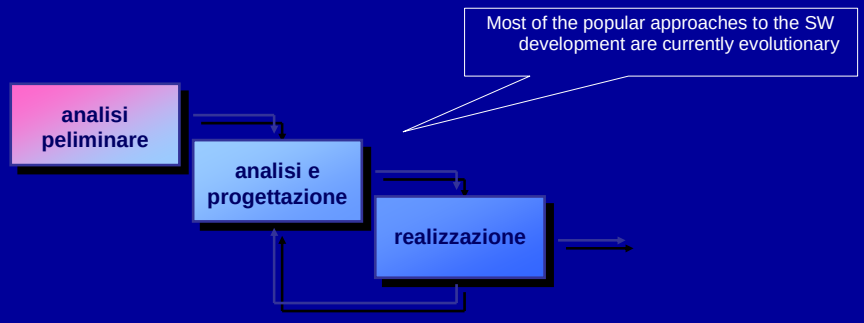
Introduzione a UML

+ Classi e oggetti

Ingegneria del Software B
a.a. 2009/2010
Laura Semini

Ruolo dell'analisi e della progettazione



- Strumenti automatici di supporto alla realizzazione
- Necessità di un maggior supporto alle attività di analisi e progettazione.

UML

- Unified Modelling Language
  - Metodo Booch OOD
  - OMT, Object Modelling Technique (Rumbaugh)
  - OOSE (Jacobson): linguaggio di modellazione OO
- Standard OMG (Object Management Group)
  - consorzio di aziende (CORBA)
  - standard non proprietario
  - UML 2.0 ufficializzato nel 2005

Generalità

- Modellazione in tutte le fasi del processo di sviluppo
- Applicabile a molti tipi di progetti e domini
- Indipendente
  - dal linguaggio di sviluppo
  - dal modello di processo
- Unificazione a livello di linguaggio *non* a livello di metodo

Obiettivi di UML

- **Visualizzazione**
  - **Comunicazione e comprensione**
- **Specifica e documentazione**
  - **Descrizione del sistema in tutti i suoi livelli**
- **Realizzazione**
  - **Supporto all'automazione della codifica**

Concetti di UML

- **Modello: astrazione di (parte di) un sistema**
  - ESEMPIO: Auto
    - il modellino mignon,
    - il progetto della stessa
    - la formula per il calcolo dello spazio di frenata
  - **Modello statico: Descrive gli elementi del sistema e le loro relazioni**
  - **Modello dinamico: Descrive il comportamento del sistema nel tempo**
- **Progetto (o anche disegno)**
  - **Insieme dei modelli: le diverse dimensioni del sistema**
- **Esistono disegni con preponderanza**
  - **modelli statici: DB**
  - **dinamici: sistemi di controllo, di calcolo**
- **Vista: descrizione di un aspetto di un modello**

Caratteristiche dei modelli UML

- Tolleranza a inconsistenze e incompletezza
  - Supporto al dialogo tra le parti interessate
- Meccanismi di strutturazione (package)
- Personalizzazione mediante stereotipi
- Strumenti di supporto disponibili sul mercato

Come usare UML

- Un progetto può essere realizzato come:
  - Abbozzo (sketch)
  - Progetto dettagliato (blueprint)
  - Eseguibile (UML come linguaggio di programmazione)
- Secondo due prospettive
  - Software: elementi UML corrispondono a el.ti sw
  - Concettuale: " " " dominio

Definire un linguaggio formale

Linguaggio_formale: *Sintassi* + *Semantica*

Sintassi = Le regole attraverso cui gli elementi del linguaggio (ad es. le classi) sono raggruppati in espressioni (ad es. i diagrammi)

Semantica = Le regole che assegnano un significato alle espressioni sintattiche

Metamodello di UML

- **Per specificare la sintassi e la semantica**
- **Gli utilizzatori sono i costruttori dei tools e i modellatori esperti.**
- **UML descrive il proprio metamodello in UML, come alcuni compilatori che vengono utilizzati per compilare se stessi.**
- **Stile semi-formale che combina linguaggio naturale e formale.**
- **Esiste una rappresentazione XML del metamodello che si chiama XMI (XML Metadata Interchange)**

Elementi di base di UML

- ***Entità*** : classi, interfacce, componenti, casi d'uso, ecc.
- ***Relazioni*** : associazioni, generalizzazioni, dipendenze, ecc.
- ***Diagrammi*** : diagrammi delle classi, dei casi d'uso, d'interazione, ecc.

