Unified Modeling Language

Paolo Costa

Politecnico
di Milano

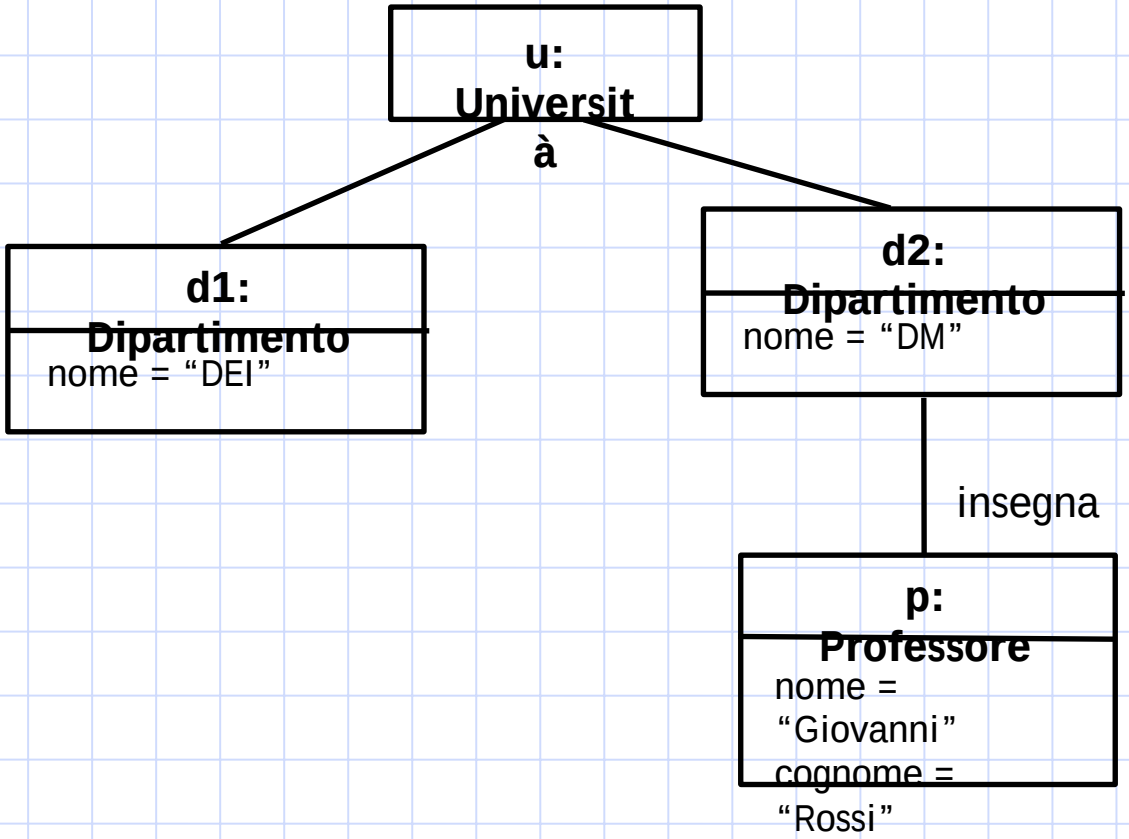
Lucidi tratti dal materiale di Luciano Baresi

Object Diagram

Object Diagram

- Rappresentano le istanze e i legami fra queste
- Si usano durante l'analisi e il progetto
  - Per capire la struttura di oggetti complessi
  - Per esplicitare la struttura di oggetti complessi
  - Per presentare "immagini" del sistema

Esempio

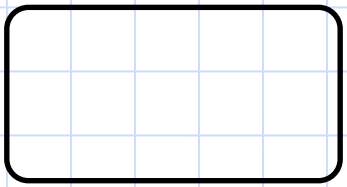

u:
Università
d1:
Dipartimento
nome = “DEI”
d2:
Dipartimento
nome = “DM”
insegna
p:
Professore
nome =
“Giovanni”
cognome =
“Rossi”

Statechart diagram

State Diagram

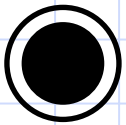
- Rappresentano il comportamento dei singoli oggetti di una classe in termini di
  - Eventi a cui gli oggetti (la classe) sono sensibili
  - Azioni prodotte
  - Transizioni di stato
    - Identificazione degli stati interni degli oggetti
- Possibilità di descrivere evoluzioni parallele
- Sintassi mutuata da StateChart (D. Harel)

Elementi grafici

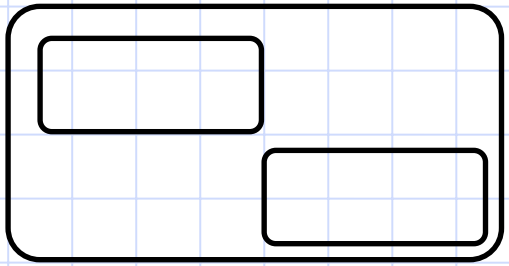


Stato

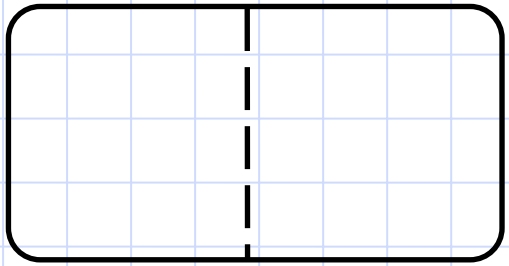
Stato iniziale



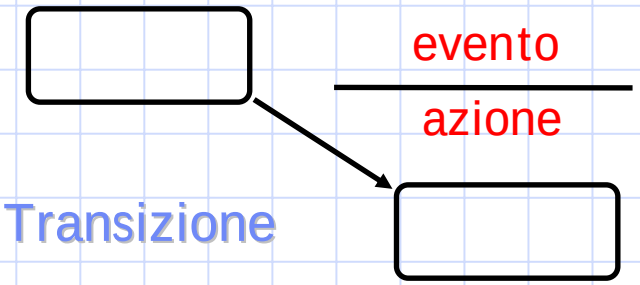
Stato finale



Decomposizione OR

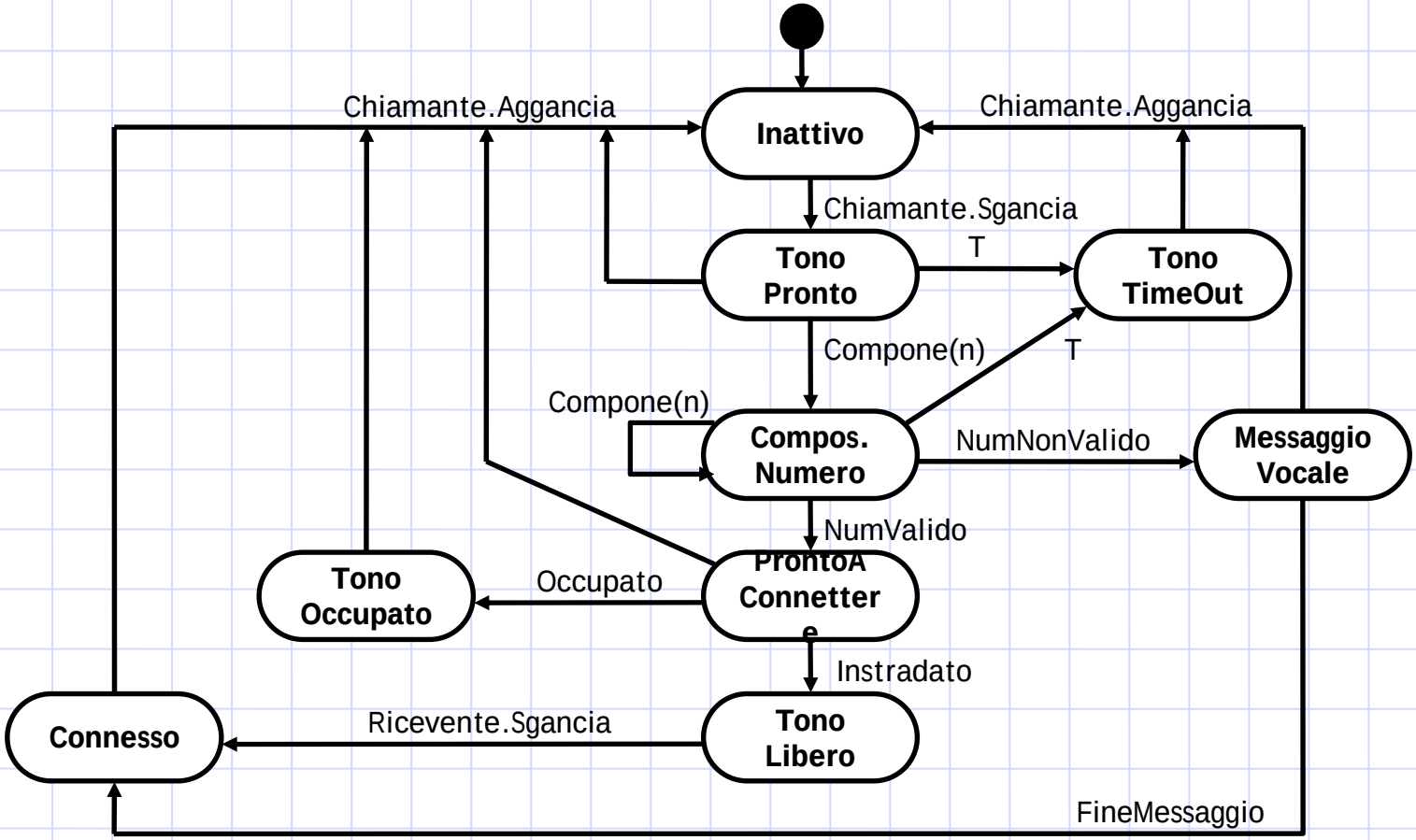


Decomposizione AND



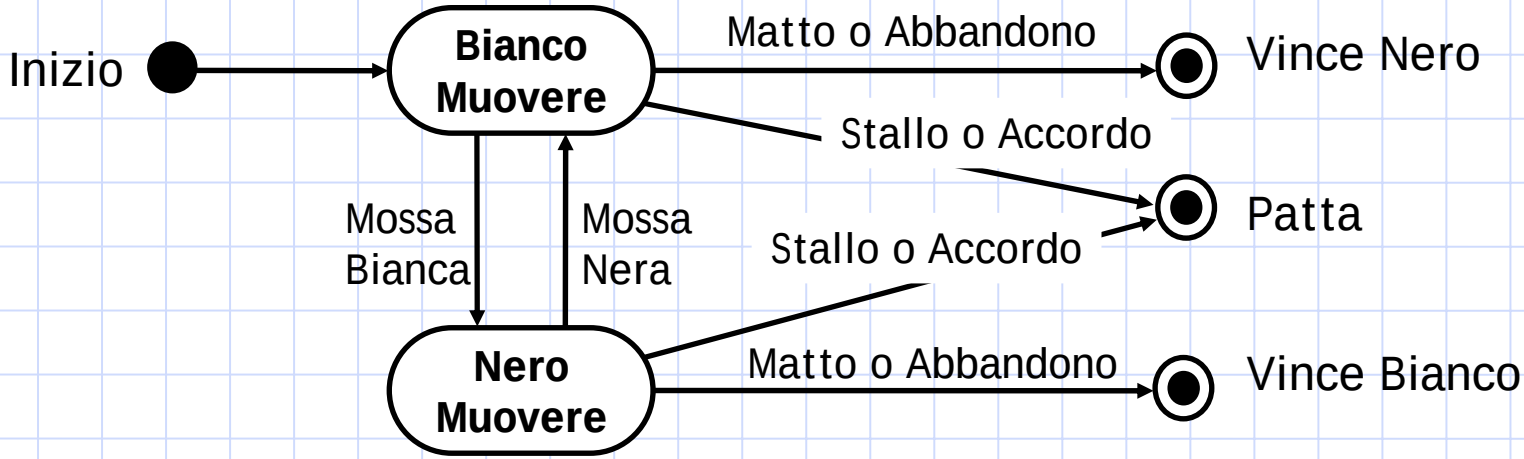
Transizione

Esempio
(Telefonata)



