

Ingegneria del Software

UniVR - Dipartimento di Informatica

Fabio Irimie

2° Semestre 2024/2025

Indice

1	Introduzione	2
1.1	Cos'è l'ingegneria del software?	2
1.1.1	Economia	2
1.1.2	Fallimento dei progetti	2
1.2	Prodotti software	2
1.2.1	Prodotti generici	2
1.2.2	Prodotti personalizzati	2
1.3	Caratteristiche di un buon software	2
1.4	Problemi che influenzano il software	3
1.5	Tipi di applicazione	3
1.6	Principi fondamentali	4
2	Processi del software	4
2.1	Modelli di processo	5
2.1.1	Modello a cascata	5
2.1.2	Modello incrementale	6
2.1.3	Modello a integrazione e configurazione	6
2.2	Attività dei processi	7
2.2.1	Specificazione	7
2.2.2	Design	7
2.2.3	Implementazione	8
2.2.4	Validazione	9
2.2.5	Evoluzione	9

1 Introduzione

1.1 Cos'è l'ingegneria del software?

L'ingegneria del software è un insieme di metodologie, teorie, metodi e strumenti, che guidano nello sviluppo di software **professionale** in modo che esso fornisca le funzionalità richieste, sia performante e mantenibile, affidabile e usabile. Il **software** non è solo il programma e l'eseguibile, ma anche **la documentazione associata**. Le attività principale dell'ingegneria del software sono:

- **Specifica del software:** Il cliente e l'ingegnere del software definiscono le funzionalità e i vincoli del software da produrre.
- **Sviluppo:** Il software viene progettato e implementato.
- **Validazione:** Il software viene verificato per assicurarsi che soddisfi i requisiti forniti dal cliente.
- **Evoluzione:** Il software viene modificato per adattarlo a nuovi requisiti del cliente o del mercato.

1.1.1 Economia

Il software spesso costa più dell'hardware, e il costo è più legato alla manutenzione che allo sviluppo. Quindi l'ingegneria del software è importante per ridurre i costi di manutenzione.

1.1.2 Fallimento dei progetti

I progetti spesso falliscono per la **crescente complessità del sistema**: nuove tecniche di sviluppo e nuove tecnologie rendono i sistemi sempre più complessi, e quindi più difficili da mantenere.

1.2 Prodotti software

1.2.1 Prodotti generici

Sono prodotti che vengono pubblicizzati e venduti a qualsiasi cliente che ne faccia richiesta. La specifica è di proprietà del produttore e le decisioni sulle modifiche sono prese dai produttori.

1.2.2 Prodotti personalizzati

Sono prodotti che vengono commissionati da clienti specifici per soddisfare le loro necessità. La specifica è di proprietà del cliente dopo una contrattazione con il produttore e le decisioni sulle modifiche sono prese dal cliente.

1.3 Caratteristiche di un buon software

- **Mantenibilità:** la facilità con cui il software può essere modificato per correggere difetti, migliorare le prestazioni o adattarlo a cambiamenti.

- **Affidabilità e sicurezza:** la capacità del software di svolgere le sue funzioni in modo corretto. I malfunzionamenti non devono causare danni fisici o economici. Utenti malintenzionati non devono poter violare la sicurezza del sistema.
- **Efficienza:** il software non deve sprecare risorse (CPU, memoria, ecc).
- **Accettabilità:** il software deve essere accettato dagli utenti per i quali è stato progettato.

1.4 Problemi che influenzano il software

- **Eterogeneità:** I sistemi devono sempre di più operare in modo distribuito, e quindi devono essere in grado di comunicare con sistemi diversi. Oppure bisogna garantire che il software funzioni su piattaforme diverse.
- **Cambiamento sociale o del business:** Il software deve essere in grado di adattarsi a cambiamenti sociali o organizzativi nelle aziende.
- **Sicurezza e fiducia:** Siccome il software fa parte della vita di tutti i giorni è essenziale che ci sia fiducia nel software.
- **Scalabilità:** Il software deve essere sviluppato a più scale, cioè soluzione che funziona in piccolo deve adattarsi anche a grandi scale senza rischiare di fallire.