Modello statico

- **I concetti del dominio**
  - **modellati mediante classi e relazioni**
- **Realizzazione del sistema**
  - **classi di realizzadione, componenti e nodi**
- **Assenza degli aspetti dipendenti dal tempo**

Modello dinamico

- Modella il comportamento delle entità descritte nel modello statico
  - Il modello dinamico si basa su quello statico

Diagrammi

- Rappresentazione grafica
  - di un insieme di elementi del modello
  - secondo una certa vista
- Grafo
  - Vertici elementi del modello
  - Archi relazioni

Diagrammi

- delle classi
- degli oggetti
- delle componenti
- di struttura composita
- di macchina a stati
- dei casi d'uso
- di attività
- di sequenza
- di dislocazione
- dei packages

Diagrammi e modelli

- I diagrammi si usano per costruire i modelli
- I diagrammi non sono i modelli
  - Un modello è un grafo di elementi *semantici*
  - Un diagramma è un grafo di elementi *visuali* che rappresentano gli elementi semantici
- Negli strumenti di supporto
  - Un modello è la struttura su cui si opera
  - Un diagramma è una struttura di presentazione
  - Non eliminare un elemento solo da un diagramma

Classi e oggetti

- Una *classe* cattura
  - un concetto nel dominio del problema o della realizzazione
- Una classe descrive
  - un insieme di oggetti con caratteristiche simili
  - cioè oggetti che hanno lo stesso *tipo*
- Un *oggetto* è un'entità caratterizzata da
  - Un'identità
  - Uno stato
  - Un comportamento

Classificatori e istanze

- Le classi sono *classificatori*
- Gli oggetti sono *istanze*
- Modellare a livello dei classificatori significa vincolare i modelli a livello di istanza

Classificatori e istanze (continua)

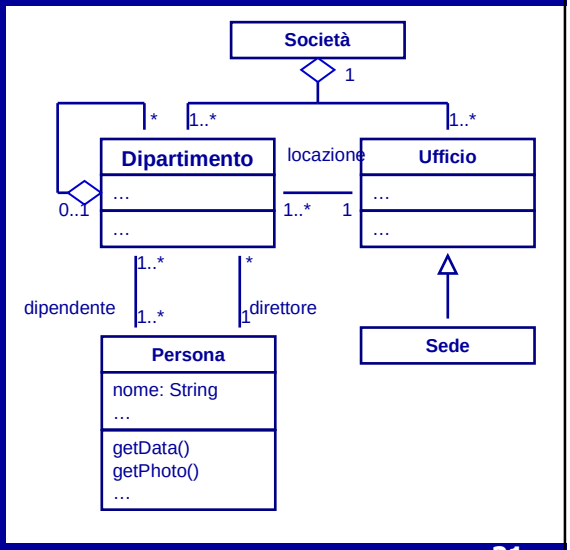
- In teoria i modelli a livello di istanza possono esistere solo in un ambiente definito dai modelli a livello dei classificatori
  - Le variabili in un linguaggio fortemente tipato possono vivere solo in un ambiente in cui il tipo è definito
- In pratica UML è più flessibile
  - Si possono introdurre oggetti senza definirne le classi
  - UML tollera le inconsistenze

Classi UML vs entità (DB)

- DB: le classi sono intese come collezioni. Sottointeso che
  - ci sono più istanze.
  - ci sono operazioni per visitare tutte le istanze.
- Da cui, per esempio, nome singolare vs nome plurale.
- La differenza è significativa più in prospettiva di progettazione che di descrizione del dominio.
  - In progettazione OO e quindi in UML uso "ListaDiQcosa" come aggregato di "Qcosa"
- Un esempio dal dominio:

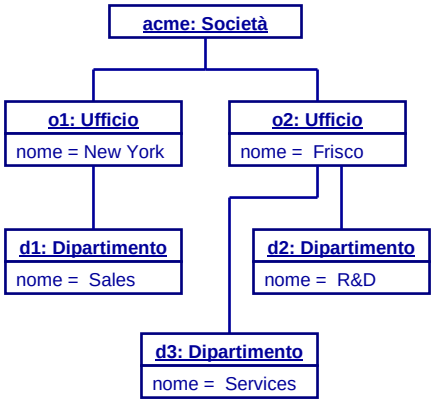
Esempio: classi

- Una società è formata da dipartimenti e uffici
- Un dipartimento ha un direttore e più dipendenti
- Un dipartimento è situato in un ufficio
- Esiste una struttura gerarchica dei dipartimenti
- Le sedi sono uffici

