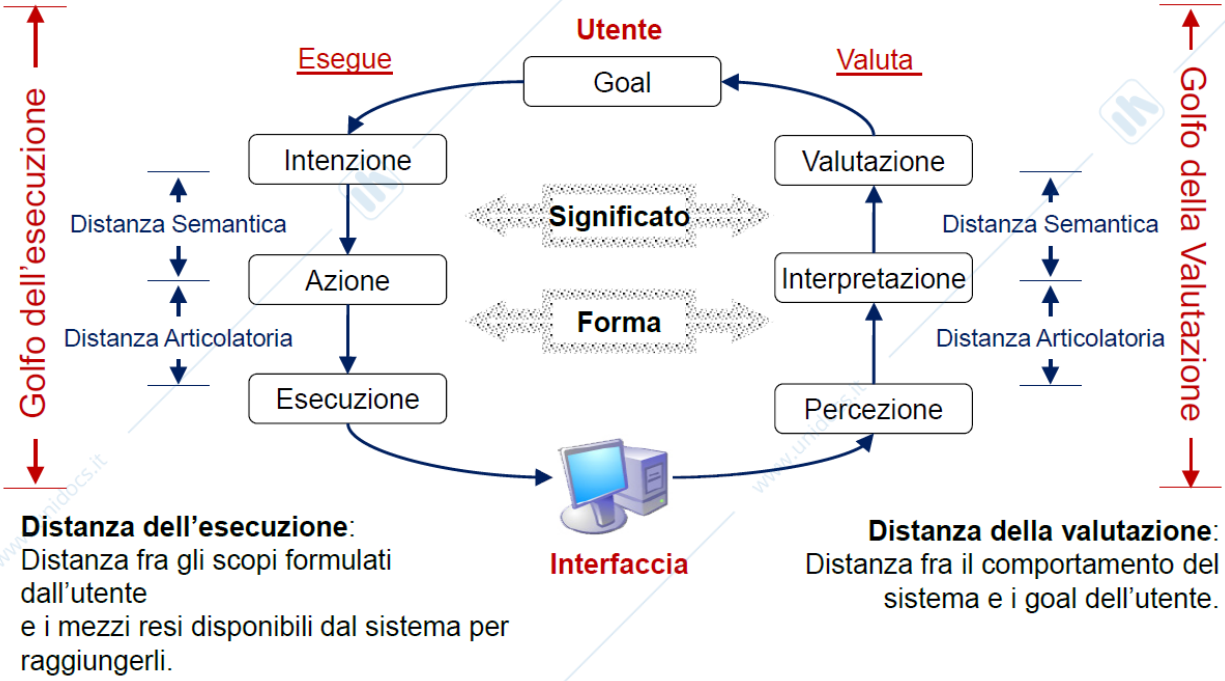
Riassunto di Interazione Uomo Macchina

Il profilo utente

Documento che **identifica** l'utente designato del sistema.
Individuando quelle **caratteristiche** che determineranno la sua **capacità di interagire** con il sistema e di usarlo adeguatamente (caratteristiche comuni e caratteristiche che variano: **individuali**, **ambientali**, **stile cognitivo**, **attitudine**, **motivazione**).

Il sistema interattivo deve: **minimizzare** le emozioni negative e **massimizzare** l'appagamento del lavoro (motivazione ed attitudine). Deve inoltre cercare di trovare il "flow" emotivo (armonia tra le varie emozioni, ansia, noia…).

Modello di Hutchins, Hollan e Norman



Golfi: **errori** dovuti alla distanza tra il modello mentale dell'utente e il mondo esterno, reale, degli oggetti.

Distanza **semantica** in **esecuzione**: quando gli oggetti sono a un **livello** troppo **basso** o troppo **alto** per le intenzioni dell'utente.
Distanza **articolatoria** in **esecuzione**: quando l'azione **NON** è **collegata** direttamente al **significato**.
Distanza **articolatoria** in **valutazione**: quando la **forma** dell'output **NON** è "**disegnata** bene" per il suo significato.
Distanza **semantica** in **valutazione**: quando l'utente deve **interpretare** l'output prima di decidere se lo scopo è stato raggiunto.

Nello sviluppo è necessario tenere conto della: **visibilità**, **affordance** (fruibilità), **affordance percepita** e **relazione persona-strumento**.

Punti di **forza** del modello HHN:

- Individua i concetti base dell'interazione dal punto di vista dell'uomo: ciclicità, golfi, distanze e forma dello strumento.
- Permette una prima definizione di usabilità (definizione operativa)

Punti di **debolezza**:

- Vago ed allusivo
- Concentrato sull'uomo e sull'interfaccia
- Non si focalizza sulla comunicazione
- Dimentica che uno strumento interattivo ha una dinamica e dei limiti tecnologici
- Lo strumento vincola cosa può fare la persona e le sue strategie

La progettazione secondo HHN è vincolata dalle caratteristiche dell'utente e dall'ambiente e sviluppata in base ai modelli da seguire. Ma il modello non considera le **specificità dell'utente!**

Personas

Realistica rappresentazione di possibili/gruppo di utenti e servono a conoscere **chi** potrebbe usare il sistema e **come** progettare il sistema.

