Modelli di sviluppo SW

H1 # Introduzione

H2 Vediamo ora i dettagli riguardanti il **processo** di sviluppo di un SW. In particolare analizzeremo le **fasi di sviluppo** e i **modelli di ciclo di vita** del SW.

H3 ## Definizioni preliminari

Processo di produzione

H5 > La sequenza di attività svolte per *progettare, realizzare, consegnare e modificare* un prodotto SW

Ciclo di vita di un SW

> È l'*insieme delle fasi* in cui si trova il sistema

H5

Modello di processo

H5 > Esso determina l'**organizzazione** del *processo di produzione* e quindi anche del *ciclo di vita*

H3 ## Fasi di sviluppo

Studio di fattibilità

H4 > Si tratta della produzione di un documento detto *Feasibility Study Document* (***FSD***), il quale valuta i **costi** e **benefici** dell'applicazione.

In particolare, l'FSD deve contenere:

- la definizione del problema

- le possibili soluzioni
- per ogni soluzione, una stima di:
  - benefici
  - costi
  - risorse richieste
  - tempi di consegna

In genere, l'FSD, scritto in linguaggio naturale e spesso pieno di ambiguità e contraddizioni, è ciò che viene consegnato all'ingegnere del SW, e a partire da questo egli dovrà produrre le **specifiche**. Tale fase è detta **analisi dei requisiti** 👇

Analisi dei requisiti (o Specifica)

H4

> Si tratta del processo di produzione delle specifiche, in un documento detto *Requirements Analysis and Specification Document* (***RASD***). Esse hanno lo scopo di determinare le **funzionalità**, e le **proprietà** del SW in termini di *performance, facilità d'uso, portabilità etc.*

Per poter produrre un documento così complesso e importante, occorre:

- Capire il **dominio** dell'applicazione
- Capire l'**interfaccia** tra l'applicazione e l'ambiente esterno
- Identificare i principali **stakeholder** e comprendere le loro aspettative
- Stabilire le **qualità** che devono essere raggiunte

La **specifica** è un *processo incrementale* che richiede *continua* ***interazione*** *col cliente*.

Un buon modo per scrivere le specifiche è quello di seguire le linee guida delle
"**5 W**":

Definizione del **dominio**:

Cioè **chi** userà il sistema e **a quale scopo** dovrebbe usarlo

- ***Who*** *will use the system*
- ***Why*** *should it be developed +* ***why*** *will the users use it*

Requisiti funzionali:

Descrivono **cosa** fa il sistema (non come!)

- ***What*** *will the system provide (not how!)*

Requisiti non funzionali:

Descrivono *affidabilità, prestazioni, interfaccia, limiti fisici, limiti operativi, etc.*

- ***Where*** *will the system be used, on which architecture*

Requisiti del processo e manutenzione:

Descrivono i tempi del processo e dell'eventuale manutenzione del SW

- ***When*** *and how long will the system be used*

Per quanto riguarda in dettaglio il RASD, esso:

- Deve permettere al cliente di verificare le caratteristiche specificate
- Deve consentire al progettista di procedere con lo sviluppo
- Deve essere:
  - facilmente comprensibile
  - preciso, completo, coerente, non ambiguo
  - modificabile (la specifica è un processo incrementale!)
- Dovrebbe includere:
  - *user manual* preliminare
  - *system test plan* (definizione delle modalità di *testing* del sistema)

Progettazione (*Design*)

> H4
>
> Si tratta delle definizione dell'**architettura del SW** sia a livello *globale* che a livello di singole **componenti**, producendo un documento detto ***Design Document***

Il *Design Document* contiene:

- la ***definizione*** *delle componenti*

- le ***relazioni*** *tra le componenti*
- l'***interazione*** *tra le componenti*

La fase di **progettazione** consente lo *sviluppo concorrente* e la *separazione delle responsabilità*, cioè la possibilità di sviluppare in parallelo e in modo indipendente i singoli componenti del SW.

Implementazione e test dei componenti

H4

> È la fase in cui i programmi vengono *effettivamente* realizzati e i programmatori effettuano i *test sulle funzionalità relative ai singoli componenti*

Per ogni componente viene prodotto:

- Il **codice sorgente**
- La **documentazione**
- **Specifiche dei test** effettuati

Integrazione e test del sistema

H4

> È la fase in cui il codice delle singole componenti si *assembla* e ne si verifica l'effettiva *compatibilità*, *risolvendo eventuali errori* derivanti dalle interazioni fra i componenti

Spesso i problemi di integrazione nascono nell'incompatibilità dei parametri di input e output delle funzioni o metodi del programma.

