UML: Unified Modeling Language

Gianluca Pironato

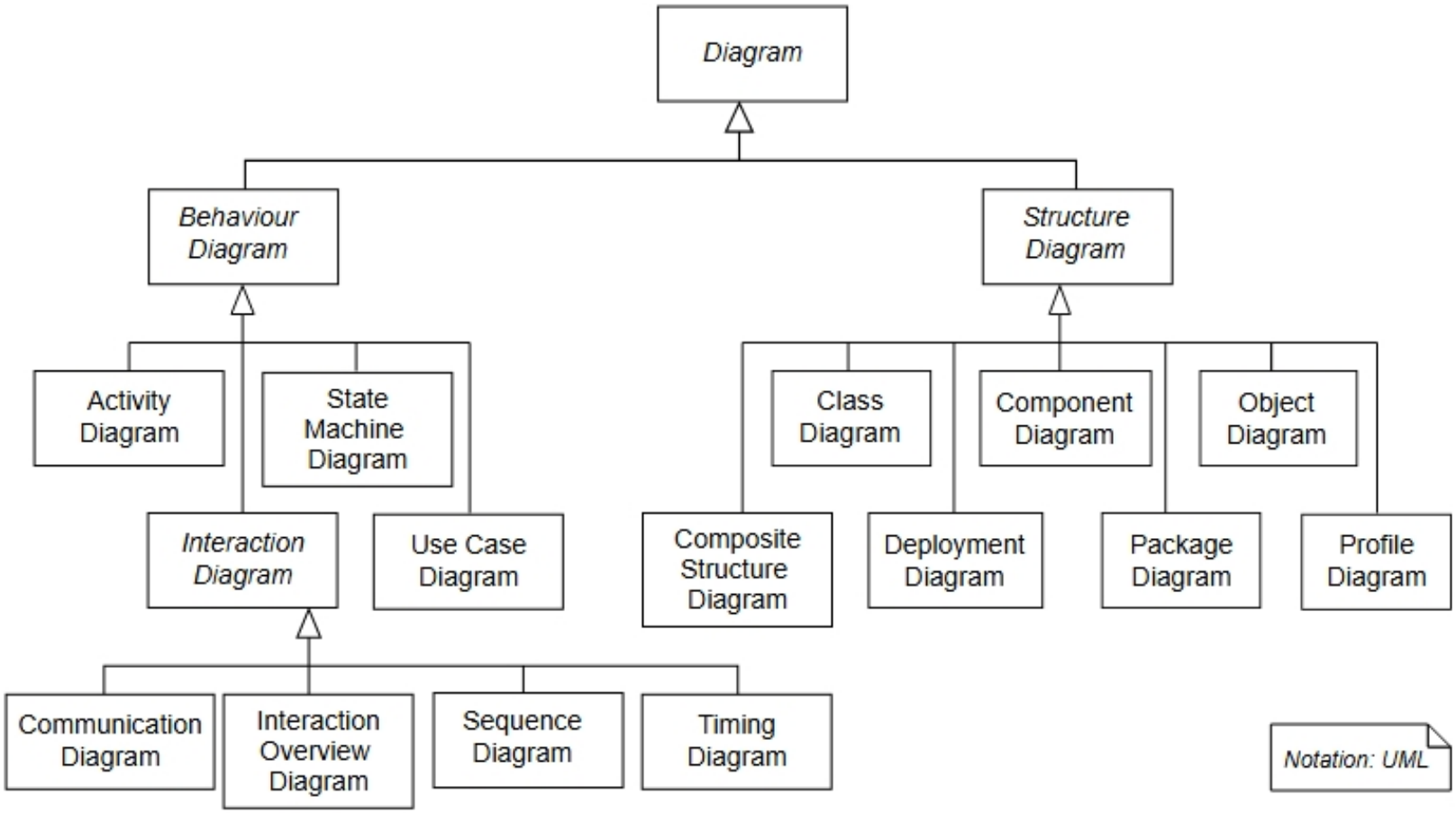
Introduzione a UML

- UML è un linguaggio di modellazione visuale che aiuta a capire, descrivere e documentare le caratteristiche strutturali, funzionali e dinamiche di sistemi software complessi.
- Il suo sviluppo iniziò a metà degli anni 90 grazie a Grady Booch, James Rumbaugh e Ivar Jacobson per unificare diversi approcci alla progettazione di **software object oriented**.
- Lo standard UML, nella sua versione 2.0, venne pubblicato nel 2005 nella **ISO/IEC 19501** ed è gestito dal consorzio OMG.
- Doveroso è specificare che il suo "scope" non si limita alla descrizione di sistemi software, bensì viene proposto come linguaggio per modellare qualsiasi *"business and similar processes"* [pag. 43].