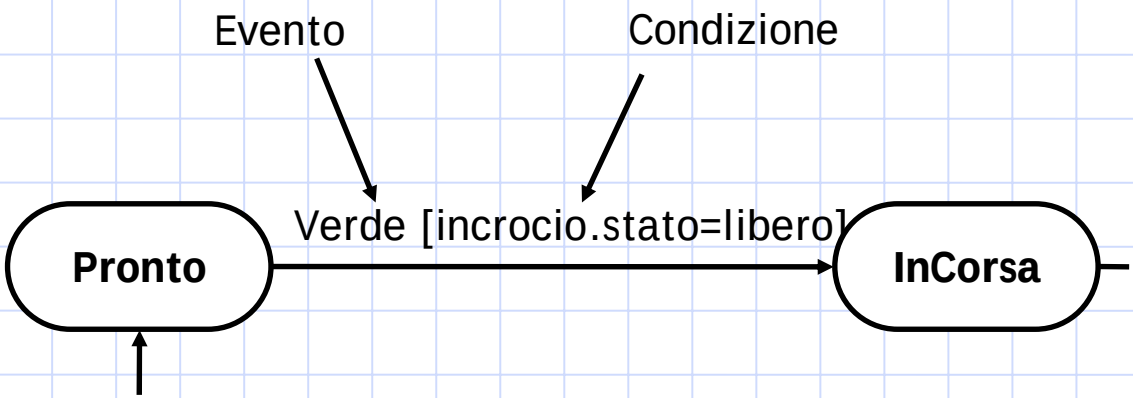
UML: Unified Modeling Language

Inizio e Fine



Condizioni

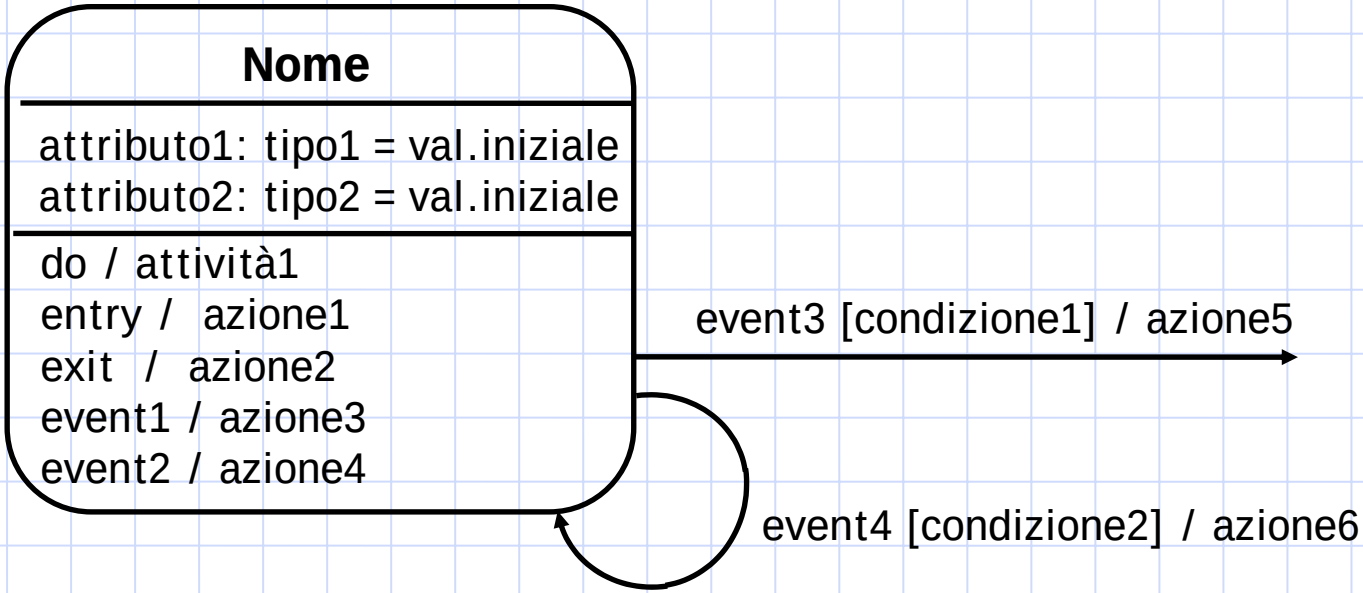
- Funzioni booleane sui valori degli oggetti
- Utili quando non basta l'evento, ma si vuole aggiungere un predicato



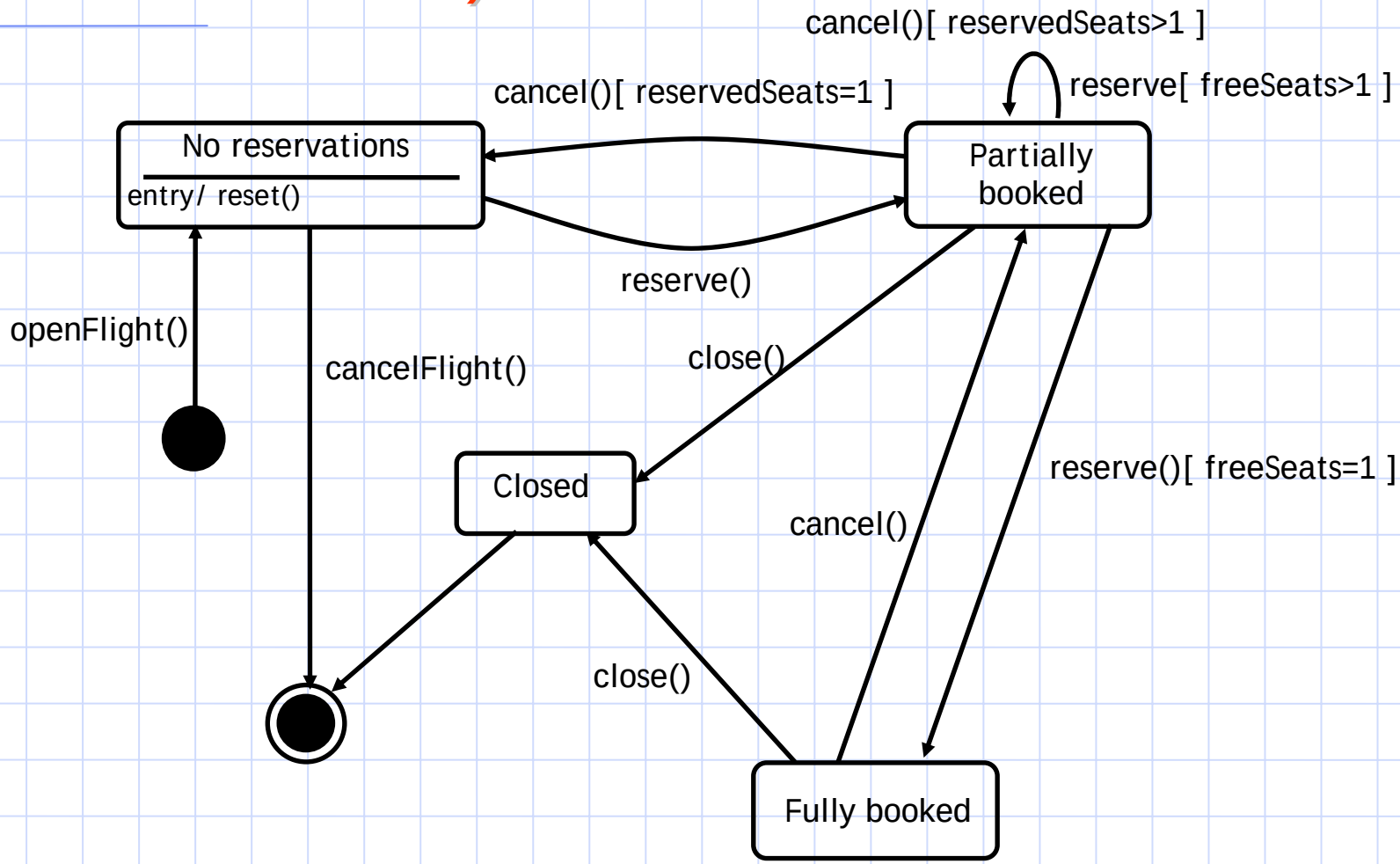
Operazioni

- Azioni
  - Operazioni che hanno durata istantanea
    - Tipicamente produzione di eventi
  - Sono associate alle transizioni di stato oppure all'ingresso o all'uscita da uno stato
- Attività
  - Sono operazioni con durata significativa
  - Sono associate ad uno stato
    - Continue o sequenziali

Stato Completo

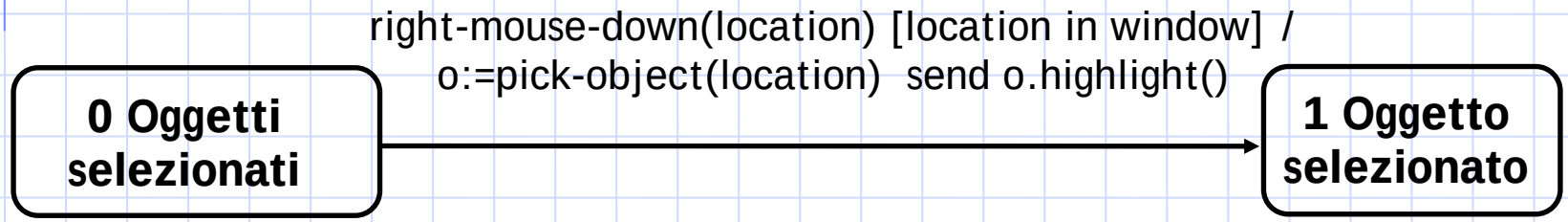


Esempio
(Prenotazione voli)