1.5 Tipi di applicazione

Ci sono diversi tipi di sistemi software e non c'è un insieme universale di tecniche applicabili a tutti i sistemi. Quindi i metodi e gli strumenti utilizzati dipendono dal tipo di applicazione che si deve sviluppare, dalle richieste dei clienti e dalle competenze degli sviluppatori. I principali tipi di applicazione sono:

- **Applicazioni stand-alone:** Sono applicazioni che si eseguono su un singolo computer locale e includono le funzionalità necessarie per l'utente.
- **Applicazioni interattive transaction-based:** Sono applicazioni che sono eseguite su un computer in remoto e sono accessibili dagli utenti dai propri computer. Queste includono le applicazioni web come gli e-commerce.
- **Sistemi embedded:** Sono applicazioni che controllano e gestiscono dispositivi hardware. Questi sistemi sono i più numerosi.
- **Sistemi batch:** Questi sistemi elaborano grandi quantità di dati per produrre un output.
- **Sistemi di intrattenimento:** Questi sistemi sono per uso personale e servono ad intrattenere l'utente.
- **Sistemi di modellazione e simulazione:** Questi sistemi sono sviluppati da scienziati per modellare processi fisici o situazioni che includono tanti oggetti separati che interagiscono tra di loro.
- **Sistemi di collezione di dati (o IOT):** Questi sistemi raccolgono dati da sensori e li inviano ad un sistema centrale per l'elaborazione.

- **Sistemi di sistemi:** Questi sistemi sono composti da più sistemi software che collaborano tra di loro.

1.6 Principi fondamentali

- I sistemi devono essere sviluppati utilizzando un processo strutturato e ben pensato.
- L'affidabilità e la performance sono importanti per ogni tipo di software.
- Capire e gestire i requisiti del software è essenziale.
- Dove appropriato si dovrebbe riutilizzare software che è già stato sviluppato al posto di svilupparne uno da zero.

2 Processi del software

I **processi** del software sono un insieme di attività strutturate che sono necessarie per sviluppare un sistema software. Ci sono diversi tipi di processi, ma tutti coinvolgono le seguenti attività:

- **Specificazione**
- **Sviluppo**
- **Design e implementazione**
- **Validazione**
- **Evoluzione**

Un modello di processo software è una rappresentazione astratta di un processo da un punto di vista particolare.

Per descrivere e discutere i processi di solito si parla delle attività all'interno di questi processi e dell'ordine in cui queste attività vengono svolte. Alcune descrizioni possono anche includere:

- **Prodotti:** cioè i risultati di un'attività.
- **Ruoli:** cioè le responsabilità di un individuo o di un gruppo.
- **Pre e post condizioni:** cioè le condizioni che devono essere soddisfatte prima e dopo un'attività.

Alcuni esempi di processi sono:

- **Processi plan-driven:** sono processi dove tutte le attività sono pianificate in anticipo e tutti i progressi sono misurati rispetto al piano.
- **Processi agili:** sono processi in cui la pianificazione è incrementale ed è più facile adattarsi ai cambiamenti.

Nel pratico si utilizza un approccio ibrido tra i due.

2.1 Modelli di processo

- **Modello a cascata:** è un modello plan-driven con fasi di specificazione e di sviluppo distinte e separate
- **Modello incrementale:** la specifica, lo sviluppo e la validazione sono intercalate. Può essere plan-driven o agile.
- **Integrazione e configurazione:** i sistemi sono sviluppati da componenti già esistenti e configurabili.

Nel pratico, sistemi molto grandi sono sviluppati utilizzando un processo che incorpora elementi di tutti e tre i modelli.

2.1.1 Modello a cascata

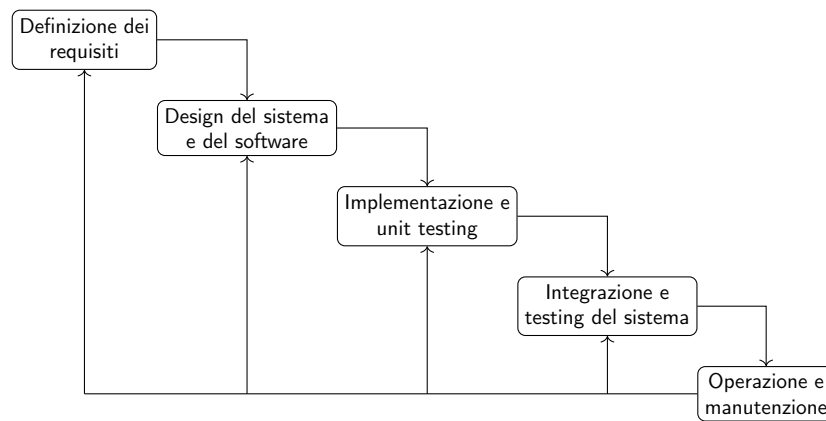


Figura 1: Modello a cascata

Le diverse fasi del modello a cascata sono:

- Analisi e definizione dei requisiti
- Design del sistema e del software
- Implementazione e unit testing
- Integrazione e testing del sistema
- Operazione e manutenzione

Il vantaggio del modello a cascata è la partizione inflessibile del progetto in fasi distinte rende difficile rispondere ai cambiamenti dei requisiti, quindi questo modello è adatto solo per progetti dove i requisiti sono ben definiti e i cambiamenti rimangono limitati. Però pochi progetti soddisfano queste condizioni.