UML per lo sviluppo software

- UML è un linguaggio di modellazione utile per:
  - la progettazione del software;
  - la verifica dello stato dei lavori di sviluppo;
  - la verifica del funzionamento del software;
  - la documentazione del software.
- I formalismi grafici di UML seguono una notazione semplice e priva di ambiguità.
- UML definisce, per diverse fasi del processo di sviluppo (analisi, progetto, codifica), tredici diagrammi suddivisi in due categorie principali:
  - **Structure diagrams**: forniscono le viste Logical, Development e Physical (tutto ciò che è **statico**)
  - **Behavior diagrams**: forniscono le viste Process e Use case (ovvero, la parte **dinamica**)

UML per lo sviluppo software



Le viste di UML

- È quindi possibile concentrarsi su alcuni aspetti precisi del ciclo di vita SW:
  - **Aspetti strutturali**: si concentrano sui moduli che compongono l'architettura SW; in particolare, noi faremo attenzione alle strutture dati che riflettono **classi e relazioni** in gioco;
  - **Aspetti funzionali**: riportano le necessità del cliente e si concentrano su ciò che il software deve fare, ovvero sui **casi d'uso**;
  - **Aspetti comportamentali**: rappresentano in dettaglio quanto avviene nel sistema per un caso d'uso specifico, evidenziando le **interazioni tra gli attori in sequenza temporale**.

Quali diagrammi UML

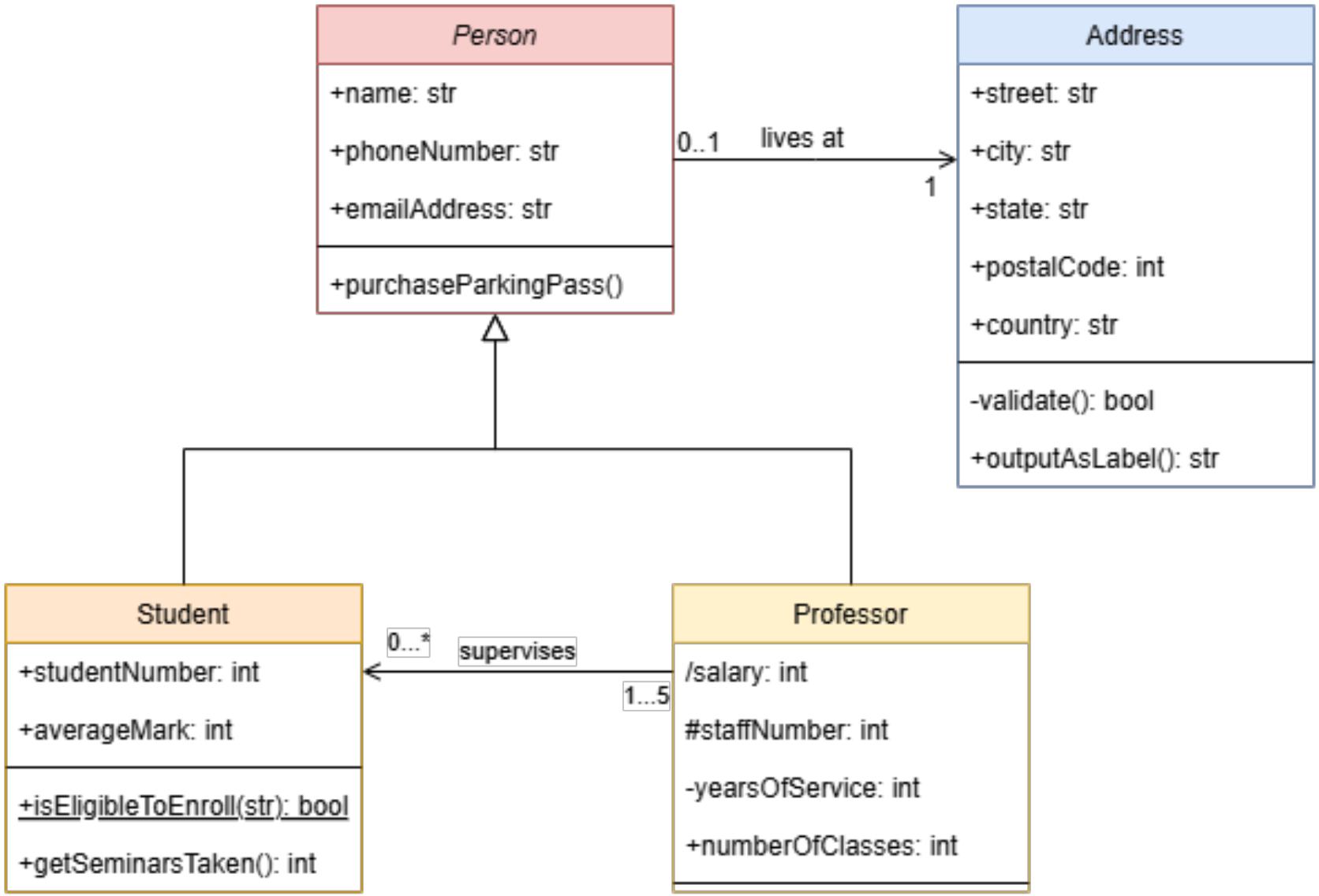
Per analizzare gli aspetti presentati nella slide precedente, utilizzeremo i seguenti:

