare in prossimità di un lettore innesca azioni specifiche (es. piazzare i souvenir su una superficie apre l'album di foto associato).

Il **modello utilizzato dalle TUI** è **MCRit (Model-Control-Representation)**, sia per tangible che intangible e permette di integrare rappresentazioni e controlli fisici in interfacce utente tangible.
Oggetti tangible sono accoppiati tramite funzionalità digitali a dati digitali (**computational coupling**): Gli oggetti tangible rappresentano il mezzo del controllo interattivo. Spostare e manipolare oggetti è la forma di controllo principale quindi, spostando un oggetto si ha la percezione di dare un input al sistema e ci si aspetta un output digitale. Possiamo dire che questi oggetti sono associati percettivamente con rappresentazioni digitali (ad esempio, audio ed effetti visivi). Lo stato degli oggetti tangible rappresenta l'intero stato del sistema (**representational significance**). In questo modo, il sistema rimane parzialmente utilizzabile se la corrente salta.

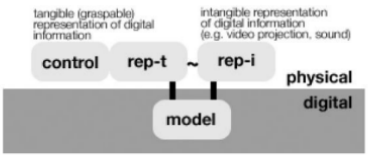


Fig. 5.1 The MCRit model (redrawn based on [238]) eliminates the distinction between input and output devices.

Le TUI possono essere classificate in vari modi:

- Superfici interattive (ad esempio tangible bits);
- Constructive assembly (ad esempio sandscape);
- Token + vincoli (ad esempio marble answering machine).

Una **grande limitazione delle TUI** è la specifica di comandi complessi (astratti, concatenati o che agiscono su un insieme di oggetti) è generalmente un punto debole di tutte le interfacce di manipolazione diretta. In particolare si può notare la difficoltà nell'annullare gli errori in quanto non esiste una specifica funzione di annullamento quindi, se vuole rimediare da un errore, lo stato precedente del sistema deve essere ricordato e ricostruito manualmente.

Alcuni domini di applicazione sono: Apprendimento, problem Solving e Planning, visualizzazione delle informazioni, programmazione tangible, intrattenimento, gioco ed intrattenimento a scopo educativo (edutainment), musica e spettacolo, social communication e tangible reminders e tags.

Le **tecnologie principali** per lo sviluppo delle tangible interface sono:

- RFID (Radio-frequency IDentification): Attraverso radiofrequenze si mandano informazioni (tags) al sistema, ed esso risponde alle interrogazioni;
- Computer Vision: Attraverso opportuni dispositivi si può interagire con la realtà "aumentata" con informazioni, immagini o video.
- Micro-controlli, Sensori e Attuatori: Attraverso i micro-controllori e i sensori viene registrato l'input e attraverso gli attuatori viene inviato l'output.

Le TUI hanno proprietà ben precise ed ogni tecnologia ha delle caratteristiche che soddisfano le proprietà in modo differente. Nella tabella in basso vengono riportate le caratteristiche di ogni tecnologia e in che modo soddisfano ogni proprietà:

- **Proprietà fisiche rilevate**: Quali proprietà fisiche possono essere rilevate utilizzando una particolare tecnologia?

Property	RFID	Computer Vision	Microcontrollers
Physical properties sensed	Identity, presence.	Identity, presence, shape, color, orientation, position, relative position, and sequence.	Light intensity, reflection, motion, acceleration, location, proximity, position, touch, temperature, gas concentration, radiation, etc.

- **Costo**: Qual è il costo relativo dei diversi componenti che compongono una tecnologia di rilevamento?

Cost	Tags are cheap and abundant. The cost of readers varies, but is generally inexpensive (short distance readers).	Fiducial tags are practically free. The cost of high-quality cameras continuously decreases. A high-resolution projector is relatively expensive.	Generally inexpensive. The cost of sensors and actuators vary according to type.

- **Prestazioni**: Il sistema è efficiente in termini di elaborazione e tempi di risposta? Quali fattori influenzano l'efficienza del sistema?

Performance	Tags are read in real time, no latency associated with additional processing.	Dependent on image quality. Tag-specific algorithms are typically fast and accurate. A large number of tags or low-quality image take longer processing. Motion blur is an issue when tracking moving objects.	Generally designed for high-performance. Stand-alone systems typically perform better than computer-based systems.

- **Estetica**: In che misura una tecnologia di rilevamento influisce sull'aspetto di un oggetto? L'utente può identificare quali oggetti o proprietà vengono rilevati e quali no?

Aesthetics	Tags can be embedded in physical objects without altering their appearance.	Fiducial marker can be attached to almost any object (ideally to its bottom).	Sensors and actuators can be embedded within objects. Wires may be treated to have a minimal visual affect.

