UML

Unified Modeling Language

Introduzione[1] alle **basi del linguaggio UML**

Nel mondo costantemente in fermento ed in evoluzione qual'è quello dello sviluppo di applicazioni Object Oriented, diviene sempre più difficile costruire e gestire applicazioni di alta qualità in tempi brevi.

Come risultato di tale difficoltà e dall'esigenza di avere un linguaggio universale per modellare gli oggetti che potesse essere utilizzato da ogni industria produttrice di software, fu inventato il linguaggio **UML**, la cui sigla sta per ***U**nified **M**odeling **L**anguage*.

In termini pratici, si può dire che il linguaggio UML rappresenta, in qualche modo, una cianografia, un **progetto di ingegneria edile**, riportato nel mondo dell'Information Technology.

L'UML è, dunque, un metodo per descrivere l'architettura di un sistema in dettaglio. Come è facile intuire, utilizzando una sorta di cianografia sarà molto più facile costruire o manutenere un sistema e assicurarsi che il sistema stesso si presterà senza troppe difficoltà a futuri cambiamenti dei requisiti.

La forza dell'Unified Modeling Language consiste nel fatto che il processo di disegno del sistema può essere effettuata in modo tale che i clienti, gli analisti, i programmatori e chiunque altro sia coinvolto nel sistema di sviluppo possa capire ed esaminare in modo efficiente il sistema e prendere parte alla sua costruzione in modo attivo.

Si consideri il seguente esempio: nessuno acquisterebbe un appartamento dando come requisito semplicemente il solo fatto che abbia 4 camere da letto, 3 bagni e che sia grande circa 180 metri quadri. Tutti i clienti desiderano studiare a tavolino, magari con l'ausilio dell'architetto, tutti i dettagli possibili (finestre, porte, ecc.) servendosi, a tale scopo, della mappa della casa. In modo pressoché analogo si può pensare che funzioni la "macchina" dello sviluppo di sistemi software. Invece di leggere la piantina della casa, in questo caso ci si servirà dei *diagrammi UML* che non fanno altro che ***documentare*** in ***modo esauriente*** e ***a tutti i livelli*** le ***caratteristiche tecniche*** che dovranno essere implementate.

> *Come si può riuscire a discernere le **classi** da utilizzare dalla intervista con il cliente?*
>
> E' necessario, a tal fine, prestare particolare attenzione ai ***nomi*** che i clienti usano per descrivere le ***entità*** del loro business. Tali nomi saranno ottimi candidati per diventare delle **classi** nel modello UML. Si deve prestare, altresì, attenzione ai ***verbi*** che vengono pronunciati dai clienti. Questi costituiranno, con molta probabilità, i **metodi** (le operazioni) nelle classi definite. Gli attributi di una classe verranno stabiliti con le analoghe modalità utilizzate per i nomi delle classi.

---

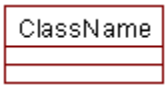
[1] Per una "Guida al linguaggio UML": http://www.html.it/guide/guida-uml/ nell'analisi e design di sistemi (http://www.analisi-disegno.com/) con chiara introduzione di A.Comai.
Per ambiente SW: Argo UML a confronto con altri *tools open source per creare diagrammi; dedicati plug-in per IDE (easyUML – plugin di NetBeans; plugin di Eclipse) o applicazione abilitata per Google Drive (draw.io)*

Rappresentazione** (definizione grafica) **di una classe in linguaggio UML

Definizione: Una classe è una categoria o un gruppo di oggetti (con questo termine includiamo, per comodità anche gli esseri viventi) che hanno ***attributi*** simili e ***comportamenti*** analoghi.

Gli oggetti possono essere suddivisi in categorie e, quindi, in classi.

Una classe viene rappresentata da un **rettangolo**. Il nome della classe, per convenzione, è una parola con l'iniziale maiuscola ed appare vicino alla sommità del rettangolo. Se il *nome della classe* definita consiste di una parola composta a sua volta da più parole allora viene utilizzata la notazione in cui tutte le iniziali di ogni parola sono scritte in maiuscolo.

