Ingegneria del software

Roberto Pietrantuono

a.a. 2023-2024

Author
Alessio Romano

July 1, 2025

Contents

1 Qualità del software

Le qualità si distinguono in qualità interne (white box), e qualità esterne (black box). Le principali qualità del software sono:

- **Correttezza**: Un software è corretto se soddisfa i requisiti funzionali
- **Affidabilità**: È la probabilità che un software operi come atteso in un intervallo di tempo determinato. **MTBF** (mean time between failure) e **MTTF** (mean time to failure) sono due metriche importanti da misurare
- **Robustezza**: Un software è robusto se si comporta in maniera accettabile anche in situazioni esterne ai requisiti funzionali
- **Prestazioni**: è una qualità esterna basata sui requisiti dell'utente e sull'utilizzo efficace delle risorse
- **Usabilità**: Un sistema è usabile se i suoi utenti lo reputano facile da utilizzare
- **Manutenibilità**: È la facilità con cui è possibile eseguire attività di manutenzione e l'economicità di queste ultime. Si divide in
  - Manutenzione correttiva: volta alla risoluzione di problematiche del software
  - Manutenzione adattiva: riguarda le modifiche dell'applicazione in risposta a cambiamenti dell'ambiente (hardware, os, DBMS)
  - Manutenzione perfettiva: riguarda i cambiamenti nel software per migliorare alcune qualità
- **Interoperabilità**: capacità di coesistere e cooperare con altri sistemi
- **Produttività**: fattore di qualità del processo di produzione
- **Dependability**: composto da Affidabilità, Manutenibilità, Disponibilità, Safety, Integrità
- **Confidenzialità**: assenza di alterazioni improprie

2 Ciclo di vita del software

Il ciclo di vita del software è l'insieme delle fasi e delle attività che guidano la produzione, il rilascio e la manutenzione di un'applicazione software. L'obiettivo è garantire qualità, efficienza e soddisfazione delle esigenze degli utenti. Le attività necessare allo sviluppo corretto del software sono:

- **Acquisizione e specifica dei requisiti**: in questa fase si passa dal software come entità informale ad una serie di specifiche formali che lo modellano. Ciò avviene tramite la produzione di documentazione atta ad estrarre le informazioni dal committente, e alla creazione di un **Documento di specifica dei requisiti**.
- **Progettazione**: La progettazione è l'attività dattraverso la quale i progettisti strutturano l'applicazione a diversi livelli di dettaglio
- **Codifica, Test e Rilascio**: Si implementano le soluzioni individuate, si testano e si mantiene il software

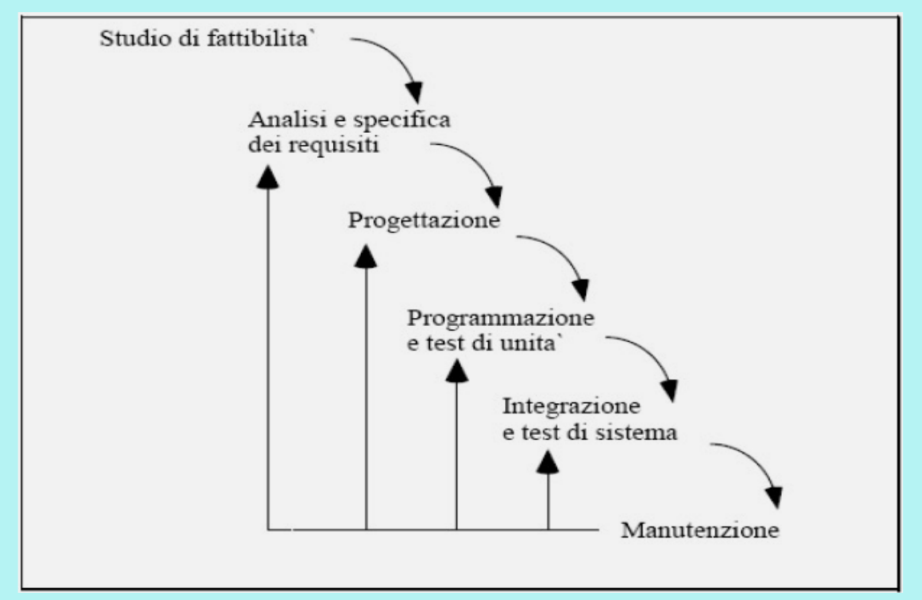
Si identificano diversi modelli di sviluppo

2.1 Code and Fix

Un approccio primitivo alla produzione del software, basato sullo scrivere codice ed aggiustarlo di conseguenza. È causa di molte difficoltà e carenze in quanto il codice diventa rapidamente disorganizzato e difficile da seguire e modificare.