Questo modello è più utilizzato in sistemi molto grandi e sviluppati da molte persone. In quelle circostanze la natura plan-driven di questo modello è utile per coordinare il lavoro.

2.1.2 Modello incrementale

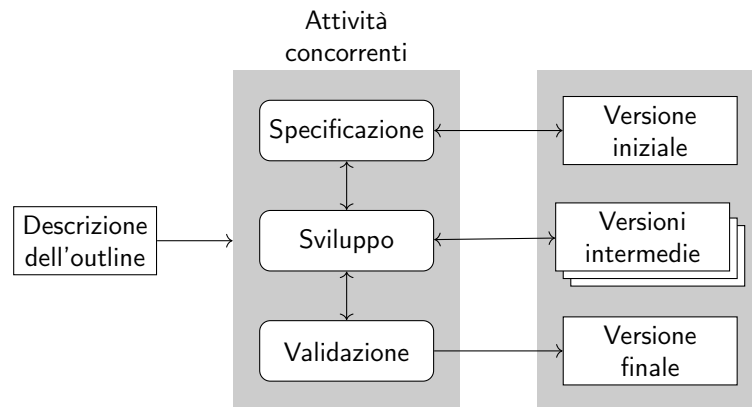


Figura 2: Modello incrementale

I vantaggi del modello incrementale sono:

- Il costo del cambiamento dei requisiti è ridotto e la quantità di analisi e documentazione da rifare è molto meno rispetto al modello a cascata.
- È più facile ottenere un feedback dagli utenti sul lavoro svolto.
- È possibile consegnare software utile più velocemente al cliente.

Gli svantaggi del modello incrementale sono:

- Il processo non è visibile, cioè è difficile valutare il progresso. I manager hanno bisogno di costanti "deliverables", cioè risultati intermedi, per misurare il progresso e non sempre è poco costoso produrre documenti che riflettono lo stato di ogni versione del software.
- La struttura del sistema tende a decadere con l'aggiunta di nuovi incrementi. A meno che il tempo e i soldi non vengano spesi per il refactor e il miglioramento del software, i cambiamenti regolari tenderanno a corrompere la struttura del sistema e quindi incorporare nuovi cambiamenti diventerà sempre più difficile e costoso.

2.1.3 Modello a integrazione e configurazione

Questo modello si basa sulla realizzazione di sistemi integrati tramite componenti software già esistenti e configurabili. Il riutilizzo è l'approccio standard per creare molti tipi di sistemi.

I tipi di software riutilizzabile sono:

- Applicazioni o sistemi stand-alone (a volte chiamati COTS, Commercial Off-The-Shelf) che sono configurati per essere utilizzati in un certo ambiente
- Collezioni di oggetti o librerie che sono sviluppati come pacchetti per l'integrazione con vari framework

- Servizi web sviluppati secondo standard che permettono l'interazione tramite internet

Le attività principali di questo modello sono:

- Specificazione dei requisiti
- Ricerca e analisi del software riutilizzabile
- Adattamento dei requisiti
- Configurazione del sistema
- Adattamento e integrazione dei componenti

I vantaggi di questo modello sono:

- Riduzione dei costi e del rischio siccome meno software è sviluppato da zero
- Deployment più veloce

Gli svantaggi sono:

- È inevitabile fare compromessi per i requisiti, quindi i sistemi potrebbero non essere più ciò che aveva richiesto il cliente
- Perdita di controllo sull'evoluzione dei componenti riutilizzati

2.2 Attività dei processi

I processi software reali sono sequenze interconnesse di attività tecniche, collaborative e manageriali con l'obiettivo di specificare, progettare, implementare e testare un sistema software. Questi quattro processi base sono organizzati in modo differente in processi di sviluppo diversi.

2.2.1 Specificazione

Il processo di stabilire che processi sono richiesti e quali sono i vincoli sull'operazione e lo sviluppo del sistema è chiamato **specificazione del software** e consiste in:

- **Ricavo e analisi dei requisiti:** cosa vogliono o cosa si aspettano gli utenti interessati dal sistema
- **Specifica dei requisiti:** definizione formale dei requisiti del sistema
- **Validazione dei requisiti:** verifica che i requisiti siano completi e consistenti

2.2.2 Design

Questo processo consiste nella conversione delle specifiche del sistema in un sistema eseguibile e consiste in:

- **Design del software:** progettazione di una struttura software che realizzi i requisiti del sistema
- **Implementazione:** traduzione del design in un programma eseguibile

Queste attività sono interconnesse e spesso si intrecciano.