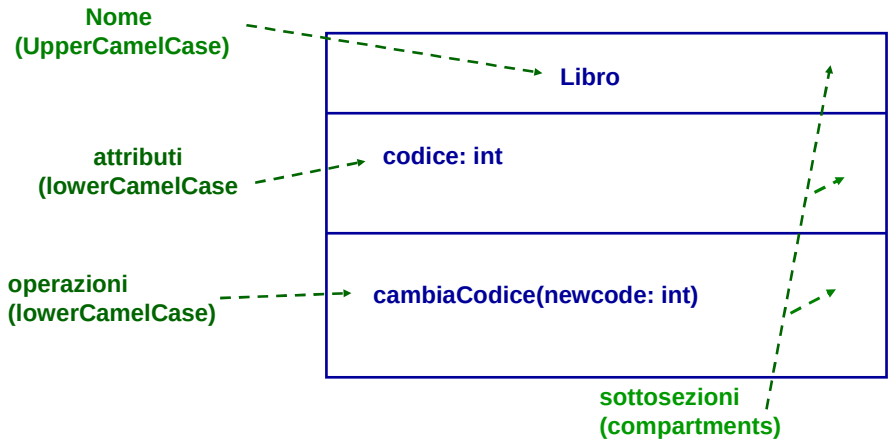


Esempio: oggetti

- Una società reale, ACME
- Ha due Uffici
- Il dipartimento vendite è a New York
- I dipartimenti Ricerca&Sviluppo e Servizi sono a San Francisco



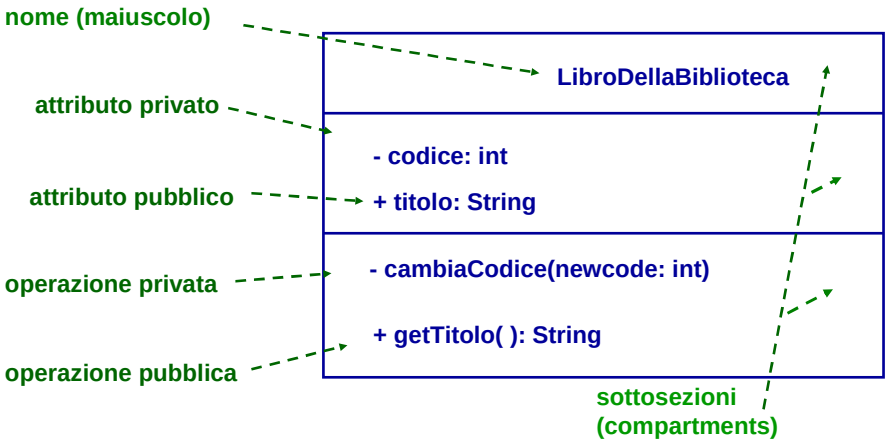
Sintassi della classe



Se non interessano attrib. e operazioni

Libro

Sintassi della classe



Semantica

- Un *oggetto* è un'entità caratterizzata da
  - Un'identità, uno stato, un comportamento
- L'*identità* si vede a livello di istanza
- Gli *attributi* definiscono lo stato dell'oggetto
- Le *operazioni* definiscono il suo comportamento

Spazio dei nomi

- Un classificatore è uno spazio di nomi
  - Gli elementi contenuti hanno un nome unico.
- Per esempio un Package
  - è un costrutto di strutturazione
  - può contenere package innestati e altri elementi (classi, diagrammi …).