- **Class diagram**: descrive la struttura dei dati e le relazioni tra le classi (esempio, fino a 21:52).
- **Use-case diagram**: rappresenta le funzionalità e le interazioni con gli utenti o altri sistemi (esempio, fino a 09:40).
- **Sequence diagram**: illustra la sequenza di eventi o messaggi tra gli oggetti durante l'esecuzione (esempio, fino a 19:31).

Panoramica - Class diagram

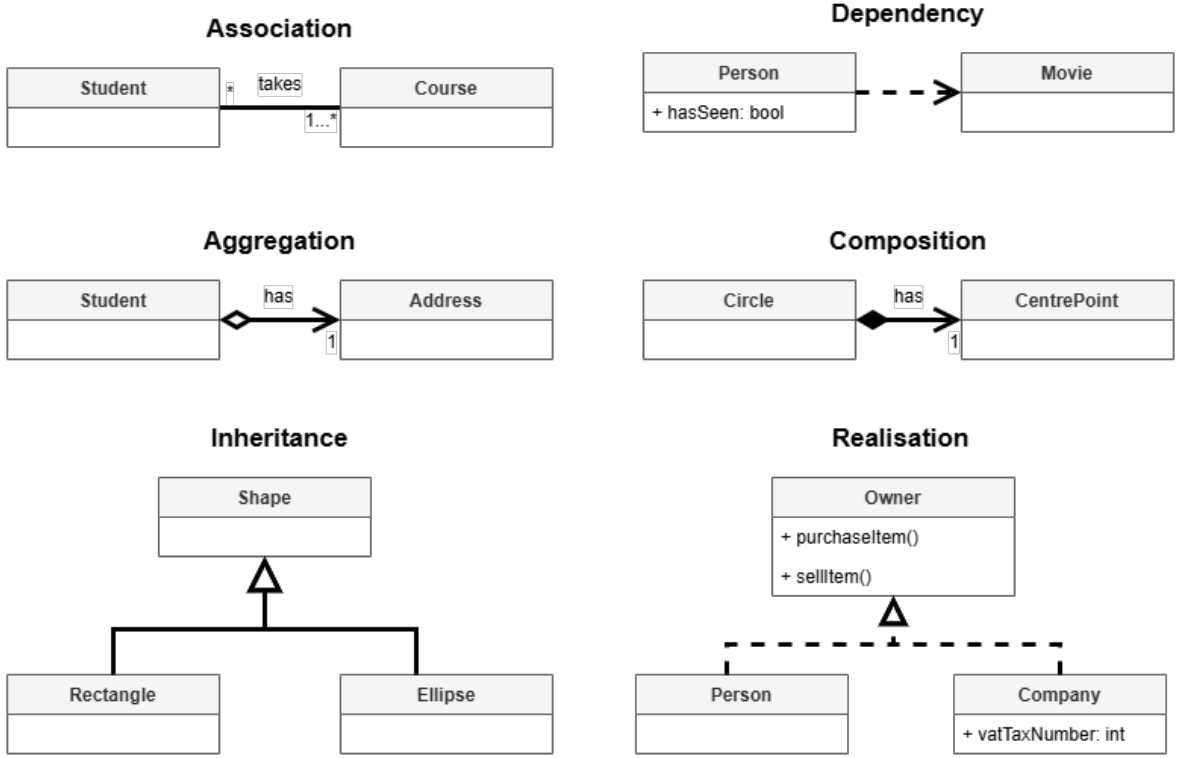
- Ogni **classe** è composta da tre parti:
  1. nome;
  2. attributi;
  3. metodi.
- Graficamente si presenta come un rettangolo suddiviso in tre sezioni: nome in alto, attributi al centro e metodi in basso.
- Per i metodi è possibile specificare:
  - Il **ruolo** (metodi di input `(in)` o output `(out)`)
  - Il **tipo del parametro restituito** (indicato dopo il nome del metodo)
- Davanti ad attributi e metodi è indicata la **visibilità**:
  - \- (private)
  - \+ (public)
  - \# (protected)
- **N.B. esiste in UML un Object diagram pensato per rappresentare un'istanza specifica di una certa classe (solo nome e attributi con valori)**