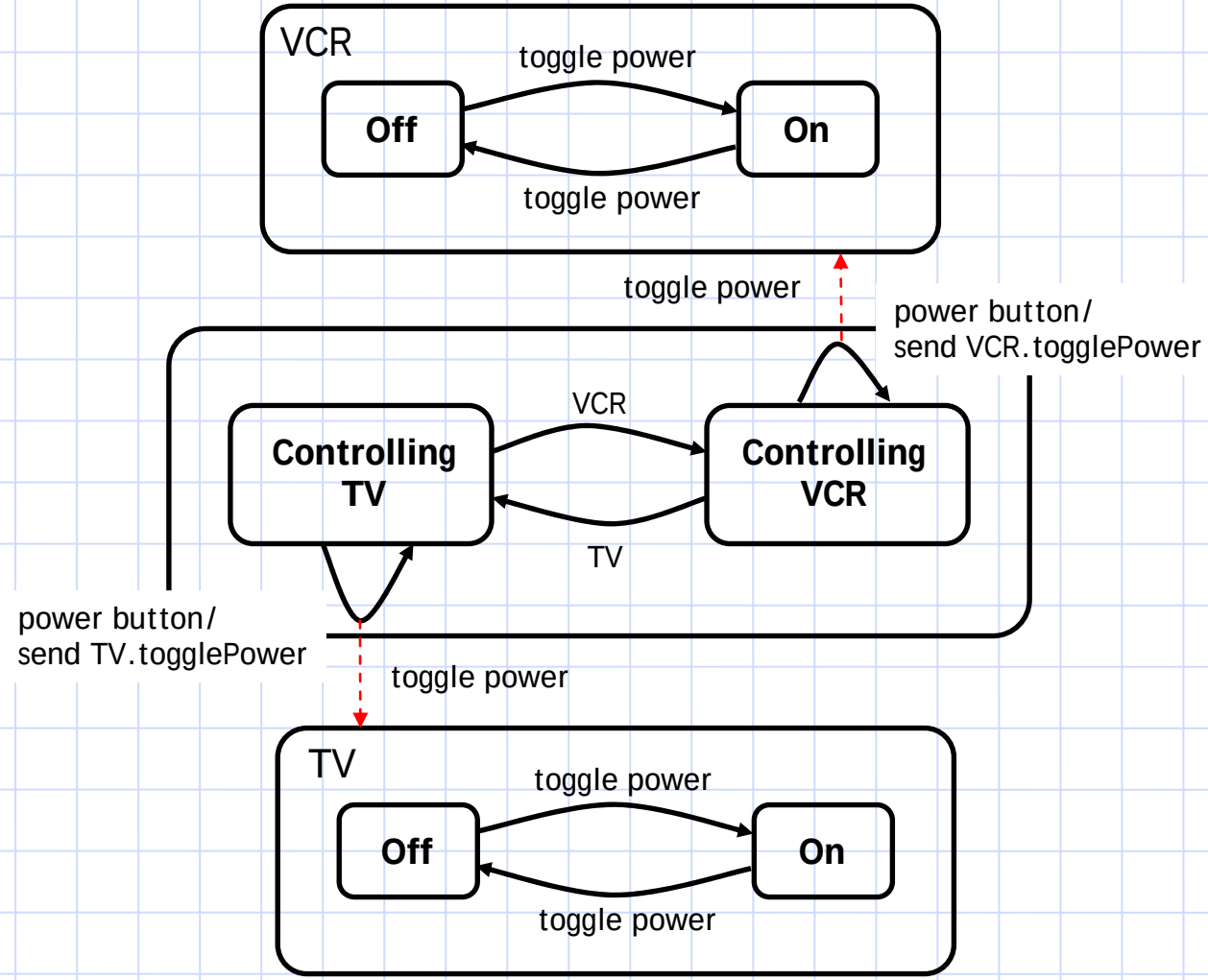


Eventi
(Generati da Azioni)

- Spesso le azioni consistono nell'inviare un evento ad un altro oggetto
  - evento(arg) [cond] / azione send t.azione(arg)

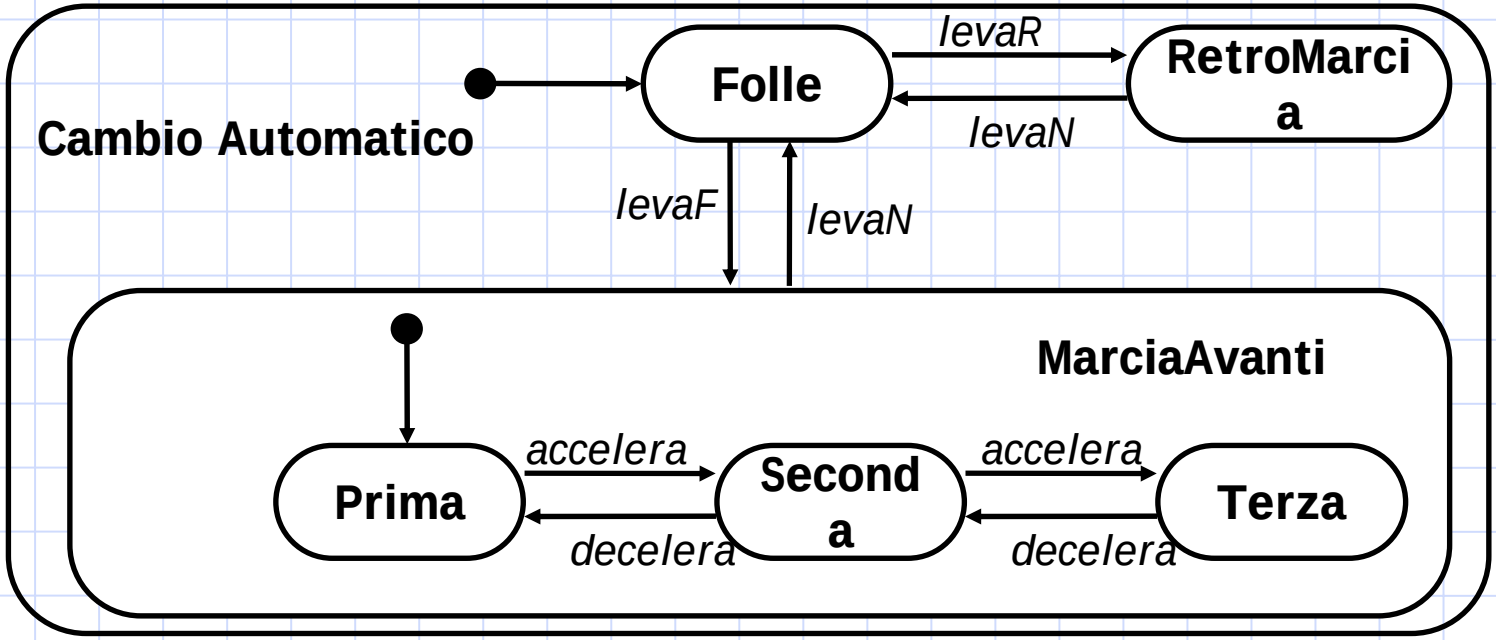


Esempio
(TV, Video Registratore e Telecomando)

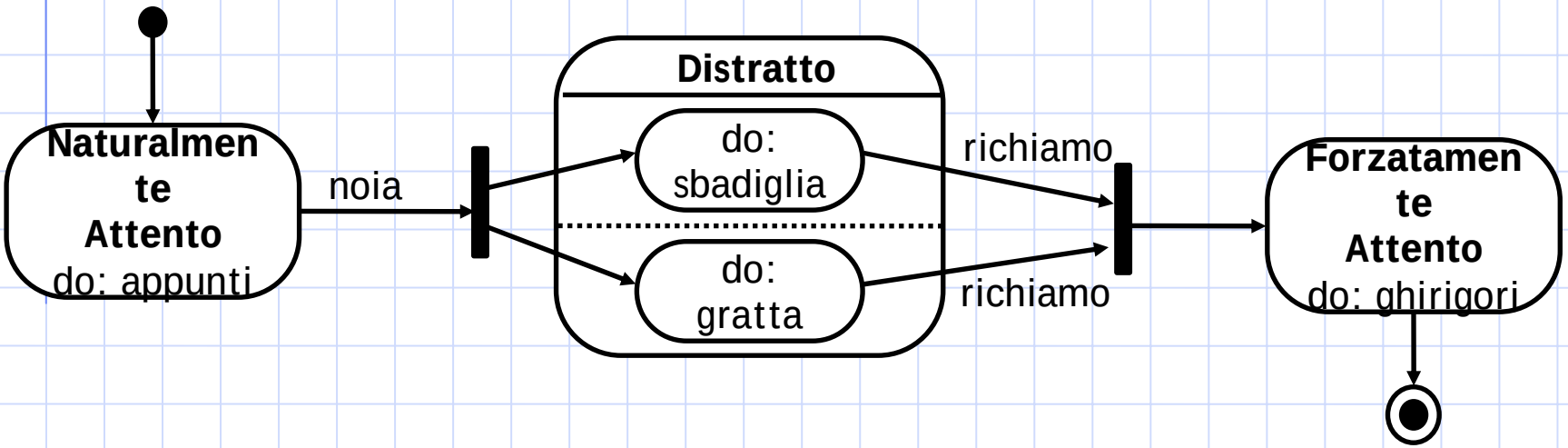


Decomposizione OR

- Un macro stato equivale ad una scomposizione OR degli stati
- I sottostati ereditano le transizioni dei loro superstati

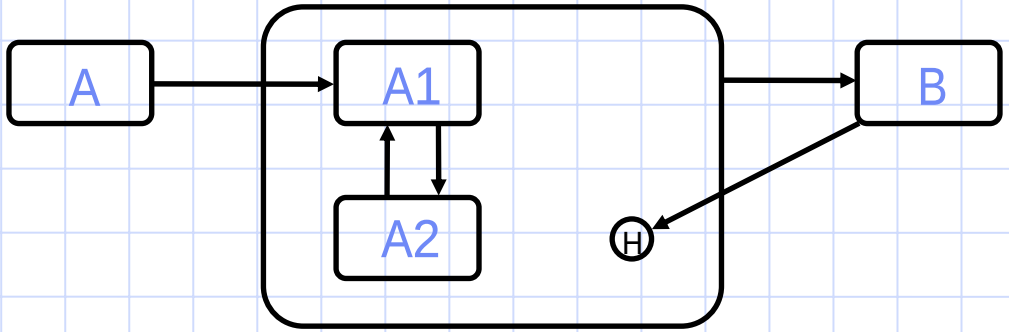


Decomposizione AND

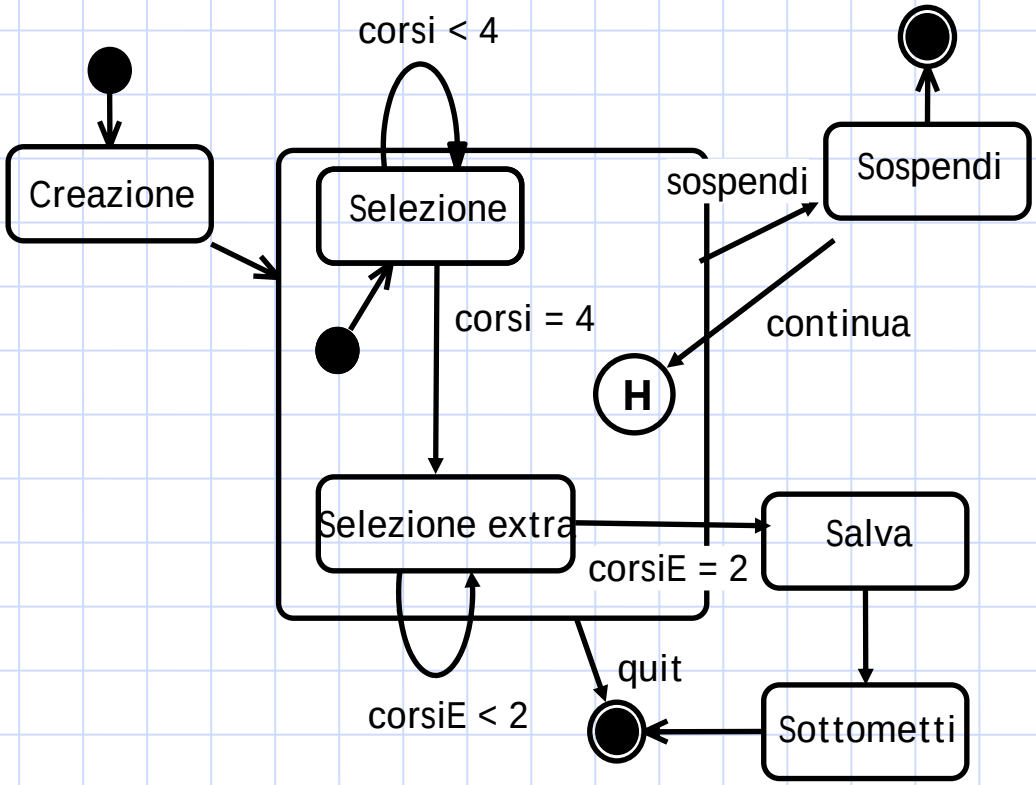


History

- History può essere associata a stati non foglia
- Quando l'esecuzione lascia uno stato S con history
  - Si salva l'ultimo stato visitato S_1 in S
- Quando l'esecuzione ritorna in S
  - Si riparte da S_1



