Analisi e Progettazione del Software

spitfire

A.A. 2023-2024

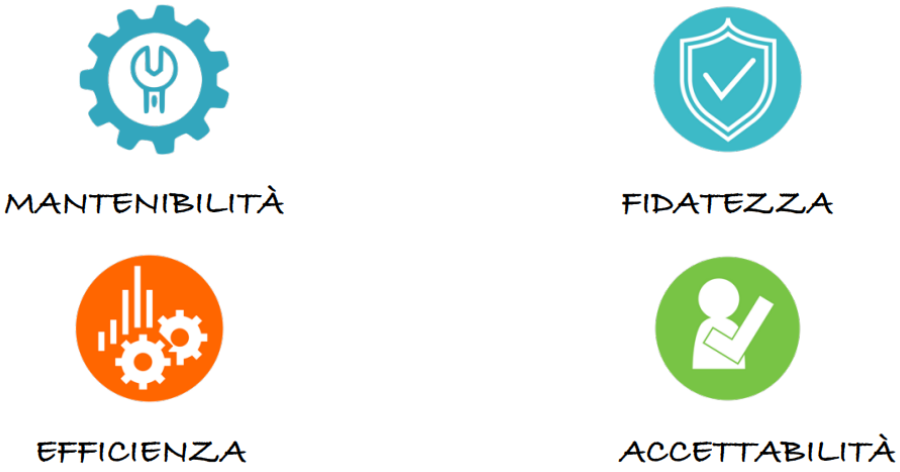
Contents

1 Introduzione

Che cos'è il **software**? Esso è **un programma per computer** unito alla **documentazione ad esso associata**, la quale specifica e comprende **requisiti, modelli di progetto, manuale utente,...**
I prodotti software possono essere:

- **Generici**: sviluppati per un ampio insieme di clienti (elaboratori di testo, database,...)
- **Personalizzati** (custom): sviluppati per un singolo cliente in base alla sue esigenze specifiche

Un nuovo prodotto software può essere **creato da zero, personalizzando software già esistenti o riusando parti o software già esistente**. Le caratteristiche essenziali di un buon software sono:

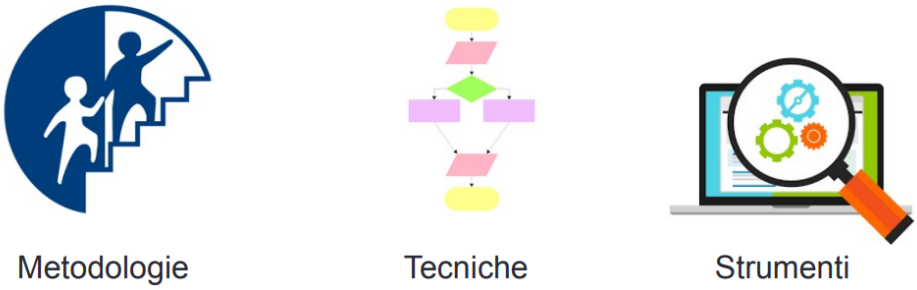


1.1 Introduzione all'ingegneria del software

Che cos'è **l'ingegneria del software**? **L'ingegneria del software** è una disciplina ingegneristica che si occupa di tutti gli aspetti della produzione del software di buona qualità, dalle **prime fasi della specifica del sistema fino alla manutenzione del sistema** dopo la messa in uso. Vediamo cosa si intende per **disciplina ingegneristica** e "**Tutti gli aspetti della produzione del software**":

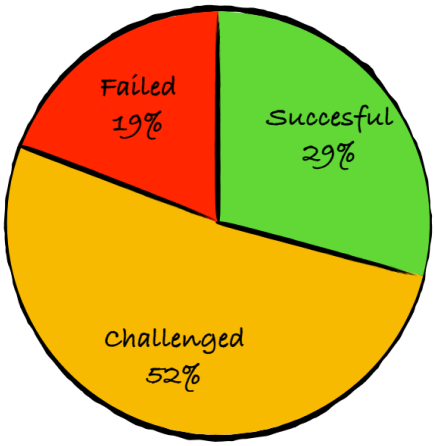
- **Disciplina ingegneristica**: Utilizzare metodi e teorie **appropriati** per risolvere i problemi tenendo conto dei vincoli **organizzativi e finanziari**
- **Tutti gli aspetti della produzione del software**: Non solo il **processo tecnico di sviluppo**. Anche la **gestione del progetto** e lo sviluppo di **strumenti** ,metodi ecc... per supportare la produzione del software

