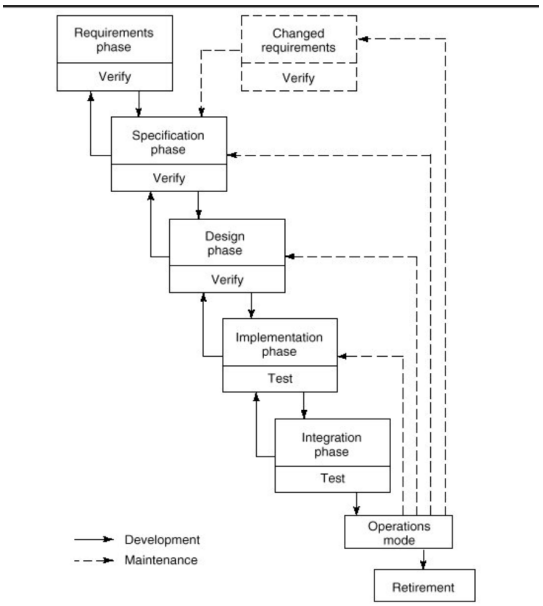
- ***<u>Lezione 4 – (21/10/2024)</u>***

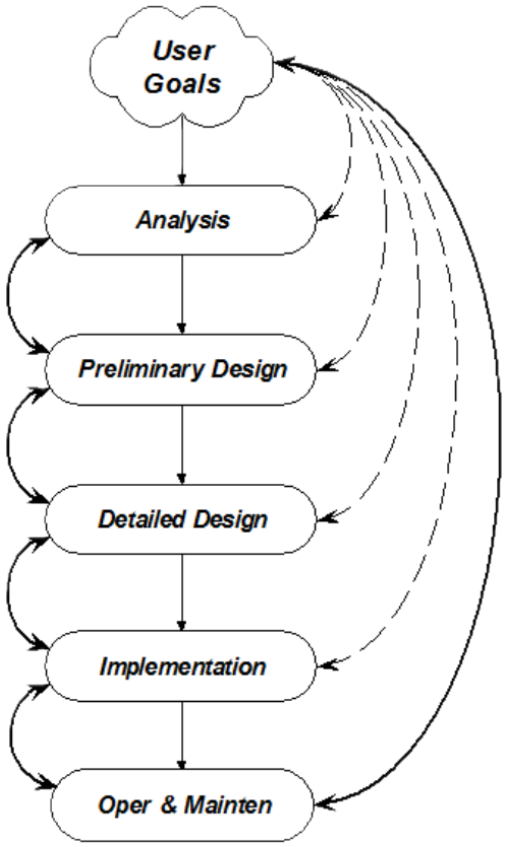
Il modello Modello Waterfall o Modello a Cascata, che si distingue nettamente dal modello Build & Fix. Viene chiamato "a cascata" poiché le fasi del processo sono presentate in sequenza, come in una cascata. Questo modello può essere considerato un punto di riferimento rispetto al quale vengono confrontati altri modelli di sviluppo software.

Nel Modello Waterfall, ogni fase non può essere considerata completata senza una verifica dei requisiti e della correttezza di quanto realizzato. In ogni rettangolo che rappresenta una fase specifica del ciclo di vita, è presente una riga con l'indicazione "Test" o "Verifica". Questo si deve al fatto che:

- Verify indica una verifica statica del lavoro svolto (controllo della correttezza, coerenza dei documenti e della progettazione),
- Test indica una verifica dinamica (esecuzione pratica del software per rilevare errori funzionali).

Un aspetto critico del Modello Waterfall è che i requisiti di un prodotto software tendono a cambiare continuamente durante il processo di sviluppo. Per questo motivo, il modello include meccanismi per il ritorno a fasi precedenti o l'integrazione di cambiamenti nei requisiti, come indicato dalle linee tratteggiate nel diagramma.