Visibilità

- Un elemento è visibile all'esterno dello spazio di nomi che lo contiene, in accordo con il suo tipo di visibilità
  - + public: tutti
  - # protected: i discendenti
  - - private: solo nell'elemento stesso
  - ~ package: nello stesso package

Sintassi attributi

visibilità nome: tipo [molteplicità]= valoreIniziale {proprietà}

obbligatorio

molteplicità: array di valori

colore: Saturazione [3]

nome: String [0..1] 0 per permettere valore null

[1] può essere omesso

proprietà

{ordered}

{>=0}

Visibilità di attributi e operazioni

- In modo simile alla visibilità della classe
  - \+ public: accessibile ad ogni elemento che può vedere e usare la classe
  - # protected: accessibile ad ogni elemento discendente
  - - private: solo le operazioni della classe possono vedere e usare l'elemento in questione
  - ~ package: accessibile solo agli elementi dichiarati nello stesso package

Esempi

- numero di tipo intero

  n: Integer

- numero intero positivo

  n: Integer{>= 0}

- contatore positivo, inizialmente a 0

  n: Integer =0 {>= 0}

Esempi

- Punti del quadrante positivo, pubblico

+ puntiQuadPos : Integer [2] {>= 0}

- sequenza ordinata di 10 interi compresi tra 3 e 33, privato

- seq: Integer[10] {>= 3, <= 33, ordered}

Sintassi operazioni

visibilità nome (listaParametri) : tipoRitorno {proprietà}

obbligatori

può essere vuota

listaParametri

default

direzione nome: tipo = default

in, out, inout

valore assegnato al parametro in assenza di argomento

Esempi

- Metodo pubblico che restituisce la somma di 2 interi

  + sum (a: Integer, b: Integer) : Integer

- sum con 10 valore di default del secondo parametro

  + sum (a: Integer, b: Integer =10) : Integer

- Metodo privato che restituisce un oggetto di tipo Gra

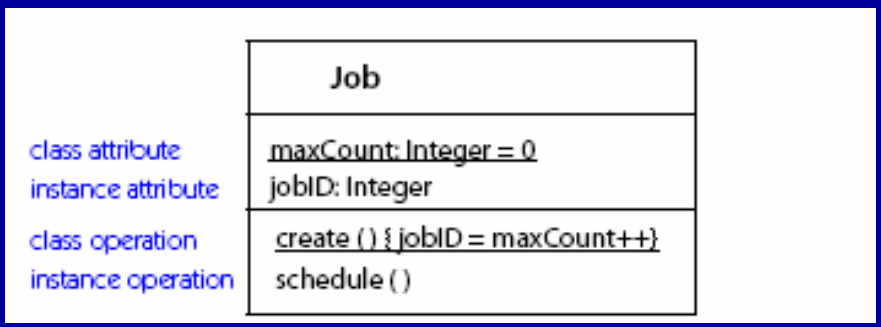
  - gra () : Gra

attributi e operazioni con ambito di classe (statici)

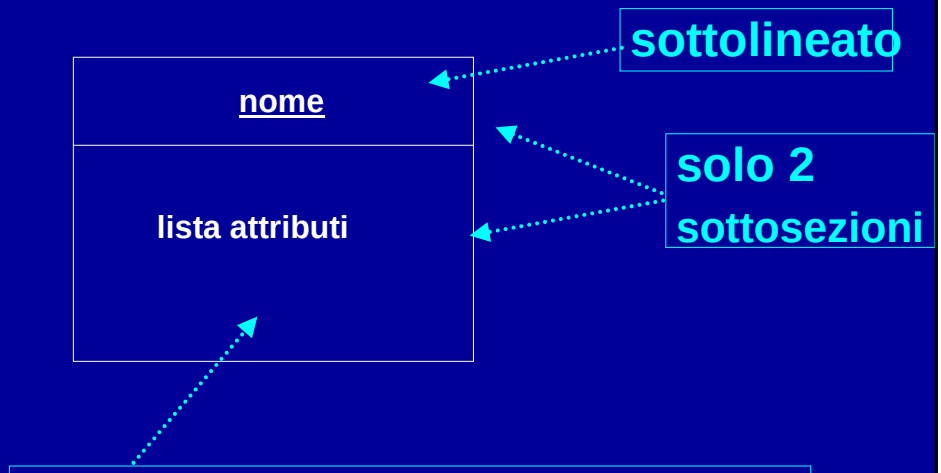
sottolineati

Classe
+ numeroIstanze : Integer =0
+ sum (a: Integer, b: Integer) : Integer

Esempio



Diagrammi degli oggetti



Diagrammi degli oggetti: nome

nomeoggetto: Nomeclasse

minou: Gatto

nomeoggetto

minou

: Nomeclasse

: Gatto

nomeoggetto: Nomeclasse, Nomeclasse

minou: Gatto,Cantante

Diagrammi degli oggetti: attributi

Classi e Oggetti

Individuare le classi di analisi

- Cosa sono le classi di analisi
- Come sono fatte
- Tecniche per individuarle
  - Nome-verbo
  - CRC

