

INGEGNERIA DEL SOFTWARE

Alessandro Zappatore

Università del Piemonte Orientale
Anno accademico 2024/2025, 1° semestre

Indice

1	Introduzione	2
1.1	Cos'è l'Ingegneria del software	2
1.2	Sistemi software	2
1.3	Il processo software	2
1.4	Modelli di processo	2
1.4.1	Modello a cascata (waterfall)	2
1.4.2	Modello iterativo	3
1.4.3	Altri modelli	3
1.5	Gestione del processo	3
1.6	Strumenti CASE	3
2	Specifica	4
2.1	Requisiti funzionali	4
2.2	Requisiti non funzionali	4
2.2.1	Requisiti di prodotto	4
2.2.2	Requisiti organizzativi	5
2.2.3	Requisiti esterni	5
2.3	Sistemi critici	5
2.4	Costo dell'affidabilità	5
2.5	Processo di specifica	6
2.5.1	Studio di fattibilità	6
2.5.2	Deduzione dei requisiti	6
2.5.3	Analisi dei requisiti	6
2.5.4	Validazione dei requisiti	7
3	Progettazione	7
3.1	Attività di progettazione	7
3.1.1	Progettazione architetturale	8
3.1.2	Progettazione delle strutture dati	8
3.1.3	Progettazione degli algoritmi	8
3.1.4	Progettazione dell'interfaccia utente (grafica)	8
3.2	Progettazione architetturale	8
3.2.1	Strutturazione	8
3.2.2	Deployment	9
3.2.3	Metodo di controllo	9
3.2.4	Modellazione del comportamenti ad oggetti	10

1 Introduzione

1.1 Cos'è l'Ingegneria del software

L'ingegneria del software è l'applicazione del processo ingegneristico allo sviluppo di prodotti software.

1.2 Sistemi software

Un *sistema software* è un insieme di componenti software che funzionano in modo coordinato allo scopo di informatizzare una certa attività. La realizzazione di un sistema software richiede l'impiego di un gruppo di lavoro, nel quale ogni persona ricopre un ruolo ben preciso e le attività dei vari gruppi vanno coordinate, e tempo da dedicare alle varie fasi di sviluppo.

Esistono due categorie di sistemi software:

- i **sistemi generici**, realizzati per essere utilizzati da tanti clienti, venduti sul mercato. Requisiti dettati dalle tendenze di mercato;
- **sistemi customizzati**, richiesti da uno specifico cliente (il committente).

1.3 Il processo software

Con *ingegneria del software* si intende l'applicazione del processo dell'Ingegneria alla produzione di sistemi software. Il processo è suddiviso in:


- **specifica**: definizione dei requisiti funzionali e non funzionali
- **progettazione**: si definiscono architettura, controllo, comportamento dei componenti, strutture dati, algoritmi, struttura del codice, interfaccia utente
- **implementazione**: scrittura del codice e integrazione dei moduli
- **collaudo**: si controlla se il sistema ha difetti di funzionamento e se soddisfa i requisiti
- **manutenzione**: modifiche del sistema dopo la consegna

1.4 Modelli di processo

1.4.1 Modello a cascata (waterfall)

Esegui le 5 fasi in modo sequenziale (modello tradizionale):

specifica → progettazione → implementazione → collaudo → manutenzione



Foto/Waterfall1.png

1.4.2 Modello iterativo

1. Si esegue un po' di specifica, un po' di progettazione, un po' di implementazione e un po' di collaudo;
2. Viene generato un prototipo che realizza un sottoinsieme dei servizi;
3. Si ripete dal punto 1 fino a realizzare tutti i servizi richiesti.

1.4.3 Altri modelli

- Modello agile;
- Modello a spirale;
- Tanti altri.

1.5 Gestione del processo

L'ingegneria del software si occupa anche della gestione del progetto, attraverso il **project manager**, che si svolge in parallelo al processo software.

Le principali attività di gestione sono l'**assegnazione** di risorse (umane, finanziarie...), la **stima del tempo** necessario per ogni attività, la **stima dei costi** e la **stima dei rischi**.

1.6 Strumenti CASE

CASE = Computer Aided Software Engineering

Sono tutti i tool informatici che supportano le fasi dell'ingegneria.

- **Specifica** : editor dei requisiti;
- **Progettazione** : editor dei diagrammi di progettazione;
- **Implementazione** : IDE, strumenti per condivisione dei file;
- **Collaudo** : strumenti per ispezione e testing automatico del codice;

- **Manutenzione** : strumenti per la gestione delle versioni del codice e della documentazione.