- **Robustezza e affidabilità**: Il sistema può eseguire le funzionalità richieste per un lungo periodo di tempo? Il sistema può resistere a condizioni mutevoli?

Robustness and reliability	Tags do not degrade over time, impervious to dirt, but sensitive to moisture and temperature. Nearby technology may interfere with RFID signal. Tags can only be embedded in materials opaque to radio signals.	Tag-based systems are relatively robust and reliable. However, tags can degrade over time. Detection only within line of sight.	Typically designed for robustness and reliability. Batteries need to be charged. The robustness and reliability of sensors and actuators vary. Wiring may need to be checked.

- **Configurazione e calibrazione**: Cosa è necessario per portare il sistema in una modalità utilizzabile?

Setup and Calibration	Minimal. No line of sight or contact is needed between tags and reader. The application must maintain a database that associates ID with desired functionality.	Address a variety of factors including occlusion, lighting conditions, lens setting, and projector calibration.	Connect microcontroller to computer; wire sensors and actuators; embed hardware in interaction objects; fabricate tailored interaction objects to encase hardware.

- **Scalabilità**: Il sistema può supportare un numero crescente di oggetti o utenti?

Scalability	The number of simultaneously detected tags is limited by the reader. No practical limitation on the number of tagged objects.	The maximal number of tracked tagged objects depends on the tag design (typically a large number).	Typically constrained by the number of I/O ports available on a microcontroller.

Le **TUI hanno delle limitazioni importanti**, non sono scalabili (spesso si è in possesso di pochi dati e in più per la loro natura hanno bisogno di molto spazio per essere utilizzate), non sono versatili (ogni tui è progettata per un singolo utilizzo e non può generalizzare a più di quello), l'utilizzo stesso della tui può risultare faticoso (c'è bisogno di effettuare un movimento e la dimensione degli oggetti e i continui spostamenti possono creare disagio) e non sono malleabili (siccome gli oggetti fisici rappresentano almeno parzialmente lo stato del sistema se viene effettuato uno spostamento che di conseguenza cambia lo stato del sistema esso non può essere annullato automaticamente, non è presente la l'undo, quindi per tornare allo stato precedente bisogna spostare manualmente tutti gli oggetti. Inoltre non esiste uno storico delle azioni che sono state effettuate, bisogna ricordarsi ogni movimento per essere passati da uno stato ad un altro).

9. Gestural Interaction

Le **gesture** sono un tipo di comunicazione né verbale, né vocale in cui si possono comunicare messaggi particolari tramite azioni o movimenti eseguiti con il corpo, eseguiti con le mani, il viso o di altre parti del corpo. Grazie ad algoritmi matematici (**gesture recognition**) è possibile riconoscere ed interpretare le gesture quali posture, andamenti, comportamenti e più in generale la **prossemica** (la prossemica è la disciplina semiologica che studia i gesti, il comportamento, lo spazio e le distanze all'interno di una comunicazione, sia verbale sia non verbale). Ci sono due tipi di gestures:

- Touch gestures: Quelle legate all'utilizzo del touchscreen dello smartphone (swipe, pin, tap, ecc..);
- **Touchless gestures**: Interazione con i dispositivi senza toccarli;

Le **touchless gesture** sono delle gestures che servono ad interagire con i dispositivi senza toccarli: sono basate sulla computer vision e l'image processing attraverso sensori di misurazione quali accelerometro, giroscopio e il magnetometro che insieme permettono di rilevare movimenti ed effettuare le opportune elaborazioni. Queste gestures vengono utilizzate insieme a diverse tecnologie, tra cui troviamo le videocamere stereoscopiche (fotocamera che consente di simulare la visione binoculare umana e quindi di creare immagini tridimensionali), infrarossi, guanti che catturano i movimenti delle mani, guanti con i cavi, scanner laser (con il laser che viene riflesso sulle superfici è possibile misurare il time of flight cioè il tempo di percorrenza tra due oggetti). Un problema delle gestures è che devono essere **socialmente accettabili** e quindi l'utilizzo delle gesture è legato anche al contesto/ambiente (es. alzare le mani in aria per aprire una porta potrebbe provocare disagio), ed inoltre alcune gesture possono essere faticose da effettuare. Un esempio di prodotto che utilizza le Touchless gestures è il Kinect di Microsoft che consente di tracciare i movimenti del corpo.