Nel caso di modelli di sviluppo ***agili***, le fasi di implementazione dei singoli componenti e la fase di integrazione non sono distinte.

Consegna

La consegna è divisibile in 3 sottofasi:

H4 **Alpha testing**

H5

> Il sistema viene distribuito ad un insieme di utenti interni alla software house per testarne il funzionamento, e eventualmente correggere gli errori

Beta testing

H5 > Il sistema viene distribuito ad un insieme di utenti esterni alla software house, i quali sono a conoscenza che il prodotto è in fase beta, per testarne il funzionamento, e eventualmente correggere gli errori

Distribuzione

> Il SW viene *definitivamente* rilasciato agli utenti

H5

Manutenzione

H4 > Si tratta della fase che include tutti i cambiamenti, *correzioni* o *evoluzioni*, che *seguono la consegna* del SW

È importante notare come la fase di manutenzione sia spesso la fase *più costosa* dello sviluppo

H3 Modelli di processo

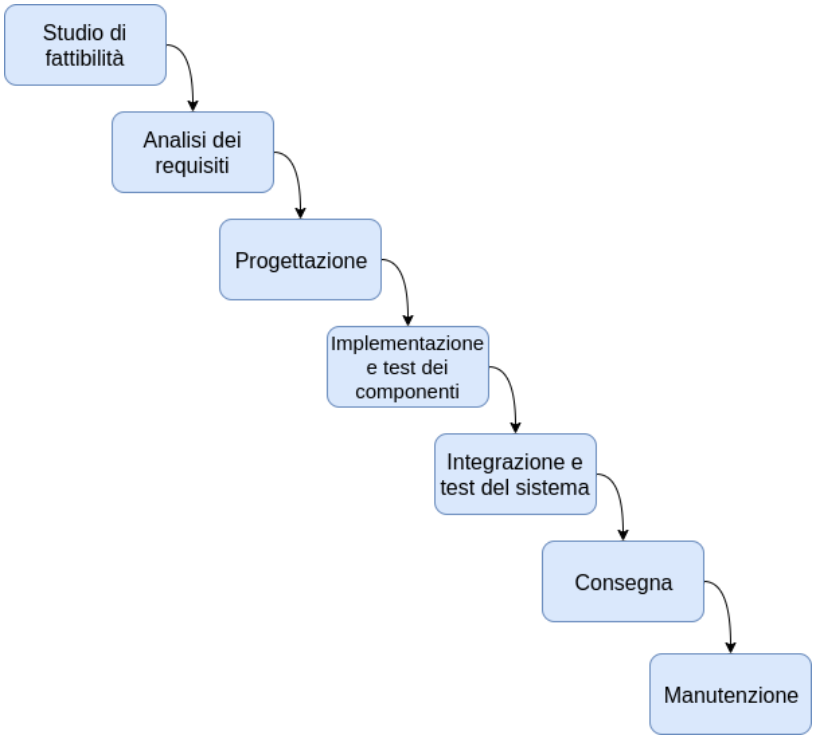
Vediamo ora i diversi modelli per organizzare le fasi di sviluppo, cioè i diversi **modelli di processo**.

Essi sono:

- Modelli **a cascata**
- Modelli **evolutivi**
- Modelli **agili**

Modelli a cascata

H4 > Si tratta semplicemente di organizzare le fasi di sviluppo una di seguito all'altra, dove *l'output di ogni fase rappresenta l'input della successiva*



Esso fu introdotto e standardizzato nel 1970.