Cosa indagare: stile cognitivo, attitudine, motivazione, livelli di alfabetizzazione, titolo di studio, esperienza, lingua, varie abilità informatiche, manualità, daltonismo, sesso, caratteristiche del lavoro e dei compiti.

Strategie di raccolta dati

Utili per raccogliere **requisiti** degli utenti, per **valutare** i sistemi.

Interviste

Buone per **esplorare** i problemi; indicazioni **qualitative**. Utilizzo di **supporto** (es: scenari d'uso, prototipi…). Richiedono **tempo** e **NON tutte** le classi utenti sono **intervistabili.** Raggiunge un numero **limitato** di utenti. Strumento di **semantizzazione progressiva**.

Utile per: **conoscere** l'utente, dopo **l'osservazione** degli utenti, per ottenere **feedback**, nella **task analysis**.

Problema: interpretare i risultati, gente risponde **ciò che pensa di dover rispondere**. Opinioni **soggettive** dell'utente.

Tipi di interviste:

- **Strutturata**: domande **predefinite**, adatta ad **utente esperti**, **l'intervistatore** può **NON** essere **esperto**.
- **Non strutturata**: domande **flessibili**, l'**intervistatore** deve essere **esperto**, strategia di indagine **chiara**, non deve mettere a disagio gli utenti.
- Semi strutturata: gruppi di domande predefinite.
- Prompted: utilizzo di **solleciti** come schizzi, metafore grafiche, linguaggio di dominio.

Focus group

Tecnica **informale**: **6-9 utenti discutono** in presenza di un **moderatore** su concetti, strumenti per un tempo di circa **due ore** prima e dopo l'implementazione di prototipi. Gli utenti hanno l'impressione di una **discussione libera**.

Il moderatore deve: **prima:** preparare la lista di temi e gli scopi da raggiungere (**copione**); **durante**: deve mantenere la discussione focalizzata, garantire che tutti si esprimano; **dopo**: **analizzare** i dati e creare il **rapporto**.

Pro: fa emergere idee e reazioni spontanee.

Contro: costosi in termini di utenti.

Questionari

Usati insieme ad altre tecniche, danno sia dati **qualitativi** che **quantitativi**. Sono **poco flessibili** ma buoni per rispondere a **specifiche questioni** da un gruppo vasto di utenti. Strumento di **semantizzazione progressiva**.

Problemi: domande **NON ambigue**, costi, problema dell'**analisi dati**, test **pilota**. Opinioni **soggettive** dell'utente.

Scale adottabili:

- Lista di controlli (senza ordine, checklist)
- Scala di valutazione a punti (zero e distanza non definiti)
- Scala di **Likert** (struttura matematica)
- Semantica differenziale (struttura bipolare)
- Ordinale secondo rango (ordinamento di elementi simili)

Osservazioni dirette

Osservatore assiste all'attività e annota ciò che vede. Utili per capire come un utente **si comporta** e per **valutare** un sistema, le **esigenze** ed i **bisogni** nello svolgere determinati task. Molto utile nelle fasi **iniziali** o di valutazione.

Problemi: coinvolgimento **full-time** di un membro del team. **Effetto Hawthorne** (coscienza di essere osservati altera il comportamento).

Osservazioni indirette

Utili per il **logging** di task attuali. Utilizza strumenti come video camere, protocolli verbali (raccolta audio, utente parla ad alta voce, poco adatto per utenti silenziosi ma poco invasivo), data log. Possibilità di rivedere i filmati **ma** si hanno problemi di tempo, sincronizzazione ed effetto Hawthorne. Rischio di raccogliere **troppi** dati, comporta **un'analisi lunga**.

Monitoraggio utenti con log

Osservare utenti che interagiscono col software tramite **programmi**, utilizzato dai programmatori in fase di **debug**. Occorre **modificare** programmi esistenti. Strumento di basso livello: produce **troppi** dati e sono **difficili** da esaminare.

Studio di documentazione

Procedure e regole scritte nei **manuali**. Descrive gli **step e regole** di ogni attività/task e utile per capire **informazioni** di background. **MA** non può essere usato come **tecnica isolata**.

Problemi generali con la raccolta dati

Identificazione stakeholders, comunicazioni tra le varie parti, difficoltà varie di **dominio** (conoscenza), problemi **politici**/organizzativi, cambiamento ambiente di **business**.

Linee guida: coinvolgere gli **stakeholders** e rappresentanti di ogni gruppo (**personas**), usare una **combinazione** delle **tecniche** di raccolta dati, eseguire test **pilota**, utilizzare differenti approcci.