2.2 Waterfall

Nel modello a cascata, il processo struttura le attività come una cascata lineare di fasi in cui l'output di una fase diventa l'input della seguente



La fine di ogni fase si definisce **milestone**, e l'output di ogni fase è detto **deliverable**. I due maggiori contributi apportati dal modello a cascata sono

- Il processo di sviluppo è soggetto a disciplina, pianificazione e gestione
- L'implementazione del prodotto dovrebbe essere rimandata fino a quando non sono chiari gli obiettivi

Il modello a cascata è un modello ideale, in quanto può essere solo approssimato nella realtà. È possibile caratterizzarlo mediante tre proprietà: linearità, rigidità, monoliticità. Il modello si basa sull'assunzione che lo sviluppo è lineare dall'analisi alla produzione di codice, e nella pratica ciò può non succedere a causa di cicli di feedback come alpha e beta testing, oltre che si assume che i requisiti e le specifiche di progetto possano essere congelati nelle prime fasi di sviluppo, cosa che nella pratica non accade. Dunque le problematiche di questo modello sono

- è difficile stimare le risorse in maniera accurata quanso sono disponibili solo informazioni limitate
- la specifica dei requisiti produce un documento scritto immutabile che guida e vincola il prodotto da sviluppare
- l'utente spesso non conosce i requisiti esatti dell'applicazione
- non sottolinea sufficientemente il bisogno di anticipare possibili cambiamenti

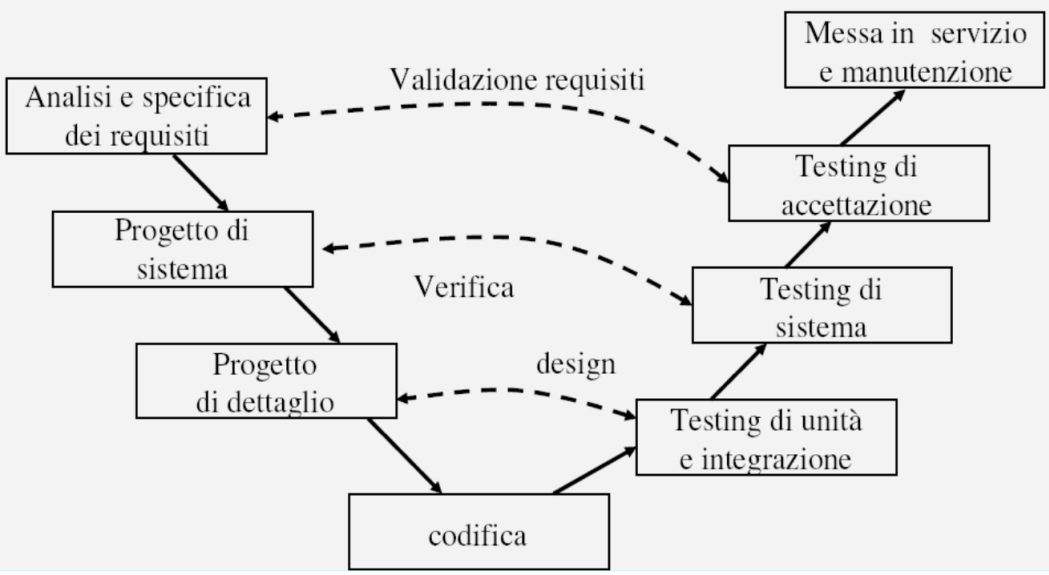
Si tratta di un processo document driven.

2.3 Verification and Validation

Si tratta di una variante del Waterfall con retroazione, ossia dopo ogni attività si verifica l'attività precedente

2.4 Modello a V

Nel modello a V tutte le attività del ramo di sinistro sono collegate con quelle del ramo a destra:



Se si trova un errore in una fase a destra, si riesegue la fase a sinistra collegata

3 Patterns

3.1 Pattern di progettazione

Per pattern di progettazione si intende una soluzione comprovata a un problema tipico che sorge nello sviluppo software in uno specifico contesto. Si dividono in 3 categorie principali

- **Creazionali**: astraggono il processo di creazione di oggetti
- **Strutturali**: trattano la composizione di oggetti e clasi per formare strutture più complesse
- **Comportamentali**: algoritmi di interazioni reciproche tra classi o oggetti e di distribuzione di responsabilità

Un ulteriore suddivisione dei pattern è quella relativa allo **scope**:

- **Classi**: trattano relazioni statiche determinate a compile-time
- **Oggetti**: trattano relazioni dinamiche che variano a run-time

3.1.1 Design patterns creazionali

Nei design pattern creazionali con scope le classi, la creazione di oggetti è affidata a sottoclassi basandosi sul principio dell'ereditarietà, mentre nei design pattern creazionali con scope gli oggetti, questa responsabilità si affida ad altri oggetti. Ciò permette al sistema di essere indipendente da come sono creati, rappresentati e gestiti gli oggetti. Fanno parte di questa categoria:

- **Abstract Factory**: un oggetto che serve a creare istanze di altri oggetti
- **Factory method**: un oggetto che serve a creare istanze di diverse classi derivate
- **Prototype**: istanza completa di un oggetto che serve per essere clonato
- **Singleton**: un oggetto che restituisce una singola istanza di se stesso
- **Builder**: separa costruzione e rappresentazione per oggetti complessi