Attraverso questi strumenti è possibile creare una **3D User interfaces** che è tipo di interazione uomo-macchina in cui le azioni dell'utente vengono eseguite direttamente in un **contesto spaziale tridimensionale**, le interazioni avvengono attraverso movimenti in uno spazio fisico tridimensionale oppure utilizzando strumenti, sensori o dispositivi all'interno dello spazio tridimensionale. L'interazione 3D è una scelta naturale per contesti con display molto grandi e in cui si devono mostrare molte informazioni: la tecnologia che permette questo tipo di UIs è lo **spatial tracking**, ma da solo non è sufficiente in quanto la maggior parte dei controller includono qualche altro input (spesso con bottoni e joystick), perché è difficile mappare le azioni in posizione, orientamento o movimento

del tracker. Un esempio di prodotto che utilizza le 3D UI è l'Oculus Rift della Sony che è formato da un headset per visualizzare l'ambiente virtuale, dei controller touch per interagire con l'ambiente e due sensori per tradurre i movimenti fisici in virtuali.

La **realtà virtuale** è interamente creata da un software e l'utente deve utilizzare dei dispositivi per interagire con essa: per questo definiamo la VR come completamente immersiva, nel senso che una volta entrato nell'ambiente virtuale non si ha la cognizione del mondo reale. I feedback che arrivano dalla VR sono di tipo audio, video, haptic, cioè tutto ciò che può raggiungere facilmente l'utente. Abbiamo due tipi di realtà virtuale, supportati da dispositivi diversi:

- Desktop VR: Ad esempio i videogiochi con visuale in prima persona;
- Head-mounted display, accessibile con un VR headset, come l'oculus rift. Il VR headset è composto da due schermi su cui vengono mandate immagini differenti, l'audio è stereo, e il movimento della testa viene tracciato dai sensori. Questo tipo di dispositivi offre un haptic feedback. L'interazione con un VR head-mounted avviene tramite dispositivi come mouse 3D, guanti con cavo, motion controller e sensori di tracciamento ottico. Per quanto sembri di avere raggiunto un obiettivo surreale progettuale, questa tecnologia ha incontrato un limite umano, la motion sickness o virtual reality sickness, ciò nasce dal fatto che il corpo si muove e l'occhio vede un mondo in continua evoluzione ma la posizione del corpo non cambia: questo provoca un senso di nausea all'utente (non tutti ne sono affetti).

Alcuni campi di applicazione della Virtual Reality sono: Intrattenimento, belle arti, robotica, ingegneria, salute e sicurezza, benessere, istruzione e allenamento e così via.

Nella **realtà aumentata** (augmented reality) tutto ciò che è presente nel mondo reale viene "arricchito" da informazioni generate dal computer. La differenza principale tra AR e VR è che la AR modifica la percezione dell'ambiente nel mondo reale, mentre la VR rimpiazza completamente il mondo reale con uno generato dal computer. Le informazioni del mondo reale che circonda l'utente diventano interattive e manipolabili digitalmente. Alcuni esempi sono i filtri di Snapchat che permettono di mostrare elementi digitali nel mondo reale come avatar animati, videogiochi come Pokémon Go oppure gli head-up display che vengono utilizzati sugli aerei per avere informazioni senza guardare i vari strumenti nella cabina di pilotaggio.

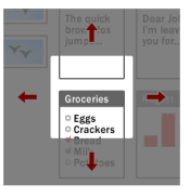
Esiste anche la **realtà mista** (mixed reality) (anche chiamata Augmented Virtuality), che risulta in una fusione degli oggetti del mondo reale in oggetti del mondo virtuale quindi si vedono sia oggetti virtuali che oggetti del mondo reale. Un esempio di Mixed Reality sono le Microsoft HoloLens in cui è possibile interagire con ologrammi con la gestualità istintiva con cui si manipolano gli oggetti presenti nel mondo reale.

Per quanto riguarda la **fedeltà delle interazioni** che vengono realizzate, devono avere un grado oggettivo di coerenza: le azioni (movimenti, forze, parti del corpo in uso, etc) usate per completare un certo task nella UI devono corrispondere alle azioni utilizzate per quello stesso task nel mondo reale e per questo si parla di **realismo continuo** (continuum realism). Talvolta si può voler estendere le abilità dell'utente oltre ciò che è possibile nel mondo reale (quindi effettuare azioni iper-naturali) (es. se salto, salto per 10 metri nella VR).

