

Ingegneria del Software

Linguaggi di Modellazione

Prof. Paolo Giorgini

A.A. 2024/2025

Modello

- Rappresentazione di un'entità in modo semplificato
 - astrazione di aspetti o dettagli irrilevanti per lo scopo a cui è destinato il modello - focus sulle informazioni importanti

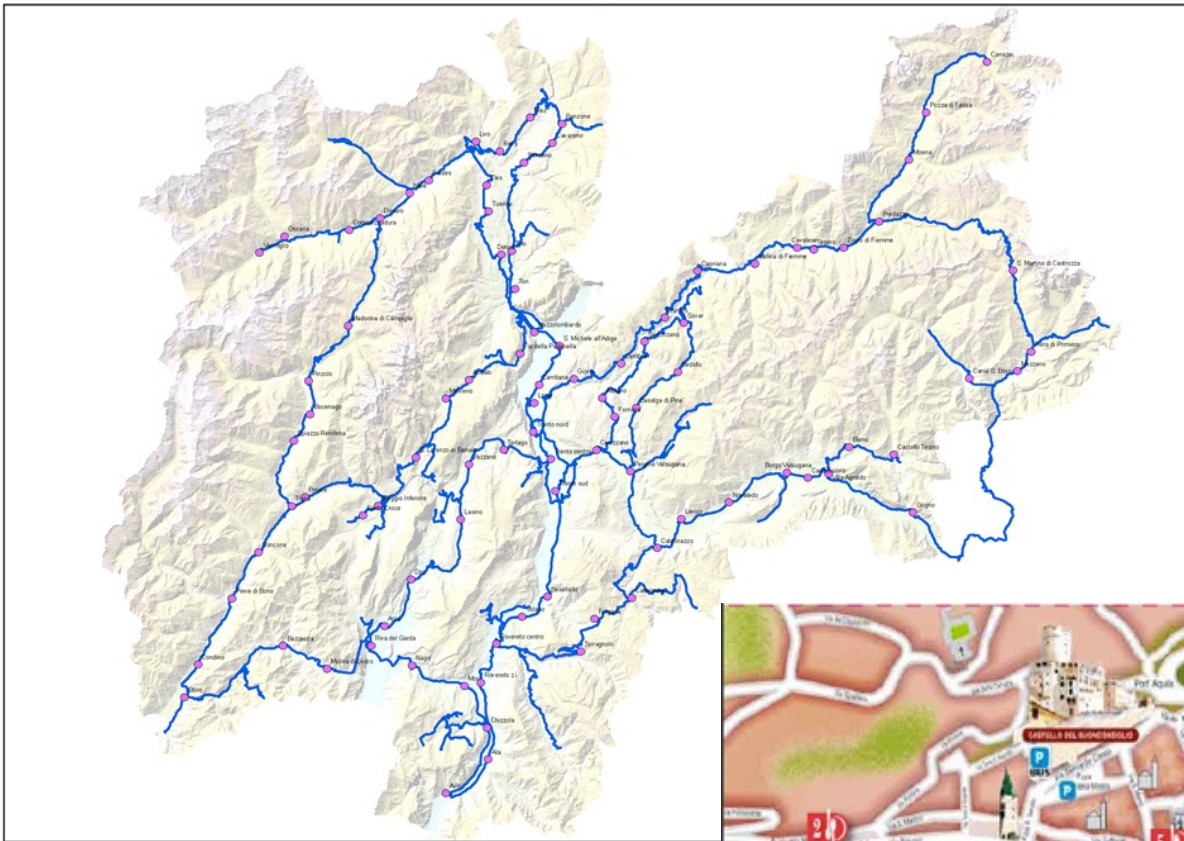


Notazione del modello



Quali linguaggi?

- Un ingegnere civile è in grado di leggere i calcoli relativi alla statica di un edificio elaborati da un qualsiasi collega
 - Linguaggio comune della matematica
- Un ingegnere software?
 - Quale linguaggio? UML?
 - Dipende da cosa si intende modellare!
 - Vocabolario di termini che rappresentano i concetti da modellare – utilizzabili solo per alcuni contesti
 - ES. simboli usati per la mappa della città di Trento non sono adatti a modellare una mappa per gli scavi di posa della fibra ottica in Trentino

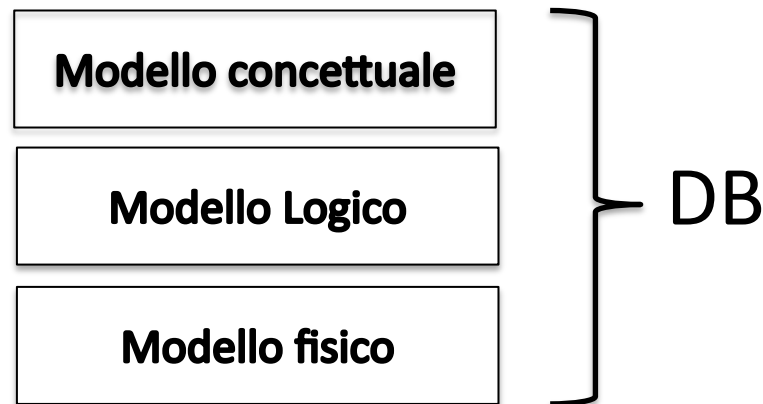


Scelta del Linguaggio




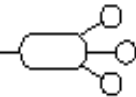


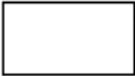

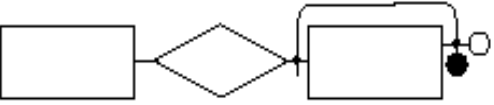
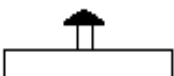

- Facilità d'uso
 - Semplicità di espressione (quanto sia facile produrre un modello) e l'immediatezza della comprensione (quanto sia facile capire il modello)
- Comprensibili anche da non esperti (tecnici)
 - Un'utile strumento di comunicazione con utenti e clienti
- Quanto più formale e preciso è il linguaggio, tanto più il modello può essere utilizzato per analisi ed elaborazioni
 - Formale e preciso non significa necessariamente di difficile da utilizzare
- Linguaggi diversi possono essere utilizzati all'interno delle varie fasi dello sviluppo del software
 - Diversi livelli di astrazione concettuale

Linguaggio Entity-Relationship

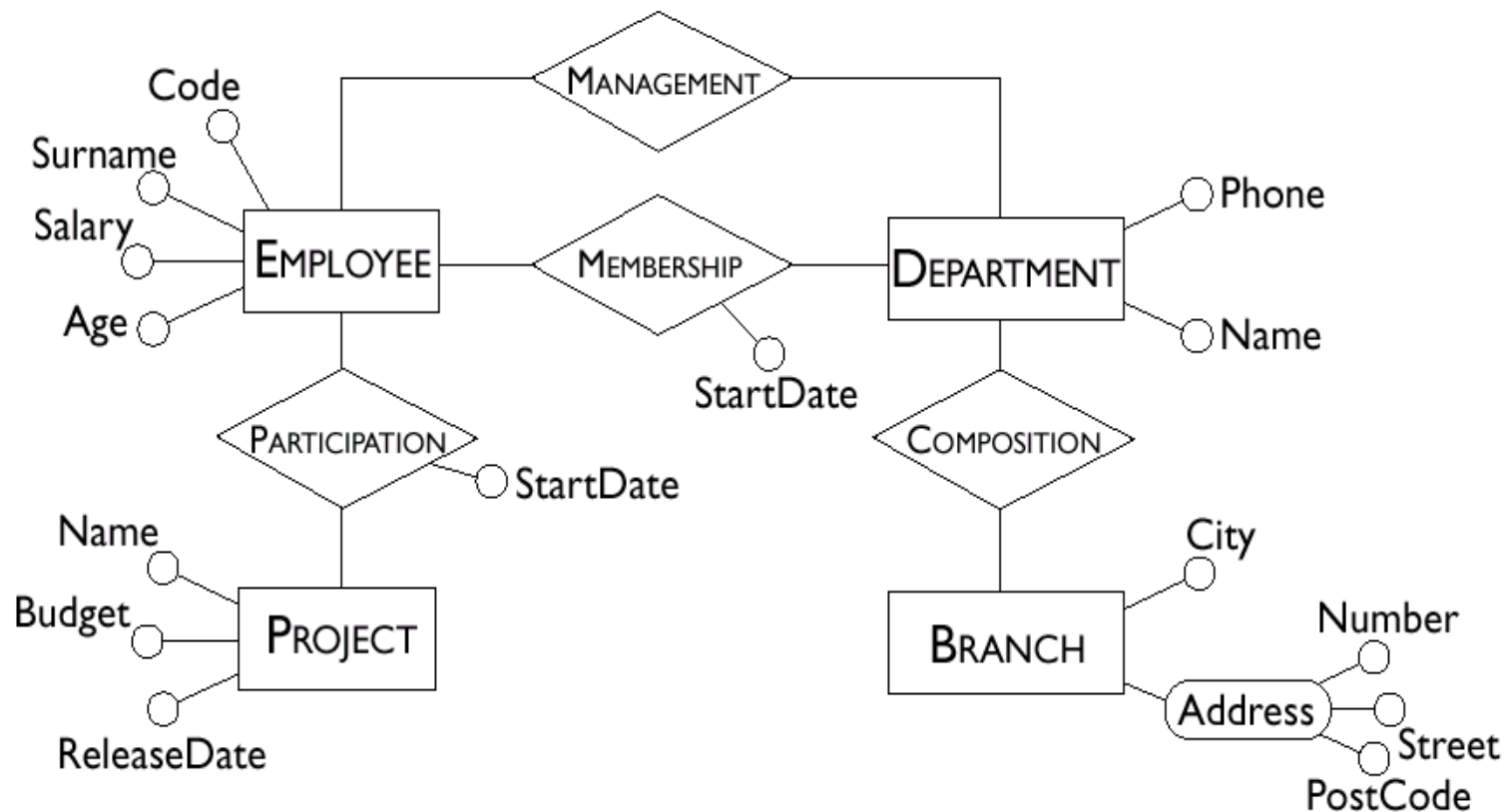
- Utilizzato prevalentemente nell'analisi e progettazione di sistemi informativi e DB
 - Modellazione concettuale (prima di quella logica – modello relazionale)