La disciplina dell'ingegneria del software si occupa di:

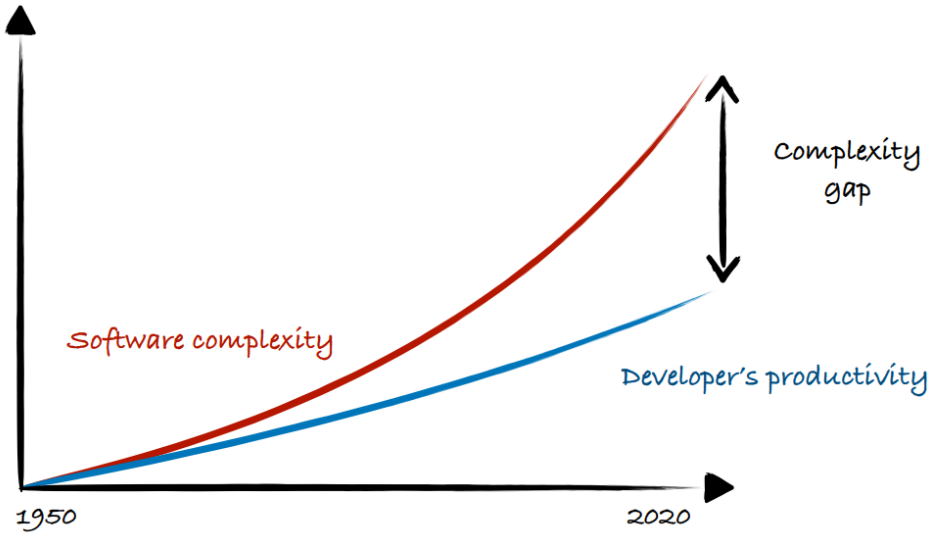


1.2 La crisi del software

Il termine **crisi del software** (o software crisis) è usato nell'ambito dell'ingegneria del software per descrivere l'impatto della **rapida crescita** della potenza degli elaboratori e la **complessità** dei problemi che dovevano esseri affrontati. Le parole chiavi della software crisis erano **complessità, attese e cambiamento**. Il concetto di software crisis emerse negli anni '60.



Standish Group 2015 Chaos Report



Le cause della crisi del software erano legate alla **complessità dei processi software** e alla **relativa immaturità dell'ingegneria del software** Per superare la crisi infatti si dovettero introdurre:

- **Management**
- **Organizzazione**, attraverso **analisi e progettazione**
- **Teorie e tecniche** come la **programmazione strutturata e ad oggetti**
- **Strumenti**, come gli IDE
- **Metodologie**, tra cui il **modello a cascata e il modello agile**

1.3 Analisi e progettazione

Che cosa sono **analisi e progettazione**?
L'analisi enfatizza un'**investigazione del problema e dei requisiti** invece che una soluzione: per esempio, se si vuole realizzare un nuovo sistema di trading online, bisognerà capire **come questo sistema verrà utilizzato** e **quali sono le sue funzioni**.
"Analisi" è un termine ampio con più accezioni, tra cui:

- **Analisi dei requisiti**, cioè un'investigazione dei requisiti del sistema
- **Analisi orientata agli oggetti**, cioè un'investigazione degli oggetti di dominio

La **progettazione** enfatizza una soluzione **concettuale** (software e hardware) che **soddisfa i requisiti**, anziché la relativa implementazione. Per esempio, la descrizione di uno schema di base di dati e di oggetti software. Nella progettazione vengono spesso **esclusi dettagli di basso livello o "ovvi"** (o almeno "ovvi" per coloro a cui è destinato il software).
Infine i progetti possono essere **implementati** e la loro implementazione (ovvero il codice) esprime il progetto realizzato vero e completo. Come nel caso dell'analisi, anche "progettazione" è un termine con più accezioni, tra cui:

