Introduzione a UML

obiettivo di questa introduzione

- fornire elementi di base su UML
- introdurre i principali diagrammi
- fornire indicazioni sulle modalità di utilizzo di UML nello sviluppo delle applicazioni

- questi temi sono trattati in modo approfondito, con esercitazioni, nel corso “Sviluppo di applicazioni con UML”

http://www.analisi-disegno.com/a_comai/corsi/sk_uml.htm

unified modeling language (UML)

- linguaggio (e notazione) universale, per rappresentare qualunque tipo di sistema (software, hardware, organizzativo, …)
- standard OMG (Object Management Group), dal nov.1997
- autori:
  - Grady Booch
  - Ivar Jacobson
  - Jim Rumbaugh

cos'è UML (e cosa non è)

- è un linguaggio di rappresentazione dei sistemi, non di programmazione (come Java, VisualBasic, C++, …)
- serve a progettare un nuovo sistema, o per documentarne uno esistente, senza perdersi nei dettagli dei linguaggi di programmazione
- è universale: può rappresentare sistemi eterogenei per architettura (dai web ai "legacy"), per tecnologie, per tipologia applicativa (gestionale, real-time)

...cos'è UML (e cosa non è)

- è un linguaggio, non un metodo completo
- notazione e semantica sono standard
- ma UML non è legato ad uno specifico processo, e non fornisce indicazioni sul proprio utilizzo
- quindi può essere (ed è) utilizzato da persone e gruppi che seguono approcci diversi (è "indipendente dai metodi")

UML come standard

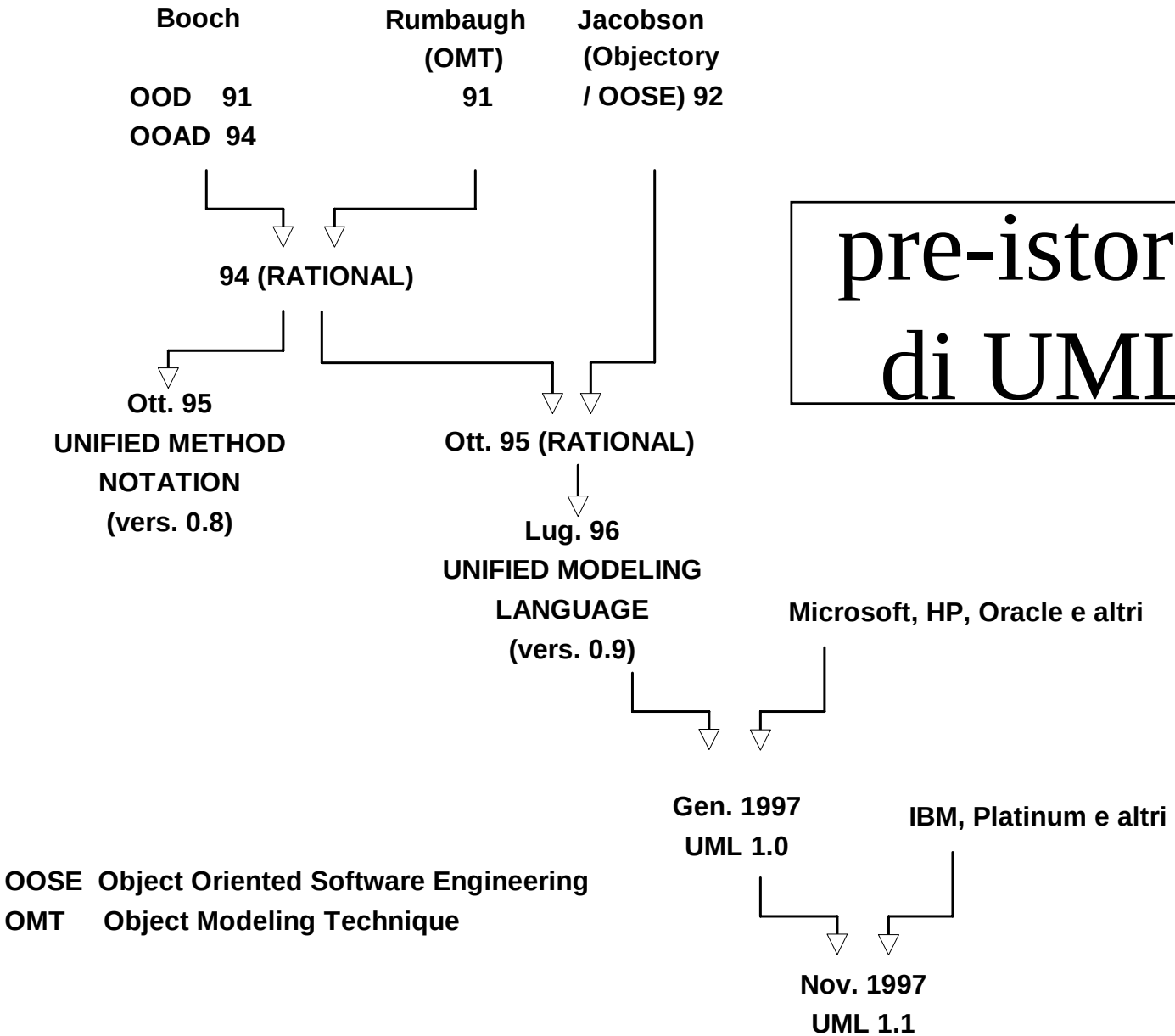
- i suoi autori (Booch, Rumbaugh e Jacobson) non hanno il copyright su UML
- la versione diventata standard OMG ha ricevuto i contributi di molti altri metodologi, e delle più importanti società di software mondiali
- la sua evoluzione è a carico dell'OMG, e soggetta a procedure ben definite per ogni cambiamento

  - versione attuale: **1.4**
  - documenti ufficiali: www.omg.org

nuove tecniche e diagrammi ?