Alcune **limitazioni dell'input spaziale** (relative alla **precisione nello spatial input**), dato che le interazioni 3D sono realizzate in aria e non su una superficie, non abbiamo attrito, né un supporto fisico per realizzare movimenti più controllati e precisi. Gli umani hanno un tremore naturale alle mani che rende i movimenti in aria molto agitati. Le interfacce basate su puntatori tridimensionali che usano il ray-casting (metafora del puntatore laser) amplificano il tremore della mano sempre di più man mano che ci si allontana dal bersaglio nella VR (es. voglio puntare un oggetto, più mi allontano e più è difficile puntarlo). Inoltre, i trackers non si possono posare (come il mouse), in quanto non si è sicuri che una volta posati restino nella stessa posizione. Le soluzioni possibili a questi punti deboli sono:

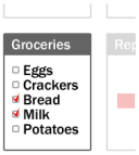
- Filtrare l'output dei trackers per ridurre il rumore e stabilizzare il movimento dell'utente;
- Modificare il rapporto control/display (C/D) (quindi quanto cambia il movimento virtuale rispetto al movimento reale). Se il rapporto tende a 1, si sta riducendo il movimento nel mondo virtuale rispetto a quello nel mondo reale;
- Non richiedere di essere più precisi di quanto sia assolutamente necessario (a volte essere estremamente precisi nei movimenti non è necessario);
- Progressive refinement: Si tratta di creare l'interfaccia in maniera tale da aiutare l'utente ad effettuare un certa azione nel momento esatto e ottenere il feedback in maniera tale da evitare sorprese o azioni inaspettate.

10. Zooming Interfaces



Panning across objects in two directions.

Zooming out for an overview.



Zooming in for greater detail.

Per poter comprendere bene l'argomento che si sta per affrontare è bene prima sapere cosa sia una **Graphic User Interface** (Interfaccia Grafica Utente). La GUI è basata sulle views (viste) tra le quali l'utente può navigare e in esse troviamo bottoni e schede (tabs) con cui egli può interagire per muoversi tra le views. Bisogna fare attenzione a quante views si impiegano all'interno dell'interfaccia, poiché l'utente può avere difficoltà a ricordare informazioni o opzioni presenti su vecchie viste in cui ha navigato in precedenza quindi meno sono, maggiore è la probabilità che l'utente si ricordi. L'impiego delle GUI può risultare così in un **effetto "labirinto"**, dove le tante viste diventano soffocanti e difficili da usare per l'utente, in quanto l'utente potrebbe non ricordare i passaggi che ha compiuto durante la navigazione.
L'idea delle **Zooming User Interface (ZUI)** si pone come una soluzione a questo problema. Formalmente, una ZUI è un **piano infinito con risoluzione infinita**. In particolare, su ogni piano l'utente può effettuare:

- Pan, "volare" sul piano in modo da avere una vista generale del piano stesso;
- Climb, passare al piano precedente facendo uno zoom-out;
- Dive In, passare al piano successivo facendo uno zoom-in.

Quindi l'utente può **muoversi sul piano** e **cambiare la scala** della vista che sta osservando.
Nelle ZUI l'utente tende a ricordarsi le posizioni dei punti di riferimento più importanti e ciò grazie a una associazione basata sulla prossimità degli oggetti, inoltre egli può organizzarli nella vista come preferisce. Quindi la **relazione gerarchica** nelle ZUI è basata sullo zooming, ovvero l'avvicinamento o l'allontanamento nella view, in quanto se l'utente si allontana (zoom-out) in essa si muove verso la sua radice (root), mentre al contrario se si avvicina (zoom-in) in essa si muove verso le sue foglie (leaves).
Un esempio di ZUI è GoogleMaps.

11. HCI in the car

Mentre si è in auto, l'utente rimane focalizzato sulla guida e le distrazioni rappresentano un pericolo. Oggi le auto sono tecnologicamente avanzate e possiedono molte utilità, dalla CPU, a sensori, attuatori e si può interagire con esse anche connettendosi con loro tramite, ad esempio, bluetooth o wifi. Questo permette di avere a disposizione diversi sistemi interattivi, quali sistemi di navigazione, HVAC (Heating, Ventilation & Air Conditioning) e persino di usare il cellulare. Quest'ultimo è una grande fonte di distrazione per il guidatore.
In generale, le potenziali **fonti di distrazione** sono:

- **Attenzione visuale**: L'utente viene disturbato da segnali visivi che gli fanno distogliere lo sguardo dalla strada, come il lampeggiamento dello schermo del cellulare;
- **Carico cognitivo**: L'utente viene disturbato da input esterni che lo distraggono, quali una conversazione con un'altra persona che distoglie parte del suo interesse dalla strada.