Panoramica - Class diagram



Panoramica - Class diagram

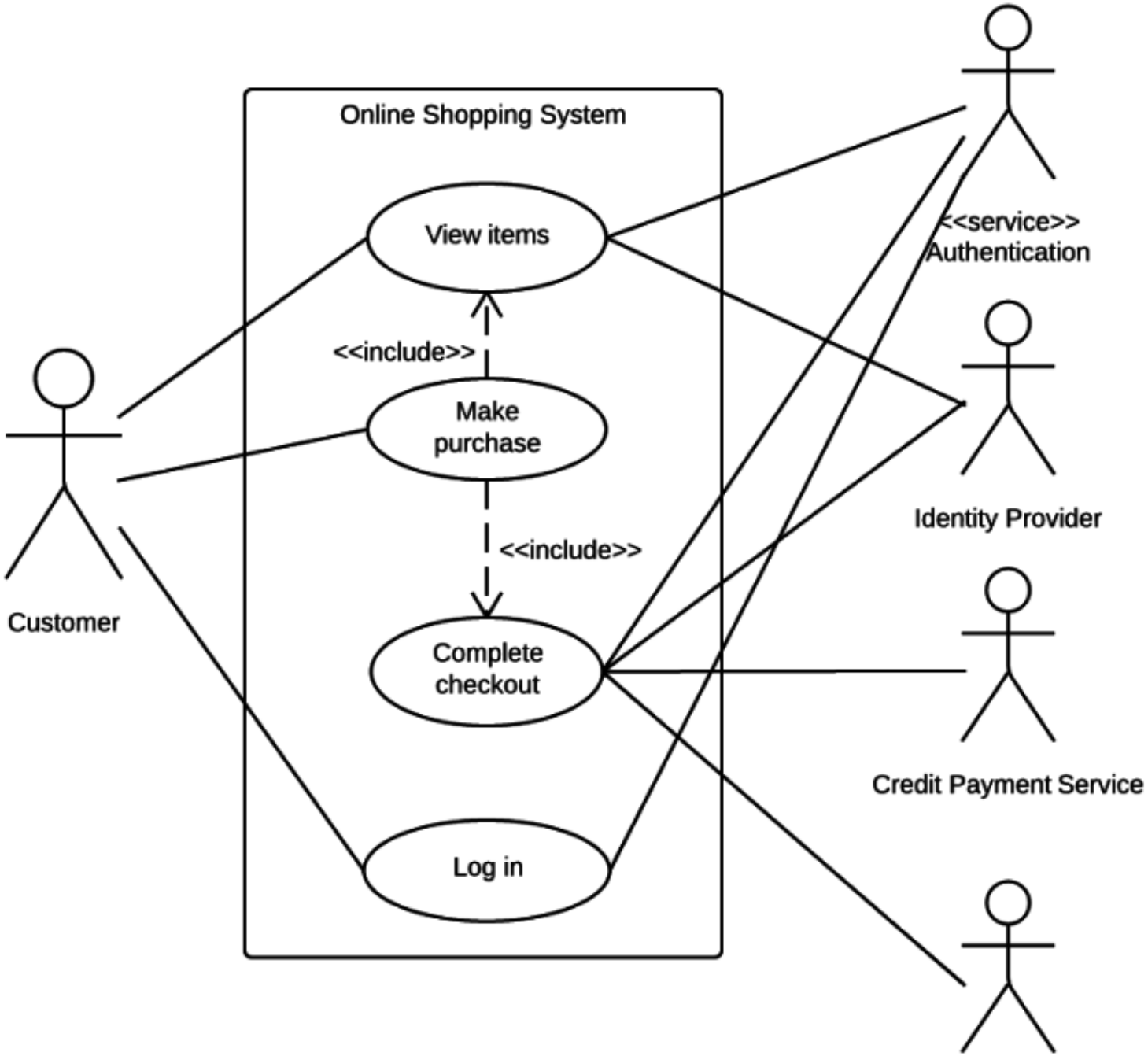
- In questi diagrammi si possono rappresentare anche le **relazioni tra classi**: associazione, aggregazione, composizione, dipendenza, ereditarietà, implementazione, ecc.
- Ciascuna con notazione grafica diversa.