Fenomeni che caratterizzano l'HCI (5 fenomeni)

Gap comunicazionale

I progettisti e gli utenti hanno differenti **culture**, **conoscenze**, **tecniche** di **ragionamento** e **modelli mentali** (problemi di **dominio**).

Conseguenze per il CD: **conoscere** l'utente, **progettazione** modello concettuale, **valutazione** del modello concettuale.

Varietà degli utenti

Gli utenti sono vari, caratterizzati da diverse **culture**, **scopi** e **compiti**.

Conseguenze per il CD: i sistemi devono essere **localizzati**, **adattabili** a situazioni specifiche, **personalizzabili** e **modulari**.

Influenza nascosta delle tecnologie

Ogni strumento favorisce lo svolgimento di **certe** attività a **scapito** di **altre**. Se il **modello concettuale** non chiarisce lo **stile** di **interazione**, si propone uno stile che rende **difficile** l'attività dell'utente.

Conseguenze per il CD: il modello concettuale deve rendere **valutabile** ogni aspetto dell'iterazione. Occorre **verificare** e valutare ogni **strumento** usato o sviluppato rispetto **all'attività** dell'utente.

Co-evoluzione Utente – Sistema

L'utilizzo del sistema, **cambia il comportamento dell'utente.** Inoltre, inizierà ad utilizzare il sistema con **nuove metodologie** (user moving target).

Conseguenze per il CD: il sistema **evolve nel tempo** (con CD responsabile dell'evoluzione), il CD deve **progettare** il sistema in modo che possa **evolvere** (adattabile alle nuove esigenze dell'utente).

Tacita conoscenza ed informazione implicita

Tacita conoscenza:

È una **conoscenza NON codificata**, non contenuta in testi, non gestita attraverso flussi comunicativi strutturati ma una conoscenza che risiede nella testa degli individui e **nasce dall'esperienza** lavorativa e difficile da comprendere ai non esperti di dominio.

Informazione implicita:

Portata nel messaggio, dipende da **contesto e dall'attività**, incorpora la conoscenza tacita dell'autore, interpretando un messaggio. **Esplicitabile** solamente **da chi possiede conoscenza tacita**.

Conseguenze per il CD: creare strumenti che permettono l'utilizzo della **tacita conoscenza** e dell'**informazione implicita**.

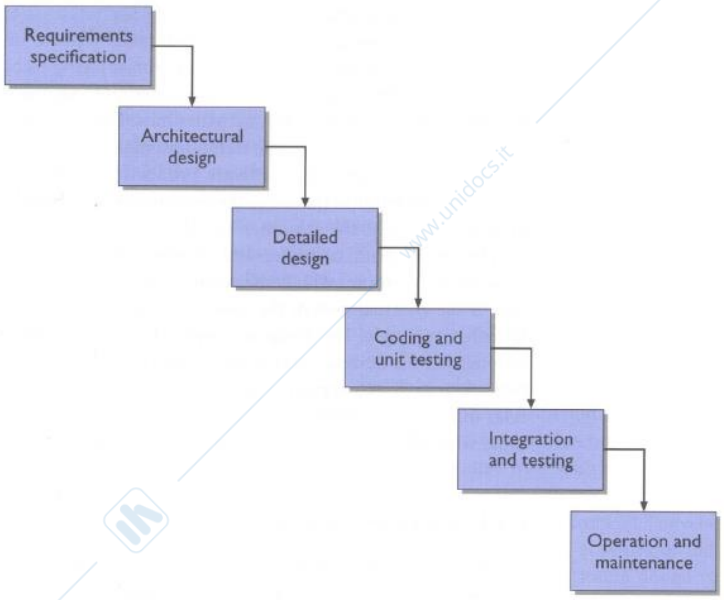
Modelli di Cicli di vita del software

Modo in cui una metodologia di sviluppo o un modello di processo **scompongono** l'attività di realizzazione di prodotti software in **sotto attività** fra loro **coordinate**, il cui risultato è il **prodotto** stesso e tutta la **documentazione** ad esso associata (**management tools** e **semplificate** versioni della realtà).