- UML è un’evoluzione di modelli e diagrammi preesistenti, non una rivoluzione
- affinità con altri diagrammi molto noti:
  - Entity - Relationship
  - Flow Chart
  - diagrammi di stato
  - varie notazioni object oriented

pre-istoria di UML



storia di UML

novembre 1997: versione 1.1
dicembre 1998: versione 1.2
giugno 1999: versione 1.3
maggio 2001: versione 1.4

? 2002 : versione 2.0

UML: meta-modello e diagrammi

- UML è basato su un meta-modello integrato, composto da numerosi elementi, collegati tra loro secondo regole precise
- utilizzando gli elementi del meta-modello è possibile creare i modelli per i sistemi da rappresentare
- molti elementi hanno una icona che li rappresenta graficamente
- gli elementi del meta-modello possono comparire in diagrammi di diverso tipo
- le regole permettono verifiche di correttezza

diagrammi UML

diagrammi “logici”:

- diagramma dei casi d’uso (use case)
- diagramma delle classi (class)
- diagramma di sequenza (sequence)
- diagramma di collaborazione (collaboration)

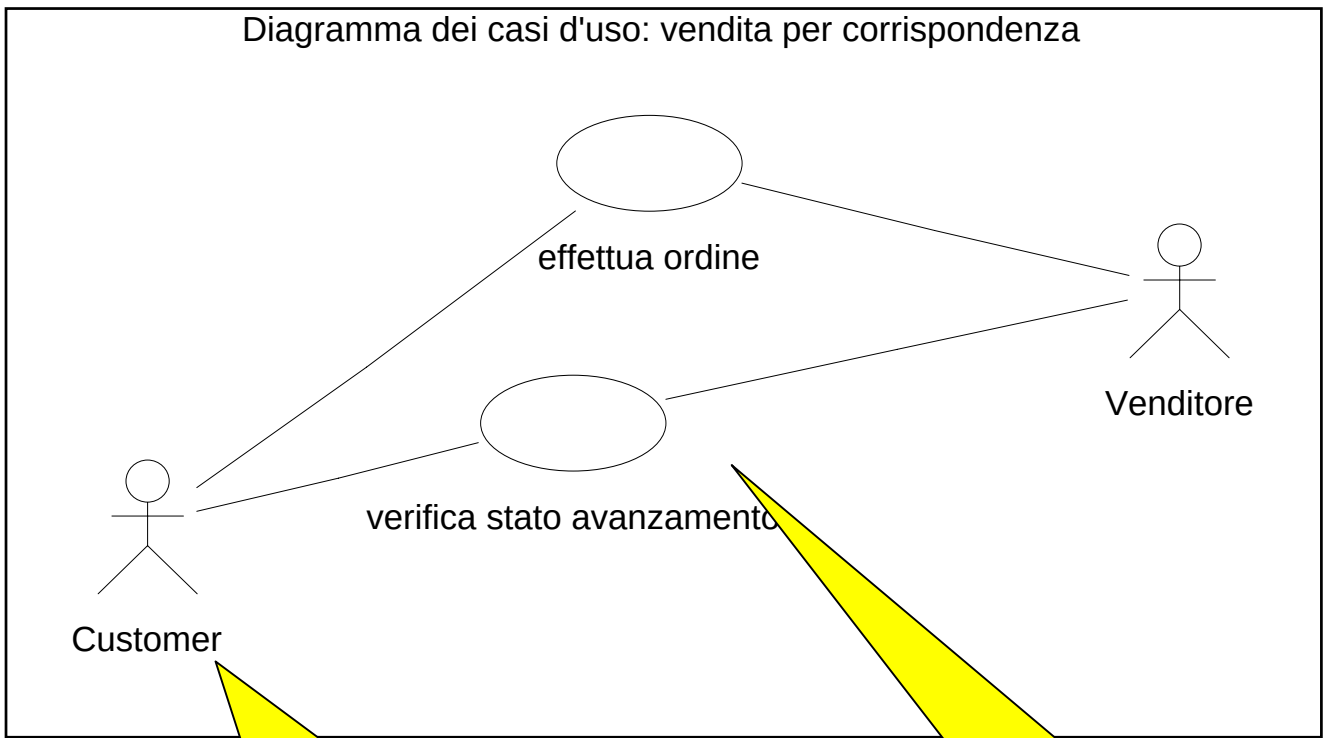
} interaction

- diagramma di stato (statechart)
- diagramma delle attività (activity)

diagrammi di implementazione (implementation) :

- diagramma dei componenti (component)
- diagramma di distribuzione (deployment)

diagramma dei casi d'uso



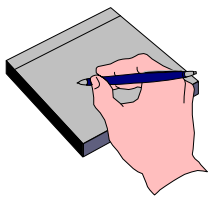
attore: un soggetto esterno al sistema (essere umano, altro sistema, …)

caso d'uso: una particolare modalità di utilizzo del sistema

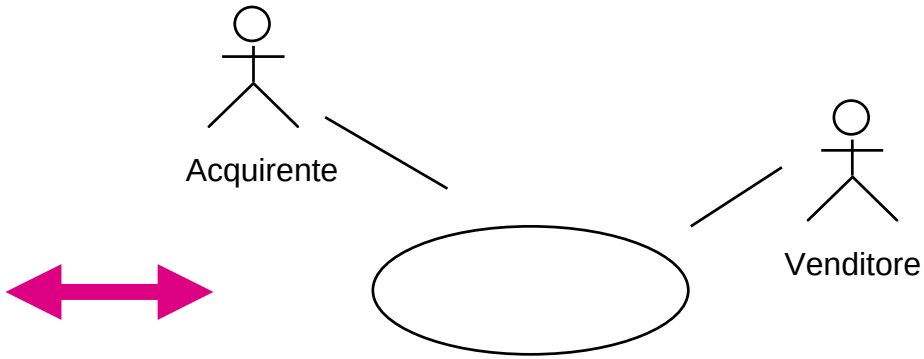
casi d’uso : a cosa servono

- rappresentano le modalità di utilizzo del sistema da parte di uno o più utilizzatori (attori)
- descrivono l’interazione tra attori e sistema, senza rivelare l’organizzazione interna del sistema
- sono espressi in forma testuale, comprensibile anche per i non “addetti ai lavori”
- possono essere definiti a livelli diversi (sistema o parti del sistema)
- ragionare sui casi d’uso aiuta a scoprire i requisiti