Example
(Selezione corsi)



Esercizio

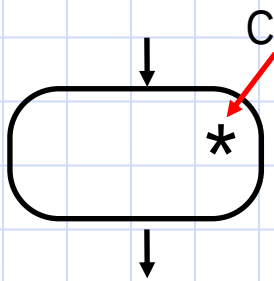
- Si definisca uno State Diagram che rappresenti gli stati in cui si potrebbe trovare un'automobile in fase di parcheggio in un parcheggio a pagamento

Activity Diagram

Activity Diagram

- Forniscono la sequenza di operazioni che definiscono un'attività più complessa
- Possono essere considerati State Diagram particolari
  - Ogni stato contiene (è) un'azione
- Un Activity Diagram può essere associato
  - A una classe
  - All'implementazione di un'operazione
  - Ad uno Use Case
- Retaggio della scomposizione "funzionale"

Elementi grafici



attività

start

end

fork

[condition]

join

branch

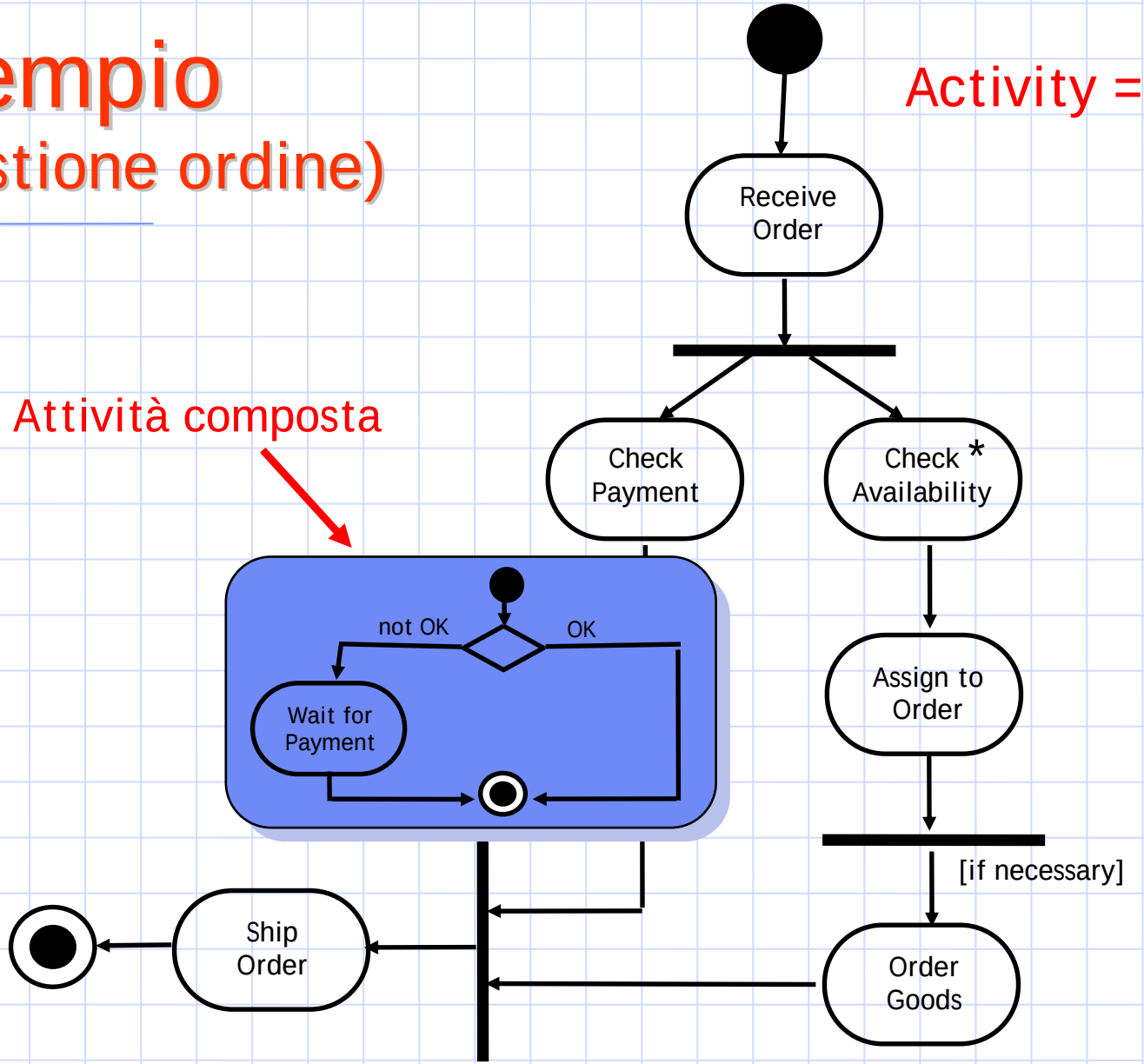
[condition]

[else]

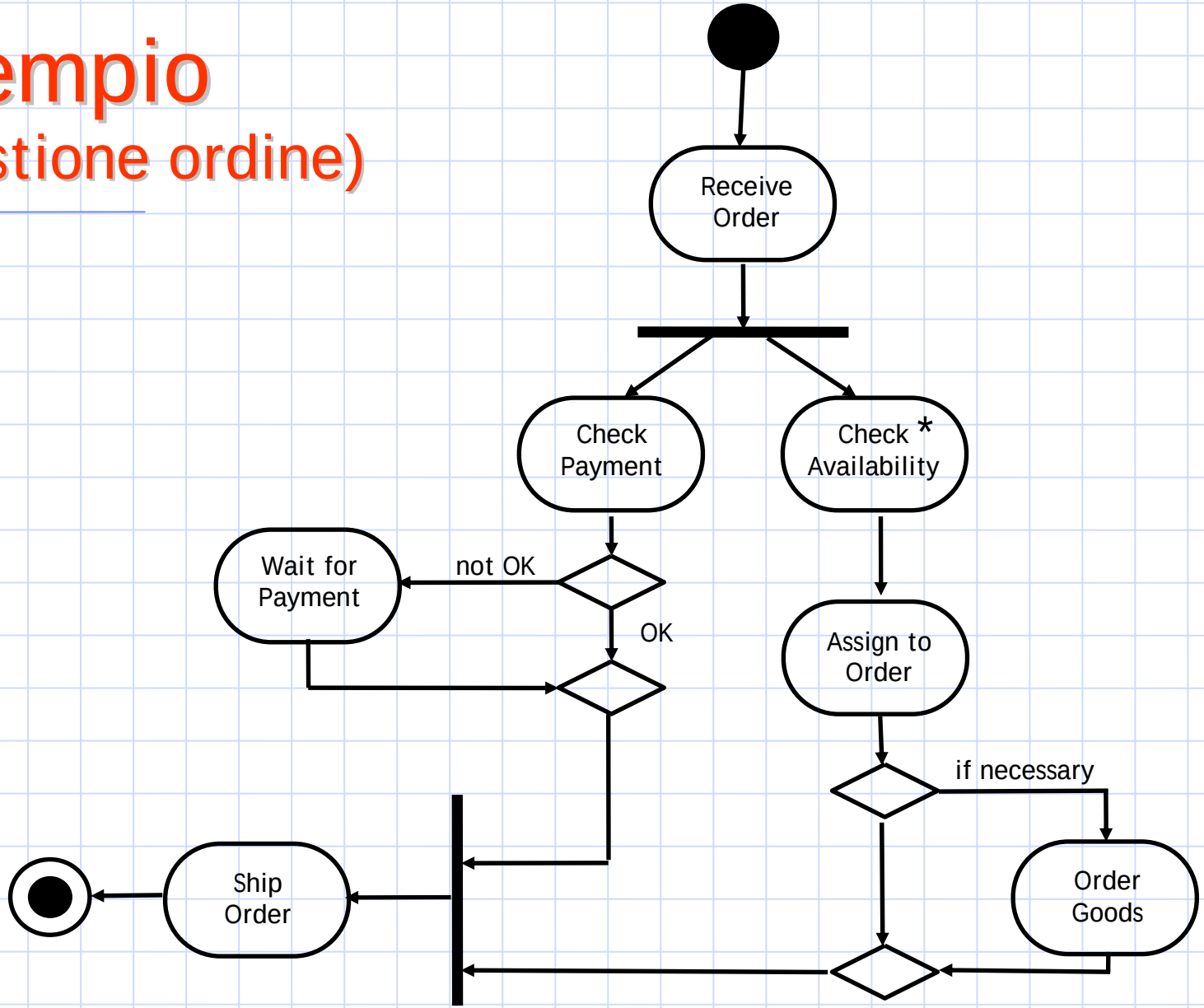
merge

Le attività possono essere gerarchiche

Esempio
(Gestione ordine)

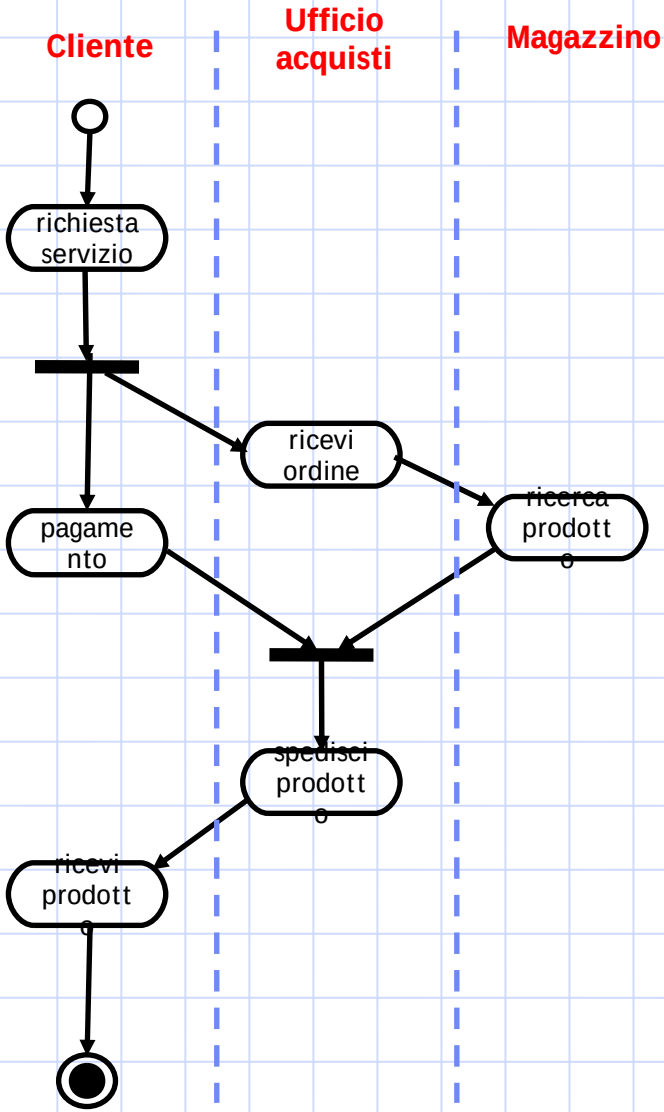


Esempio
(Gestione ordine)