2 Specifica

La **specifica** è l'insieme di attività necessarie per generare il documento dei requisiti che descrive i *requisiti funzionali* e i *requisiti non funzionali*: descrive il "cosa" il sistema deve fare, non il "come". I requisiti servono per una proposta di contratto e modellare fasi successive del processo software.

Nella specifica non viene considerata l'architettura del sistema (interfaccia utente, database, ...).

2.1 Requisiti funzionali

I requisiti funzionali sono i servizi che il cliente richiede al sistema. Per ogni servizio si descrive:

- cosa accade nell'interazione tra utente e sistema;
- cosa accade in seguito ad un certo input o stimolo;
- cosa accade in particolari situazioni, ad esempio in caso di eccezioni.

Il sistema viene considerato come un'unica entità, non viene descritto come funziona internamente il sistema, in quanto è oggetto della successiva fase di progettazione.

2.2 Requisiti non funzionali

2.2.1 Requisiti di prodotto

Descrivono la qualità del prodotto. Una **proprietà emergente** è una proprietà che "emerge" dal funzionamento del sistema, dopo che è stato implementato.

Esempi:

- **Usabilità** (requisiti che riguardano la facilità d'uso del sistema);
- **Mantenibilità** (facilità di manutenzione del sistema eg. stabilire quali tipi di commenti aggiungere nel codice);
- **Portabilità** (facilità con cui posso far migrare il sistema da una piattaforma ad un'altra);
- **Efficienza** (eg. tempo di risposta del sistema, quantità di memoria occupata);
- **Affidabilità** (grado di fiducia con cui si ritiene che il sistema funzioni correttamente);
 - *Reliability* (il sistema deve funzionare per un certo periodo di tempo senza avere dei malfunzionamenti, eg. 90% di reliability dopo un anno che non abbia problemi, MTTF);
 - *Availability* (probabilità con cui il sistema fornisce il sistema nel momento in cui è richiesto);
 - *Safety* (capacità del sistema di non causare danni alle persone, ambiente di natura economica);
 - *Security* (sicurezza dagli attacchi informatici).
- **Recoverability** (capacità del sistema di tornare disponibile dopo un crash, recuperare dati persi).

2.2.2 Requisiti organizzativi

Caratteristiche riguardanti le fasi del processo software o la gestione del progetto.

- **Requisiti di sviluppo** Sono tutti i vincoli relativi alle tecniche di sviluppo del software.
 - Metodi e tecniche (eg. no ricorsione);
 - Linguaggi;
 - Tool.
- **Requisiti gestionali**
 - Tempo di consegna;
 - Risorse da usare (eg. persone, hardware, licenze);
 - Costi di sviluppo.

2.2.3 Requisiti esterni

Tutti i requisiti che provengono dall'esterno.

- **Requisiti giuridici**
 - Safety;
 - Standard;
 - Privacy;
- **Requisiti di compatibilità** (eg. il sistema si interfaccia con un altro sistema)

2.3 Sistemi critici

Un sistema viene detto critico quando un suo malfunzionamento può causare danni:

- **Safety critical system**
 - Persone;
 - Ambiente;
 - Strutture.
- **Business critical system**
 - Economico.

2.4 Costo dell'affidabilità

Il costo cresce in modo esponenziale rispetto al grado di affidabilità richiesto.

- Tecniche di tolleranze ai guasti (fault-tolerance);
- Tecniche per proteggersi da attacchi.

2.5 Processo di specifica

Il *processo di specifica* è il processo per generare il documento dei requisiti, ed è diviso in più fasi. Lo stesso requisito viene definito con due gradi di dettaglio diversi. Il *requisito utente* è descritto ad alto livello, in linguaggio naturale, ed è il risultato della deduzione dei requisiti. Il *requisito di sistema* è descritto dettagliatamente, fornendo tutti i dettagli necessari per la fase di progettazione, ed è il risultato dell'analisi dei requisiti.

2.5.1 Studio di fattibilità

Lo *studio di fattibilità* è la valutazione della possibilità di sviluppare il sistema e dei suoi vantaggi per il committente. Si decide se la costruzione del sistema è fattibile date le risorse disponibili e se il sistema è effettivamente utile al cliente. Per svolgere lo studio si raccolgono informazioni e si prepara un rapporto di fattibilità, che contiene la valutazione della possibilità di costruire un sistema e dei vantaggi che possono derivare dalla sua introduzione.