ruolo dei casi d'uso

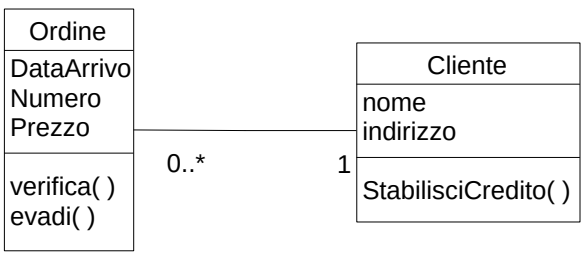


requisiti



caso d'uso:
effettua ordine

componenti software



casi prova

unità di rilascio

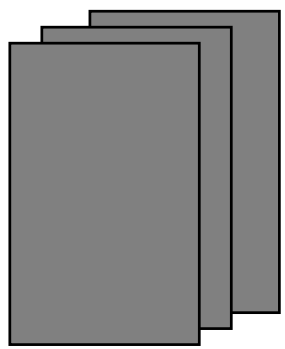


diagramma delle classi

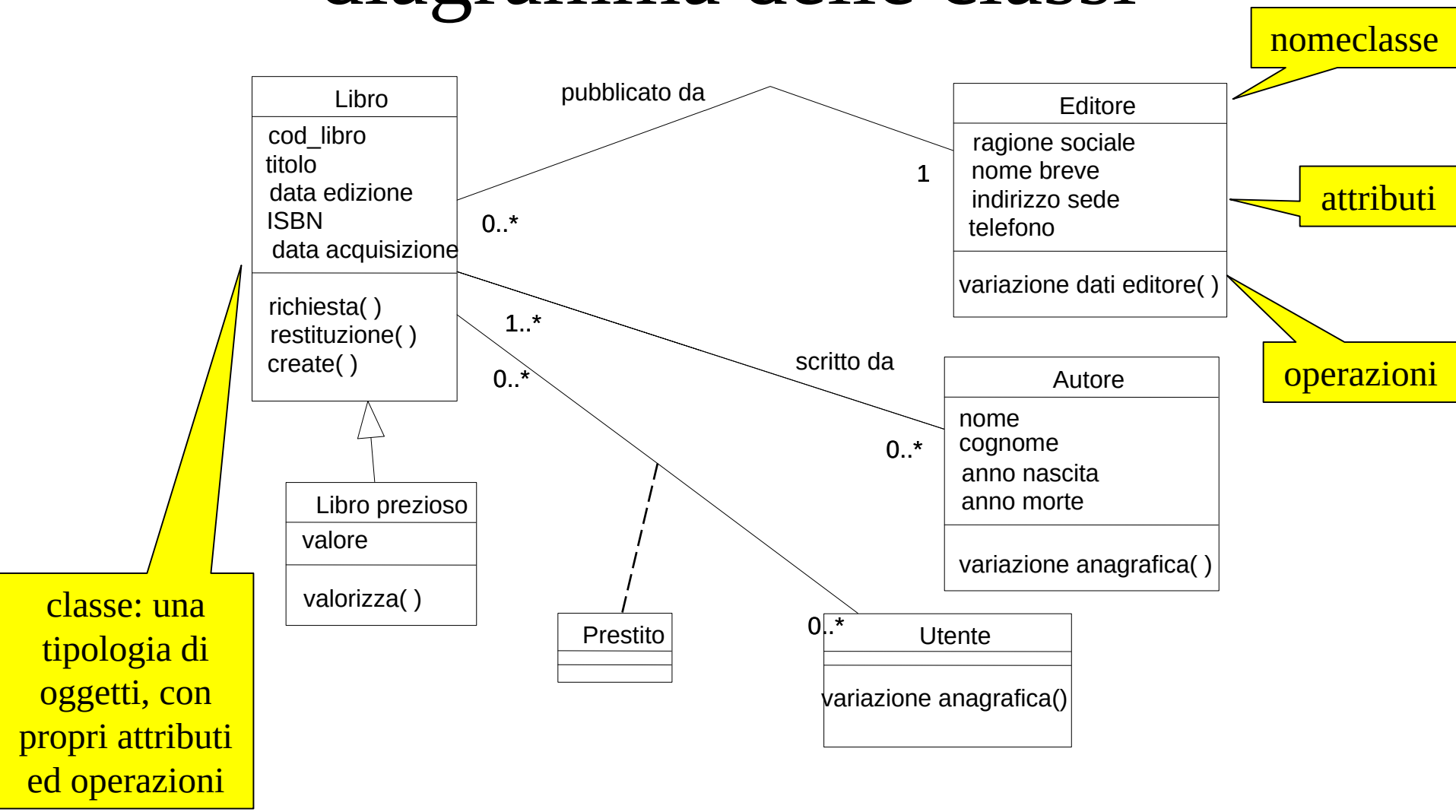


diagramma delle classi

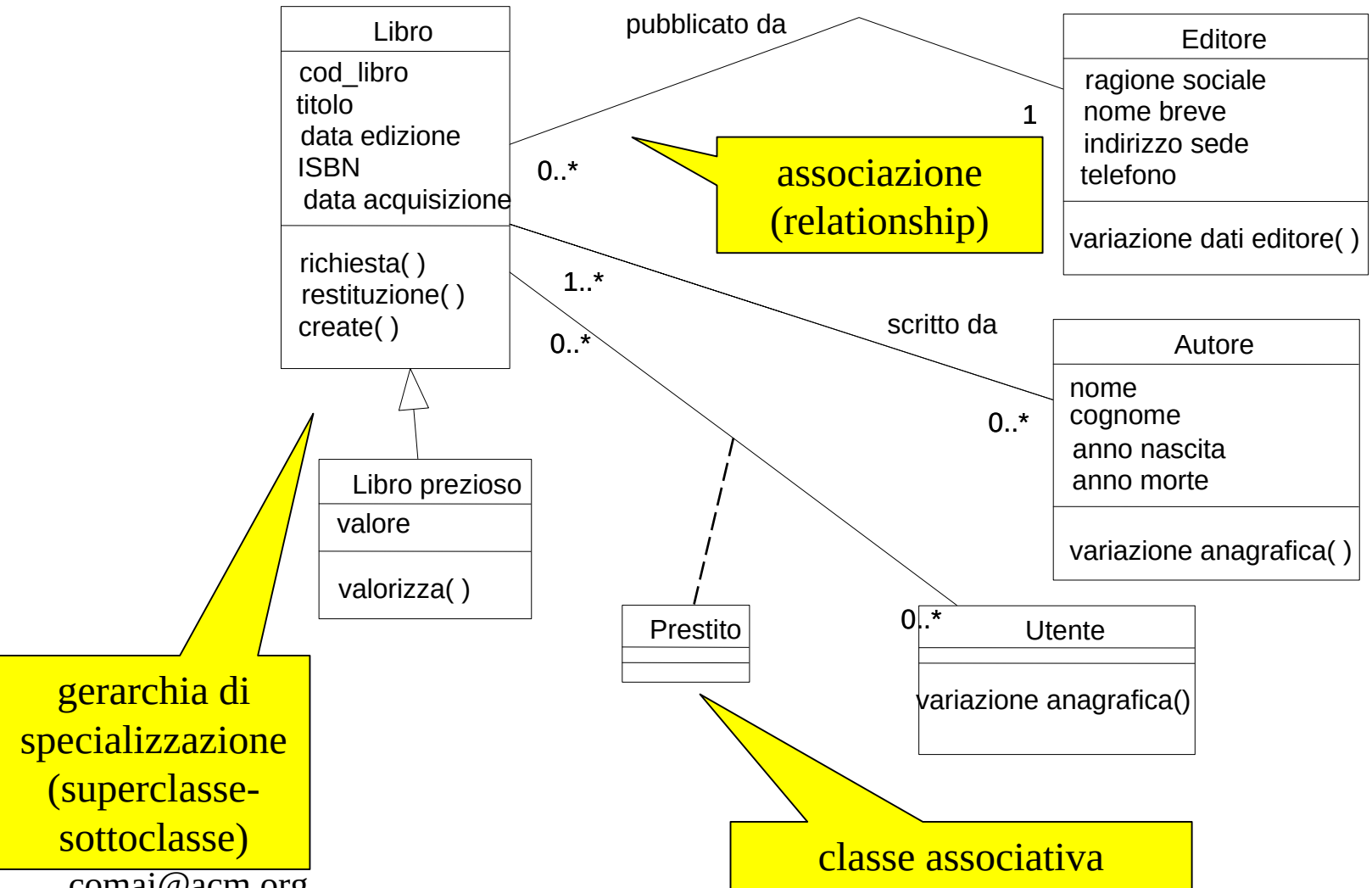


diagramma delle classi: a cosa serve

- rappresenta le classi e gli oggetti che compongono il sistema, ed i relativi attributi ed operazioni
- specifica, mediante le associazioni, i vincoli che legano tra loro le classi