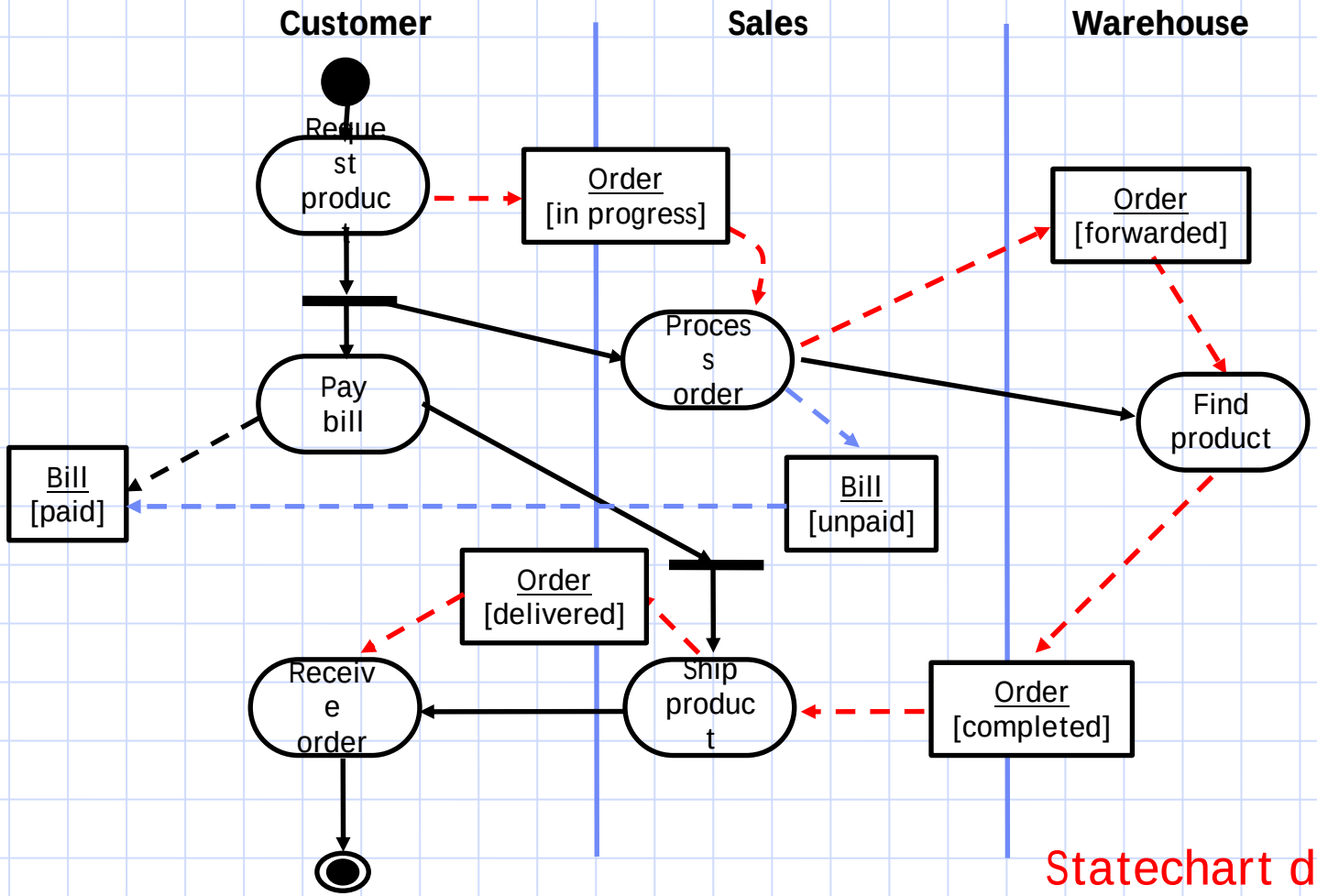


Swimlane
(Corsie)

- Identificano le responsabilità relative alle diverse operazioni
  - Parti di un oggetto
  - Oggetti diversi
- In un Business Model identificano le unità organizzative



Attività e flussi degli oggetti



Statechart diagram

UML: Unified Modeling Language

Esercizio

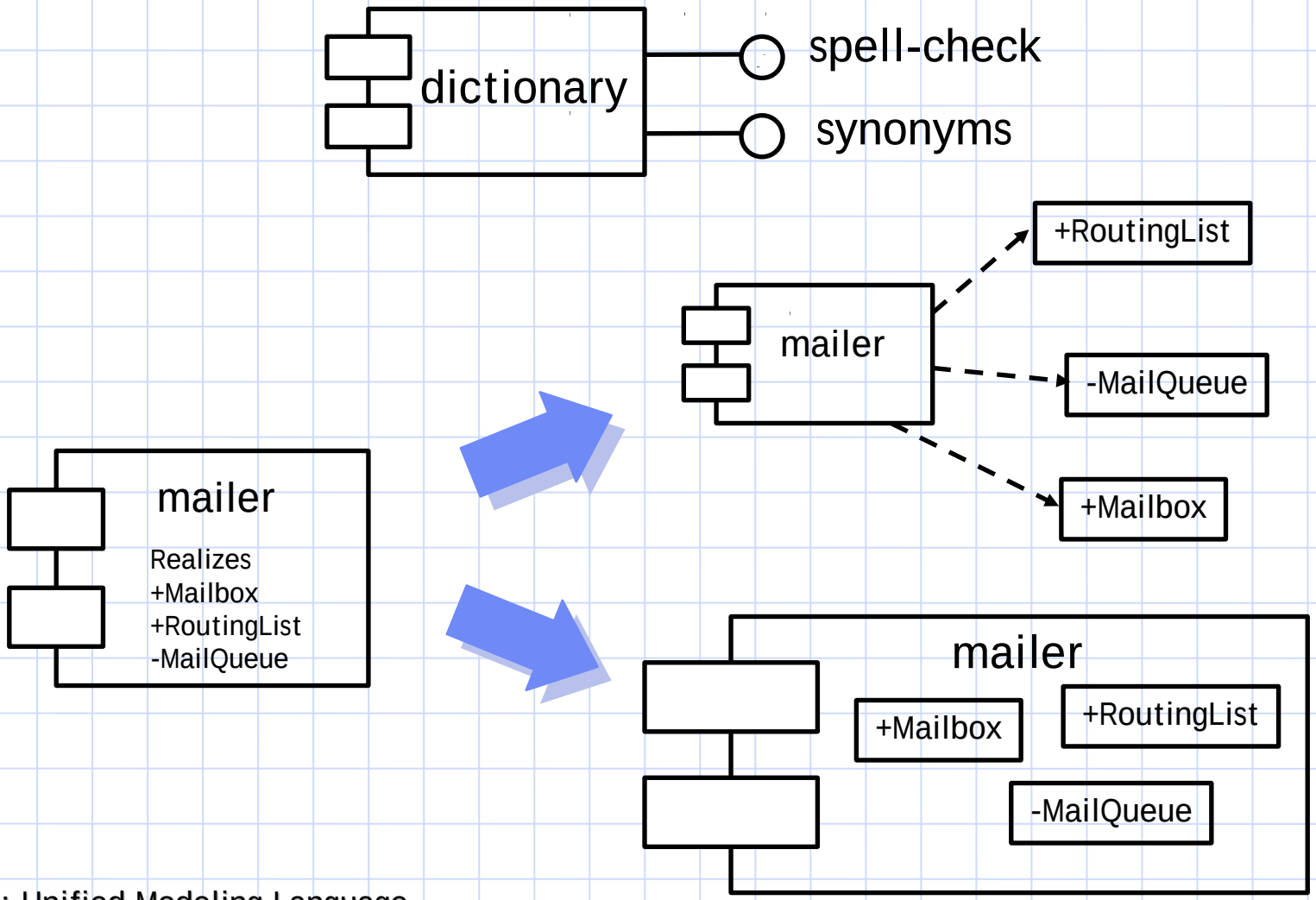
- Definire il diagramma delle attività relativo alla realizzazione di un programma C

Implementation Diagram

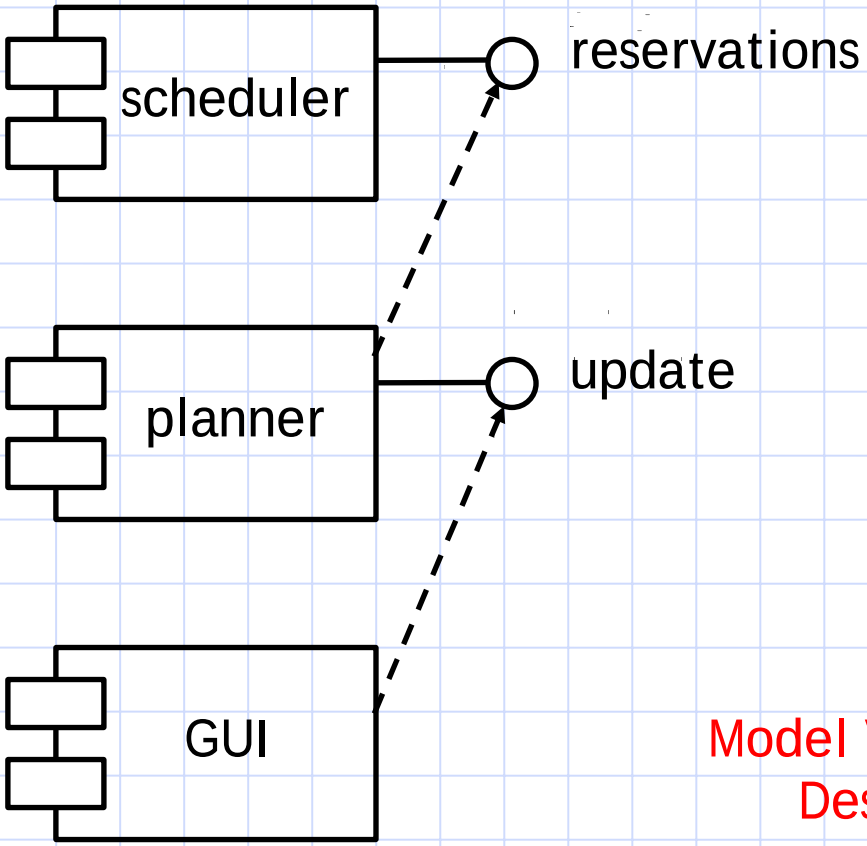
Implementation Diagram

- Component Diagram
  - Definiscono i componenti “fisici” che realizzano l’applicazione (eseguibili, librerie, ecc..)
- Deployment Diagram
  - Definiscono il partizionamento (la struttura) dell’applicazione: processi e/o processori