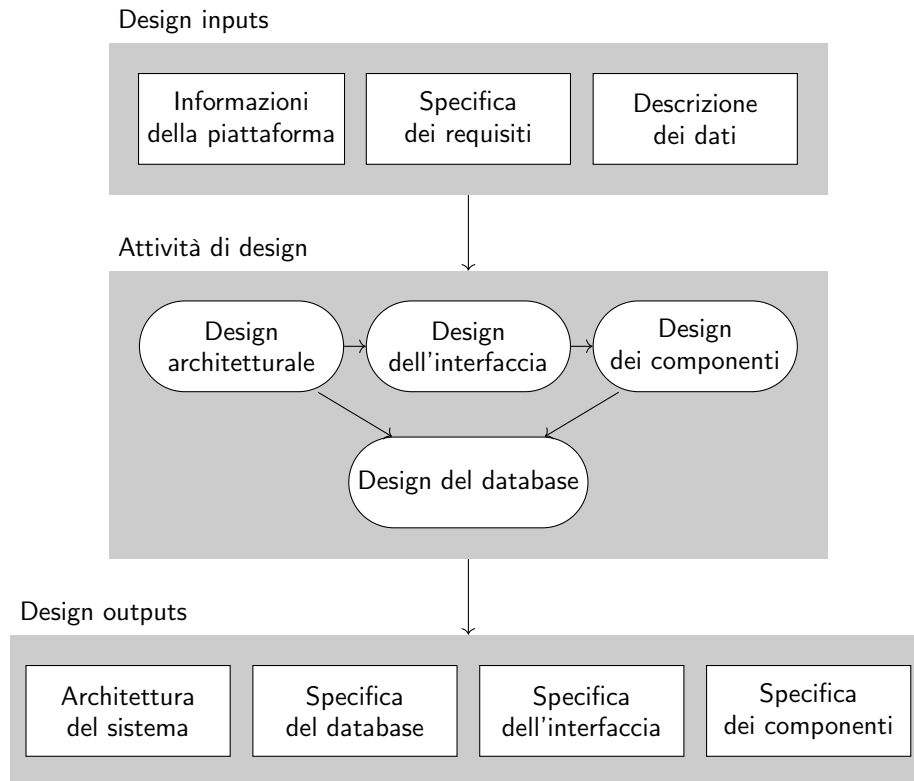


Figura 3: Modello generale del processo di design

Le principali attività di design sono:

- **Design architetturale** dove si identifica la struttura complessiva del sistema, i componenti principali, le loro relazioni e come sono distribuiti
- **Design del database** dove si progettano le strutture dati del sistema e come esse vengono rappresentate all'interno del database
- **Design dell'interfaccia** dove si definiscono le interfacce tra i componenti del sistema
- **Design e selezione dei componenti** dove si cercano componenti riutilizzabili. Se non sono disponibili si progettano nuovi componenti

2.2.3 Implementazione

Il software è implementato sviluppando dei programmi, oppure configurando un'applicazione. Il design e l'implementazione sono attività interconnesse per la maggior parte dei sistemi software.

La programmazione è un'attività individuale senza alcun processo standard, mentre invece il debugging consiste nel trovare e correggere errori nei programmi.

2.2.4 Validazione

La verifica e la validazione serve a dimostrare che un sistema è conforme alle specifiche e che soddisfa i requisiti del cliente. La validazione consiste nel controllare e verificare i processi e anche nel fare **testing**.

Il testing consiste nell'esecuzione di un sistema con dei casi di test derivati dalla specificazione dei dati reali che il software deve processare. Il testing è l'attività di validazione più utilizzata.

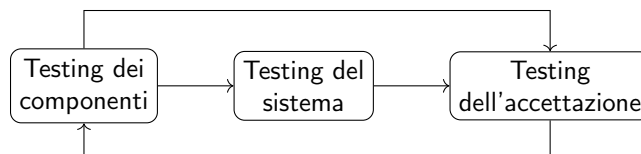


Figura 4: Fasi del testing

Le principali fasi del testing sono:

- **Testing dei componenti:** I singoli componenti vengono testati individualmente. I componenti possono essere funzioni, oggetti o gruppi coesi di tali entità.
- **Testing del sistema:** I componenti vengono integrati e il sistema viene testato come un'entità unica. È importante testare eventuali proprietà emergenti
- **Testing dell'accettazione:** Il sistema viene testato con i dati reali del cliente per verificare che soddisfi i requisiti del cliente

2.2.5 Evoluzione

Il software è flessibile e può cambiare col cambiamento dei requisiti. Anche se la linea tra sviluppo e mantenimento è spesso sfumata, questo diventa irrilevante siccome sempre meno sistemi software vengono sviluppati da zero.