Costrutti base

Construct	Graphical representation
Entity	
Relationship	
Simple attribute	
Composite attribute	
Cardinality of a	
Cardinality of an attribute	
Internal identifier	 
External identifier	
Generalization	
Subset	

Model with basic attributes



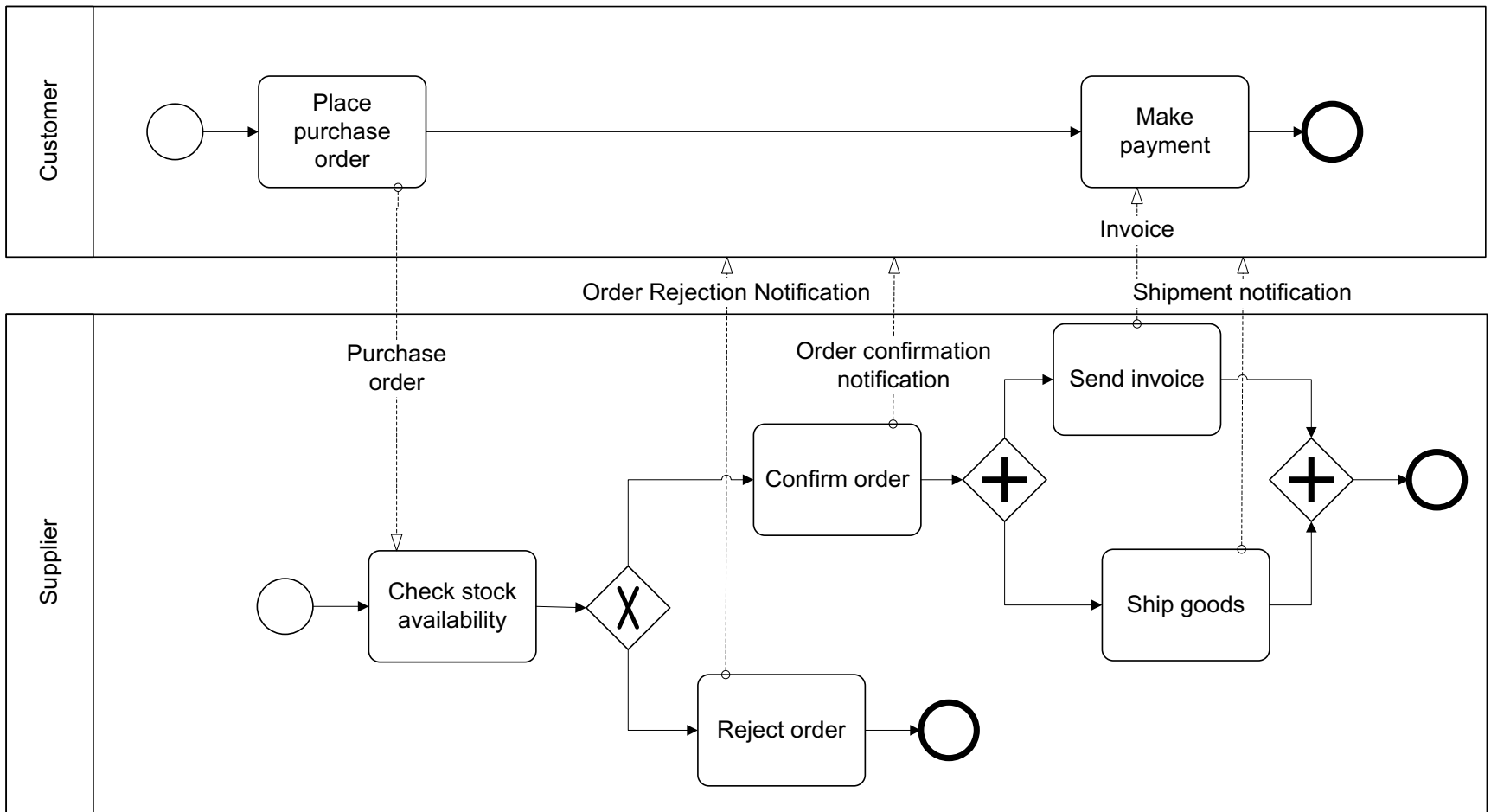
Se si vuole modellare un processo



Business Process Modeling Notation

- BPMN is the **OMG standard** for representing business processes
 - Other standards, such as activity diagrams of UML, were not accepted for process modeling in practice, because their use is restricted to the area of object-oriented software design
- There are many tools for designing a BPMN process:
 - BpmnWebModeler (Trisotech)
 - Bizagi Process Modeller (it also provides an execution engine)
 - JBPM (Eclipse plugin)
 - Signavio
 - TIBCO Business Studio (free download, quite large)
 - IBM Websphere Business Modeler
 - ARIS
 - Oracle BPA
 - Business Process Visual Architect (Visual Paradigm)
 - Progress Savvion Business Modeller

Order Management



Activities



Task

A Task is a unit of work, the job to be performed. When marked with a it indicates a Sub-Process, an activity that can be refined.



Transaction

A Transaction is a set of activities that logically belong together; it might follow a specified transaction protocol.



Event Sub-Process

An Event Sub-Process is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.



Call Activity

A Call Activity is a wrapper for a globally defined Task or Process reused in the current Process. A call to a Process is marked with a .

Activity Markers

Markers indicate execution behavior of activities:

Sub-Process Marker

Loop Marker

Parallel MI Marker

Sequential MI Marker

Ad Hoc Marker

Compensation Marker

Task Types

Types specify the nature of the action to be performed:

Send Task

Receive Task

User Task

Manual Task

Business Rule Task

Service Task

Script Task

Sequence Flow

defines the execution order of activities.

Default Flow

is the default branch to be chosen if all other conditions evaluate to false.

Conditional Flow

has a condition assigned that defines whether or not the flow is used.

Conversations



A Conversation defines a set of logically related message exchanges. When marked with a it indicates a Sub-Conversation, a compound conversation element.

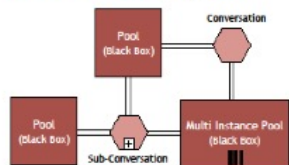


A Call Conversation is a wrapper for a globally defined Conversation or Sub-Conversation. A call to a Sub-conversation is marked with a .

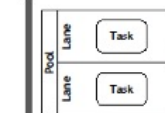
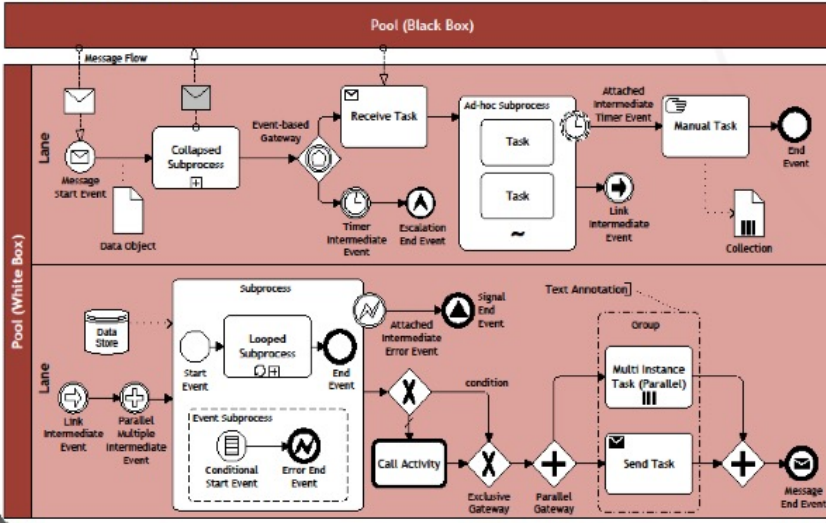


A Conversation Link connects Conversations and Participants.

Conversation Diagram



Collaboration Diagram

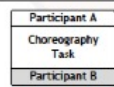


Pools (Participants) and Lanes represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.

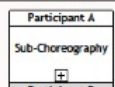
Swimlanes

Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events. The Message Flow can be decorated with an envelope depicting the content of the message.

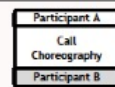
Choreographies



A Choreography Task represents an Interaction (Message Exchange) between two Participants.