- può essere definito a livelli diversi (analisi, disegno)
- può rappresentare diverse tipologie di oggetti (boundary, control, entity …)

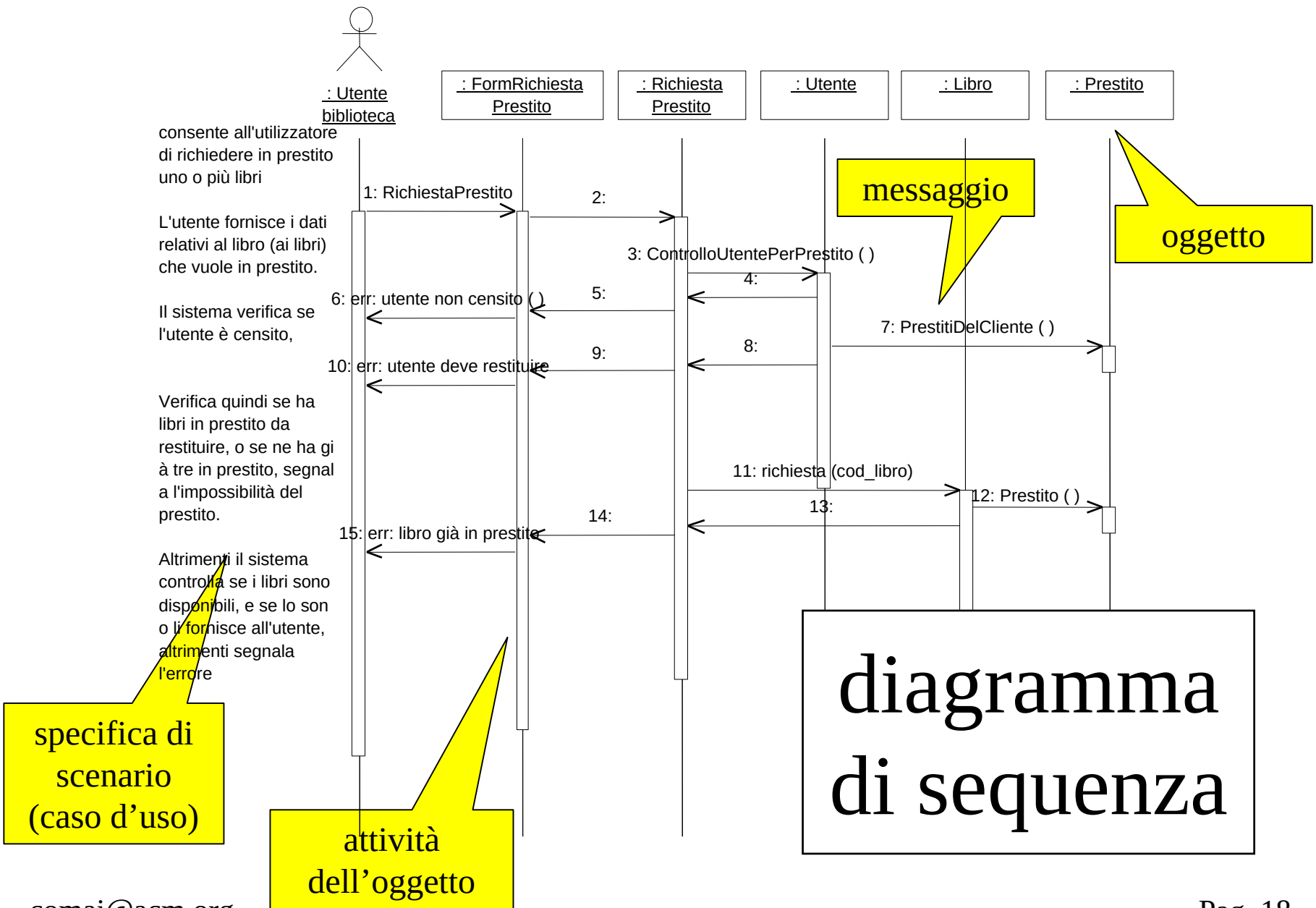


diagramma di sequenza: a cosa serve

- evidenzia il modo in cui uno scenario (uno specifico percorso in un caso d'uso) viene risolto dalla collaborazione tra un insieme di oggetti
- specifica la sequenza dei messaggi che gli oggetti si scambiano
- può specificare nodi decisionali e iterazioni
- diagrammi di sequenza e diagrammi di collaborazione esprimono informazioni simili, ma le evidenziano in modo diverso

