💚

APS - Introduzione

Created	@April 16, 2024
Tags	APS

1.1 Introduzione

> 💡 ***Cos'è un software?***
>
> *Un programma per computer e la documentazione ad esso associata.*

I prodotti software possono essere:

- **Generici** – sviluppati per un ampio insieme di clienti
- **Personalizzati** (custom) – sviluppati per un singolo cliente in base alle sue esigenze

Un buon software deve avere determinate caratteristiche:

- Mantenibilità
- Fidatezza
- Efficienza
- Accettabilità

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 sua messa in uso. Utilizza delle teorie e metodi appropriati per risolvere i problemi tenendo conto dei vincoli organizzativi e finanziari.

La realizzazione di un prodotto software è costituita dai **processi software.**

> 💡 ***Cos'è un processo software?***
>
> *Un processo software è un insieme di attività, metodi, pratiche e strumenti utilizzati per sviluppare e mantenere sistemi software. Esso definisce il "come" di un progetto software, ovvero come il software verrà sviluppato, gestito, distribuito e mantenuto nel tempo. I processi software forniscono una struttura che aiuta i team di sviluppo a organizzare e coordinare le loro attività per raggiungere gli obiettivi del progetto.*

Un processo software è costituito da una serie di componenti:

1. Analisi dei Requisiti
2. Progettazione
3. Implementazione
4. Evoluzione
5. Validazione

I processi software vengono tipicamente descritti attraverso quattro elementi principali:

1. **Artefatti** - prodotti tangibili generati durante il processo di sviluppo del software. Questi possono essere documenti, codice, modelli, ecc.
2. **Attività** - azioni intraprese dai membri del team di sviluppo per produrre gli artefatti. Queste attività costituiscono il "lavoro" del processo di sviluppo software.
3. **Ruoli** - responsabilità e i compiti dei membri del team all'interno del processo di sviluppo software.
4. **Pre e post condizioni** - definiscono lo stato del processo prima e dopo l'esecuzione di una specifica attività. Queste condizioni aiutano a garantire che le attività siano eseguite in modo corretto e che gli artefatti generati siano conformi agli standard previsti.

In questo corso si affronteranno principalmente la parte di **Analisi** e **Progettazione**.

1.2 Che cosa sono analisi e progettazione

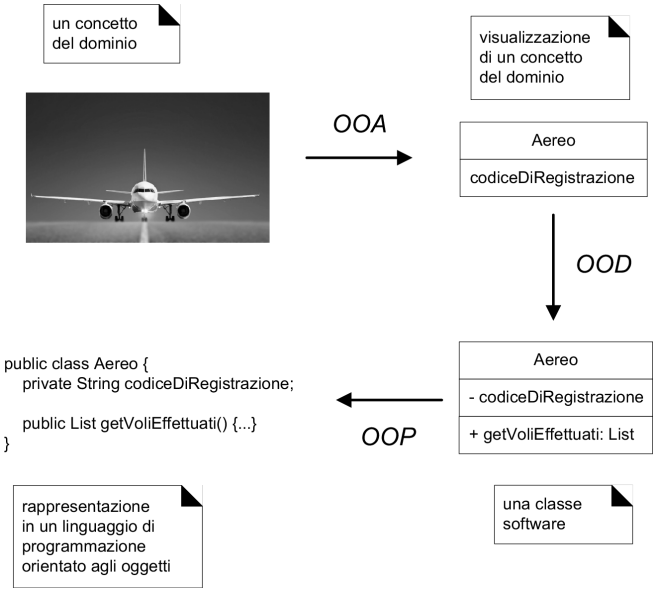
- L'**analisi** enfatizza un'*investigazione* del problema e dei requisiti, anziché di una soluzione.
  "Analisi" è un termine ampio, con più accezioni, come
  analisi dei requisiti (un'investigazione dei requisiti) o *analisi orientata agli oggetti* (un'investigazione degli oggetti di dominio).

- La **progettazione** enfatizza una *soluzione concettuale* (software e hardware) che soddisfa i requisiti, anziché la relativa implementazione.
  Come nel caso dell'analisi, questo termine ha più accezioni, come
  progettazione orientata agli oggetti o *progettazione di basi di dati*.

L'analisi e la progettazione possono essere riassunte nell'espressione *fare la cosa giusta (analisi) e fare la cosa bene (progettazione)*.

1.3 Che cosa sono analisi e progettazione orientata agli oggetti

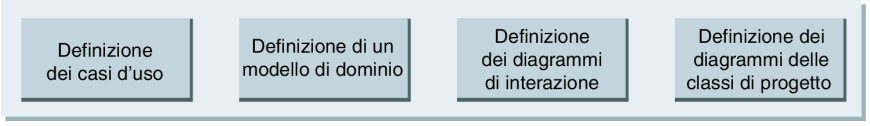
- Durante l'**analisi orientata agli oggetti** c'è un'enfasi sull'identificazione e la descrizione degli oggetti, o dei concetti, nel dominio del problema.
- Durante la **progettazione orientata agli oggetti** (o semplicemente progettazione a oggetti) l'enfasi è sulla definizione di oggetti software e del modo in cui questi collaborano per soddisfare i requisiti.



Dunque, analisi e progettazione hanno obiettivi diversi, che vengono perseguiti in modi diversi. Tuttavia, come mostrato da questo piccolo esempio, analisi e progettazione sono attività fortemente sinergiche, che sono correlate fra loro e con le altre attività dello sviluppo del software.