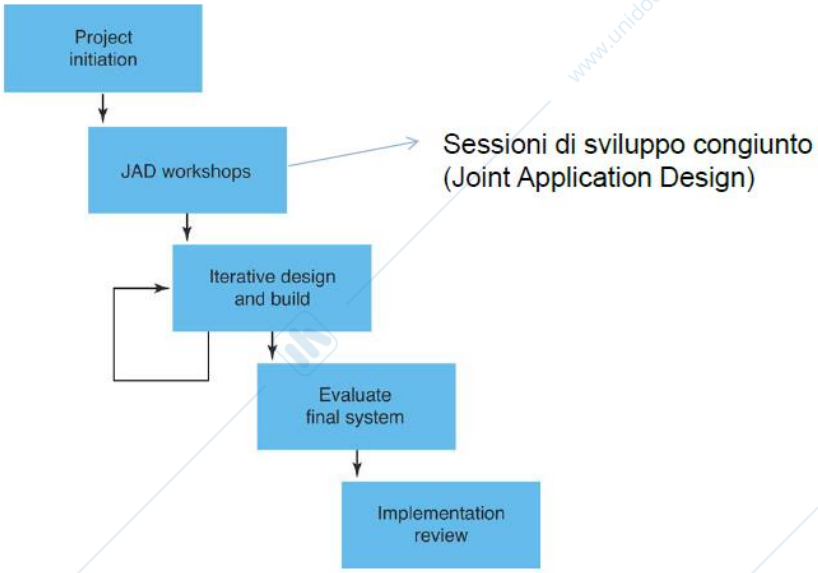
Modello a cascata

Lo sviluppo del software consiste in una **sequenza** di processi e rappresentazioni del sistema (**top-down**). Nessun **feedback**.

Problemi: Molto spesso il mondo reale impone la **revisione** di una o più fasi.

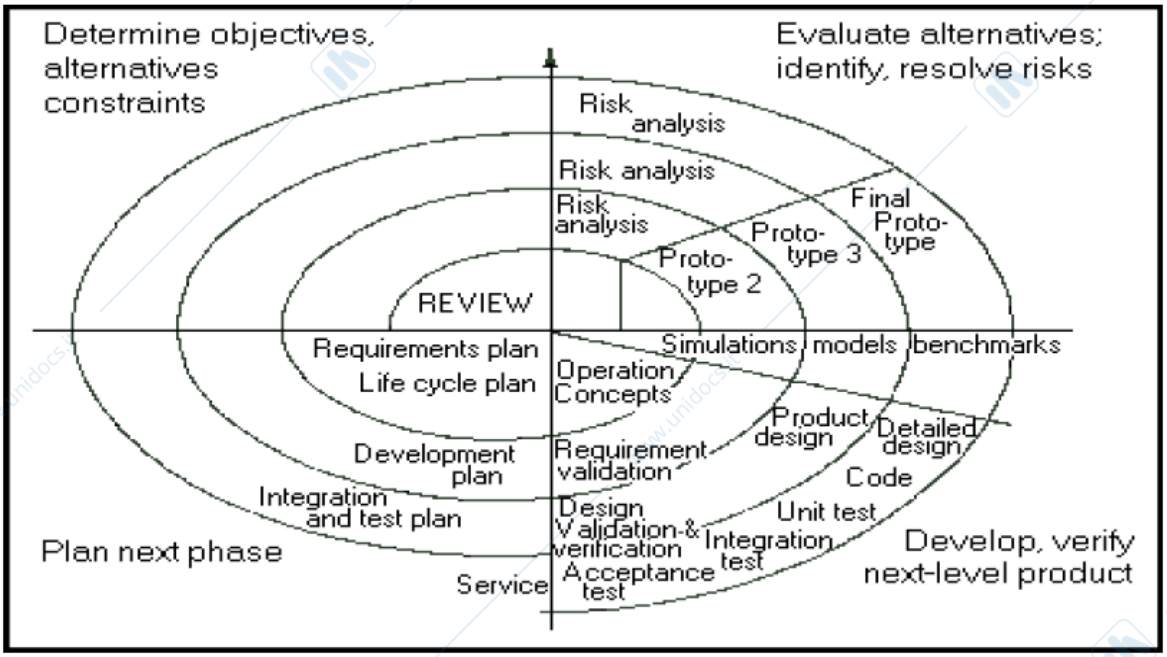


Modello RAD (Rapid applications development)