diagramma di collaborazione

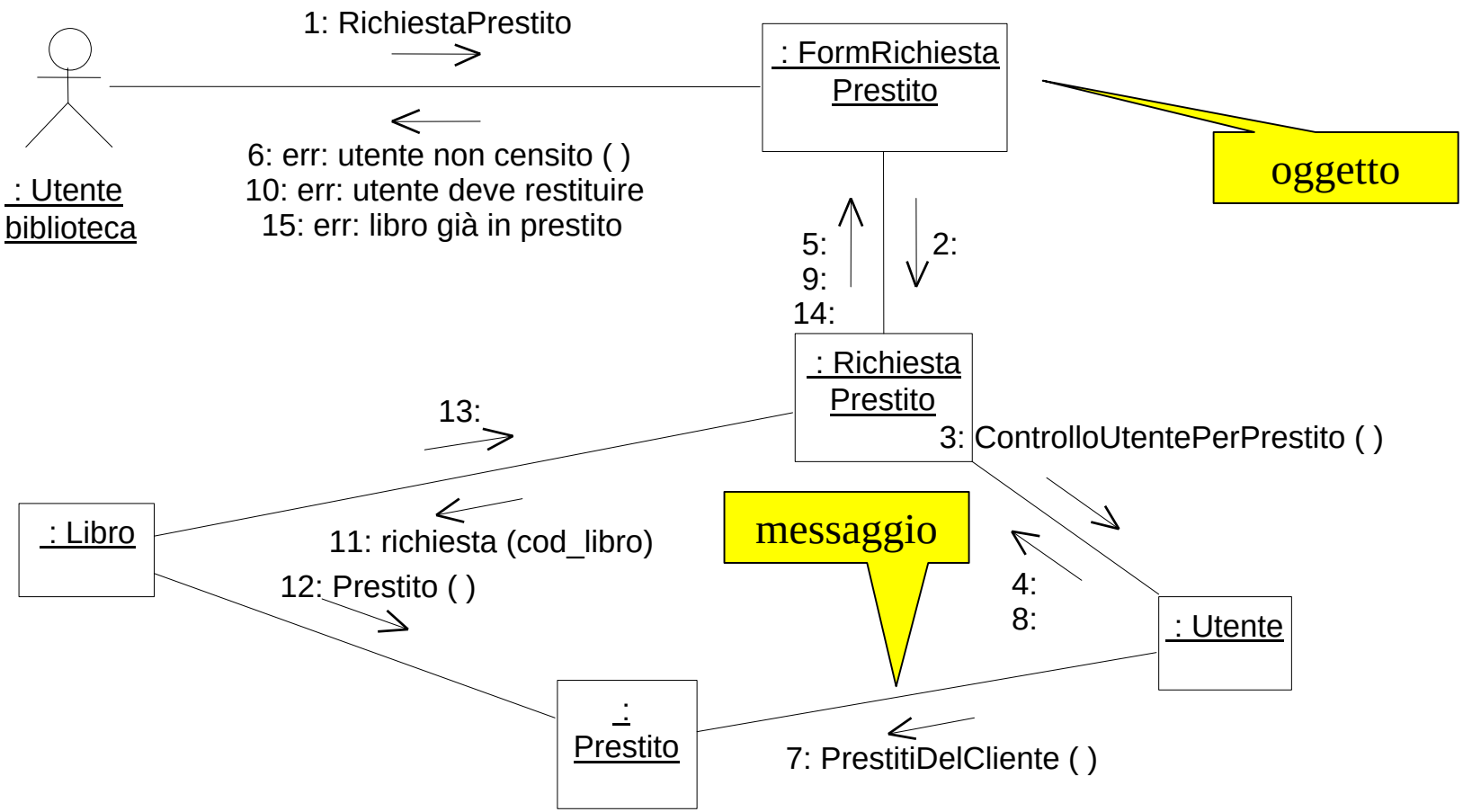


diagramma di collaborazione: a cosa serve

- specifica gli oggetti che collaborano tra loro in un dato scenario, ed i messaggi che si indirizzano
- la sequenza dei messaggi è meno evidente che nel diagramma di sequenza, mentre sono più evidenti i legami tra gli oggetti
- può essere utilizzato a livelli diversi (analisi, disegno), e rappresentare diverse tipologie di oggetti