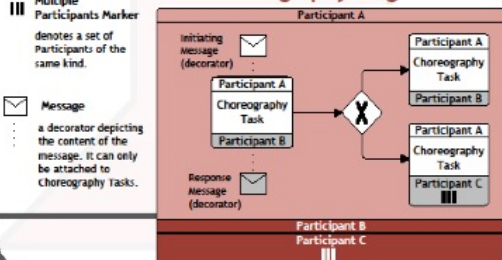


A Sub-Choreography contains a refined choreography with several interactions.



A Call Choreography is a wrapper for a globally defined Choreography Task or Sub-Choreography. A call to a Sub-Choreography is marked with a .

Choreography Diagram



Events

	Start	Intermediate	End
Standard			
Event Sub-Process Interrupting			
Event Sub-Process Non-Interrupting			
Catching			
Boundary Interrupting			
Boundary Non-Interrupting			
Throwing			
Standard			
None: Untyped events, indicate start point, state changes or final states.			
Message: Receiving and sending messages.			
Timer: Cyclic timer events, points in time, time spans or timeouts.			
Escalation: Escalating to an higher level of responsibility.			
Conditional: Reacting to changed business conditions or integrating business rules.			
Link: Off-page connectors. Two corresponding link events equal a sequence flow.			
Error: Catching or throwing named errors.			
Cancel: Reacting to cancelled transactions or triggering cancellation.			
Compensation: Handling or triggering compensation.			
Signal: Signalling across different processes. A signal thrown can be caught multiple times.			
Multiple: Catching one out of a set of events. Throwing all events defined.			
Parallel Multiple: Catching all out of a set of parallel events.			
Terminate: Triggering the immediate termination of a process.			

Data



A Data Object represents information flowing through the process, such as business documents, e-mails, or letters.



A Collection Data Object represents a collection of information, e.g., a list of order items.



A Data Input is an external input for the entire process. A kind of input parameter.



A Data Output is data result of the entire process. A kind of output parameter.



A Data Association is used to associate data elements to Activities, Processes and Global Tasks.



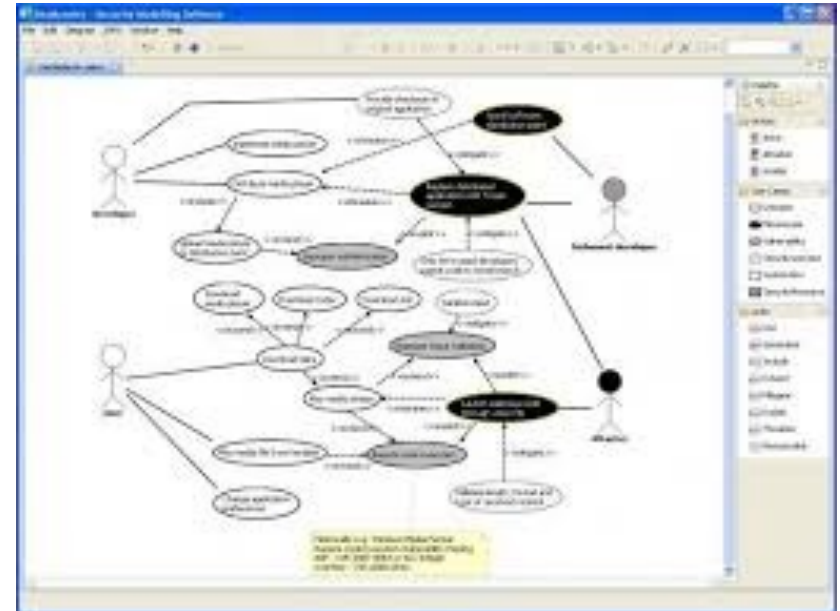
A Data Store is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

Sicurezza?



Linguaggi per la sicurezza

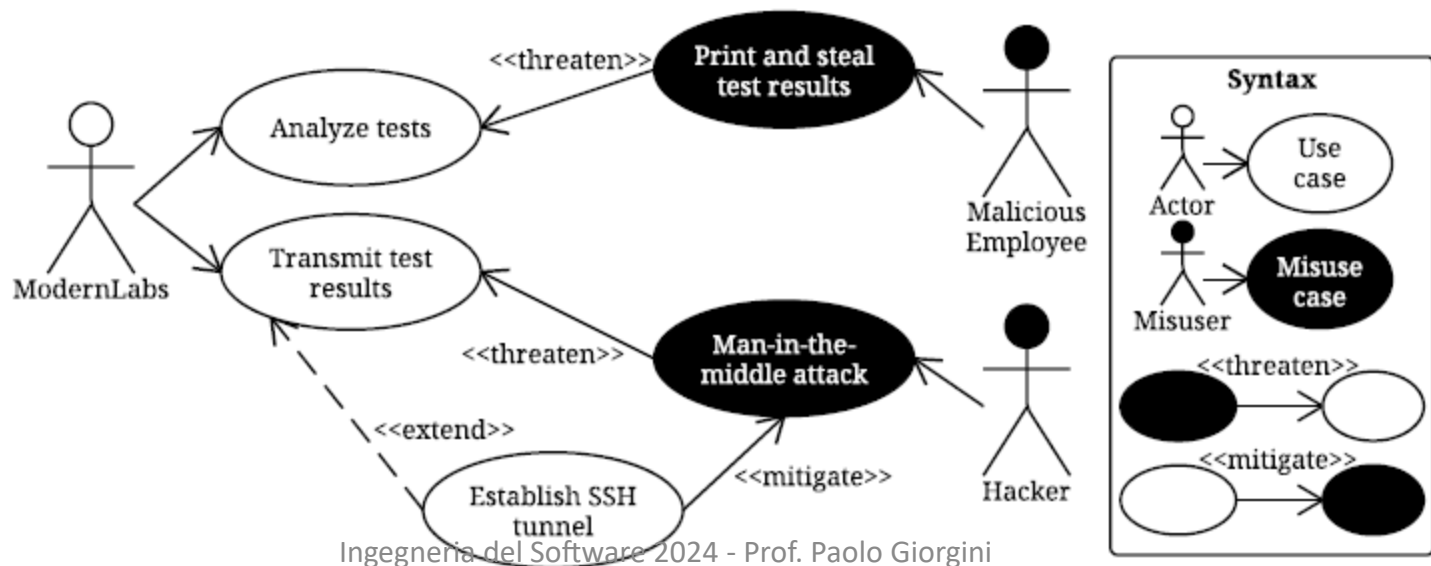
- Molti approcci alla Security Requirements Engineering (SRE) usano linguaggi per costruire modelli del sistema
 - documentazione precisa e rigorosa
 - forme di analisi automatica
- Diversi tipi di modelli
 - UML-based
 - Goal-oriented
 - Machine-oriented
 - Business processes
 - Socio-technical systems



Linguaggi UML-based

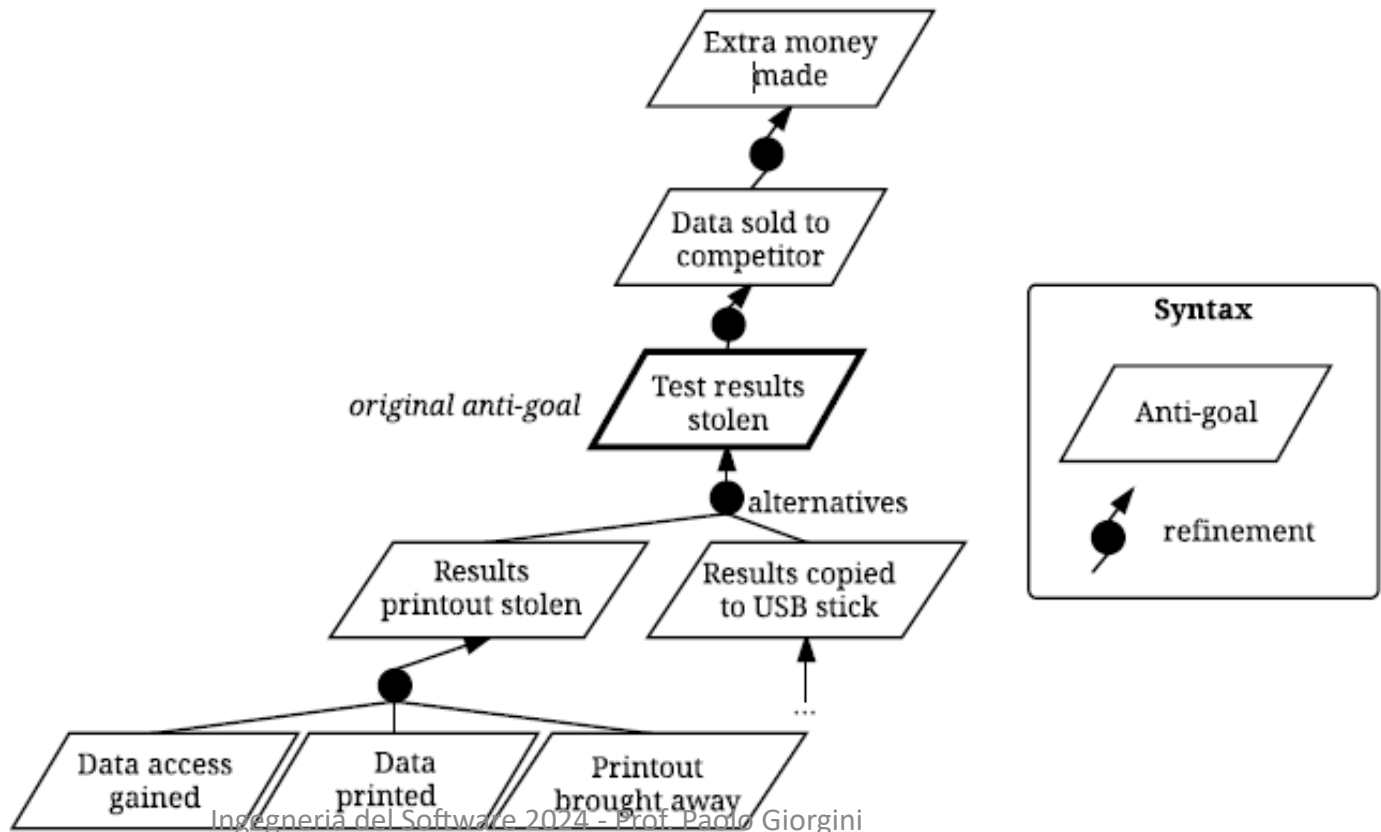
- **Misuse case**

- Alcuni attori sono “malicious” e tentano di sfruttare il sistema
- Altri attori progettano soluzioni di sicurezza per evitare il problema