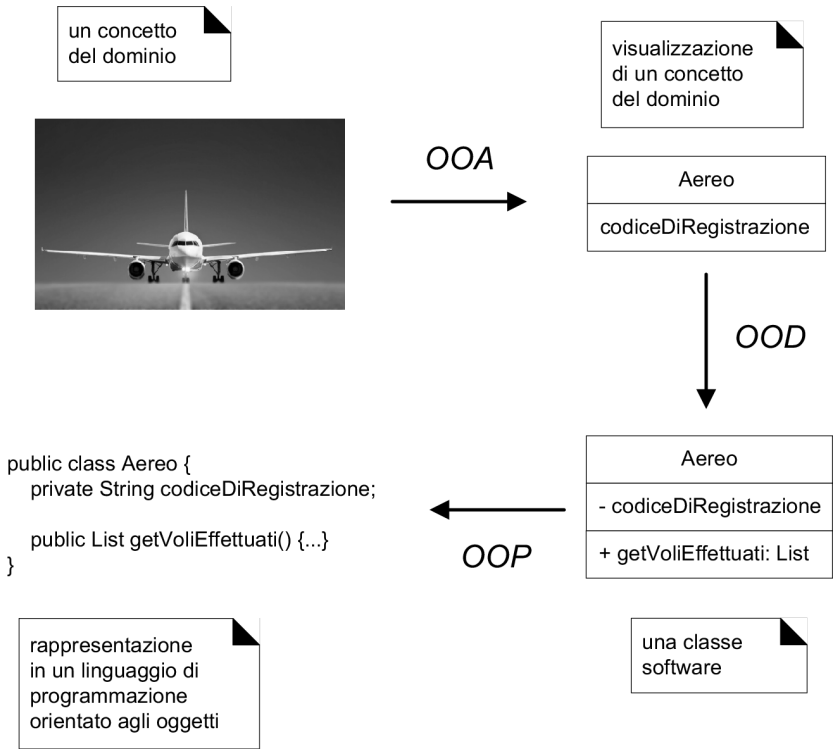
- **Progettazione orientata agli oggetti**
- **Progettazione di basi di dati**

L'analisi e la progettazione possono essere riassunti con la seguente frase:

Fare la cosa giusta(analisi) **e fare la cosa bene**(progettazione)

1.3.1 Analisi e progettazione orientata agli oggetti

Durante **l'analisi orientata agli oggetti** c'è un enfasi sull'**identificazione** e la **descrizione degli oggetti**, o dei **concetti**, nel **dominio del problema**. Per esempio, nel caso di un sistema informatico per voli aerei, alcuni dei concetti possono essere *Aereo, Volo* e *Pilota*.
Durante **la progettazione orientata agli oggetti** (o più semplicemente **progettazione a oggetti**) l'enfasi è sulla **definizione di oggetti software** e sul **modo in cui questi collaborano per soddisfare i requisiti**. Per esempio un oggetto software *Aereo* può avere un attributo *codiceDiRegistrazione* e un metodo *getVoliEffettuati*.



Infine durante **l'implementazione** o la **programmazione orientata agli oggetti**, gli oggetti progettati vengono implementati, per esempio implementando la classe *Aereo* in un linguaggio ad oggetti. Dunque, analisi e progettazione **hanno obbiettivi diversi che vengono perseguiti in maniera diversa**. Tuttavia, come mostrato dall'esempio sopra, esse sono **attività fortemente sinergiche** che sono **correlate fra loro** e con le **altre attività dello sviluppo del software**.

1.4 Introduzione ai diagrammi e ai passi fondamentali dello sviluppo software

Vediamo una breve introduzione dei **vai diagrammi e dei passi fondamentali** legati allo sviluppo software.

1.4.1 Definizione dei casi d'uso

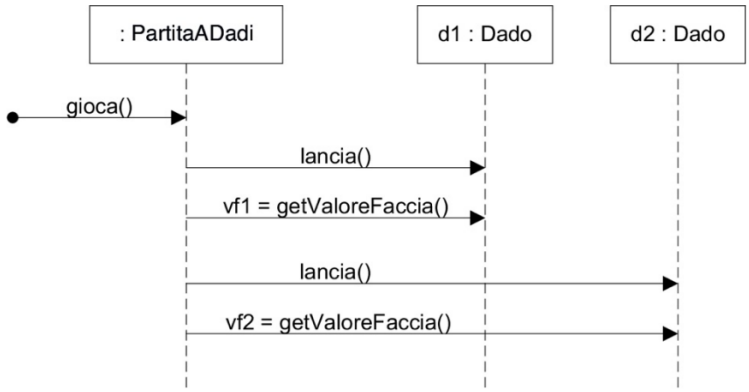
L'analisi dei requisiti può comprendere **storie o scenari** relativi al modo in cui l'applicazione può essere utilizzata dagli utenti; queste storie possono essere scritte come **casi d'uso**. I casi d'uso **non sono un elaborato ad oggetti** ma semplicemente delle storie scritte. Sono tuttavia uno strumento **diffuso nell'analisi dei requisiti**. Facciamo un'esempio:
Gioca una partita a Dadi: Il giocatore chiede di lanciare i dadi. Il Sistema presenta il risultato: se il valore totale delle facce dei dadi è sette, il giocatore ha vinto; altrimenti ha perso.