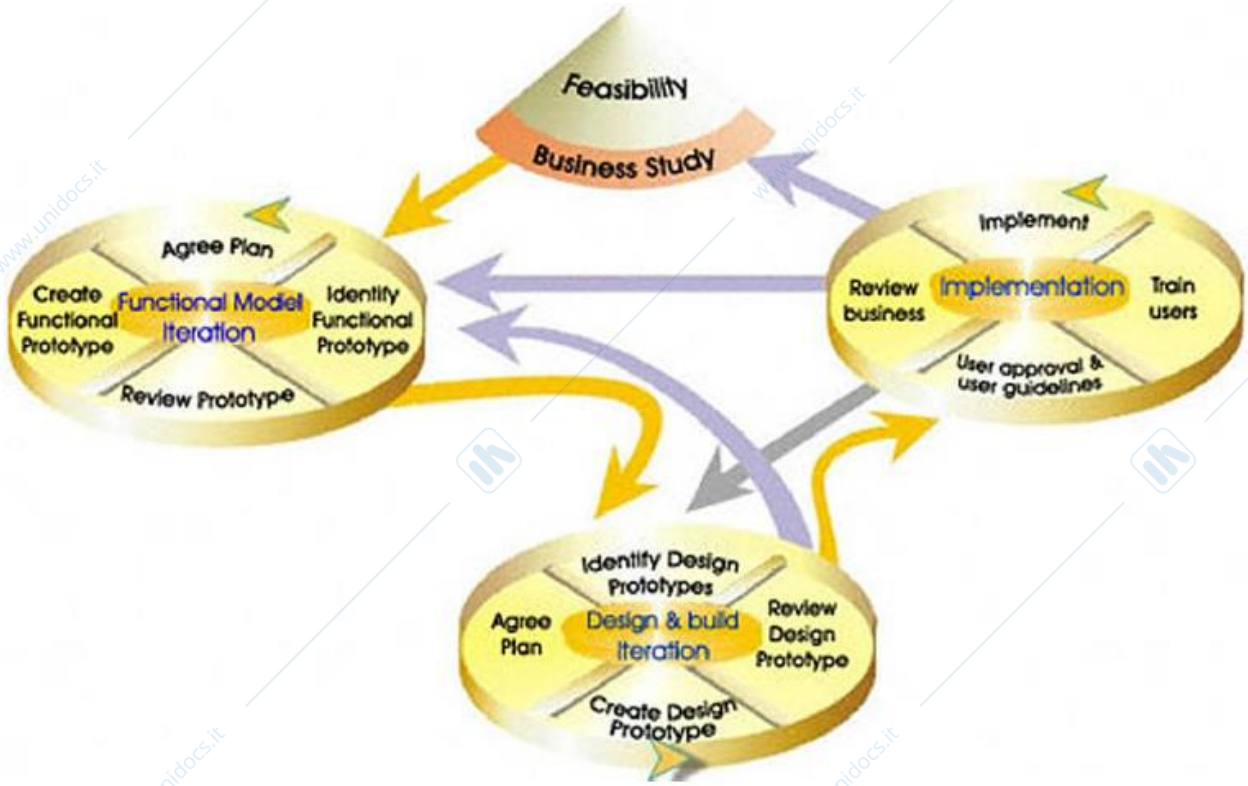


Modello a spirale (Barry Boehm)

Caratteristiche: **analisi dei rischi**, **prototyping**, **framework interattivo** in modo da valutare e controllare le idee, stimoli per le alternative e utile per **progetti grandi** e complessi, ma non per quelli semplici.