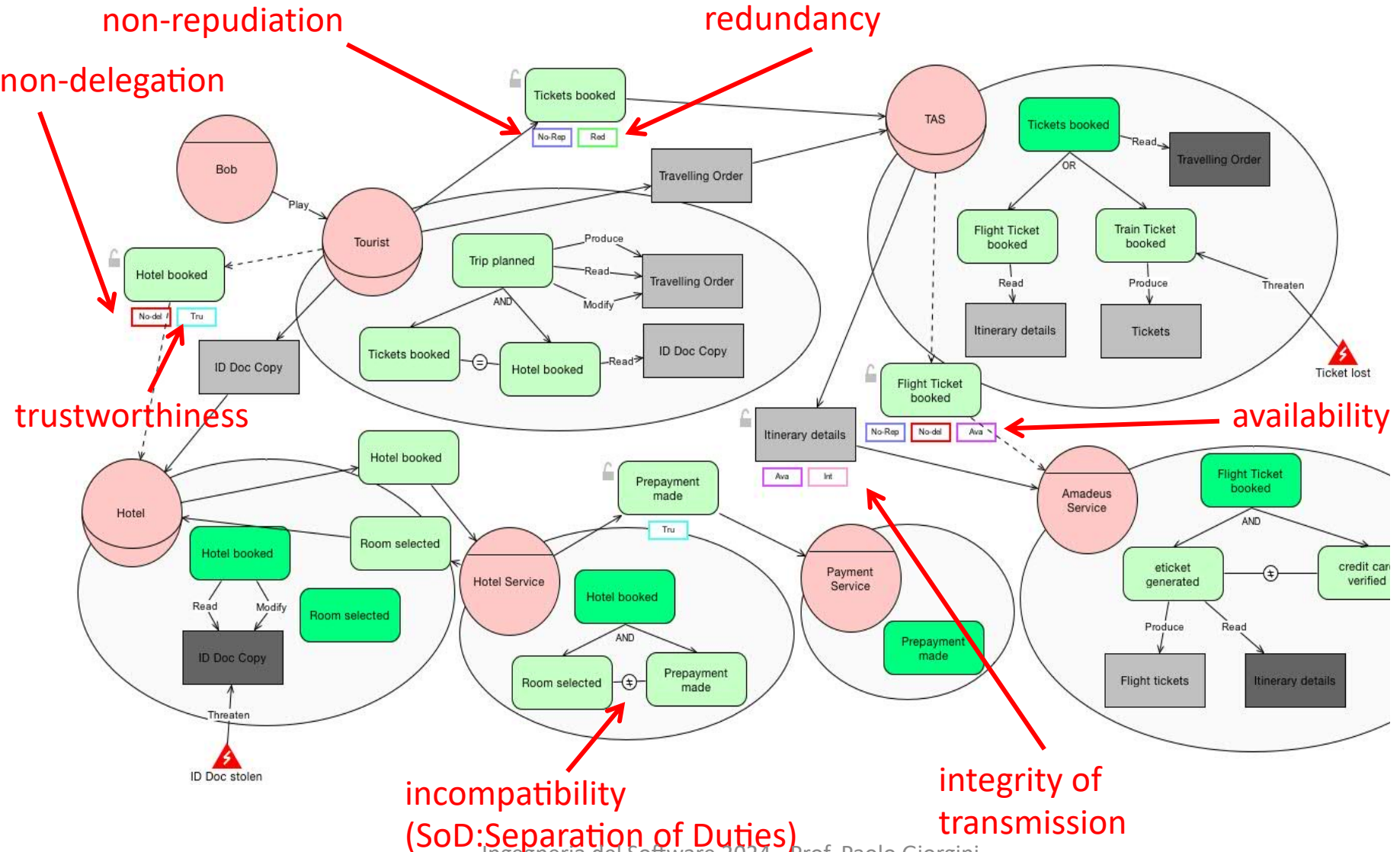


Linguaggi goals-based

- **Anti-goals:** quali sono gli obiettivi degli "attacker"



Social view: expressing security needs



UML

- UML = **U**nified **M**odeling **L**anguage
- Famiglia di notazioni grafiche per il supporto all'analisi e alla progettazione di un sistema software
 - In particolare sistemi OO, ma non solo
- OMG standars
 - Object management group
 - Consorzio di aziende IT
 - No profit
 - Creare degli standard nel contesto IT



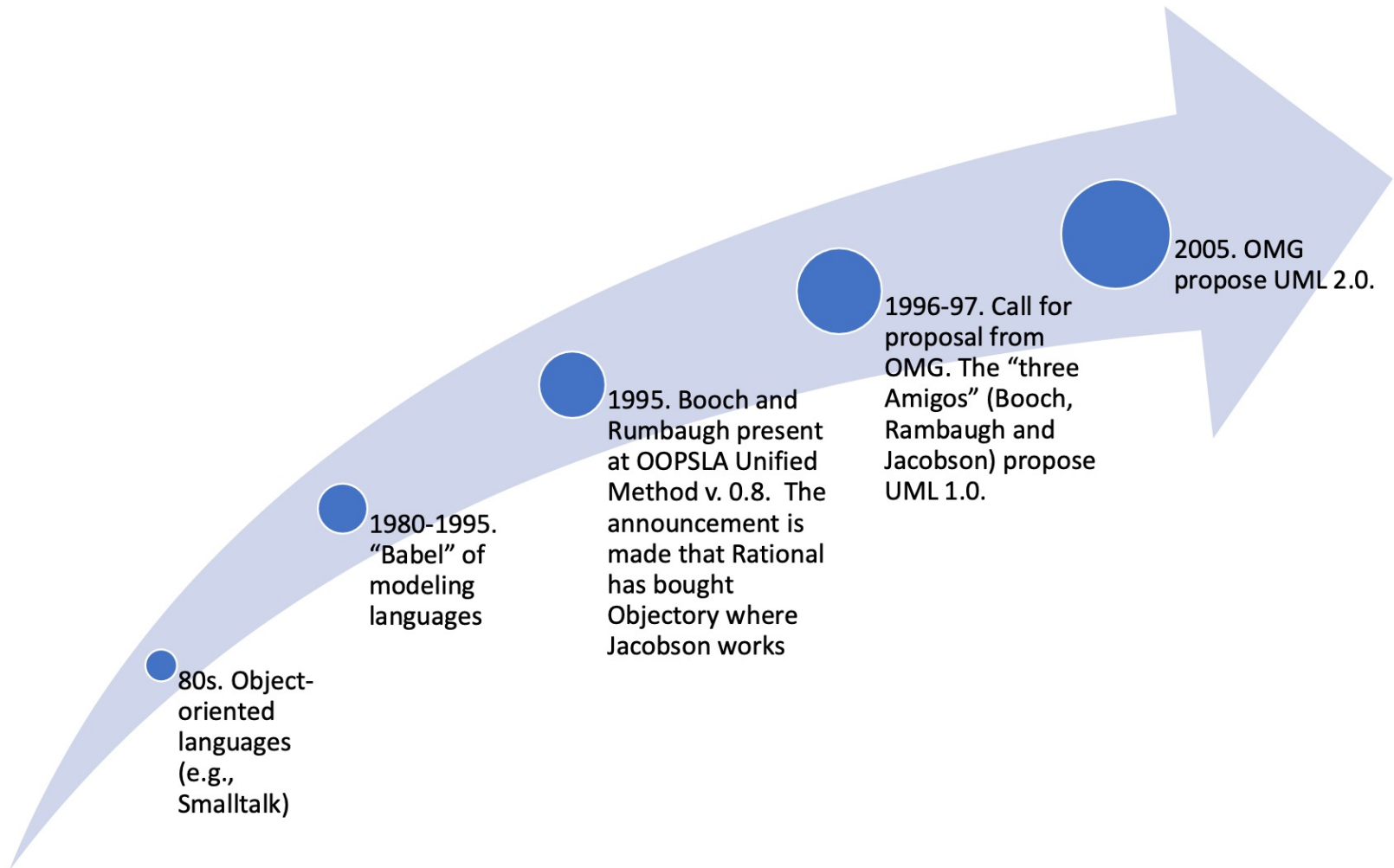
Che cosa non è UML

- Non è un metodo di sviluppo (metodologia)!
- E' un linguaggio / notazione
- Non è un linguaggio di programmazione
- E' un linguaggio di modellazione
- Non è legato ad una metodologia particolare
- Può essere usato dai metodi più classici fino a quelli agile