L'UI nelle auto è molto diverso da quello che troviamo nell'ambiente desktop o mobile (ad esempio non abbiamo mouse o tastiera) ed eventuali interfacce touch screen possono risultare molto pericolose perché l'utente è molto più portato a distrarsi (contrariamente, ad esempio, a bottoni statici di cui l'utente può memorizzare la posizione o la forma e quindi manovrarli con maggiore facilità).

Le UI in auto devono adattarsi al contesto di guida per non causare problemi all'utente, in quanto egli non può concedere la massima attenzione né muovere liberamente le mani mentre sta al volante. I **problemi** quindi sono che l'utente deve muovere lo sguardo dalla strada al dispositivo e di nuovo alla strada, e può solo compiere rapidi gesti con le mani e tutte queste azioni richiedono tempo. Per questo motivo il contesto viene ripristinato dopo ogni interazione.
Una soluzione ai problemi che si verificano quando si interagisce con i sistemi della macchina (radio, musica, ecc..) è il posizionamento di **bottoni sul volante** del veicolo che permettono di semplificare l'interazione (es. quando si prende il bottone per alzare il volume non abbiamo una variazione di contesto, non abbiamo bisogno di guardare il livello del volume). Un altro sistema molto avanzato è il **HDS (Heads Up Display System)** che permette all'utente di avere informazioni in tempo reale mostrate sul parabrezza del veicolo mentre guida.
Esistono anche delle interazioni che non richiedono l'uso delle mani (**Hands-free Interaction**) e che utilizzano il riconoscimento vocale o le gesture (es. rispondere al telefono facendo una sorta di swipe davanti allo schermo della macchina), e output audio per ottenere un feedback in risposta all'interazione. Alcuni **sistemi di controllo** che non richiedono l'attenzione dell'utente sono:

- Automatic Parking: Sistema di sicurezza che compie automaticamente una manovra di parcheggio del veicolo senza interazione dell'utente.
- Automatic Payment: Sistema che tramite sensori percepisce la necessità di pagamento di una tassa (esempio: telepass) ed effettua il pagamento senza l'intervento dell'utente.
- Automatic Transmission: Cambio della marcia in maniera automatica;
- Distance from vehicle ahead/behind/to the side: Sistema di sicurezza che tramite i sensori percepisce la vicinanza di ostacoli e avvisa l'utente quando l'auto vi si avvicina (distanza di sicurezza);
- Lane-Keeping: Sistema di sicurezza context-aware che tramite le videocamere dell'auto si rende conto e avvisa se il guidatore sta andando fuori dalle linee orizzontali (carreggiata).
- Stability controls: Sistema di sicurezza che percepisce il modo di guidare dell'utente e le condizioni circostanti (strada bagnata, ecc..) per correggere la guida;
- Braking: Frenata automatica se l'auto percepisce un ostacolo nelle immediate vicinanze.

Tutti questi controlli hands-free non richiedono l'attenzione dell'utente. Nelle automobili di ultima generazione, l'azione di guidare è diventata molto meno incentrata sull'utente, ed è necessario che l'utente creda nelle decisioni prese dall'automobile che alleggeriscono il lavoro del conducente. Però, a sua volta, l'automobile deve **ridurre al minimo i falsi allarmi**, ovvero l'automobile deve riconoscere in maniera accurata le situazioni di pericolo (altrimenti il conducente tenderà a non utilizzare i controlli hands-free). Se il conducente è troppo sicuro di sé, tenderà a usare i controlli in modo sbagliato. Inoltre, l'automobile deve ridurre le situazioni in cui è necessaria una decisione del conducente in quanto sono situazioni in cui non c'è il tempo di far capire al conducente il contesto e aspettare che venga presa la decisione.

12. Context-Aware Interactions

Per poter bene comprendere l'argomento bisogna prima capire cosa è un contesto. Il **contesto** è l'insieme delle informazioni riguardanti la situazione in cui l'utente interagisce con il sistema. Quindi ad ogni interazione possiamo avere un contesto differente e la sua comprensione è vitale per capire l'utilità del sistema. Ugualmente importante è capire quali siano gli attori. Un esempio di contesti usando l'app Zoom è che questa si deve adattare a utenti che la usano per lavoro, da casa, sul cellulare, mentre sono in movimento, su schermi diversi (piccoli e grandi); tutti questi sono diversi contesti.