2.5.2 Deduzione dei requisiti

La *deduzione dei requisiti* è la raccolta di informazioni da cui dedurre quali sono i requisiti. Le informazioni si possono raccogliere mediante uno studio del dominio applicativo del sistema richiesto, il dialogo con stakeholder, studio di sistemi simili già realizzati e studio di sistemi con cui dovrà interagire quello da sviluppare.

Dominio applicativo è l'insieme di entità reali su cui il sistema software ha effetto.

Stakeholder è, in ambito economico, il soggetto che può influenzare il successo di un'impresa o che ha interessi nelle decisioni dell'impresa; in ambito del processo software sono persone che possono influenzare il processo o che hanno interesse nelle decisioni assunte in esso.

È possibile dialogare con gli stakeholder tramite *interviste*, nelle quali viene chiesto di raccontare attraverso degli esempi reali come l'attività lavorativa funziona realmente, e tramite *etnografia*, l'osservazione dei potenziali utenti nello svolgimento delle loro mansioni.

Il dialogo con gli stakeholder presenta vari problemi, in quanto non sono in grado di indicare chiaramente cosa vogliono dal sistema, omettendo informazioni ritenute ovvie ed utilizzando terminologia non adatta. Inoltre, lo stesso requisito può essere espresso da più stakeholder in maniera differente, ed addirittura essere in conflitto.

Molti problemi scaturiscono dal *linguaggio naturale*, in quanto una descrizione ad alto livello di un requisito può generare confusione. La soluzione è quella di utilizzare il linguaggio in modo coerente, evitando gergo tecnico ed illustrando i requisiti tramite semplici diagrammi.

2.5.3 Analisi dei requisiti

L'*analisi dei requisiti* è l'organizzazione, negoziazione e modellazione dei requisiti.

Comprende:

- **classificazione e organizzazione dei requisiti**
- **assegnazione di priorità ai requisiti:** si stabilisce il grado di rilevanza di ogni requisito
- **negoziazione dei requisiti**
- **modellazione analitica dei requisiti:** produzione di modelli che rappresentano o descrivono nel dettaglio i requisiti

Requisiti di sistema sono l'espansione dei requisiti utente, e formano la base per la progettazione. Il linguaggio naturale non è adatto alla definizione di un requisito di sistema, quindi è necessario usare template, modelli grafici o notazione matematica.

Modello data-flow detto anche pipe & filter, permette di modellare il flusso e l'elaborazione dei dati, ma non prevede la gestione degli errori. L'elaborazione è di tipo batch: input → elaborazione → output.

I requisiti non funzionali si possono specificare definendone delle misure quantitative:

- *efficienza*: tempo di elaborazione delle richieste, occupazione di memoria
- *affidabilità*: probabilità di malfunzionamento, disponibilità
- *usabilità*: tempo di addestramento, aiuto contestuale

Documento dei requisiti contiene il risultato della deduzione e dell'analisi, ed è la dichiarazione ufficiale di ciò che si deve sviluppare. Il documento contiene una breve introduzione che descrive le funzionalità del sistema, un glossario contenente le definizioni di termini tecnici, i requisiti utente e i requisiti di sistema, correlati con modelli UML. Il documento è letto da tutte le figure coinvolte nella realizzazione del progetto.

2.5.4 Validazione dei requisiti

La *validazione dei requisiti* è la verifica del rispetto di alcune proprietà da parte del documento dei requisiti, serve ad evitare la scoperta di *errori di specifica* durante le fasi successive del processo software. Sono da verificare le seguenti proprietà:

- **completezza**: tutti i requisiti richiesti dal committente devono essere documentati;
- **coerenza**: la specifica dei requisiti non deve contenere definizioni tra loro contraddittorie;
- **precisione**: l'interpretazione di una definizione di requisito deve essere unica;
- **realismo**: i requisiti devono essere implementati date le risorse disponibili;
- **tracciabilità**: quando si modifica un requisito bisogna valutarne l'impatto sul resto della specifica: è quindi necessario tracciarlo. Vari tipi:
 - *tracciabilità della sorgente*: reperire la fonte d'informazione relativa al requisito
 - *tracciabilità dei requisiti*: individuare i requisiti dipendenti;
 - *tracciabilità del progetto*: individuare i componenti del sistema che realizzano il requisito;
 - *tracciabilità dei test*: individuare i test-case usati per collaudare il requisito.

Per validare i requisiti si può impiegare un gruppo di *revisori* che ricontrolli i requisiti e *costruire dei prototipi*.