Come usare UML

- Sketch
 - Scopo: facilitare la comunicazione e la discussione - solo le informazioni più importanti (diagrammi incompleti)
 - Tool: editor grafico o lavagna
- Progetto dettagliato (blueprint)
 - Scopo: fornire un modello completo/dettagliato da implementare, in modo che la programmazione sia principalmente un'attività meccanica
 - Tools: modellatore UML
- Verso il linguaggio di programmazione
 - Scopo: fornire un “modello eseguibile”
 - Tools: specifici per eseguire/trasformare il modello

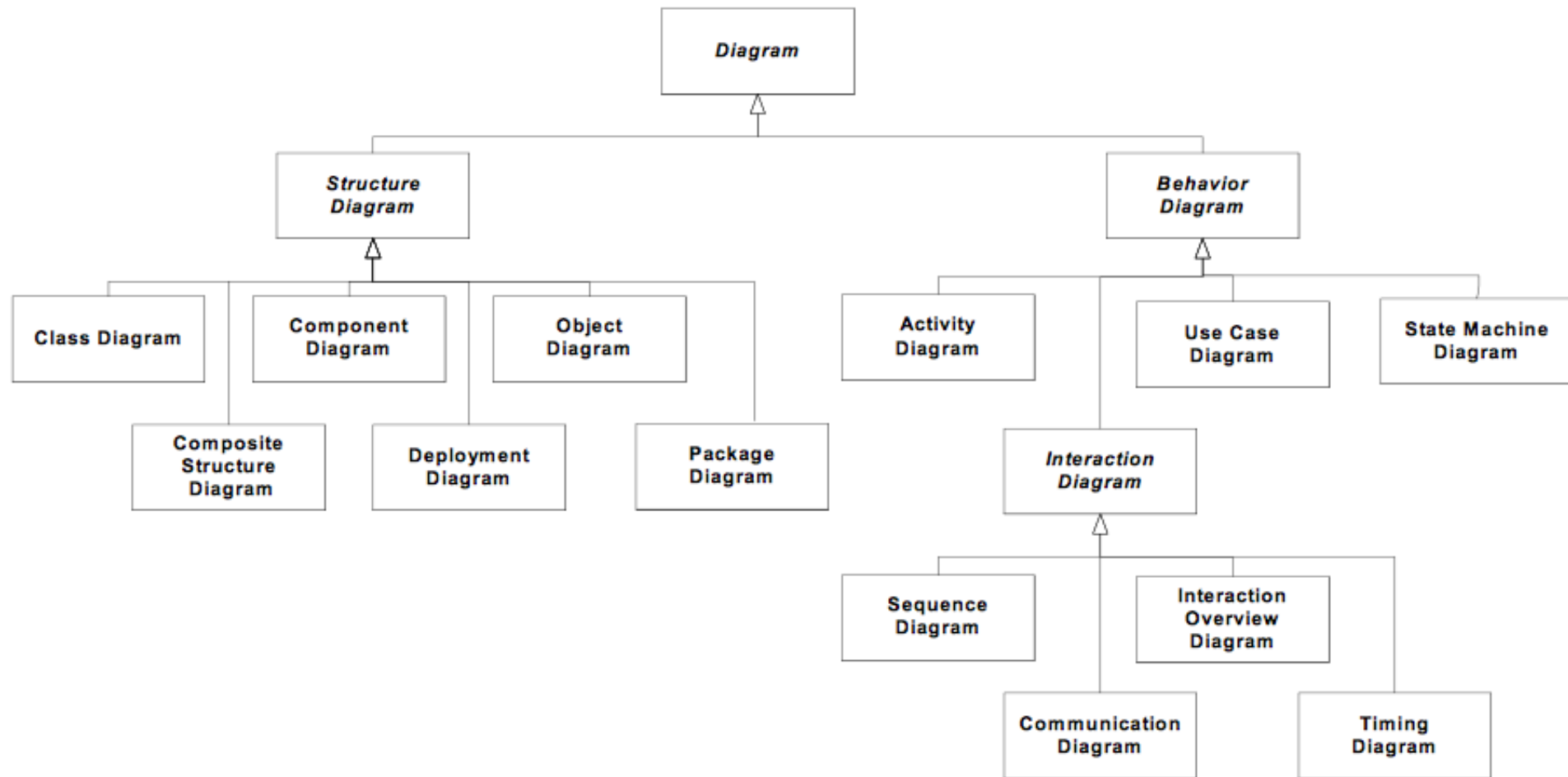
Un pò di storia



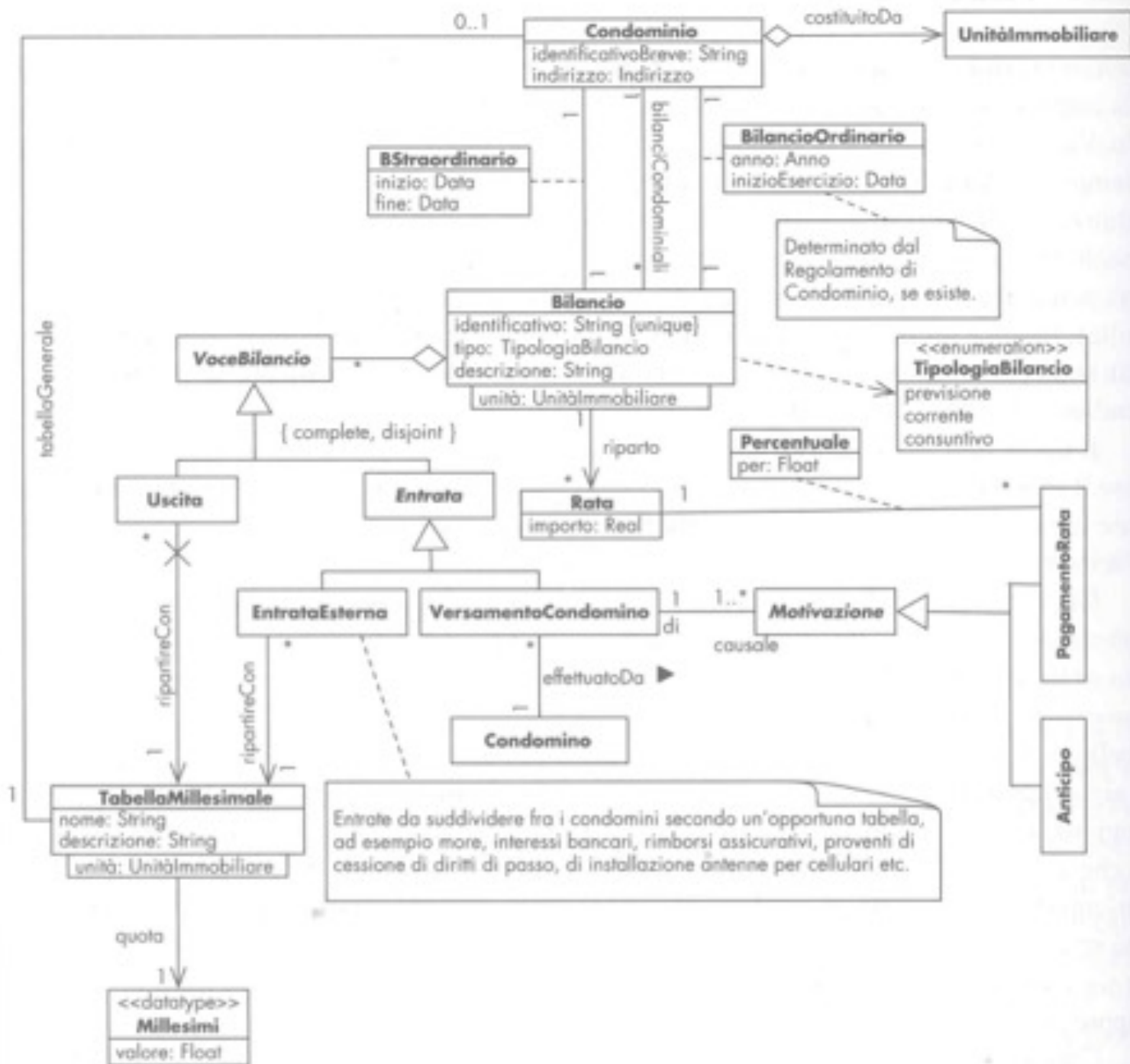
Perché studiare UML

- **Uno Standard Diffuso**
 - è uno standard riconosciuto a livello globale
 - Svolge il ruolo di "lingua franca" nel mondo dell'ICT, facilitando la comunicazione tra team e stakeholders
- **Strumento di Comunicazione Visiva**
 - Un diagramma UML può spiegare concetti complessi in modo molto più efficace rispetto a una descrizione verbale o scritta
 - "Un'immagine vale più di mille parole" - UML aiuta a visualizzare e chiarire i requisiti e le architetture
- **Richiesto dall'Industria**
 - Molto usato a livello industriale, soprattutto per la fase di schizzo dei progetti
 - Richiesto in alcuni bandi pubblici e contratti industriali
- **Supporto alla Progettazione Dettagliata**
 - UML fornisce un modello completo che rende la fase di implementazione più facile e riduce gli errori nel codice