Ben presto se ne capirono i limiti:

❌ È un processo ***black box***, cioè un processo in cui le varie fasi sono *"chiuse"* rispetto alle altre, e non vi e quindi possibilità di tornare facilmente a vecchie fasi dello sviluppo

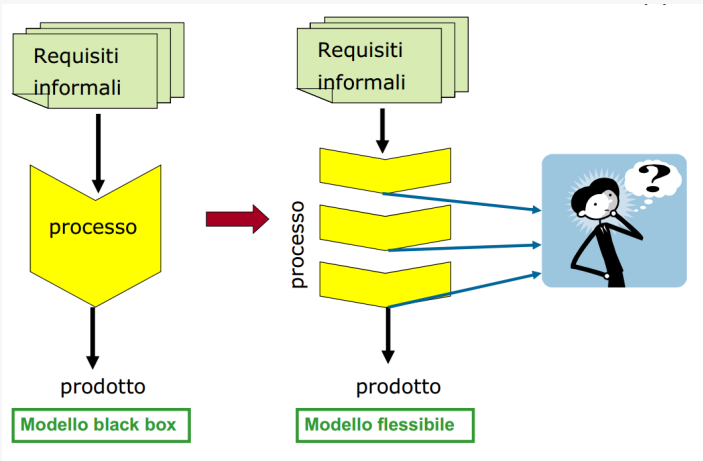
❌ Non tiene in considerazione il fatto che i requisiti cambiano:

- Cambia il contesto:
  - HW, dominio, algoritmi...
- Specifiche sbagliate:
  - requisiti errati
  - scarsa conoscenza del dominio

Modelli evolutivi

Per poter superare i limiti del modello a cascata, bisognava abbandonare l'idea di un modello **black-box**, e invece passare a un modello in cui le fasi dello sviluppo siano *"aperte"*, cioè analizzabili, ripercorribili etc., cioè a un **modello flessibile**.

H4



passaggio dal modello black box ai modelli flessibili

I modelli flessibili permettono l'esistenza di **feedback** per consentire cambiamenti anticipati. I modelli che adottano questo tipo di strategia sono i **modelli incrementali**. Essi sono in grado di adattarsi ai cambiamenti di:

- requisiti
- specifiche
- design

Inoltre essi prevedono la possibilità di cambiare il progetto e i documenti prodotti nelle varie fasi dello sviluppo, in quanto come abbiamo detto sono modelli flessibili. È questo il motivo per cui questi modelli si adattano bene ai cambiamenti.
Il modo in cui i modelli incrementali ottengono *feedback* dai processi è attraverso due attività: ***verifica*** e ***validazione*** (o ***convalida***).

Verifica

"Are we doing the product right?"

H6 > Si tratta di un'attività che determina se il processo di sviluppo del SW è corretto

Validazione

"Are we doing the right product?"

H6 > Si tratta di un'attività che determina se il SW soddisfa i requisiti voluti dal cliente

Vediamo in dettaglio alcuni specifici modelli incrementali.

Prototipazione

H5

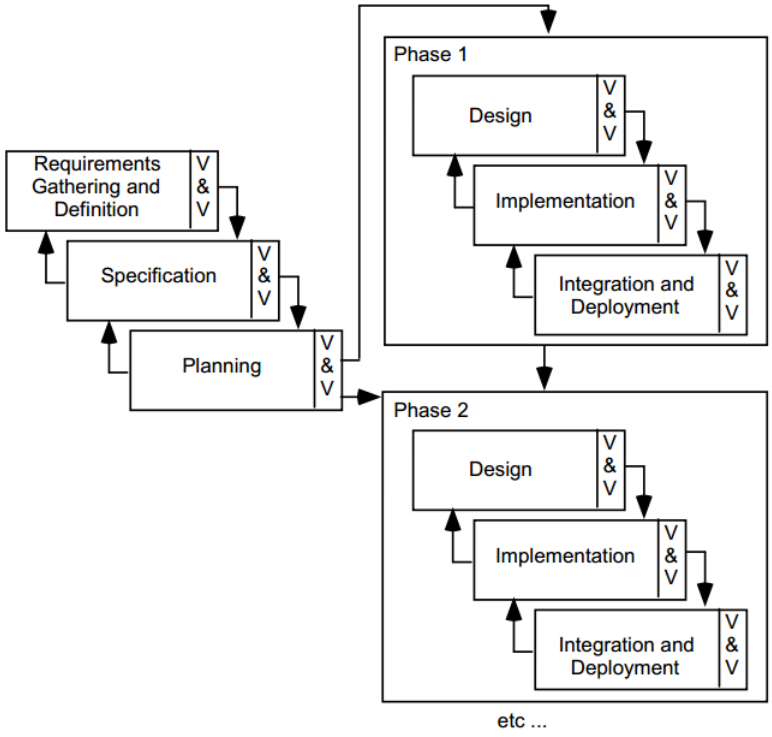
> Viene inizialmente sviluppato un **prototipo**, cioè un modello *approssimato* del sistema che ha lo scopo di *fornire feedback*, il quale dovrà evidenziare eventuali errori di progettazione.