3 Progettazione

3.1 Attività di progettazione

Durante la fase di *progettazione architetturale* viene definita la struttura del sistema, come questo verrà distribuito e come il sistema si dovrà comportare. Sono inoltre progettate le strutture dati, gli algoritmi e la GUI.

3.1.1 Progettazione architetturale

- **Strutturazione del sistema:** definire quali sono i sottosistemi e i moduli interni ad ogni sottosistema;
- **Metodo di controllo:** stabilire quali sono i componenti che invocano le operazioni e quali le eseguono;
- **Modellazione del comportamento:** per ogni servizio stabilire come i componenti interagiscono tra di loro.

Posso utilizzare diagrammi UML.

3.1.2 Progettazione delle strutture dati

Se è presente un database dovrò progettare le strutture dati. (Eg. stabilire le tabelle, chiavi, ...).
Modello entità-relazione, modello relazionale, UML (diagramma delle classi).

3.1.3 Progettazione degli algoritmi

Gli algoritmi da implementare per ogni metodo.
Diagrammi di flusso, UML (diagrammi delle attività).

3.1.4 Progettazione dell'interfaccia utente (grafica)

Se necessaria.
Disegnare su carta/tool per l'aspetto dell'interfaccia grafica.

3.2 Progettazione architetturale

3.2.1 Strutturazione

Stile stratificato Ogni strato (solitamente 3) interagisce con gli strati adiacenti

- **Presentazione:** interfaccia utente;
 - raccolta degli input dell'utente;
 - visualizzazione output all'utente.
- **Elaborazione:** applicazione;
 - elabora dati in input e produce dati in output.
- **Gestione dei dati:** database.
 - aggiunta, modifica, rimozione di dati;
 - ricerca di dati.

Sottosistema Parte del sistema dedicata a svolgere una certa attività.

Modulo Parte di un sottosistema dedicata a svolgere particolari funzioni legate alle attività del sottosistema.

- Presentazione: UI di login, UI del menu, ...;
- Elaborazione: gestore accessi, gestore ricerche, ...;
- Gestione dei dati: DB utenti, DB pagamenti, ...