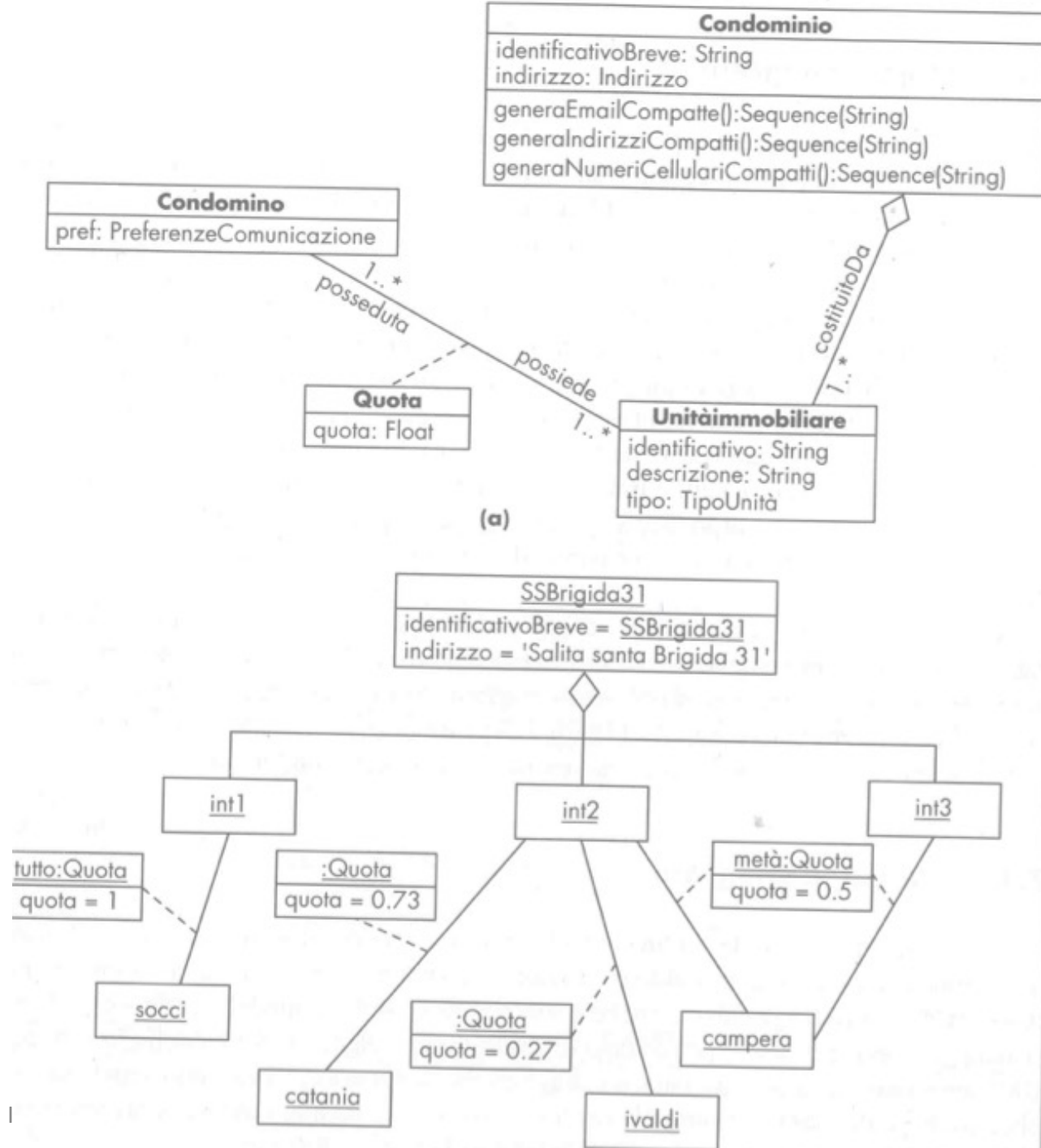
Costrutti base di UML



Class diagram che mostra i vari tipi di associazioni



Class diagram e object diagram



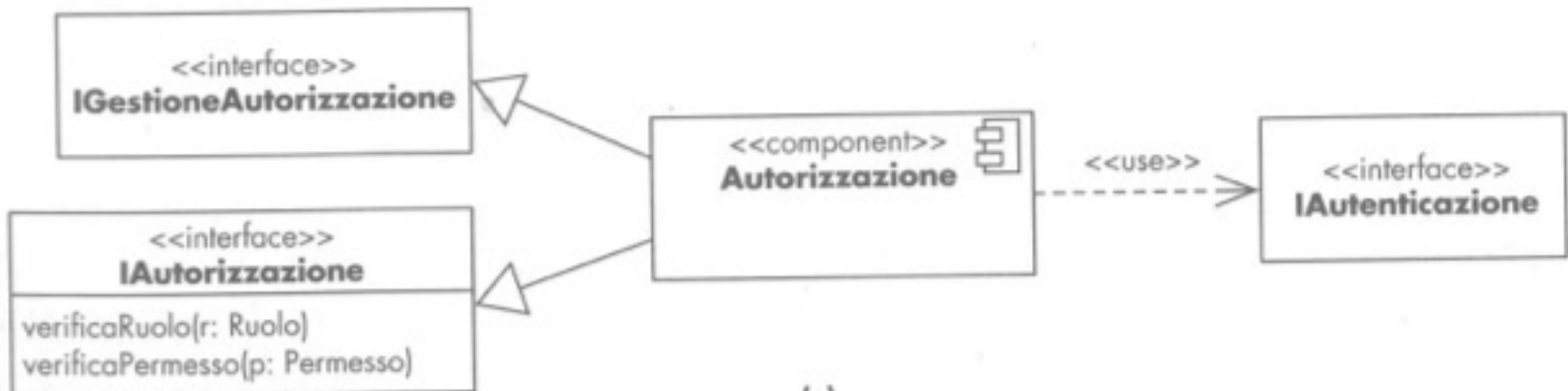
Component diagram



(a)

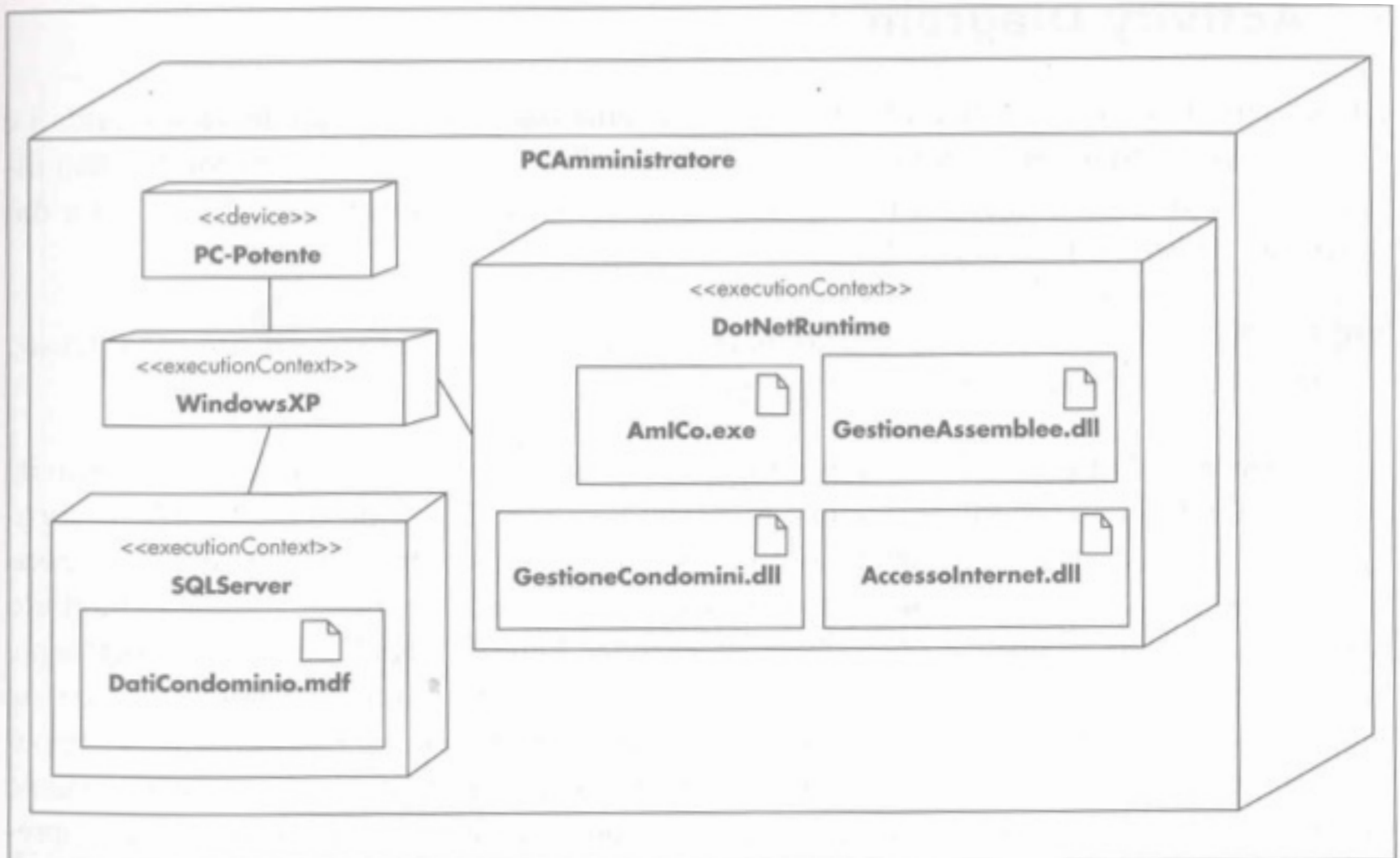


(b)