Cosa sono le classi di analisi

- Corrispondono a concetti concreti del dominio:
  - Per esempio i concetti descritti nel glossario
  - Normalmente, ciascuna classe di analisi sarà raffinata in una o più classi di progettazione.
- Evitare di introdurre delle classi di progettazione

Classi di analisi: caratteristiche

- Astrazione di uno specifico elemento del dominio
- Numero ridotto di responsabilità (funzionalità)
- Evitare le classi “onnipotenti”
  - Attenzione quando si chiamano “sistema”, “controllore”, ...
- Evitare funzioni travestite da classi
- Evitare gerarchie di ereditarietà profonde (>=3)
- Coesione e disaccoppiamento
  - Tenere responsabilità simili in una classe
  - Limitare interdipendenze tra classi

Classi di analisi: livello di dettaglio

- Operazioni e attributi solo quando veramente utili
  - Le classi di analisi dovrebbero contenere attributi e operazioni ad “alto livello”
  - Limitare la specifica di tipi, valori, etc.
  - Non inventare mai niente!

Identificazione delle classi

- Problema classico delle prime fasi di sviluppo
- Approccio data driven
  - Si identificano tutti i dati del sistema e si dividono in classi (ad esempio mediante identificazione dei sostantivi)
- Approccio responsibility driven
  - Si identificano le responsabilità e si dividono in classi (ad esempio mediante CRC Cards)

Analisi nome-verbo

- Sostantivi → classi o attributi
- Verbi → responsabilità o operazioni
- Passi:
  1. Individuazione delle classi
  2. Assegnazione di attributi e responsabilità alle classi
  3. Individuazione di relazioni tra le classi

Analisi nome-verbo

Problemi ricorrenti :

- **Tagliare le classi inutili**
  - **Trattare i casi di sinonimia**
- **Individuare le classi nascoste cioè le classi implicite del dominio del problema che possono anche non essere mai menzionate esplicitamente**
  - **In un sistema di prenotazione di una compagnia di viaggi si potrebbe parlare di prenotazione, richiesta, ma tralasciare il termine ordine**

Tagliare le classi inutili

1. **Elenco dei sostantivi**
2. **Eliminazione dei sostantivi riconosciuti come**
   - **Sinonimi**
   - **Eventi o operazioni**
   - **Metalinguaggio** (sistema)
   - **Inutili** (estranei al sistema)
   - **Attributi** (titolo del libro....)

Quidditch

Il preside di Hogwarts, Albus Silente, chiede la realizzazione di un sistema per la visualizzazione e l'archiviazione dei punti segnati durante le partite del campionato di Quidditch del collegio. Il sistema deve permettere di gestire i seguenti eventi: inizio partita, goal (con punti realizzati e nome del realizzatore), cattura del boccino (con punti realizzati e nome del cercatore) , fine partita. Vengono inseriti dall'aiuto arbitro. Siccome si possono portare le bacchette magiche in campo il sistema dovrà garantire opportuna sicurezza.

- Sinonimi
- Eventi o operazioni
- Metalinguaggio
- Inutili
- Attributi

CRC Cards

- Class, Responsibilities, Collaboration Cards
- Ideate da Ward Cunningham e Kent Beck (Smalltalk) come una tecnica per insegnare a programmatori con esperienza in linguaggi non--OO a pensare in termini di oggetti.
- Non fanno parte di UML, ma sono utili per realizzare i diagrammi UML

CRC Cards

Responsabilità

- Una funzionalità che la classe deve realizzare

➢ Collaboratore

- Le classi che partecipano alla realizzazione di una data responsabilità

Struttura delle CRC Cards

La notazione usata per rappresentare le CRC cards è la seguente: fare una tabella con il nome della classe in cima, sotto a sinistra le responsabilità e a destra, per ogni responsabilità, I collaboratori.

Copia di libro	
Responsabilità	**Collaboratori**
Mantenere lo stato di una particolare copia di un libro	
Informare il libro su prestiti e restituzioni	Libro

Syllabus

- **Arlow**
  - **Capitolo 1. tranne 1.9.4 (per ora)**
  - **Capitolo 7**
  - **Capitolo 8, tranne 8.2 e 8.4.3.**