È utile prototipare i componenti del sistema su cui si hanno maggiori dubbi in fase di progettazione.

Inoltre per minimizzare i costi è importante che lo sviluppo si basi sul *principio di massima riusabilità*

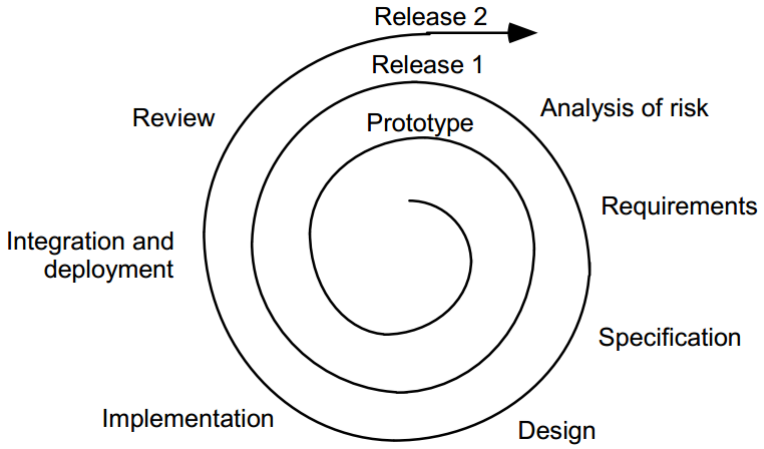
Modello a fasi di release

H5



Modello a spirale

H5



> È un processo **ciclico**, che comprende tutte le fasi dello sviluppo, e alla fine di ogni ciclo vi sarà una release sempre più perfezionata.

Il costo *aumenta* a ogni ciclo.

Esso può essere visto come un *metamodello* di sviluppo, in quanto comprende altri modelli specifici.

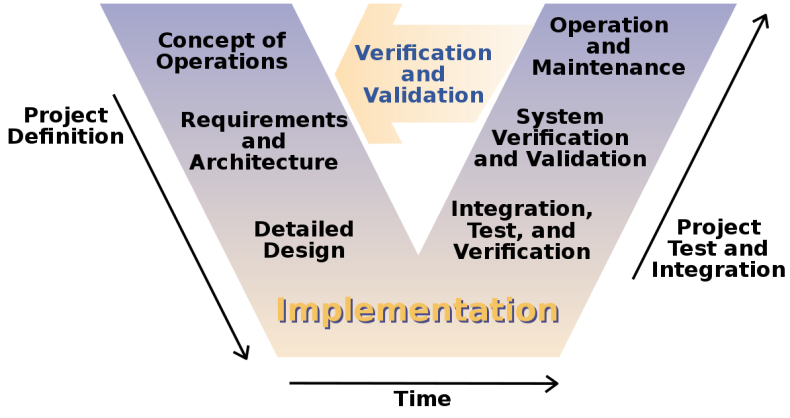
Il principio è quello di mantenere uno sviluppo continuo del sistema, senza una deadline precisa.

Modello a V

È un'estensione del modello a cascata e una semplificazione di quello a spirale.

H5

> È un modello in cui si prosegue "a cascata" fino alla fase di implementazione, dopodiché vi è un'attività di ***verifica e convalida*** *delle varie fasi precedenti*, "risalendo la cascata"



È un modello meno costoso di quello a spirale, ma anche meno rigoroso.

Modelli agili

H4

> Modelli che coinvolgono quanto più possibile gli stakeholder, ottenendo un elevata reattività e feedback.

Manifesto per lo sviluppo Agile

- Gli individui e le interazioni più che i processi e gli strumenti
- Il SW funzionante più che la documentazione esaustiva
- La collaborazione col cliente più che la negoziazione di contratti
- Rispondere al cambiamento più che seguire un piano

Questa filosofia porta in pratica alle seguenti conseguenze:

- **Rilasci frequenti** del SW
- Continua **collaborazione col cliente**
- **Documentazione ridotta**
- **Valutazione continua di possibilità di cambiamenti**