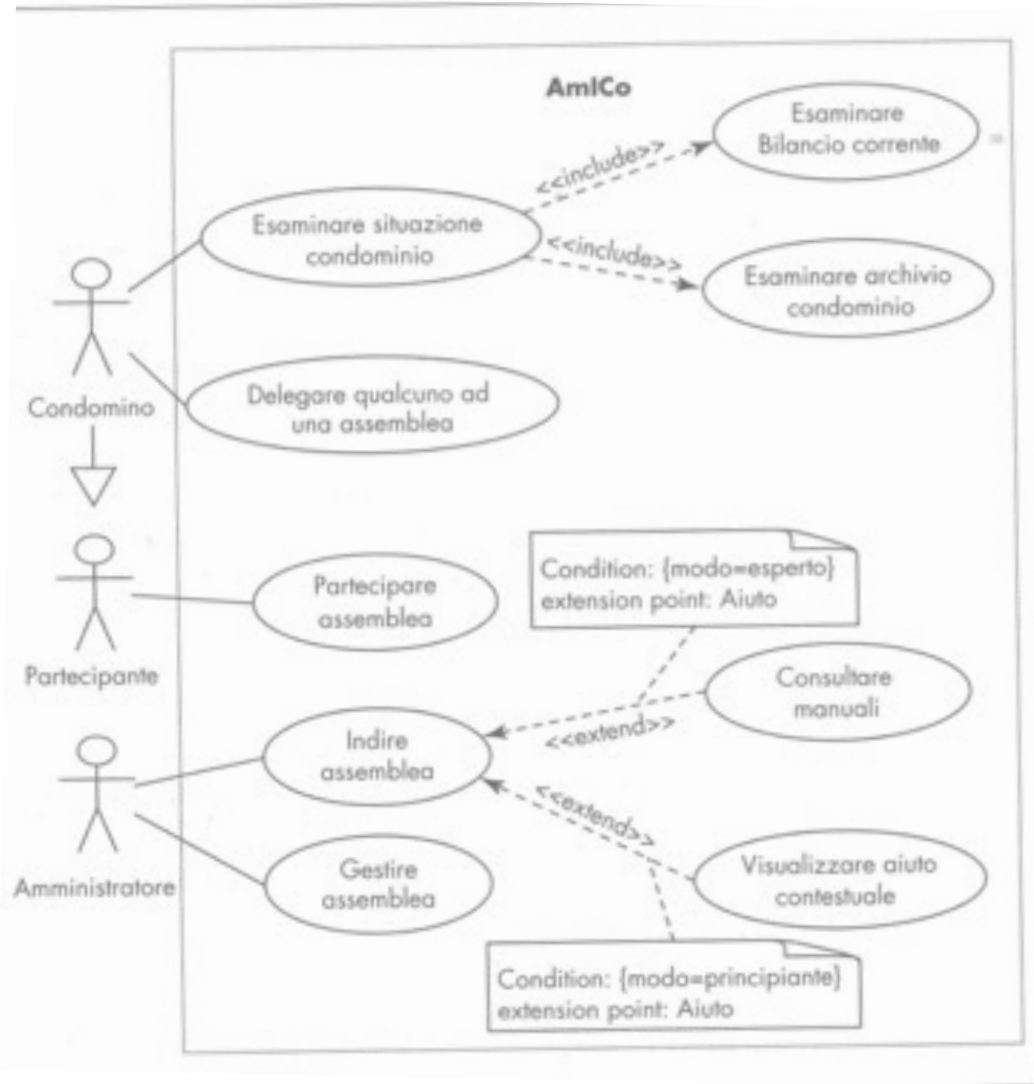


(c)