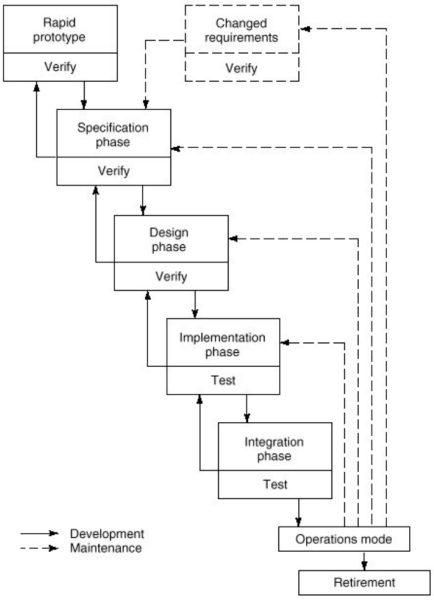


La figura illustra la differenza tra verifica e validazione nel processo di sviluppo del software.

- La verifica consiste nel controllare che un prodotto software sia stato sviluppato correttamente, ovvero che soddisfi i requisiti tecnici e di progetto.
Ad esempio: quando si passa un documento in input, si verifica che sia corretto rispetto alle specifiche, e solo in caso positivo si procede con la fase successiva.

- La validazione, invece, è un processo diverso, poiché ci si assicura che il prodotto finale soddisfi i bisogni e le aspettative dell'utente. Anche se un prodotto software può essere corretto dal punto di vista implementativo (cioè, ha superato la verifica), potrebbe non risultare utile o valido per l'utente che lo utilizzerà.

Nel corso degli anni, si è cercato di migliorare il modello a cascata (Waterfall). Uno dei primi miglioramenti fu l'introduzione del modello Rapid Prototyping.



In questo modello, nella fase di raccolta dei requisiti viene utilizzato un prototipo rapido, un prodotto che contiene principalmente l'interfaccia utente, ma che non esegue ancora tutte le funzionalità del sistema finale. Questo prototipo viene impiegato per aiutare l'ingegnere del

software a comprendere meglio i requisiti e a supportare le attività di sviluppo, consentendo di ottenere un feedback iniziale dagli utenti.

Il Prototyping Software (o **Prototipazione del software**) è una metodologia di sviluppo rapido che viene utilizzata principalmente per raccogliere e validare i requisiti di un software.

In pratica, si tratta di creare versioni preliminari del sistema, chiamate prototipi, che permettono a clienti e sviluppatori di avere un'idea concreta di come funzionerà il prodotto finale. I prototipi svolgono un ruolo fondamentale nel migliorare la comprensione reciproca tra chi utilizza il software e chi lo sviluppa.

Ci sono due usi principali dei prototipi nel processo di sviluppo software:

- *Elicitazione dei requisiti (o Requisiti dei rischi):* questo è il processo in cui i requisiti vengono scoperti o definiti. Un prototipo permette agli utenti di sperimentare una versione preliminare del sistema, aiutandoli a capire se e come il software supporterà il loro lavoro. Questo è particolarmente utile per chiarire aspetti che potrebbero non essere subito evidenti.

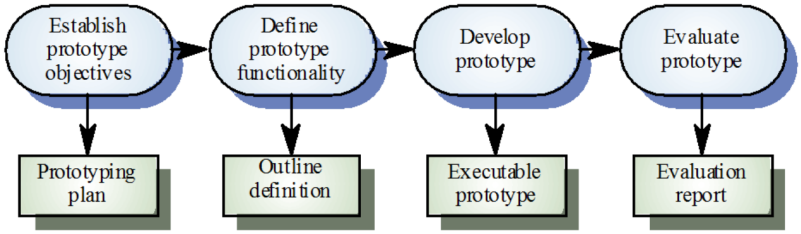
- *Validazione dei requisiti (o Convalida dei Requisiti):* una volta che i requisiti sono stati definiti, il prototipo viene utilizzato per verificare se il sistema risponde a quanto richiesto. In questa fase, si possono identificare errori o mancanze che potrebbero essere sfuggiti durante la definizione iniziale.

In sostanza, il *prototyping* serve a ridurre i rischi legati ai requisiti, rendendo il processo di sviluppo più sicuro e privo di fraintendimenti.