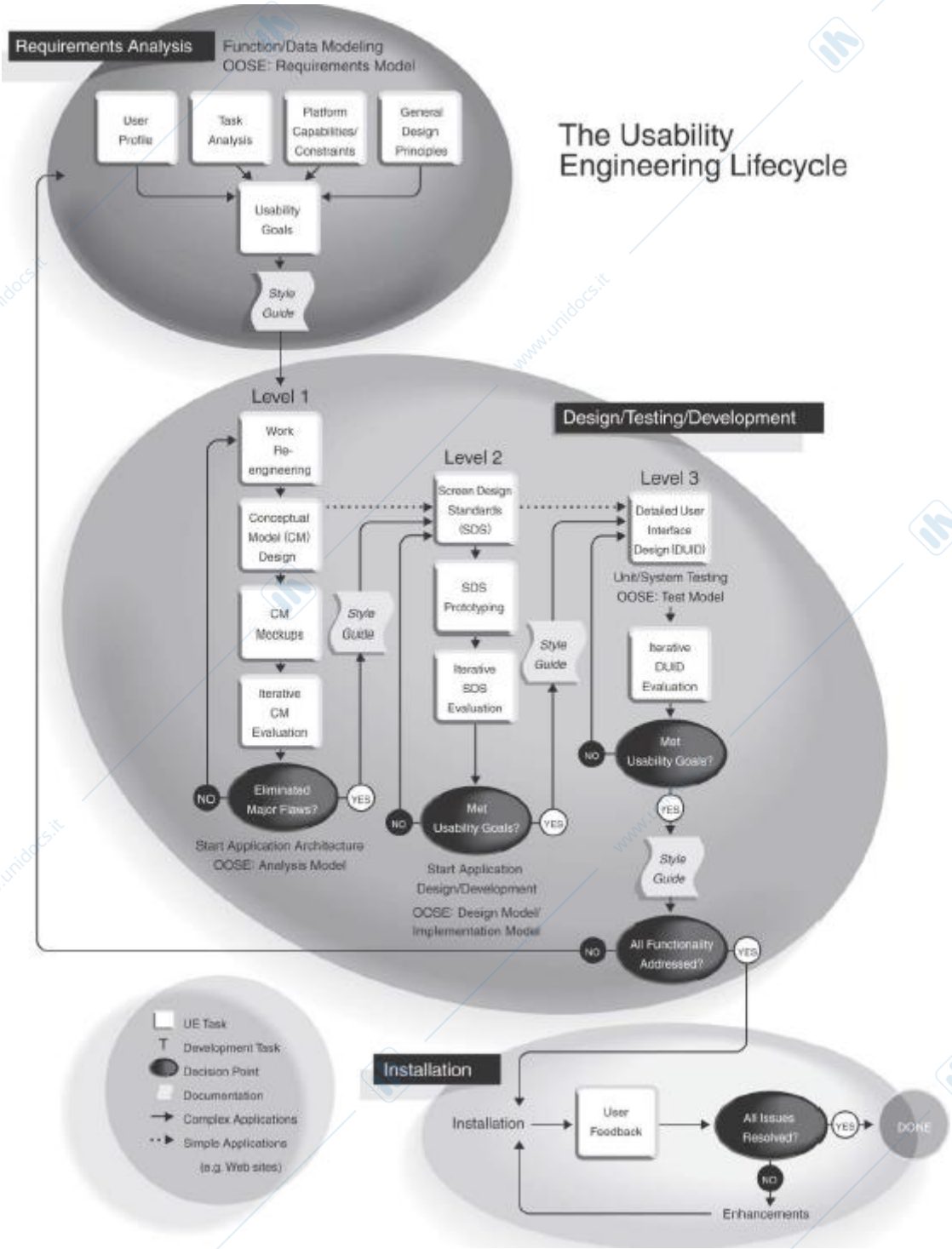


Modello di ciclo di vita DSDM (Dynamic System Development Method)



Usability engineering (Deborah Mayhew)

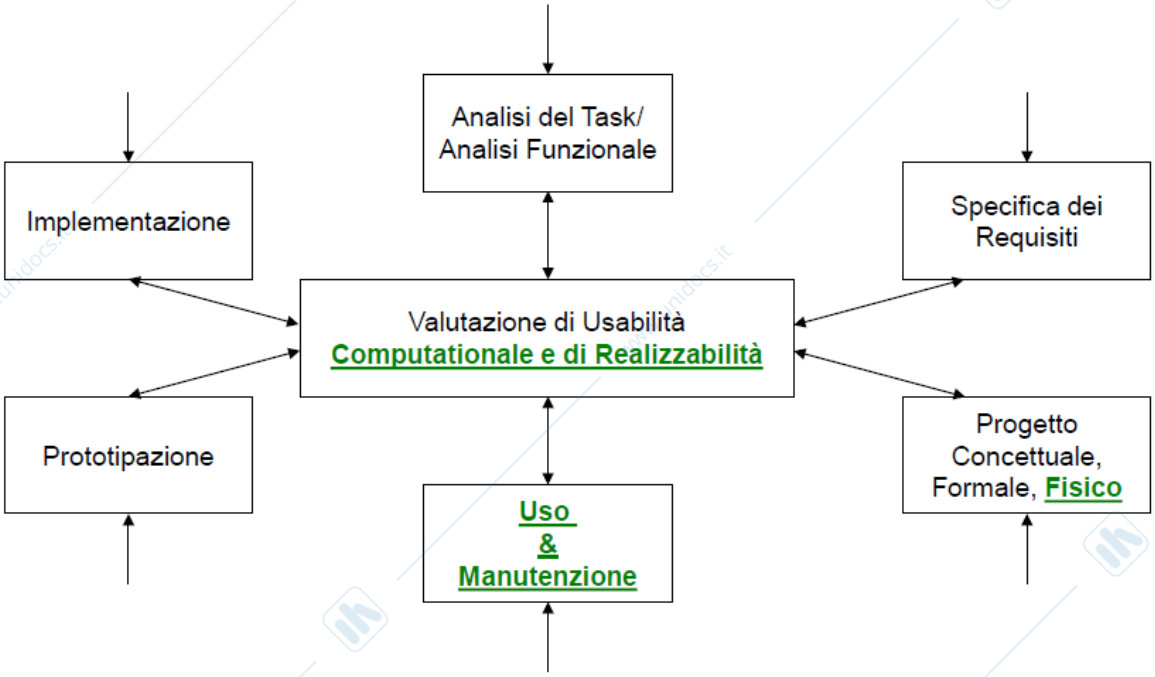
Visione **globale**, fornisce collegamento ad approcci di **software engineering**, fasi dedicate ad identificare **requisiti**, **design**, **valutazioni**, **prototyping**. È utile anche per **progetti piccoli** ed usa linee guida di stile capaci di catturare gli obiettivi di usabilità.



Modello di ciclo di vita a stella (Hartson, Hix)

Valutazione al **centro** delle attività**, non c'è ordine delle attività**, deriva da studi empirici di **interface design** (**NON** include **l'uso e la manutenzione**).

Modello di ciclo di vita a stella RIADATTATO



Specifica dei requisiti

Input: **documento** scritto del cliente di solito **vago** che contiene: richieste impossibili da soddisfare, ambiguità.

Output: **documento** (o prototipo) che: elenca i **problemi** con i sistemi attuali, specifica le **caratteristiche desiderate ed ottenibili** del prodotto finale.

Utile per capire quanto più possibile dell'utente, delle task e del contesto d'uso. Viene utilizzata la **raccolta**, **l'analisi dei dati** e attività iterative.