I sistemi cellulare devono adattarsi a molte e differenti situazioni, permettendo di utilizzare gli **stessi servizi in contesti diversi** ma anche di diversi servizi, a seconda della situazione (quindi non solo cambiamo l'interfaccia in base al contesto in cui ci trova, ma si cambia proprio il servizio).
I **Context-Aware Systems** sono quei sistemi che agiscono e prendono decisioni in base a determinate situazioni o avvenimenti, quali le luci automatiche (che possono essere attivate da un movimento o da un suono e cambiano l'intensità in base alla luce esterna), sistemi di raffreddamento o riscaldamento automatici (modificano la temperatura target o si attivano/disattivano in base alla temperatura attuale), i sistemi di navigazione (se prendiamo una strada diversa, questi si adattano e variano la posizione di conseguenza) e luci delle auto (gli abbaglianti si attivano quando non c'è alcuna macchina che arriva dall'altra corsia, contrariamente si disattivano) oppure il telefono che modifica la luminosità in base alla luce del giorno e disattiva lo schermo quando lo avviciniamo all'orecchio. Questi sistemi si basano su 3 fondamenti:

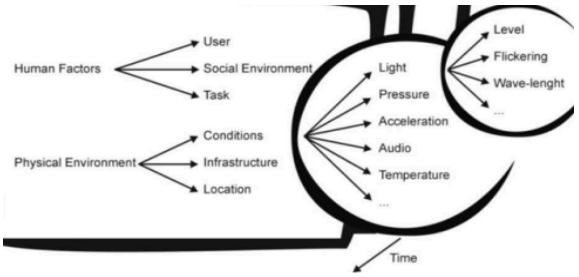
- **Sensori**: Il sistema impiega sensori per percepire quando si presenta un certo contesto per cui deve attivarsi (es. sensore di movimento che attiva un sistema come un allarme quando rileva una persona);
- **Algoritmi di Percezione** (Perception algorithms): Il sistema impiega algoritmi che sono strutturati per riconoscere la presenza di un determinato trigger che porta alla sua attivazione. I sensori così come il riconoscimento delle azioni necessitano di questi algoritmi. Inoltre, questi algoritmi permettono di classificare i vari contesti.
- **Azioni nel contesto**: Tramite un'azione dell'utente (come la presenza/vicinanza dell'utente) il sistema capisce che si trova in un contesto e quindi agisce di conseguenza.

La pipeline tipica è la seguente:

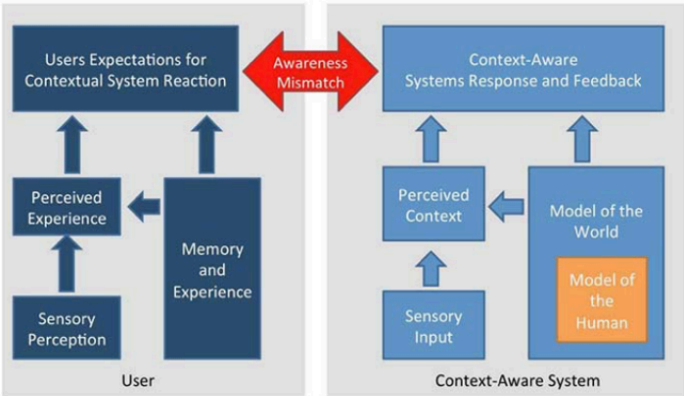
1. Il/i Sensore/i percepisce/ono un determinato avvenimento;
2. Il sensore specifico calcola l'avvenimento;
3. Il risultato dei sensori serve a capire qual è il contesto;
4. Il sistema decide le azioni/disposizioni necessarie per il contesto trovato;
5. La/Le applicazione/i corrispondenti reagiscono al contesto calcolato.

Per questi sistemi è vitale la **trasparenza**, ovvero la loro integrazione nella vita di tutti i giorni che li porta a diventare "invisibili" agli utenti, ovvero sono talmente usati che non ci si fa più caso al fatto che si tratti di

qualcosa di speciale. Per ottenere tale risultato, il sistema **non dovrebbe richiedere troppa attenzione da parte dell'utente** e deve anticipare le sue necessità (es. ABS funziona e anticipa le necessità del guidatore permettendogli di riprendere il controllo dell'auto).
Per poter funzionare adeguatamente, i Context-Aware Systems devono poter considerare molte informazioni (features) quali: la posizione, il tempo (orario), la temperatura, l'illuminazione, la velocità a cui ci si trova, l'identità dell'utente, l'attività dell'utente (se si sta muovendo, sta fermo, è in auto, ecc..). Perciò è importante creare una **gerarchia di features** (il **feature space**) con fattori che influenzano maggiormente il comportamento del sistema CA che si desidera progettare (**design hint 1**).