Il prototyping offre diversi vantaggi che lo rendono una pratica diffusa nello sviluppo software:

- Riduce fraintendimenti tra utenti e sviluppatori, migliorando la comunicazione e chiarendo meglio le aspettative.
- Individua servizi mancanti o confusi, ovvero funzionalità che magari non erano state previste inizialmente.
- Fornisce un sistema funzionante già nelle prime fasi dello sviluppo, il che aiuta a visualizzare concretamente come sarà il prodotto finale.
- Deriva la specifica software dal prototipo: il prototipo può essere usato come punto di partenza per scrivere la documentazione ufficiale del progetto
- Supporta la formazione degli utenti e i test del prodotto: gli utenti possono essere formati utilizzando il prototipo, e lo stesso può essere impiegato per eseguire test reali sul sistema (un processo chiamato "back-to-back").

Il processo di prototipazione software si articola in 4 fasi principali:



1. *Stabilire gli obiettivi del prototipo:* qui si definiscono le finalità del prototipo. Lo scopo potrebbe essere, ad esempio, verificare che una certa funzione risponda alle aspettative dell'utente. Questa fase culmina con la creazione di un piano di prototipazione.

2. *Definire la funzionalità del prototipo:* si stabiliscono le caratteristiche che il prototipo deve avere. Non tutte le funzionalità del sistema finale sono incluse, ma solo quelle più importanti o critiche per comprendere come funzionerà il software.

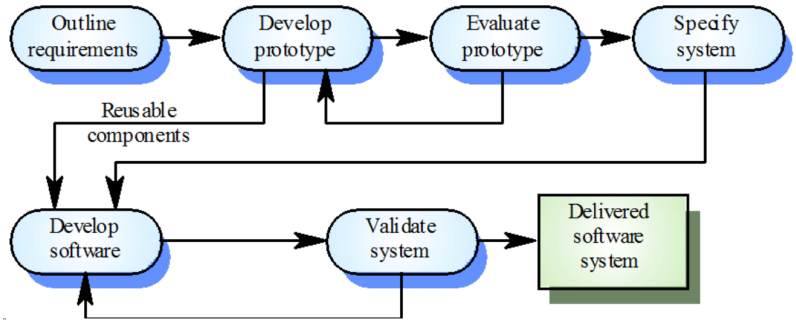
3. *Sviluppare il prototipo:* questa è la fase di costruzione vera e propria, in cui si realizza un prototipo eseguibile, basato sulle funzionalità precedentemente definite.

4. *Valutare il prototipo:* una volta sviluppato, il prototipo viene testato per verificarne l'efficacia. In questa fase si raccolgono feedback dagli utenti e si individua cosa funziona e cosa necessita di miglioramenti.

Prototipi come specifiche:

A volte i prototipi vengono utilizzati per scrivere parte della documentazione di progetto, ma ci sono alcune limitazioni. Ad esempio, le funzionalità critiche per la sicurezza non sono sempre facili da rappresentare in un prototipo, e i requisiti non funzionali (come prestazioni o sicurezza) non possono essere testati efficacemente con un prototipo.
Un'implementazione non ha valore legale come contratto.

Il prototipo usa e getta *(Throw-away)*



Il prototipo "usa e getta" è una versione preliminare del sistema che viene creata per identificare problemi nei requisiti e poi scartata una volta ottenuti i feedback necessari. Questo tipo di prototipo non è pensato per essere mantenuto o utilizzato come prodotto finale, perché:

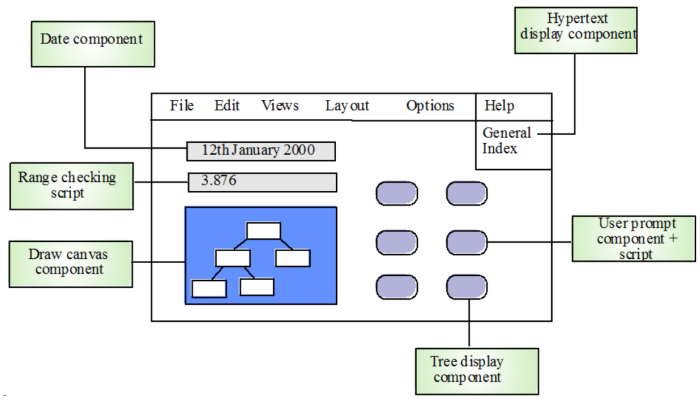
- Alcune caratteristiche potrebbero essere state omesse
- Non è pensato per la manutenzione a lungo termine, quindi non ha una struttura ottimizzata per essere ampliata o modificata nel tempo
- La struttura del prototipo è generalmente difficile da mantenere

Distribuire un prototipo come prodotto finale è sconsigliato perché potrebbe non rispettare i requisiti non-funzionali, mancare di documentazione, avere una struttura compromessa dai cambiamenti e non seguire gli standard di qualità previsti.