La specifica dei requisiti **evolve** con l'evolvere del progetto e con l'uso del sistema, il documento di output **NON** può essere considerato **definitivo**. Predisporre le attività per una co-evoluzione.

Possono essere raccolte da **diversi punti di vista**, utilizza metodi adattati dalla **Computer Science** e dal **Software Engineering**. Potrebbe essere necessaria la **revisione** del documento causata da altre attività (task analysis…). Processo di semantizzazione progressiva.

La specifica dei requisiti è una fase che se non ben curata, può causare **fallimento del progetto**.

Requisiti funzionali

Descrivono le interazioni tra il sistema ed il suo ambiente, indipendentemente dall'implementazione (**cosa deve fare il sistema**) – La piattaforma e-learning tiene traccia delle attività dello studente

Requisiti non funzionali

Proprietà del sistema misurabili dall'utente e **NON** direttamente **correlate al suo comportamento funzionale** – La piattaforma deve essere disponibile agli studenti 24/7

- Requisiti sui **dati**: quali tipi di dati devo essere memorizzati? Come verrà strutturato il database?
- Requisiti **sull'utente**: caratteristiche psicologiche, esperienza, fisiche, uso del sistema (capire la tipologia dell'utente, **NO utente** "**medio**", guardare anche i "**poli**" dell'utenza facendo **profilazione**, personas.)
- Requisiti **sull'ambiente** e **contesto d'uso**: **fisiche** (rumoroso, vibra…), **sociali** (condivisione, privacy…) ed **organizzative** (gerarchie, supporto utente…).
- Requisiti **operazionali**: quali **strumenti di I/O** adottare, quale **stile** di **iterazione** (metafore…)
- Requisiti **sull'usabilità**: **facile** da imparare, da **usare**, flessibilità, robustezza. Che **modello mentale** e che conoscenze ed abilità mette in gioco.

Task analysis

È necessario **formalizzare i requisiti funzionali** dopo averli definiti in modo da comunicare efficientemente con gli sviluppatori e con I clienti.

Task description: utili per **raccogliere**, **formalizzare** i requisiti e per **stabilire** basi per la **valutazione** dell'**usabilità**. Spesso usate per **progettare nuovi sistemi o strumenti**.

Scenari

Descrizione **narrativa informale** di persone che svolgono una certa **attività**, molto semplice, **naturale** (uso di un vocabolario non di dominio) e **non generalizzabile.** Esplora e discute dei **contesti**, **bisogni**, e dei **requisiti**. Rivelano gli aspetti degli **stakeholder** e delle loro attività che hanno **implicazioni** sul progetto.

Descrivono una **sessione singola** di iterazione, senza nessuna possibilità di **flessibilità**, utilizzando un insieme specifico di **possibilità del sistema** per ottenere un **preciso risultato** in un intervallo di **tempo certo**, di durata variabile.

Presuppone l'uso del sistema come se fosse un **prototipo**, non sempre descritto bene.

Importante: capire già cosa **funziona bene** e costruirci sopra, calcolare la quantità degli scenari e come rappresentarli.

Limiti: si focalizzano su attività specifiche ma che **nascondono questioni** più **ampie** e una visione organizzativa del **contesto**

Casi d'uso

Consiste nel valutare ogni requisito focalizzandosi sugli **attori** che **interagiscono** col **sistema**, valutandone le varie interazioni. Si dividono in **principali** ed **alternativi**.

Descrizione **formale** e NON ambigua dei vari "casi" che verranno eseguiti **nell'interazione uomo-macchina**. Infatti, si basa su quest'ultima e **NON** sul compito.

Limiti: assunzioni **sull'interfaccia** e il tipo di **interazione da progettare** a partire dal dare per scontato che si debba interagire con una certa **tecnologia**

Essential use case

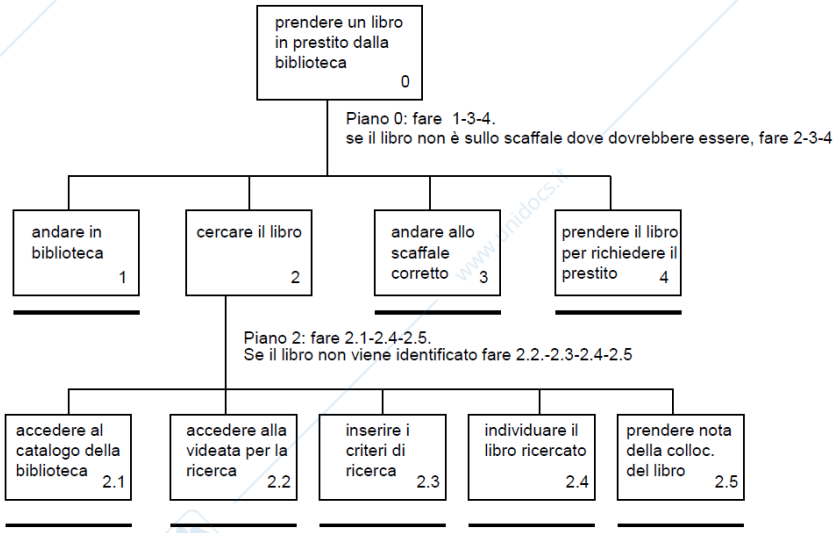
È simile al caso d'uso solo che **astrae** rispetto agli scenari. Utile per definire l'allocazione e la portata del sistema (**cosa deve fare l'utente e cosa deve fare il sistema**)