Componenti

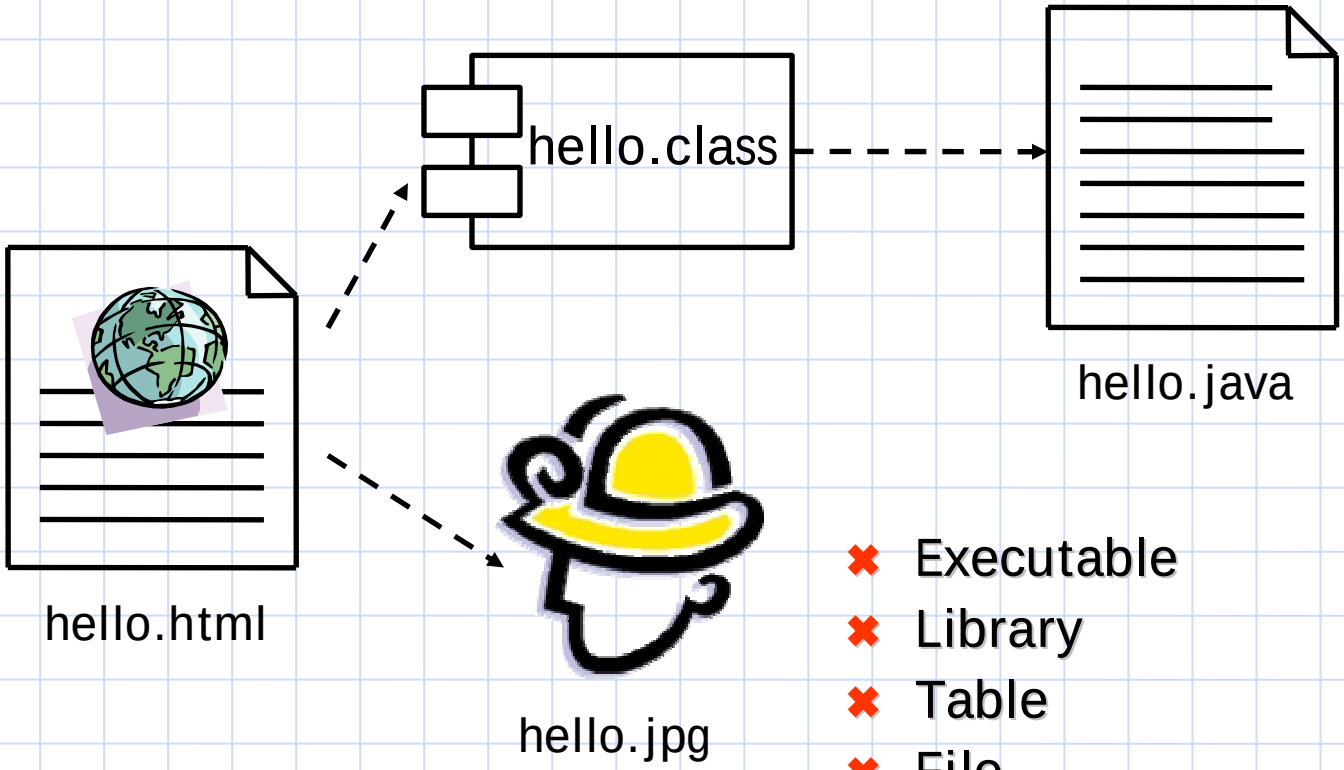

dictionary
spell-check
synonyms
+RoutingList
mailer
-MailQueue
+Mailbox
mailer
Realizes
+Mailbox
+RoutingList
-MailQueue
mailer
+Mailbox
+RoutingList
-MailQueue

Esempio
(Scheduler)



Model View Controller
Design pattern

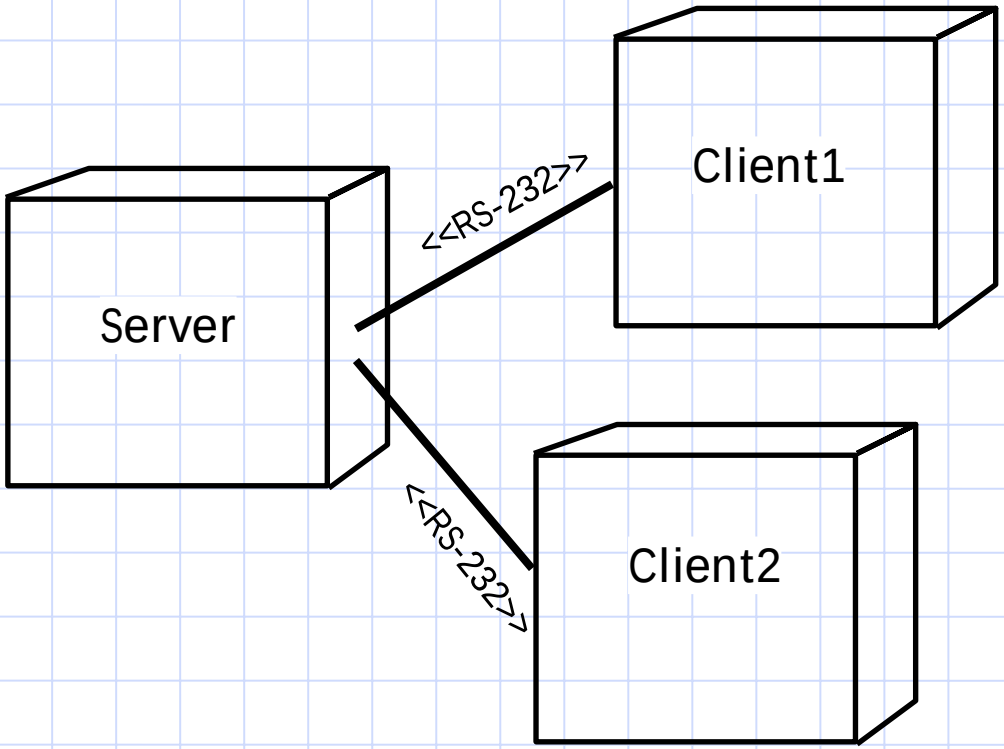
Usando icone carine
(HelloWorld)



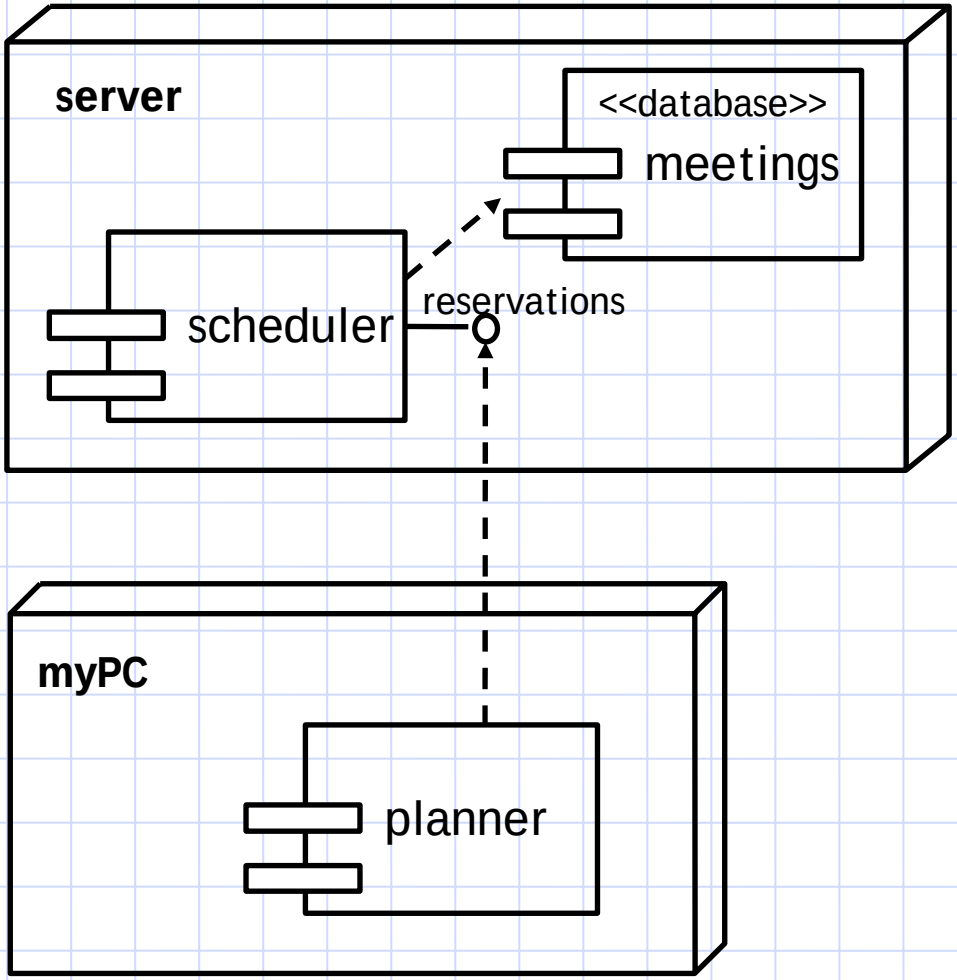
- Executable
- Library
- Table
- File
- Document
- ...

Associazioni

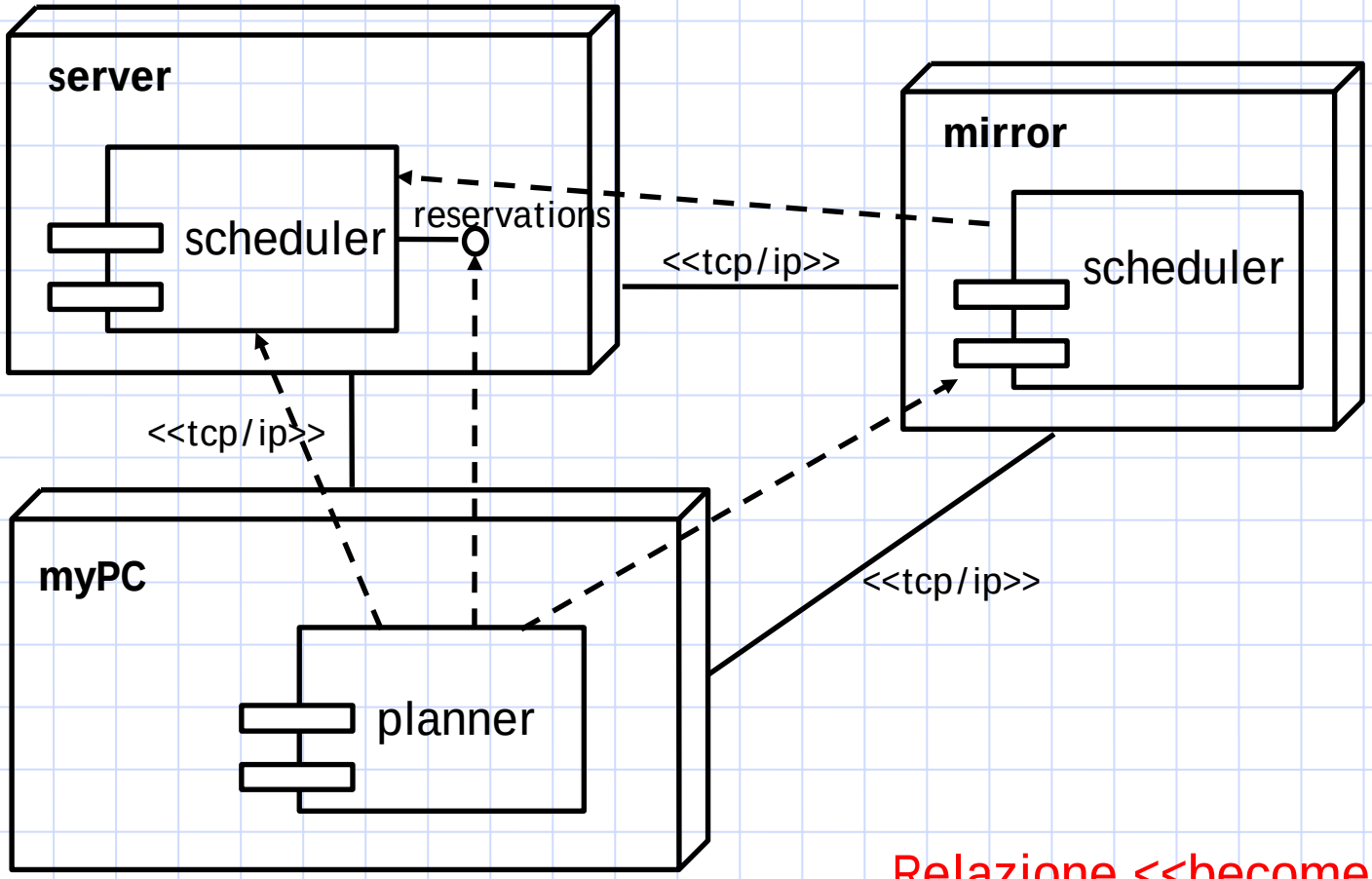
- Rappresentano le connesioni fisiche tra i nodi



Nodi e componenti
(I soluzione)



Nodi e componenti
(II soluzione)