1.4 Le fasi dell'analisi e progettazione del software

Questo paragrafo presenta una visione complessiva, a volo d'uccello, di alcuni passi e diagrammi fondamentali delle fasi di analisi e progettazione.



1.4.1 Definizione dei casi d'uso

L'analisi dei requisiti può comprendere storie o scenari relativi al modo in cui l'applicazione viene utilizzata dalle persone; queste storie possono essere scritte come casi d'uso.

I casi d'uso non sono un elaborato orientato agli oggetti, ma semplicemente delle storie scritte. Sono tuttavia uno strumento diffuso nell'analisi dei requisiti.

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** *(figura 1.1)* che mostra i concetti o gli oggetti *significativi* del dominio.

Figura 1.1: Modello di dominio parziale del gioco dei dadi.

Si noti che un modello di dominio non è una descrizione di oggetti software, ma una visualizzazione dei concetti o del modello mentale di un dominio del mondo reale. Pertanto è anche chiamato **modello concettuale a oggetti**.

1.4.3 Assegnare responsabilità agli oggetti e disegnare diagrammi di interazione

La progettazione orientata agli oggetti è interessata alla definizione di oggetti software, delle loro responsabilità e collaborazioni. Una notazione comune per illustrare queste collaborazioni è un **diagramma di sequenza** *(figura 1.2)* (un tipo di diagramma di interazione di UML). Esso mostra lo scambio di messaggi tra oggetti software, dunque l'invocazione di metodi.

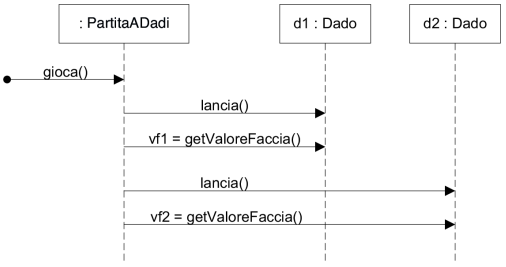


Figura 1.2: Diagramma di sequenza che illustra lo scambio di messaggi tra oggetti software.

La progettazione degli oggetti software e dei programmi si può ispirare a un dominio del mondo reale, ma non è né un modello diretto né una simulazione di questo dominio.

1.4.4 Definizione dei diagrammi delle classi di progetto

Accanto a una vista 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** *(figura 1.3)*, che mostra le classi software con i loro attributi e metodi.

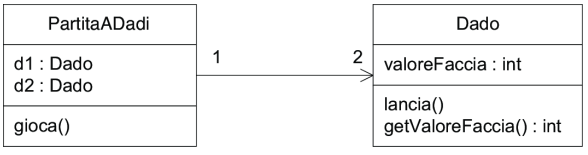


Figura 1.3: Diagramma parziale delle classi di progetto.

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 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 Che cos'è UML

Che cos'è UML?

Unified Modeling Language è un linguaggio visuale per la specifica, la costruzione e la documentazione degli elaborati di un sistema software.

Unified Modelling Language (UML) è un **linguaggio visuale** di modellazione dei sistemi software e non solo:

- Rappresenta una collezione di best practices di ingegneria, dimostratesi vincenti nella modellazione di sistemi vasti e complessi
- Favorisce la divulgazione delle informazioni nella comunità dell'ingegneria del software in quanto è standard "de facto"

UML modella i sistemi come insiemi di oggetti che collaborano tra loro

- **Struttura Statica** (*Quali tipi di oggetti sono necessari? Come sono tra loro correlati?*)
- **Struttura Dinamica** (*Com'è il ciclo di vita di questi oggetti? Come collaborano per fornire le funzionalità richieste?*)

Esistono tre modi per applicare UML:

1. **UML come abbozzo** - diagrammi informali e incompleti
2. **UML come progetto** - diagrammi di progetto relativamente dettagliati
3. **UML come linguaggio di programmazione** - La specifica completamente eseguibile di un sistema software con UML

In questo corso useremo UML come abbozzo.

UML descrive dei tipi "grezzi" di diagrammi, come i diagrammi delle classi e i diagrammi di sequenza; tuttavia, UML non impone un particolare punto di vista di modellazione per l'uso di questi diagrammi. Per esempio, la stessa notazione dei diagrammi delle classi di UML può essere utilizzata per disegnare figure di concetti del mondo reale o di classi software Java.
Una stessa notazione può essere utilizzata secondo due punti di vista (o prospettive) e tipi di modelli (*Figura 1.4*).

- **Punto di vista concettuale**. I diagrammi sono scritti e interpretati come descrizioni di oggetti del mondo reale o nel dominio di interesse.
- **Punto di vista software**. I diagrammi (che utilizzano la stessa notazione del punto di vista concettuale) descrivono astrazioni o componenti software.

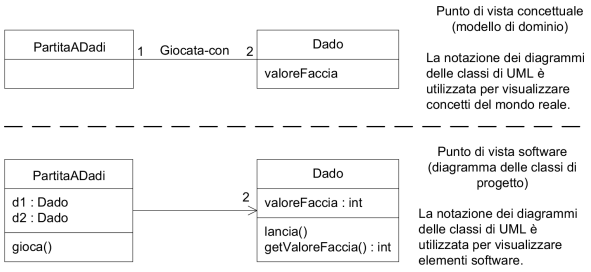


Figura 1.4 Diversi punti di vista con UML.