In ogni caso il sistema avrà delle limitazioni in quanto la rappresentazione del mondo che lo circonda è determinata solamente dai sensori di cui dispone, cosa che porta ad avere una percezione diversa tra utente e dispositivo (es. luce automatica che si spegne quando l'utente è ancora nella stanza perchè non rilevato dal sensore). Questo può portare ad un comportamento del sistema diverso dalle aspettative dell'utente.
Questo viene descritto dallo **User-Context Perception Model (UCPM)**: Per quanto riguarda l'utente, in base alla percezione sensoriale e alla memoria/esperienza l'utente percepisce una certa esperienza e si crea una aspettativa riguardo la reazione che il sistema contestuale dovrebbe avere. Il sistema, in base ai sensori e al modello del mondo, percepisce un certo contesto e invia una risposta in base a tale contesto. La reazione attesa e quella che si verificata potrebbe non corrispondere.



Per **minimizzare questa discrepanza di aspettative**, l'utente deve comprendere quali fattori hanno influenza sul sistema e, tramite una interfaccia, egli può vedere quali informazioni captate dai sensori sono usate per determinare il contesto. In questo modo l'utente crea una sorta di **modello mentale** del sistema così da poterlo comprendere a pieno (bisogna capire come funziona il sistema). Quindi, l'interfaccia utente deve fornire dettagli sulle informazioni sensoriali utilizzate per determinare il contesto al fine di ridurre al minimo la awareness mismatch (**design hint 2**). La **qualità di un CAS** è inversamente proporzionale alla discrepanza percepita, ovvero più l'errore è piccolo è più l'utente percepisce il sistema come buono e viceversa.
L'origine di queste discrepanze è dovuta alla **difficoltà di identificare adeguatamente le features** e i contesti. Se i sensori percepiscono input simili, probabilmente si trovano in contesti simili e in essi vanno identificate le caratteristiche più rilevanti. In pratica i sensori misurano, tramite parametri, le caratteristiche del contesto.
Se i sensori identificano una feature, significa che è avvenuto un match, ovvero un riconoscimento di una caratteristica che quindi serve a delineare/comprendere il contesto. Il matching accade con l'impiego di un algoritmo che può essere un modello di ML che permetta di predire se si sta verificando un certo contesto quindi permette di creare una **confusion matrix** per capire qual è l'accuratezza del modello creato. Quindi bisogna trovare i parametri (features) che sono caratteristici di un contesto che si desidera rilevare e trovare i mezzi per misurare quei parametri (sensori) (**design hint 3**).

		Perceived context by the "system"	
		Fire	No Fire
Situation in the real world	Fire	0 %	100 %
	No Fire	0 %	100 %

Un esempio è un sistema anti-incendio, che tramite la percezione di una feature nell'ambiente circostanze (quale il fumo) riconosce che si deve attivare poiché ci si trova nel contesto di un fuoco vivo, altrimenti rimane fermo poiché senza una feature sa che non deve intervenire.

Una volta rilevato il contesto, le possibili azioni che il sistema può compiere sono:

- Funzionalità di basso livello (es. provare a stabilire una connessione di rete);
- Cambiare il comportamento dell'applicazione e funzioni supportate (es. diversi privilegi di accesso da posizioni diverse ad applicazioni o web applications);
- Modifiche a interfacce utente quindi in uno stesso servizio si possono offrire interfacce differenti (es. diversi livelli di zoom nella mappa del navigatore in base al fatto che vi sia un'azione richiesta nelle vicinanze o meno).

Una categoria interessate di CASs sono i **Sistemi Context-Adaptive**, ovvero quei sistemi che agiscono **pro-attivamente** sulla base delle intenzioni dell'utente (l'utente sta tornando a casa, si attiva il riscaldamento oppure l'utente si trova alla fermata dell'autobus, viene lanciata l'app per controllare gli orari dei mezzi) oppure agiscono in maniera **adattiva** quando l'utente compie una determinata azione cioè attivano una funzione o parte dell'applicazione in base al contesto (l'utente lancia Waze nel tardo pomeriggio, il sistema mostra le indicazioni per tornare a casa). Progettare applicazioni proattive è molto difficile perché il sistema deve anticipare ciò che l'utente desidera. In molti casi, è molto più semplice non presentare "l'applicazione" ma piuttosto presentare un insieme di potenziali applicazioni interessanti che l'utente può lanciare. Ad esempio, una "schermata iniziale" sensibile al contesto può offrire una selezione di applicazioni utili in un dato contesto piuttosto che tentare di scegliere quella giusta (**design hint 4**).