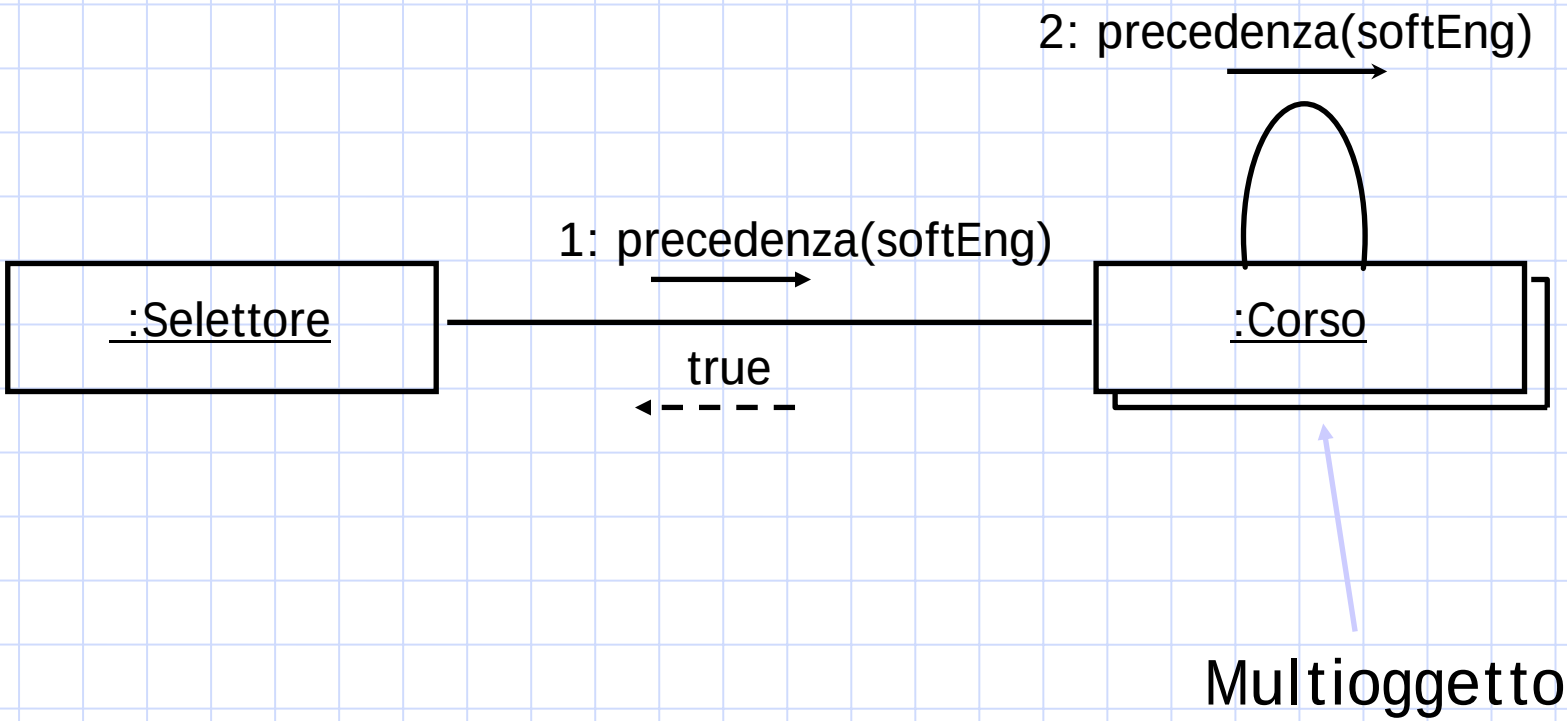


Relazione <<become>>

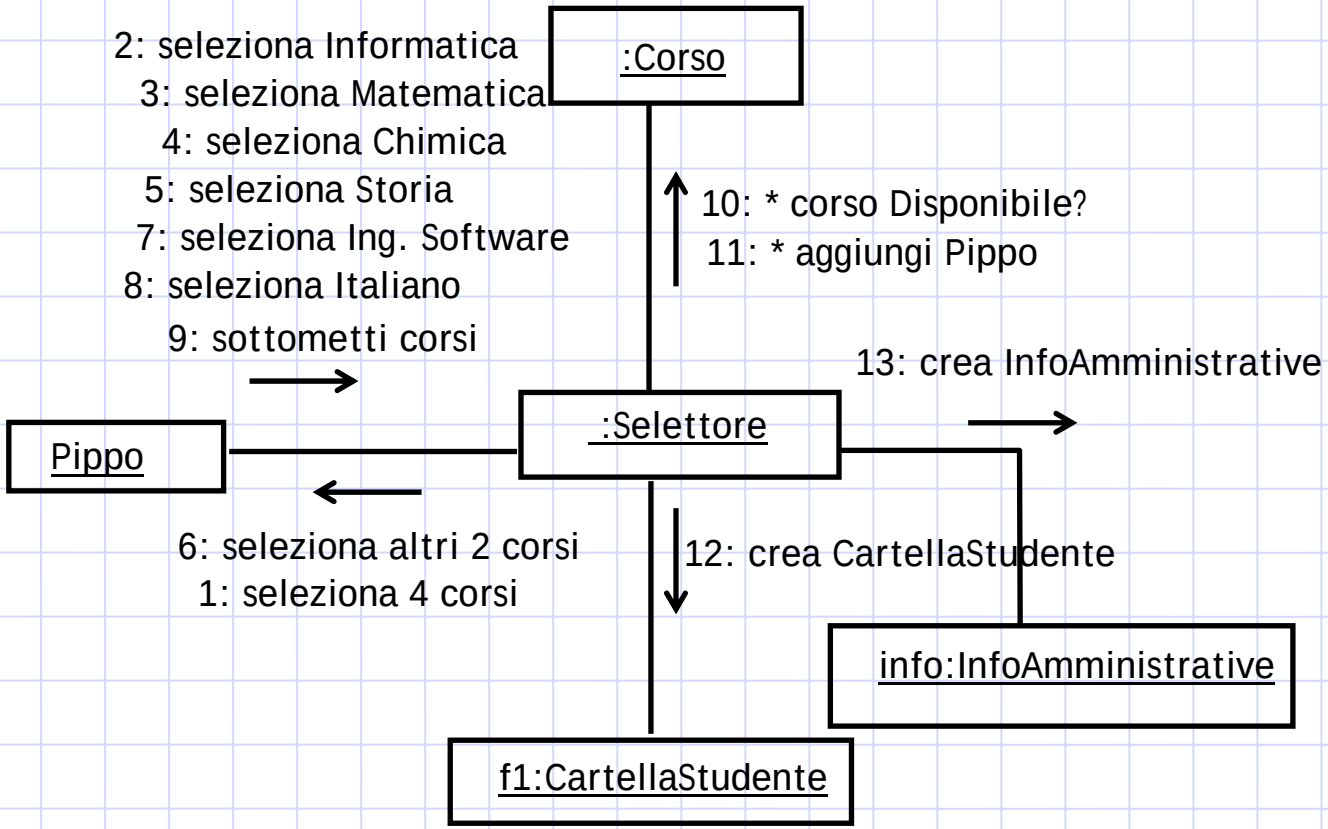
Collaboration diagram

- Simili a sequence diagram, ma
  - Evidenziano le interazioni tra le parti
  - Maggior attenzione allo scambio messaggi
  - Adatti per
    - Concorrenza e thread
    - Invocazioni innestate
    - Interazioni “sofisticate”
- Il tempo non è associato ad una dimensione precisa
  - Le sequenze si scoprono usando strane numerazioni
  - Sequence alternative possono essere modellate sullo stesso diagramma

Esempio
(Precedenza tra corsi)

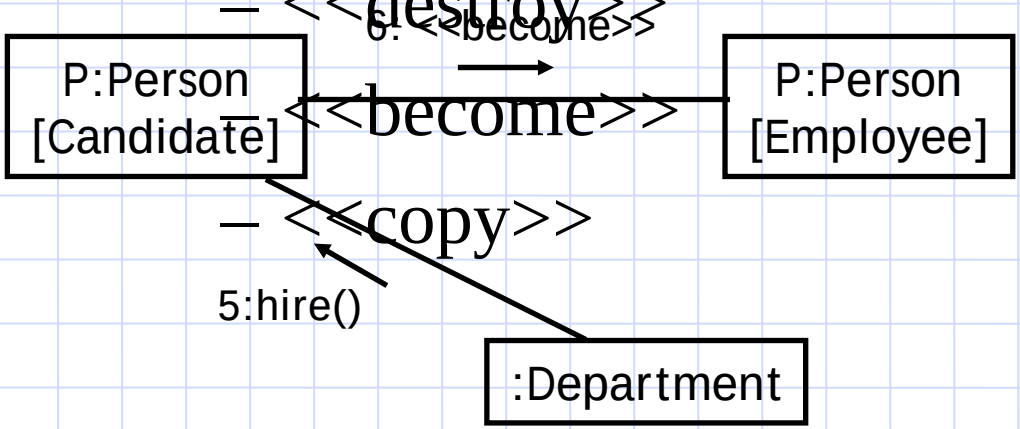


Esempio

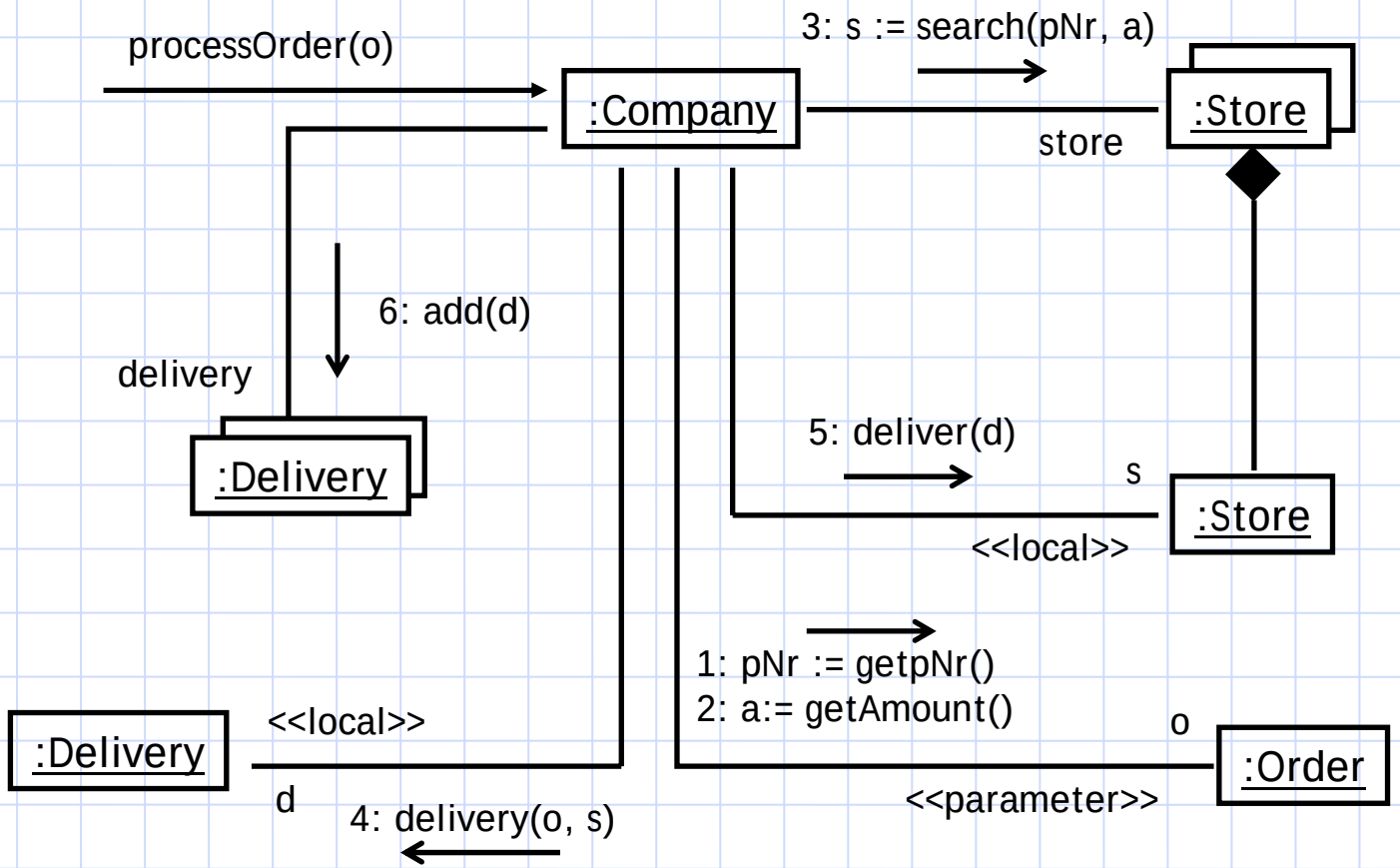

:Corso
2: seleziona Informatica
3: seleziona Matematica
4: seleziona Chimica
5: seleziona Storia
7: seleziona Ing. Software
8: seleziona Italiano
9: sottometti corsi
10: * corso Disponibile?
11: * aggiungi Pippo
13: crea InfoAmministrative
Pippo
:Selettore
6: seleziona altri 2 corsi
1: seleziona 4 corsi
12: crea CartellaStudente
info:InfoAmministrative
f1:CartellaStudente