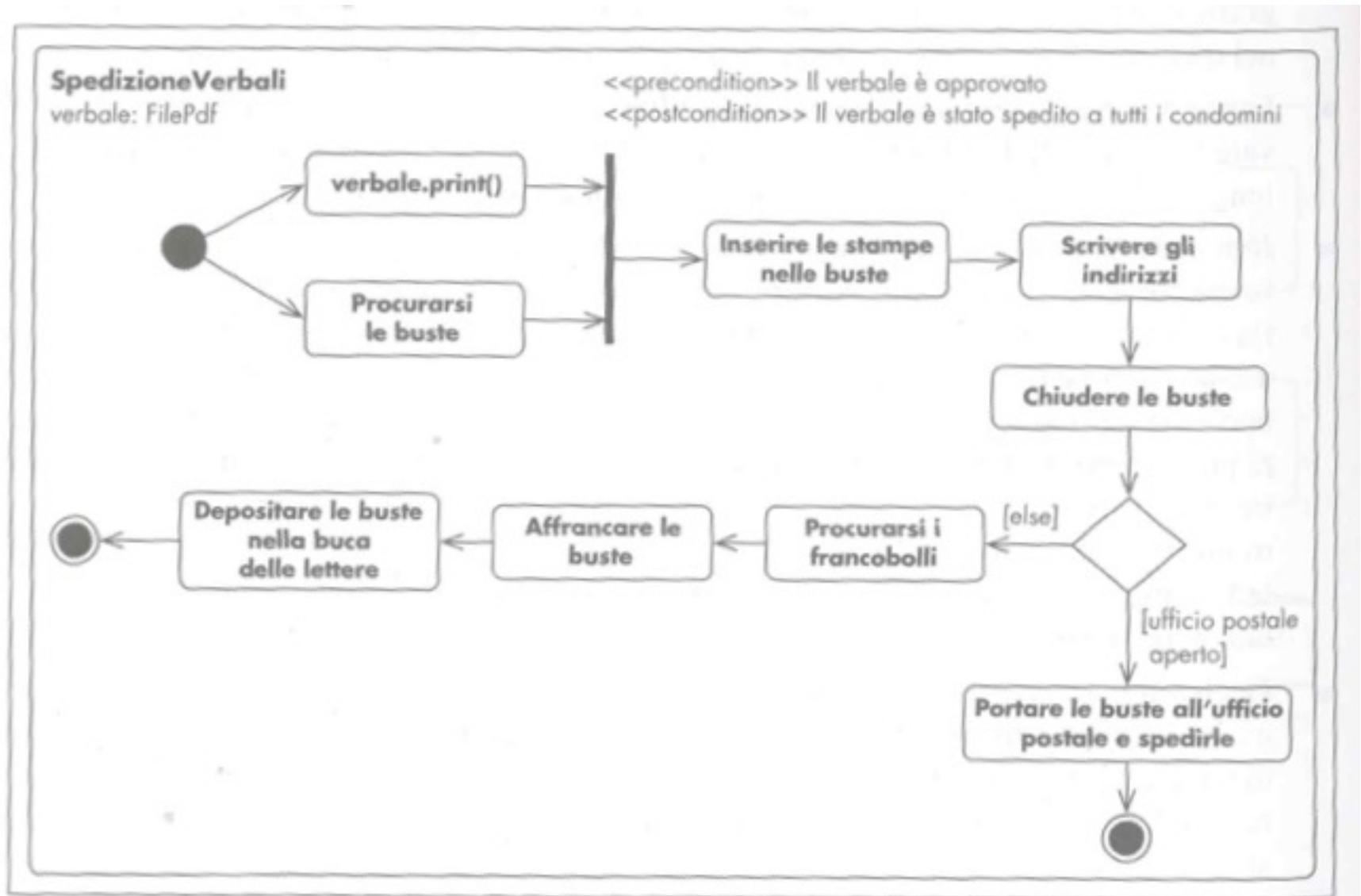
Deployment diagram



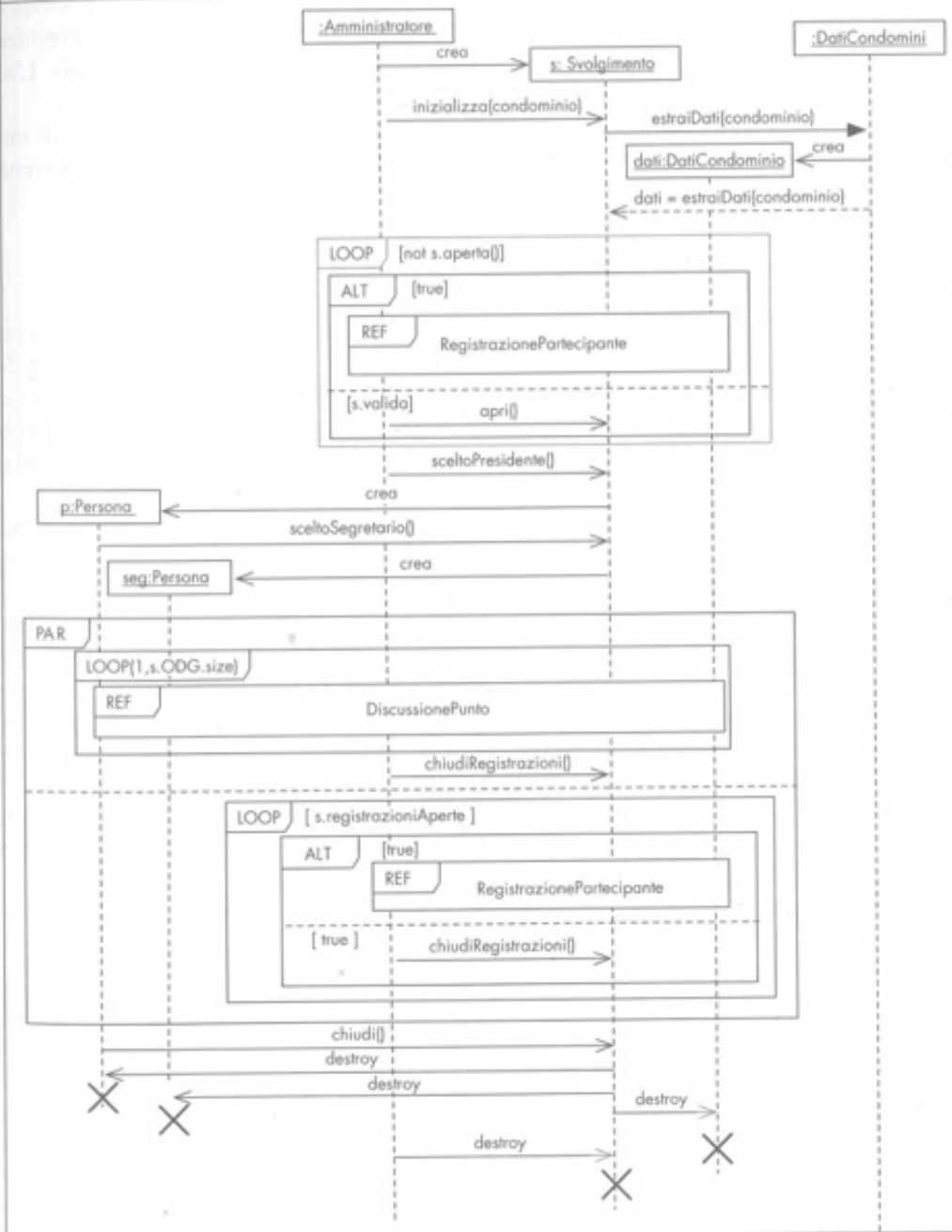
Use case diagram



Activity diagram



```
sd SvolgimentoAssemblea [condominio:Condominio]
```



Il linguaggio OCL

- Object Constraint Language
 - Linguaggio puramente funzionale per scrivere espressioni e vincoli sui modelli object-oriented
 - Proposto come estensione di UML (OMG)
- Linguaggio di query
 - Valutazione di espressioni (valore)
 - Una espressione è una specifica o riferimento ad un valore
 - Un vincolo è una restrizione su uno o più valori o parti di un modello orientato agli oggetti.
 - Es. nel caso di tipo booleano, un'espressione può essere utilizzata per esprimere un constraint, richiedendo che il suo valore sia sempre vero
 - Utilizzo di espressioni OCL come note nel diagramma (contesto)

OCL: tipi di espressioni

- Le espressioni possono essere usate in diversi diagrammi UML:
 - per specificare un **valore iniziale** di un attributo o associazione
 - specificare **valori derivati** di un attributo o associazione
 - per specificare il **corpo di un'operazione**
 - per indicare un'**istanza in un diagramma dinamico**
 - per indicare una **condizione in un diagramma dinamico**
 - per indicare il **valore attuale di un parametro** in un diagramma dinamico

OCL: Tipi di vincoli

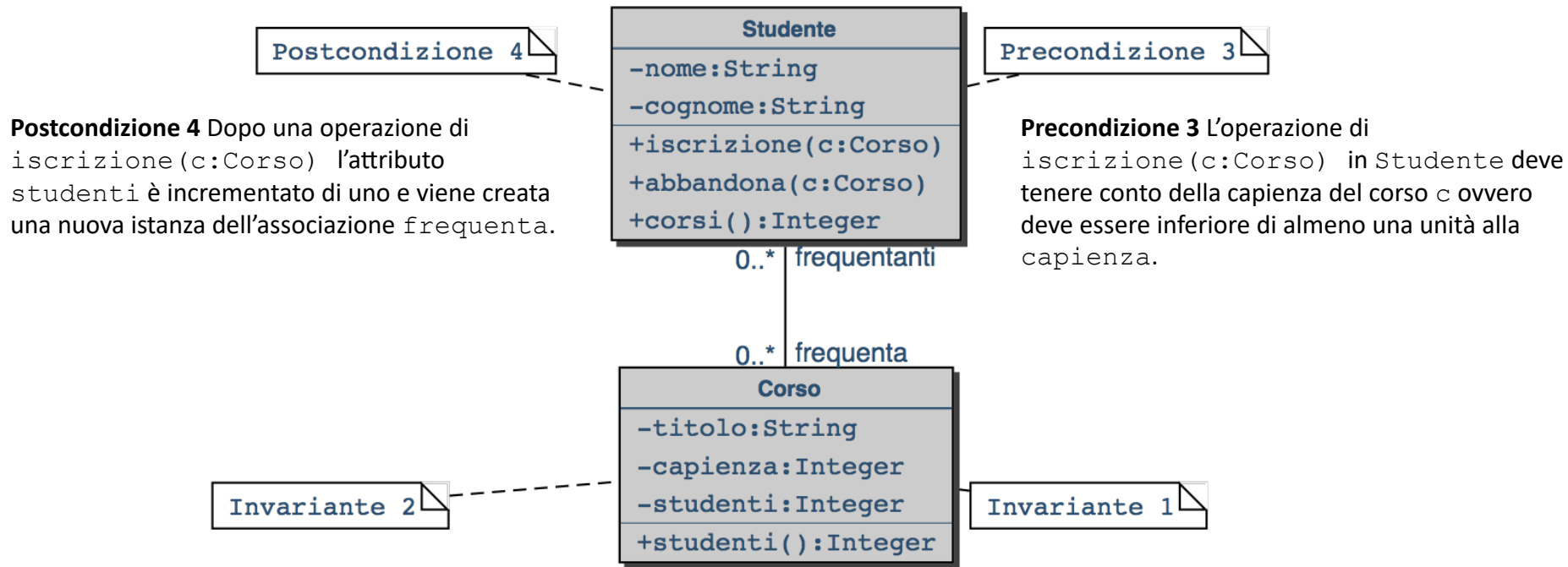
- Vi sono quattro tipi principali di vincoli:
 - Un'**invariante** è un vincolo che stabilisce una condizione che deve essere sempre valida per tutte le istanze di una classe o interfaccia. Un'invariante è descritta usando una espressione logica che valutata è sempre vera
 - una **precondizione** per una operazione è una restrizione che deve essere vera prima dell'esecuzione dell'operazione
 - una **postcondizione** per una operazione è una restrizione che deve essere vera subito dopo l'esecuzione dell'operazione
 - una **guardia** è un vincolo che deve essere vero in una fase della vita del sistema. Tipi di guardie sono `init:` e `derive:`

OCL: contesto di un'espressione

- Specifica la parte del modello su cui agisce l'espressione
 - generalmente è una classe, una interfaccia, un tipo di dato, un componente
- Molte volte la parte interessata del modello è una operazione o un attributo, raramente un'istanza
 - Il contesto indica sempre l'elemento sintattico del modello generalmente definito in UML. Questo elemento è chiamato contesto dell'espressione.
- Successivamente è importante il tipo di contesto dell'espressione. L'espressione viene valutata sempre sugli oggetti (istanze) per cui spesso viene indicato esplicitamente l'istanza mediante la parola chiave **self**.

OCL: esempio (1)

Modello degli studenti che frequentano corsi



Postcondizione 4 Dopo una operazione di `iscrizione(c:Corso)` l'attributo `studenti` è incrementato di uno e viene creata una nuova istanza dell'associazione `frequenta`.

Precondizione 3 L'operazione di `iscrizione(c:Corso)` in `Studente` deve tenere conto della capienza del corso `c` ovvero deve essere inferiore di almeno una unità alla capienza.

Invariante 2 Per ogni istanza della classe `Corso` il numero delle istanze dell'associazione `frequentatati` deve essere uguale a `studenti`.

Invariante 1 Per ogni istanza della classe `Corso` l'attributo `capienza` deve essere sempre maggiore o uguale a `studenti`

OCL: esempio (2)

-- Invariante 1

```
context Corso
inv: capienza >= frequentanti.size()
```

-- Invariante 2

```
context Corso
inv: studenti = frequentanti.size()
```

-- Pre e post condizione 3 e 4

```
context Studente::iscrizione(c:Corso)
pre: c.capienza > c.studenti
post: c.studenti = c.studenti@pre+1
      and c.frequentanti = c.frequentanti@pre+1
```