Nelle adaptive and context-aware UI, gli **elementi dell'interfaccia si possono ottimizzare in base al contesto** (es. Durante la notte si può attivare in "night shift" dove i colori dello schermo tendono al giallo per proteggere gli occhi oppure mentre si guida l'interfaccia si può semplificare per aiutare il guidatore). Si può anche avere un posizionamento adattivo o dinamico degli elementi in base al contesto (quindi cambiare posizione o struttura degli elementi come la grandezza in base al contesto corrente). A causa di queste variazioni in base al contesto, l'UI diventa più difficile da comprendere e da memorizzare. Ci sono degli stratagemmi per aiutare l'utente: ad esempio, anziché riordinare o nascondere elementi di un menù (cosa che potrebbe confondere l'utente), si possono disattivare per le funzioni non disponibili nel contesto corrente (es. disattivo, scurendolo, il bottone delle app di messaggistica mentre sto guidando). Nota che il bottone è **sempre nella stessa posizione, semplicemente viene disabilitato**. Quindi occorre usare l'adattamento nell'interfaccia utente con molta attenzione e bisogna assicurarsi che sia comprensibile per l'utente. Un buon design mantiene la stabilità e supporta l'utente nella memorizzazione dell'interfaccia utente mentre utilizza il contesto per ridurre la complessità (**design hint 5**).

Un altro concetto riguarda le **interruzioni**. Possiamo usare il contesto per decidere di posticipare/programmare le interruzioni quindi possiamo eventualmente decidere quando e come fornire **comunicazioni asincrone**, quindi possiamo optare per interruzioni programmate (schedules) o mediate (mediated) (es. sistemi di monitoraggio del sonno che non disturbano l'utente quando sta dormendo ad esempio posticipando le notifiche). Possiamo anche **condividere il contesto** per migliorare la UX nelle interruzioni (es. Informazioni sull'ultimo accesso di WhatsApp (danno un'idea dello status della persona, ad esempio se vedo che l'ultimo accesso è la sera prima allora forse la persona sta ancora dormendo) oppure condivisione di stati tipo "sono ad un meeting" come accade su Skype).

Un altro concetto riguarda i **dati generati per metadati** e il **contenuto implicitamente generato dall'utente**. **Meta dati sul contesto** sono ad esempio dove si crea un testo e chi è presente (es. Su documento in Google Drive si può vedere chi sta editando un documento e si può anche ripercorrere la storia del documento per vedere chi ha modificato cosa in passato e questi sono metadati basati sul contesto relativo al documento) mentre **dati generati implicitamente** sono ad esempio la velocità durante un viaggio in macchina (es. Google usa la velocità corrente per capire il livello di traffico e questi sono dati basati sul contesto).

Possiamo anche **organizzare risorse in base al contesto** come, ad esempio, **ottimizzare il funzionamento del dispositivo** in base al contesto (es. Risparmiare energia di una batteria in un veicolo elettrico scegliendo il percorso migliore quindi si ottimizza la batteria in base al contesto). Questo tipo di ottimizzazione è indirettamente collegata alla UX perché in qualche modo stiamo obbligando l'utente a utilizzare il sistema in un certo modo (es. Fare una strada diversa dal solito perché il sistema pensa che così si risparmierà batteria). C'è quindi un **trade-off tra trasparenza e visibilità**: ci sono dei casi in cui il sistema può agire in maniera trasparente (es. Fa caldo, la macchina accende l'aria condizionata da sola) mentre in altri casi è preferibile avere il controllo quindi si preferisce la visibilità (es. Quando fa caldo, la macchina può informare l'utente su quello che è il contesto percepito ma si può lasciare all'utente la decisione di scegliere se attivare o meno l'aria condizionata o magari abbassare il finestrino).

Un concetto importante è la **HCI implicita** che può essere vista come una generalizzazione di Context-Aware Systems, in particolare teniamo conto di:

- **Input implicito**: Azioni e comportamenti degli esseri umani compiuti per raggiungere un obiettivo e che non sono principalmente considerati come interazione con un computer, ma catturati, riconosciuti e interpretati da un sistema informatico come input.

- **Output implicito**: Output di un computer non direttamente correlato a un input esplicito e che è perfettamente integrato con l'ambiente e l'attività dell'utente.

L'**interazione esplicita** contraddice l'idea del **calcolo invisibile** (invisible computing) cioè quando interagiamo esplicitamente con un sistema, questo effettua un calcolo per arrivare ad un risultato. Al contrario nell'interazione implicita è come se non sappiamo se c'è un computer dietro dal risultato quindi è come se ci fosse appunto un calcolo invisibile (es. quando ci avviciniamo ad una porta scorrevole e questa si apre si tratta di una interazione implicita infatti non tocchiamo nulla mentre se dobbiamo toccare un bottone per aprire la porta allora questa è una interazione esplicita).