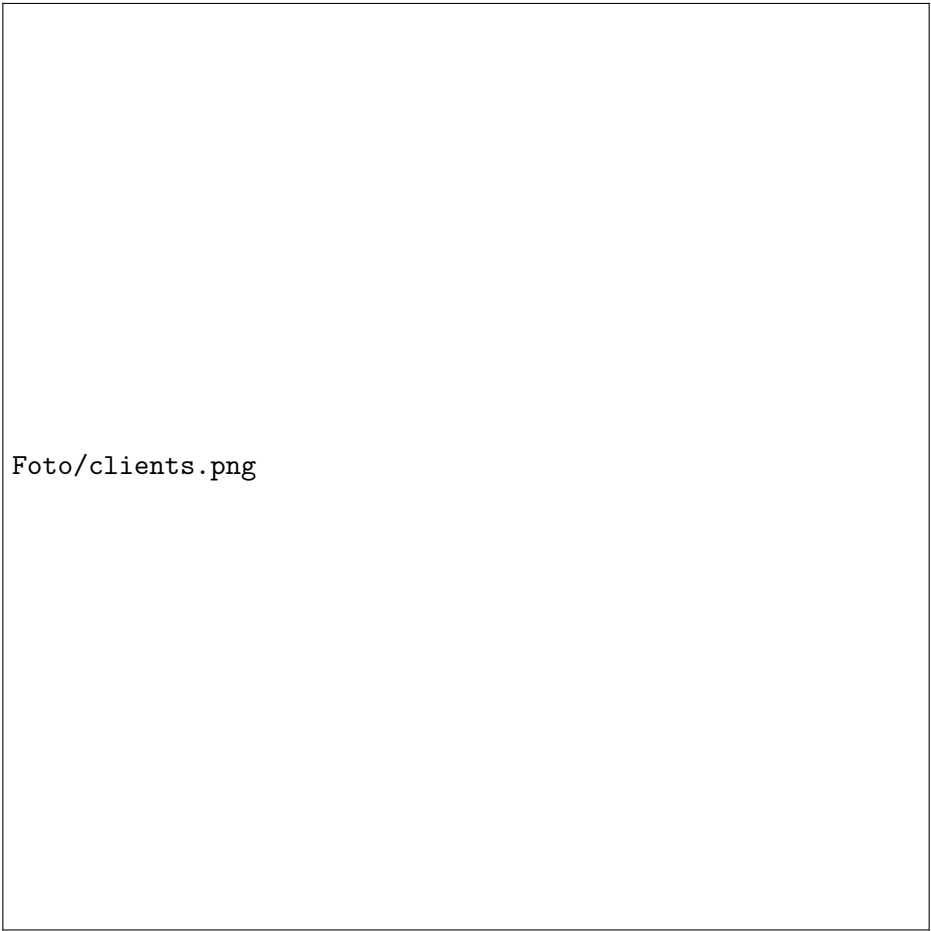
3.2.2 Deployement

Con *deployment* si intende la distribuzione dei componenti in vari dispositivi hardware.

Nel **deployment a 1-tier** i tre stati del sistema sono concentrati su un dispositivo, in quello a **2-tiers** su due e in quello a **3-tiers** su tre.

Deployement 2-tiers

- **Thin client:** il server si occupa dell'elaborazione e della gestione dei dati, mentre il client si occupa della presentazione.
 - carico spostato sulla macchina server;
 - macchina client sprecata dal punto di vista computazionale.
- **Fat client:** il server si occupa della gestione dei dati, mentre il client si occupa della presentazione e dell'elaborazione.
 - minore carico sulla macchina server;
 - il software di elaborazione deve essere aggiornato su tutte le macchine client.



Foto/clients.png

3.2.3 Metodo di controllo

Un componente fornisce servizi ad altri componenti. Un'**interfaccia** è un insieme di operazioni che il componente mette a disposizione di altri componenti ed è condivisa con i componenti che lo invocano.

Un **corpo** è la parte interna del componente e non è conosciuto agli altri componenti. La separazione tra interfaccia (pubblica) e corpo (privato) è detta **information hiding**.

Esistono diversi stili di controllo (attivazione) tra componenti:

- controllo **centralizzato**: è presente un componente detto **controllore**, che controlla l'attivazione e il coordinamento degli altri componenti detti passivi, che eseguono i comandi e restituiscono l'output al controllore;

Il controllore periodicamente esegue un **control-loop**:

- controlla se dei componenti hanno prodotto dati da elaborare;
- attiva o disattiva dei componenti a seconda dei dati raccolti;
- passa i dati ai componenti per l'elaborazione;
- raccoglie i risultati.

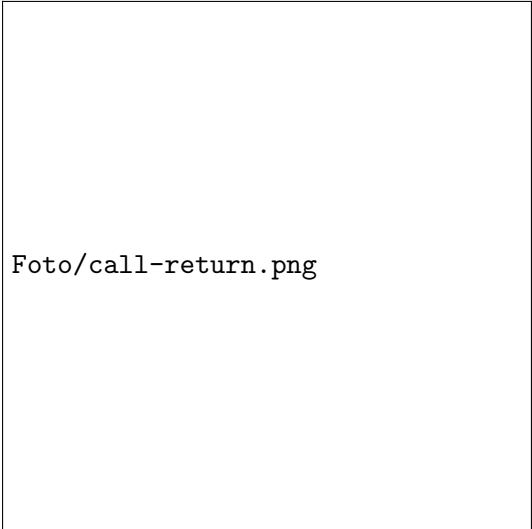
- controllo **basato su eventi**: è basato su eventi esterni (ad esempio un segnale) ed ogni componente si occupa di gestire determinati eventi; il gestore degli eventi è detto **broker**, che rileva l'evento e lo notifica tramite broadcast:

- Broadcast **selettivo**: invia una notifica ai componenti interessati.

Il gestore degli eventi mantiene un registro con gli eventi e i corrispondenti componenti di interesse. I componenti possono comunicare al gestore quali sono gli eventi di loro interesse.

- Broadcast **non selettivo**: invia una notifica a tutti i componenti.

- controllo **call-return**: il controllo passa dall'alto verso il basso (**Top-down**); Ogni operazione ritorna un risultato al livello superiore, al componente che lo aveva invocato.



Foto/call-return.png

- controllo **client-server**: un componente client (controllore) chiede un servizio ad un componente server (passivo) attraverso una chiamata di procedura e il componente server risponde.

3.2.4 Modellazione dei comportamenti ad oggetti

I componenti del sistema sono considerati come oggetti che interagiscono. Un *oggetto* è definito da:

- **attributi**: definiscono lo stato dell'oggetto;
- **operazioni**: operazioni che l'oggetto può compiere.

Gli oggetti comunicano tra di loro attraverso lo scambio di messaggi.