Alcuni punti fondamentali da ricordare sulla prototipazione sono:

- I prototipi danno agli utenti un'impressione concreta delle capacità del prodotto, aiutandoli a visualizzare come sarà.
- Il prototyping è sempre più utilizzato quando è necessario sviluppare un prodotto rapidamente.
- I prototipi usa e getta vengono creati per comprendere meglio i requisiti, ma non dovrebbero essere utilizzati come base per il prodotto finale.
- Lo sviluppo rapido dei prototipi è essenziale, anche se ciò significa dover omettere alcune funzionalità o allentare i vincoli non funzionali.
- La programmazione “visuale” è uno strumento comune nel prototyping, in cui l'interfaccia utente viene costruita utilizzando componenti standard, come avviene in linguaggi come VisualBasic.

→ Problemi dello sviluppo visuale



Nonostante i vantaggi della programmazione visuale, essa presenta alcuni problemi, come:

- Difficoltà nel coordinare il lavoro di un team: quando più persone lavorano insieme, può essere difficile mantenere tutto sotto controllo.

- Mancanza di un'architettura software esplicita, che rende complesso gestire lo sviluppo a lungo termine.
- Dipendenze complesse tra le parti del programma che possono compromettere la manutenibilità del sistema, rendendo difficili le modifiche future.

Riassumendo possiamo dire che, il prototyping è uno strumento utile per sviluppare software in modo rapido ed efficiente, specialmente nelle fasi iniziali di raccolta e validazione dei requisiti, ma è importante essere consapevoli delle sue limitazioni e dei rischi associati, soprattutto quando si tratta di prototipi "usa e getta".

- Processo di iterazione:

L'iterazione nei processi di sviluppo è essenziale, poiché i requisiti di un progetto evolvono continuamente, specialmente nei prodotti complessi. Questa iterazione può essere applicata a qualsiasi modello di sviluppo e comprende approcci come lo *sviluppo incrementale* e *lo sviluppo a spirale.*

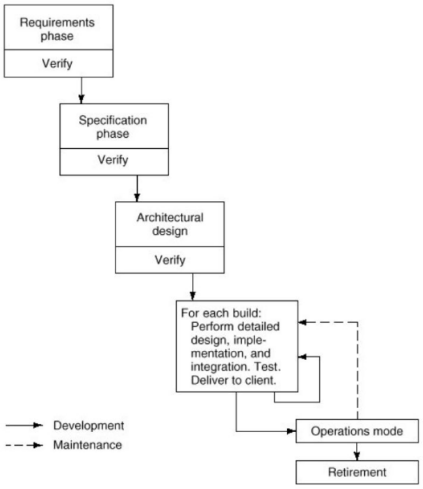
1. *Sviluppo incrementale:*

    - Il prodotto è sviluppato e lanciato con incrementi dopo aver stabilito un'architettura generale;
    - I requisiti e le specifiche per ogni aggiunta possono essere sviluppati;
    - Gli utenti possono sperimentare con gli incrementi (le aggiunte) consegnati mentre gli altri vengono sviluppati. Pertanto questo serve come una forma di prototipo;
    - Prevede di unire alcuni dei vantaggi della prototipazione ma con un processo maggiormente gestibile ed una struttura migliore;

2. *Modello incrementale:*

    - Il prodotto software viene sviluppato e rilasciato per incrementi (build) successivi;
    - Include aspetti tipici del modello basato su rapid prototyping (l'utente può sperimentare l'utilizzo del prodotto contenente gli incrementi consegnati, mentre i restanti sono ancora in fase di sviluppo;
    - Si rivela efficace quando il cliente vuole continuamente verificare i progressi nello sviluppo del prodotto e quando i requisiti subiscono modifiche;
    - Può essere realizzato in due versioni alternative:

- Versione con overall architecture;



- Versione senza overall architecture (più rischiosa).