Struttura narrativa fatta in tre parti:

- Nome che esprime l'**intenzione primaria dell'utente**
- Descrizione in passi delle **azioni dell'utente**
- Descrizione in passe delle **responsabilità del sistema**

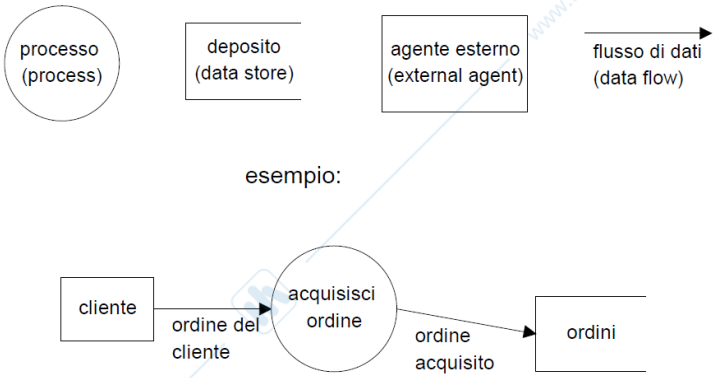
Hierarchical Task Analysis

Articola i vari **task** in **sub-task** raggruppati in **piani** che specificano come le task dovrebbero essere eseguiti. Si focalizza su **azioni osservabili** (sia collegate al software/strumento che non) e il punto di partenza è l'**obiettivo dell'utente**.



Diagrammi data flow (DFD)

Sono diagrammi che rappresentano **sistemi** e come una gerarchia di funzione (riflette l'organizzazione dei **modelli mentali**). Il diagramma è composto da insieme di processi che **elaborano** e si **scambiano** flussi di dati definiti in un **data dictionary**.



Processo

Attività di trasformazione, acquisisce **dati in input** e li trasforma in **dati di output**.

Flusso dati

È un canale che **trasporta** materiale (**dati**) omogeneo, ha una direzione, connette due punti del sistema e a uno dei due capi vi è un processo o un agente esterno che produce/acquisisce i dati

Può trasportare: **dati organizzati in strutture** o **non strutturati**, **zero, una o più occorrenze**.

Deposito

Archivio a cui i processi del sistema possono accedere per **leggere**, **aggiornare** e/o **salvare** dati. È **statico**.

Se la connessione tra due processi avviene tramite deposito, i processi si dicono **asincroni**, invece se avviene senza un deposito, si dicono **sincroni**.

Agente esterno

È un **sistema esterno** con il quale il sistema **analizzato** scambia informazioni di **I/O**. Può essere una persona, organizzazione, un sistema hardware/software, un oggetto qualsiasi. È da considerarsi come una **scatola nera**. Al progettista interessa solamente i **flussi di dati** tra l'agente esterno ed il sistema analizzato.

Diagramma di contesto

È la rappresentazione delle iterazioni tra **sistema** e il **mondo esterno**. Contiene un **solo processo** che rappresenta il sistema nella sua **interezza**, tutti gli **agenti esterni**, i **flussi di dati** che gli agenti esterni e il sistema si scambiano ed eventuali **depositi** (precisazione dei **confini** del sistema analizzato).

Scomposizione dei processi

Ogni processo può essere **scomposto in sotto processi**, originando un **nuovo diagramma**. Permette di rappresentare le funzionalità del sistema a **diversi livelli di dettaglio.**

I flussi di I/O collegati al processo **padre** devono essere collegati anche ai processi **figli**, la scomposizione è **reversibile**. La miglior tecnica di scomposizione è quella basata sugli **eventi** a cui il processo deve rispondere.

Tom De Marco propone **7±2 sotto processi** ad ogni processo.

Problemi: ogni scomposizione genera nuovo diagramma, non esiste una **regola** per la scomposizione.

Pro e contro Data Flow Diagram

Pro: focus sull'**iterazione tra sistema e mondo esterno**, capacità di rappresentare **qualunque tipo di sistema** a diversi livelli, **intuitivo**, **immediato** come **strumento di comunicazione**. Costituisce una **linea guida** per gli **analisti**.

Contro: approccio **top-down** risulta inadeguato per sistemi con **requisiti instabili**, rischi di **orientamento** alle soluzioni tecniche e le regole sintattiche sono **limitate**.