diagramma di stato

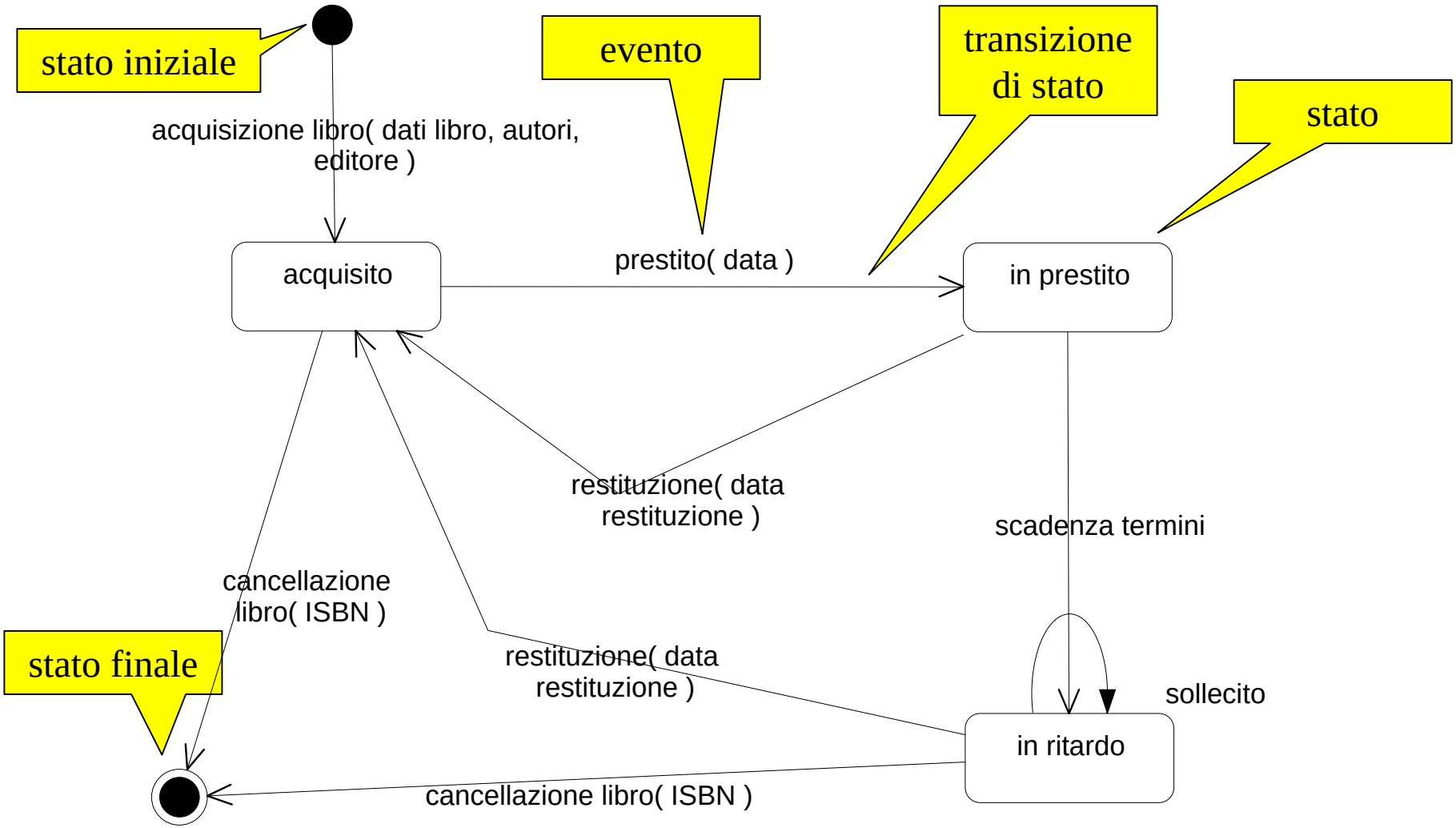


diagramma di stato: a cosa serve

- specifica il ciclo di vita degli oggetti di una classe, definendo le regole che lo governano
- quando un oggetto si trova in un certo stato può essere interessato da determinati eventi (e non da altri)
- come risultato di un evento l'oggetto può passare ad un nuovo stato (transizione)

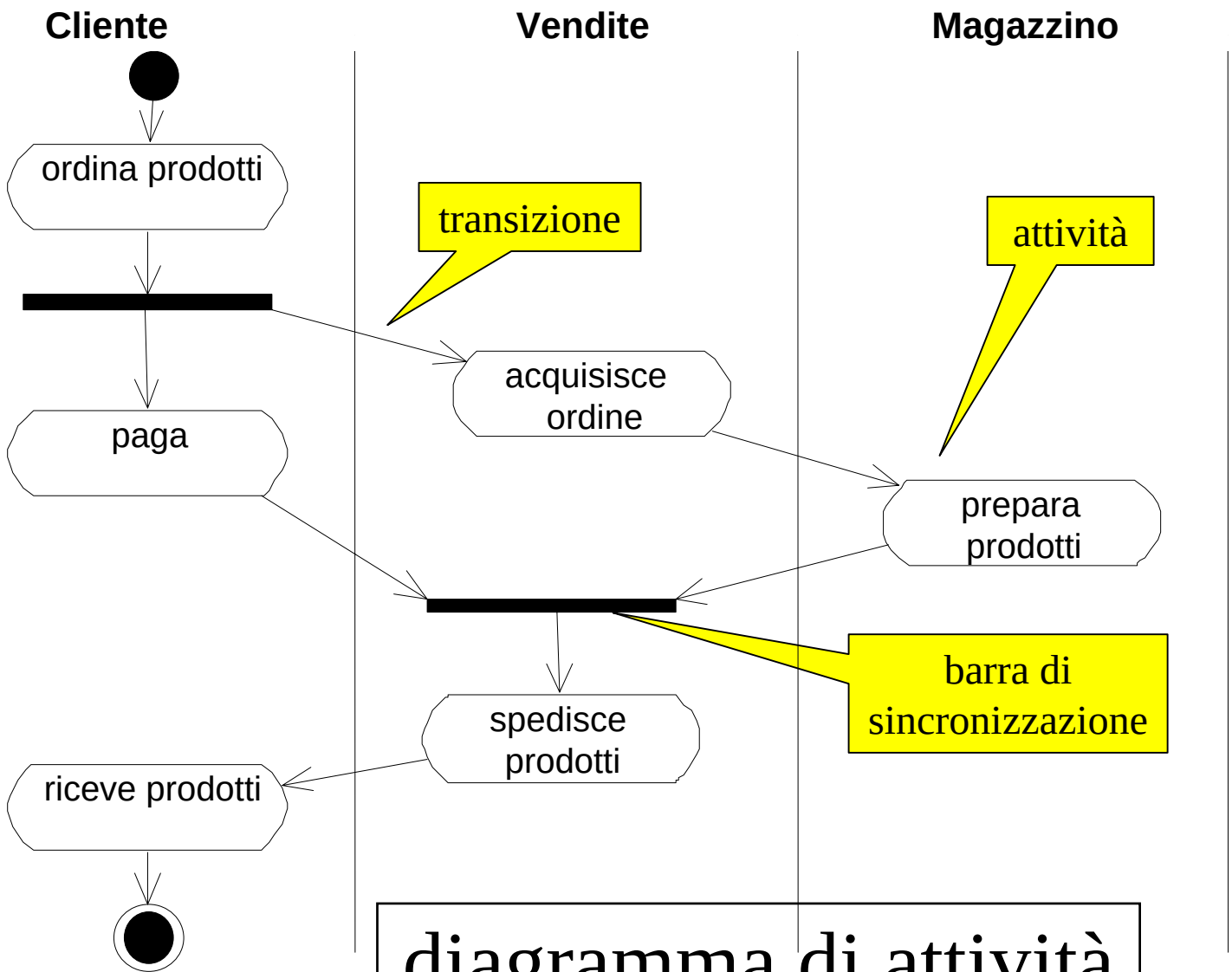
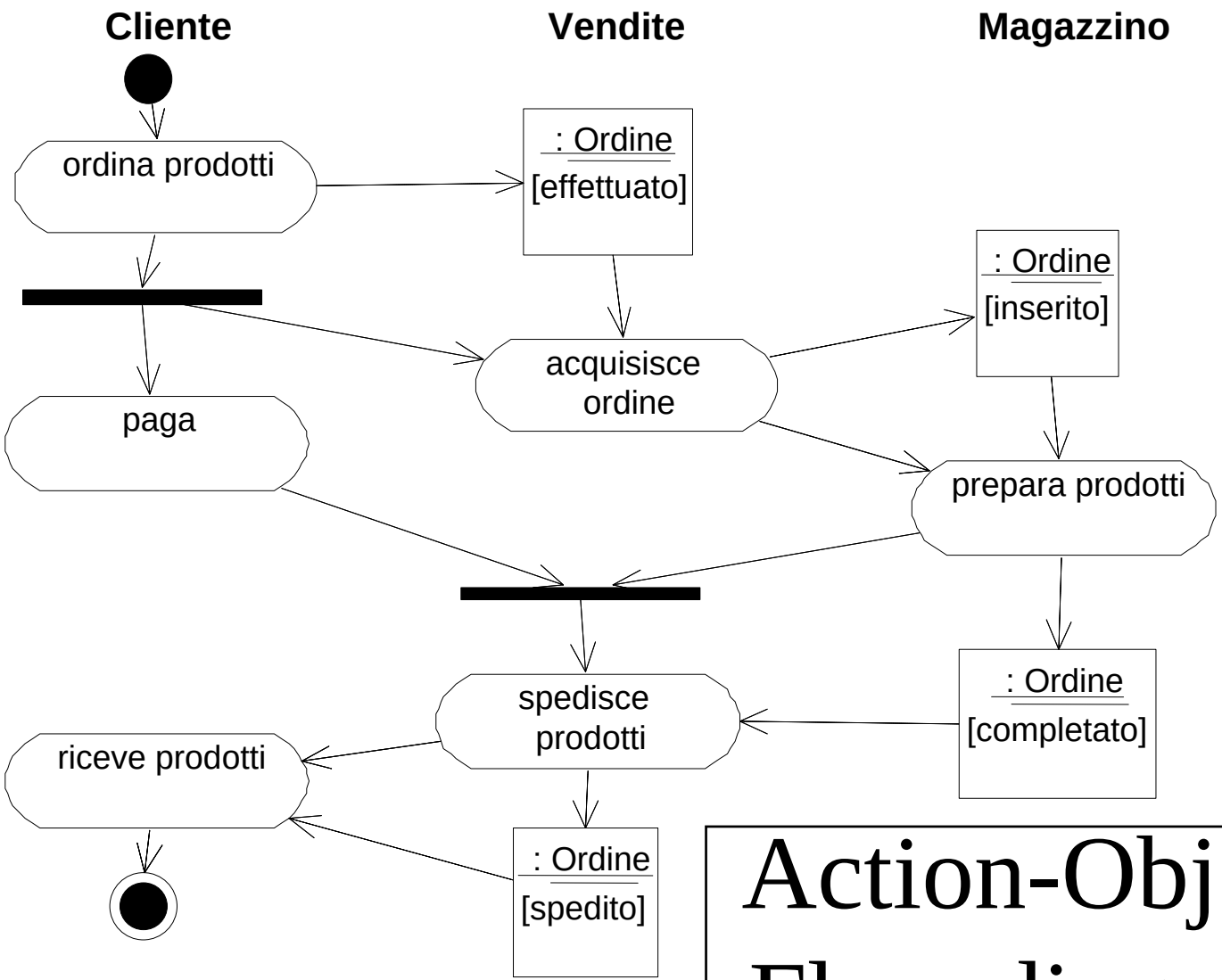