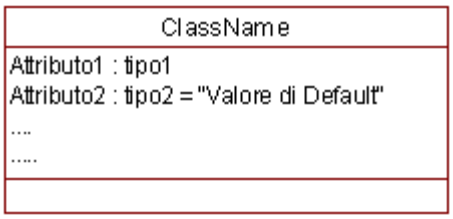


Definizione generica di una classe

Un ***attributo*** rappresenta una proprietà di una classe. Esso descrive un insieme di valori che la proprietà può avere quando vengono istanziati oggetti di quella determinata classe. Una classe può avere zero o più attributi.

Un **attributo** il cui nome è costituito da una sola parola viene scritto sempre in caratteri minuscoli. Se, invece, il nome dell'attributo consiste di più parole (es: Informazioni-Cliente) allora il nome dell'attributo verrà scritto unendo tutte le parole che ne costituiscono il nome stesso con la particolarità che la prima parola verrà scritta in minuscolo mentre le successive avranno la loro prima lettera in maiuscolo. Nell'esempio appena visto l'attributo sarà identificato dal termine: informazioniCliente.

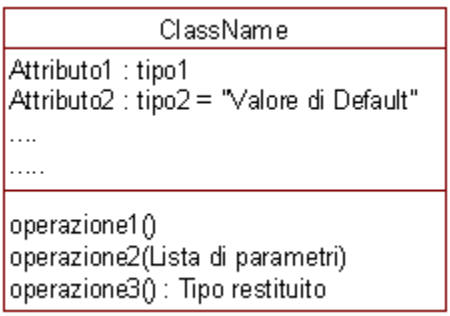
La ***lista degli attributi*** di una classe viene separata graficamente dal nome della classe a cui appartiene tramite una linea orizzontale.



Nell'icona della classe, come si vede nella figura precedente, è possibile specificare un *tipo* in relazione ad ogni attributo (string, float, int, bool, ecc.). E' anche possibile specificare un *valore di default* che un attributo può avere.

Un'**operazione** è un'azione che gli oggetti di una certa classe possono compiere.
Analogamente al nome degli attributi, il nome di un'operazione viene scritto con caratteri minuscoli. Anche qui, se il nome dell'operazione consiste di più parole, allora tali parole vengono unite tra di loro ed ognuna di esse, eccetto la prima, viene scritta con il primo carattere maiuscolo. La *lista delle operazioni* (***metodi***) viene rappresentata graficamente sotto la lista degli attributi e separata da questa tramite una linea orizzontale.

Anche i metodi possono avere delle informazioni addizionali. Nelle parentesi che seguono il nome di un'operazione, infatti, è possibile mostrare gli eventuali *parametri* necessari al metodo insieme al loro *tipo*. Infine, se il metodo rappresenta una funzione è necessario anche specificare il *tipo restituito*.



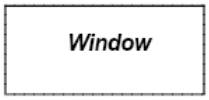
Modello Statico

Il modello **statico** di riferimento per la modellazione strutturale con **UML** permette di mostrare:

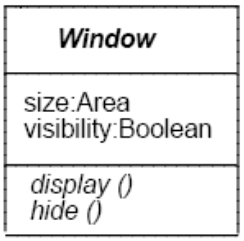
- le entità[2] presenti nel modello (classi, interfacce, componenti, nodi)
- la struttura interna
- le relazioni statiche tra entità

il livello di astrazione del diagramma (struttura statica) può essere:

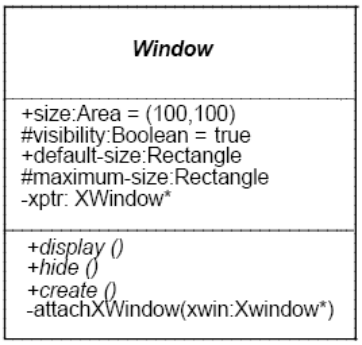
- concettuale



- di specifica



- di implementazione



[2] Per approfondire: le relazioni tra classi, rappresentazione di oggetti e tipi di relazioni