Panoramica - Use-case diagram

- Illustra in modo semplice e immediato le funzionalità che il sistema deve offrire per soddisfare le esigenze del cliente.
- L'approccio è di tipo top-down: si parte dall'identificazione generale degli attori e delle funzionalità principali, per poi procedere verso il dettaglio dei casi d'uso specifici.
- I casi d'uso vengono rappresentati con ovali, mentre gli attori con figure stilizzate poste all'esterno del rettangolo corrispondente al sistema.
- Anche i casi d'uso possono essere messi in relazione:
  - **extends**: se un caso d'uso rappresenta un'estensione di un altro caso d'uso generata da un particolare evento;
  - **includes**: se un caso d'uso racchiude in sè un comportamento specificato in un altro caso d'uso.

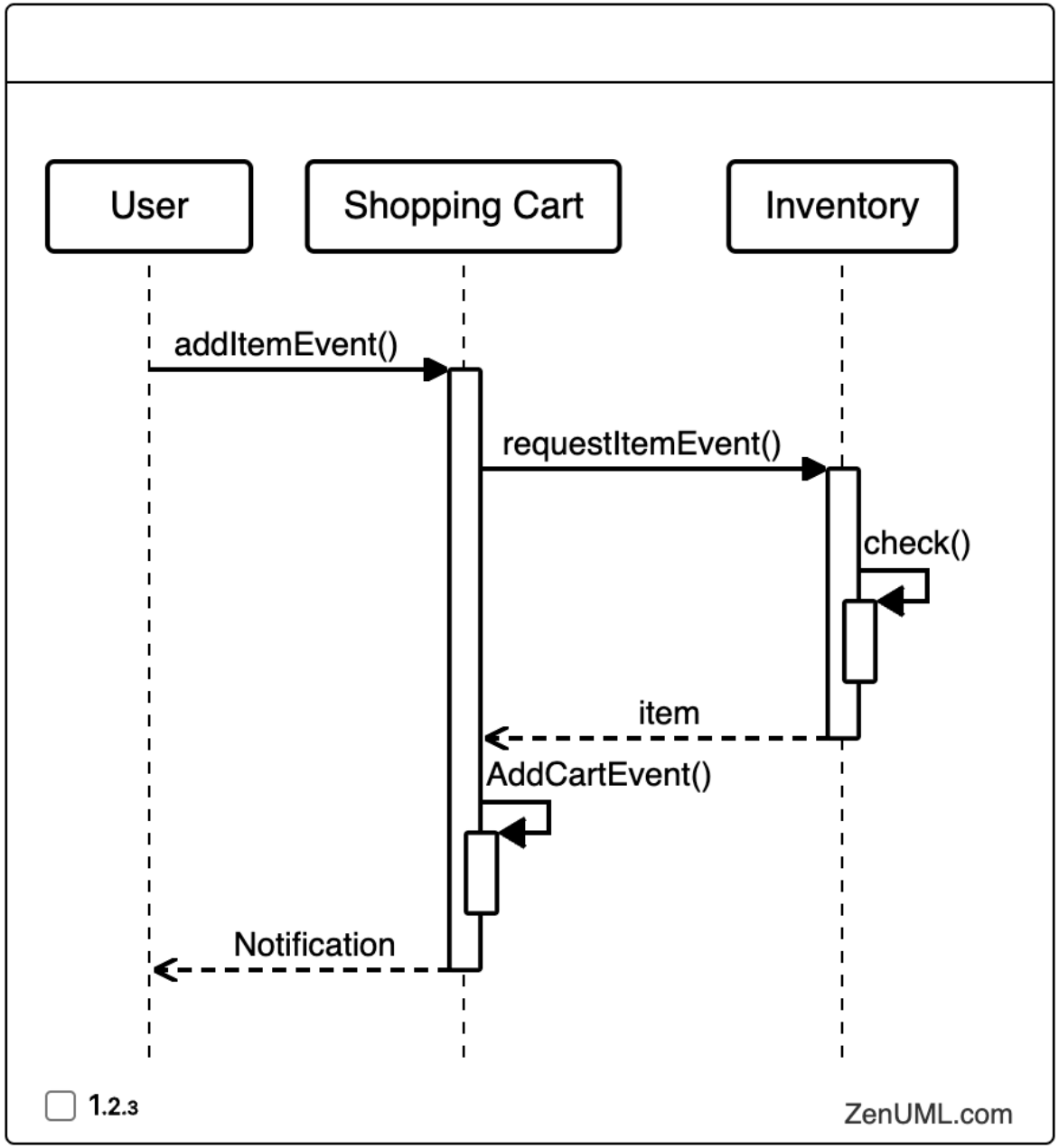
Panoramica - Use-case diagram



Panoramica - Sequence diagram

- Descrive la sequenza di messaggi o eventi scambiati tra oggetti o componenti durante un processo.
- Dal punto di vista del codice, evidenzia l'ordine temporale delle chiamate a metodi, a funzioni o dell'invio di messaggi.
- Ogni oggetto coinvolto è mostrato con una linea verticale, mentre la comunicazione si riconosce dalle frecce orizzontali.
- Si utilizza per analizzare i comportamenti dinamici del sistema, individuando le interfacce necessarie nel software (punti di contatto tra oggetti).

Panoramica - Sequence diagram



Panoramica - altri diagrammi per viste comportamentali

- **Activity diagram:** mostra il **flusso di attività** all'interno di un sistema, in modo simile a quanto avviene nei flow-chart, ma supportando l'esistenza di altri attori in parallelo.
- **State-machine diagram:** modella il comportamento di un singolo oggetto, specificandone la sequenza di stati che attraversa durante il suo ciclo di vita. Evidenzia le azioni che fanno scattare le transizioni di stato.

Conclusioni

- UML è una metodologia ampiamente utilizzata in progettazione, implementazione e documentazione di object oriented software.
- I diagrammi UML servono a visualizzare i diversi processi che rappresentano le macro-viste (strutturali, funzionali e comportamentali) del sistema.
- Il continuo scambio di informazioni tra queste diverse viste porta al progressivo affinamento e miglioramento del sistema software.
- Una buona comprensione di UML aiuta a comunicare in modo chiaro le soluzioni progettuali tra i membri del team e con gli stakeholder, facilitando la gestione dell'intero ciclo di vita del software.