diagramma di attività: a cosa serve

- a rappresentare sistemi di workflow, oppure la logica interna di un processo (di qualunque livello, dai business process ai processi di dettaglio)
- permette di rappresentare processi paralleli e la loro sincronizzazione
- è un caso particolare di diagrammi di stato, in cui ogni stato è uno stato di attività



Action-Object Flow diagram

diagramma di flusso azione - oggetto: a cosa serve

- a rappresentare le interazioni tra processi e oggetti
- è un caso particolare di diagramma di attività
- è un vero e proprio flow chart

diagramma dei componenti

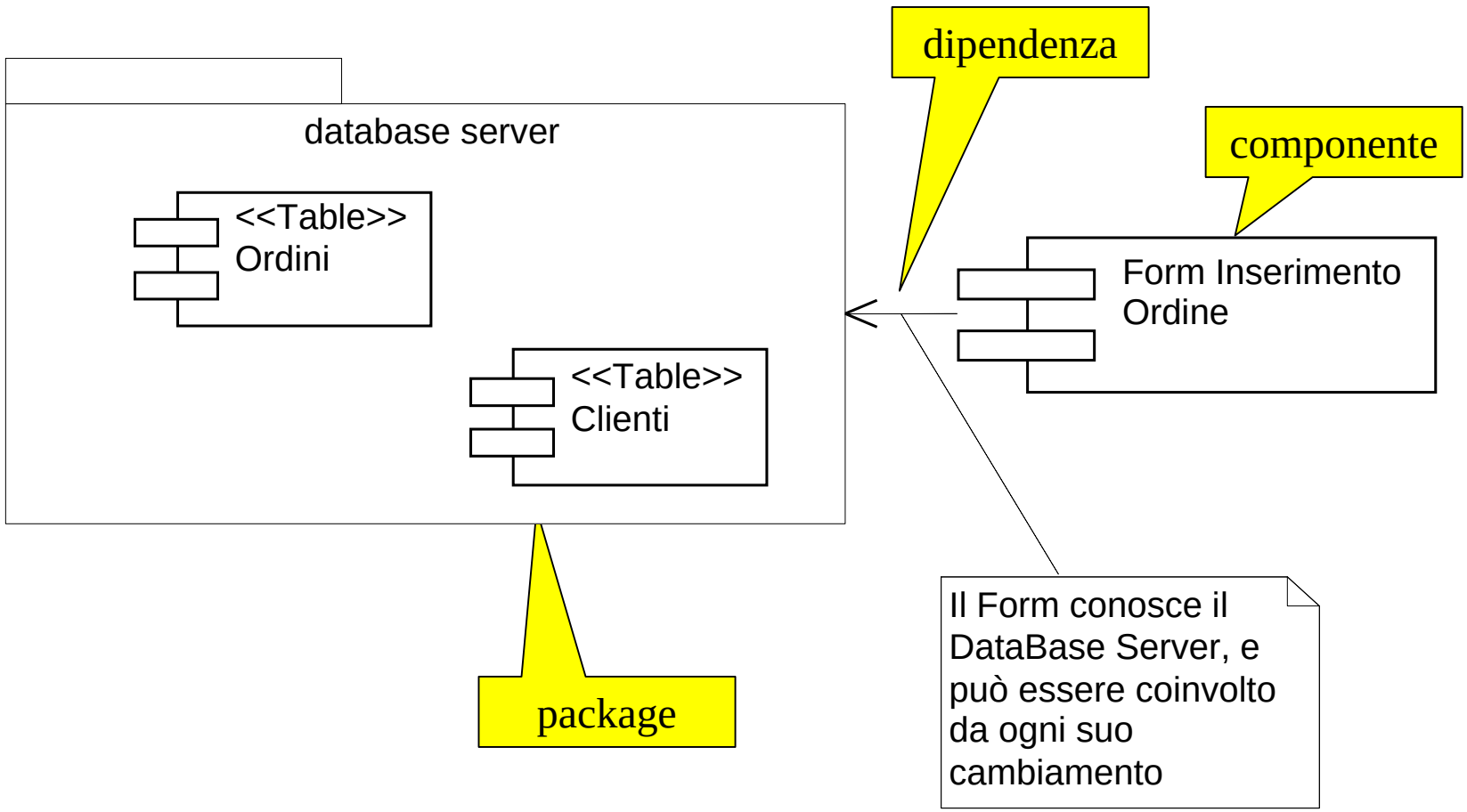


diagramma dei componenti: a cosa serve

- evidenzia l'organizzazione e le dipendenze esistenti tra componenti
- i componenti sono moduli software, dotati di identità e con un'interfaccia ben specificata
- i componenti (come a livello logico le classi) possono essere raggruppati in package