1.4.2 Definizione di un modello di dominio

L'analisi orientata agli oggetti è interessata alla **creazione di una descrizione del dominio da un punto di vista ad oggetti**. Vengono identificati i **concetti, gli attributi e le associazioni considerati significativi**. Il risultato può essere espresso come un **modello di dominio** che mostra i concetti o gli oggetti **significativi** del dominio. Esso è rappresentato nel seguente modo:

1.4.3 Definizione dei diagrammi di interazione

La **progettazione ad oggetti** è interessata alla **definizione di oggetti software, delle loro responsabilità e collaborazioni**. Una notazione comune per illustrare queste collaborazione è un **diagramma di sequenza** (un tipo di diagramma UML). Esso mostra lo scambio di messaggi **tra oggetti software**, dunque l'invocazione di **metodi**. Esso è rappresentato nel seguente modo:

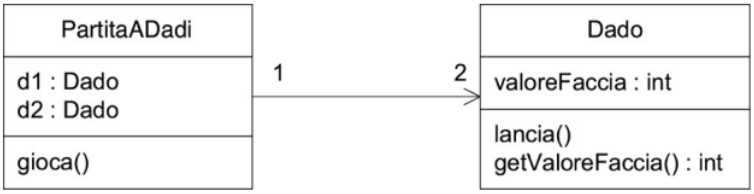


È interessante notare come **la progettazione degli oggetti software e dei programmi si può ispirare a un dominio del mondo reale**, tuttavia essa non è **nè un modello diretto nè una simulazione di questo dominio**. Quindi, per esempio, seppur nel mondo reale è il giocatore a lanciare il dado, nel progetto software è l'oggetto *PartitaADadi* che "lancia" i dadi.

1.4.4 Definizione dei diagrammi di classe di progetto

Accanto a una visione dinamica delle **collaborazioni tra oggetti**, mostrata dai diagrammi di interazione, è utile mostrare una **vista statica** delle definizioni di classi mediante un

diagramma delle classi di progetto, che mostra le classi software con i loro attributi e metodi. Il diagramma delle classi di progetto è rappresentato nella seguente maniera:



Diversamente dal modello di dominio, che illustra **classi del mondo reale**, questo diagramma mostra **classi software**. Si noti che, benché questo diagramma delle classi di progetto **non sia uguale al modello di dominio**, i nomi e il contenuto delle classi sono **simili**. In tal modo i **progetti e i linguaggi Object Oriented** (OO) sono in grado di **favorire un salto rappresentazionale basso** tra i componenti software e il nostro modello mentale di un dominio, **migliorando la comprensione**.

1.5 UML

Unified Modelling Language, abbreviato **UML**, è un **linguaggio visuale** per la **specifica, la costruzione e la documentazione degli elaborati** di un sistema software. UML rappresenta una **collezione di best practices di ingegneria**, dimostratesi vincenti nella modellazione di sistemi vasti e complessi; inoltre esso **favorisce la divulgazione delle informazioni nella comunità dell'ingegneria del software** in quanto è *standard de facto*. Bisogna però tenere a mente che **UML non è una metodologia ma un linguaggio!**
Il termine *visuale* della definizione è un punto fondamentale. UML è uno standard de facto per la **notazione di diagrammi per disegnare o rappresentare figure** (con del testo) **relative al software**, e in particolare, al software OO. A un livello più profondo, di particolare interesse per i produttori di strumenti per **MDA** (Model Driven Architecture) alla base della notazione UML c'è il **meta-modello di UML** che descrive la **semantica** degli elementi di modellazione, tuttavia non è necessario che lo sviluppatore lo conosca.