diagramma di distribuzione

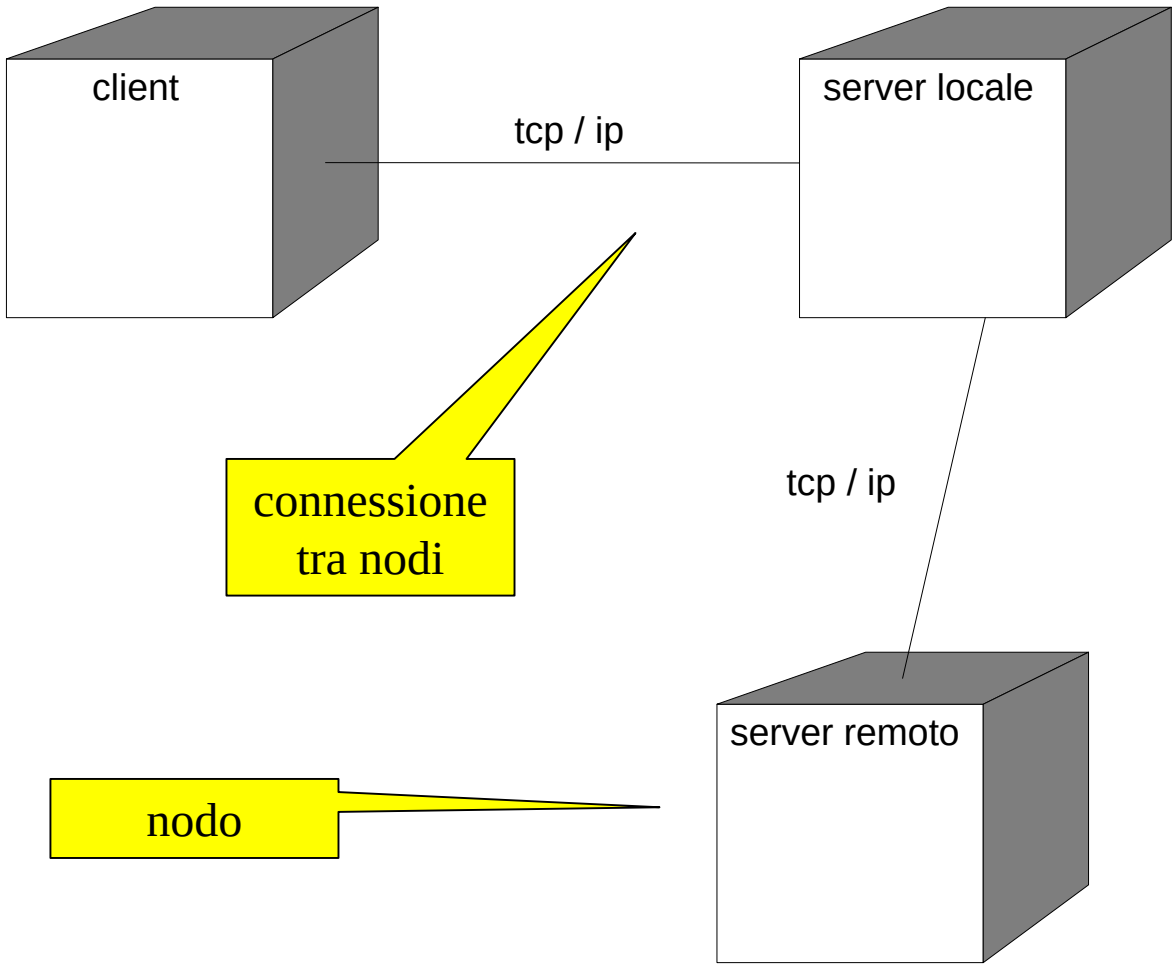


diagramma di distribuzione: a cosa serve

- evidenzia la configurazione dei nodi elaborativi in ambiente di esecuzione (run-time), e dei componenti, processi ed oggetti ubicati in questi nodi
- permette di rappresentare, a diversi livelli di dettaglio, l’architettura fisica del sistema

benefici portati da UML

- superamento della "guerra dei metodi"
  - prima di UML, vi erano decine (o centinaia?) di metodi di analisi e disegno OO proposti e praticati
  - oggi, notazione e semantica comuni a livello internazionale
- risposta ai problemi legati allo sviluppo di sistemi complessi con ambienti visuali
  - maggiore attenzione alla modellazione degli aspetti architetturali
  - il meta-modello comune favorisce le possibilità di comunicazione tra gli strumenti utilizzabili dai progettisti nello sviluppo

UML è complesso

- intende rappresentare qualunque tipo di sistema software, a livelli di astrazione differenziati
- il numero degli elementi è elevato, e in molti casi è possibile scegliere tra forme di rappresentazione diverse
- UML non suggerisce, né tantomeno prescrive una sequenza di realizzazione dei diversi diagrammi
- offre un’ampia gamma di possibili modalità di utilizzo, tra le quali i progettisti sono liberi di scegliere

UML va adattato alle proprie esigenze

tra i fattori da considerare:

- settore di attività (es. militare, finanziario)
- tipologia di progetto (rischio, complessità)
- esigenze di conformità a norme e standard
- comunicazione con committenti e stakeholders
- comunicazione con fornitori
- composizione e distribuzione del gruppo di lavoro

$\Rightarrow$ non ha senso che tutti usino UML nello stesso modo

UML in sintesi

- è uno standard: uniformità nei concetti e nelle notazioni utilizzate, interoperabilità tra strumenti di sviluppo, indipendenza dai produttori, dalle tecnologie, dai metodi
- è articolato: può rappresentare qualunque sistema software, a diversi livelli di astrazione
- è complesso: va adattato ("ritagliato") in base alle specifiche esigenze dei progettisti e dei progetti, utilizzando solo ciò che serve nello specifico contesto