Qualcosa in più sui link

- I link possono essere
  - Invocazioni
  - <<create>>
  - <<destroy>>
  - <<become>>
  - <<copy>>
- Le estremità possono essere dichiarate
  - <<association>>
  - <<self>>
  - <<global>>
  - <<local>>
  - <<parameter>>



Esempio



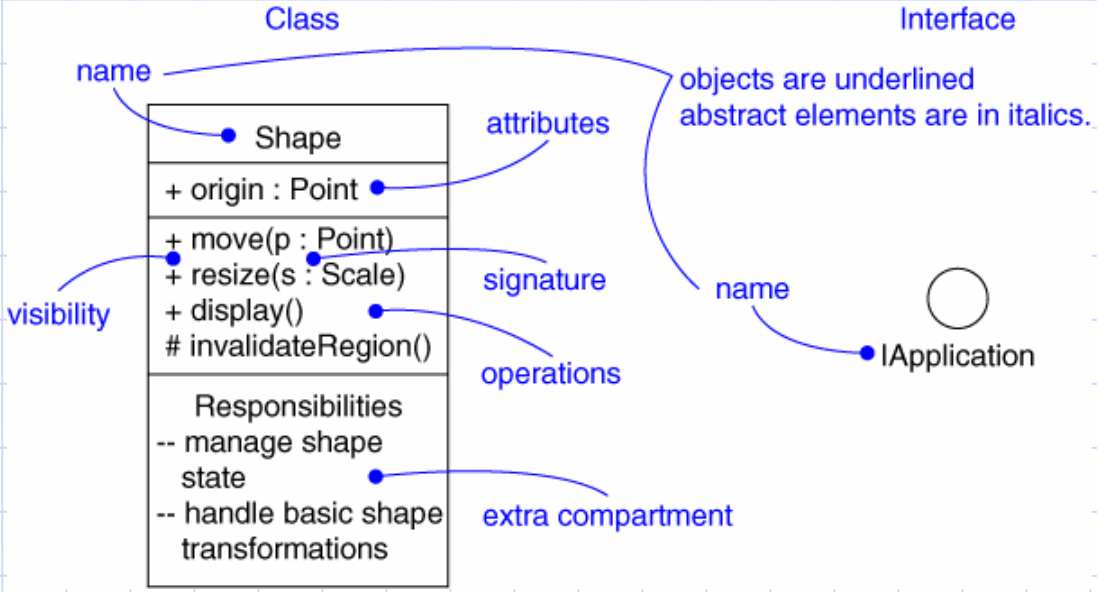
Gunnar Overgaard, "A Formal Approach to Collaborations in the Unified Modeling Language". In Proceedings of UML 99

UML: Unified Modeling Language

Riassumendo

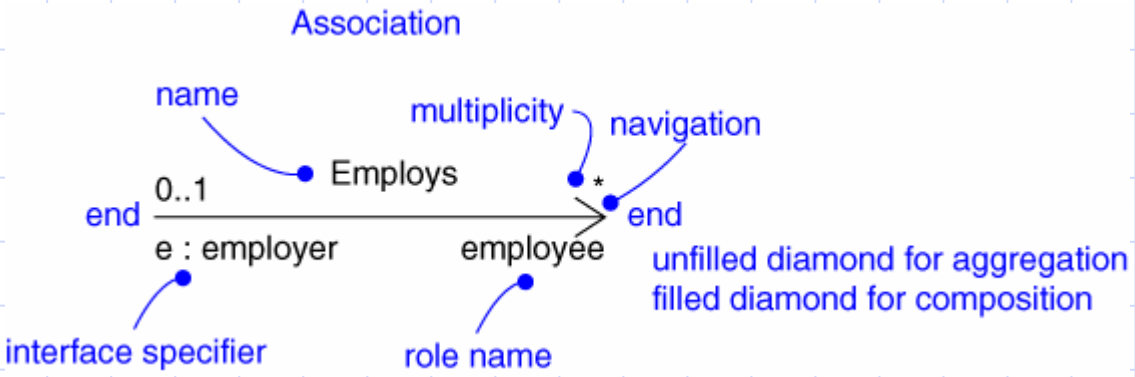
Elementi

- Elementi strutturali
  - classi, interfacce, collaboration, use case, classi attive, componenti, nodi
- Elementi “dinamici” (behavioral)
  - interaction, automi a stati
- Raggruppamenti
  - package, sottosistemi
- Altri elementi
  - note



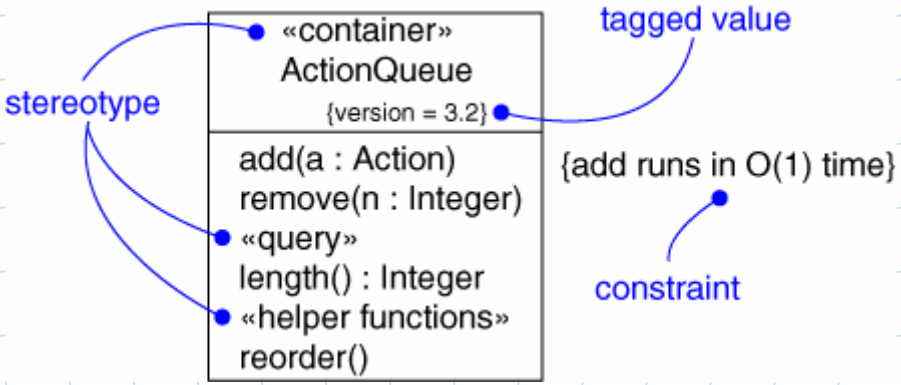
Relazioni

- Dipendenze
- Associazioni
- Generalizazioni
- Realizzazioni



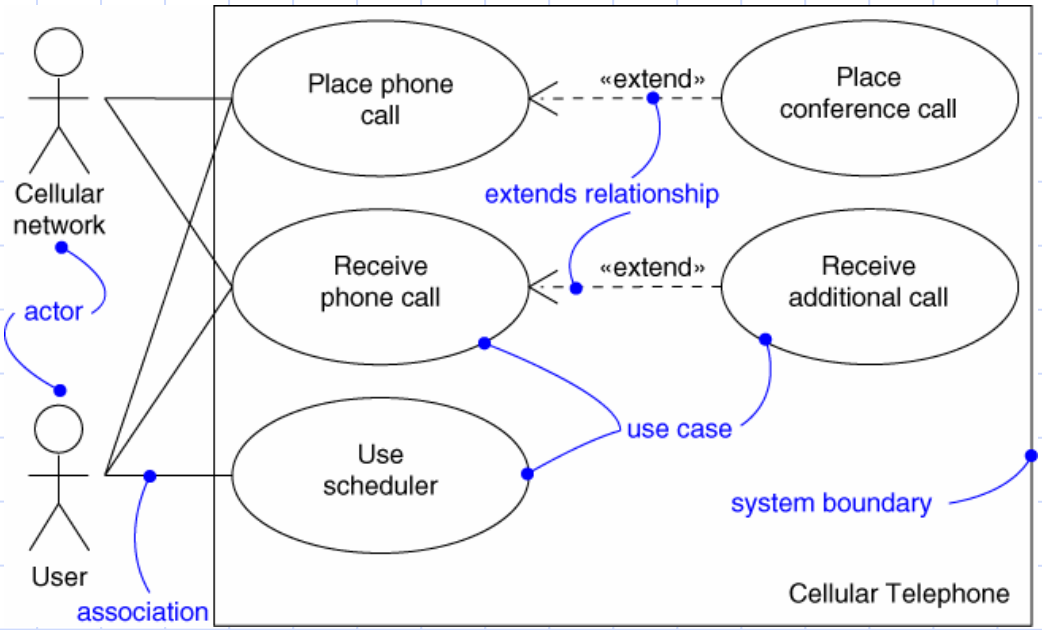
Estendibilità

- Stereotipi
- Restrizioni (tagged value)
- Vincoli (constraint)



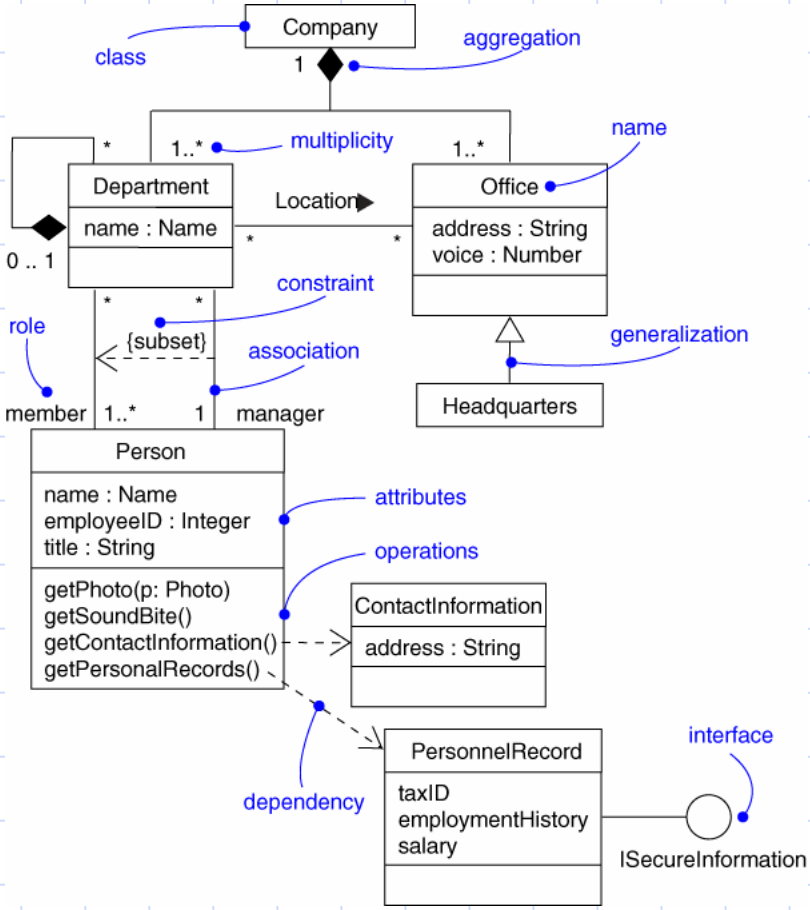
Use case diagram

- Catturano le funzionalità del sistema (vista utente)



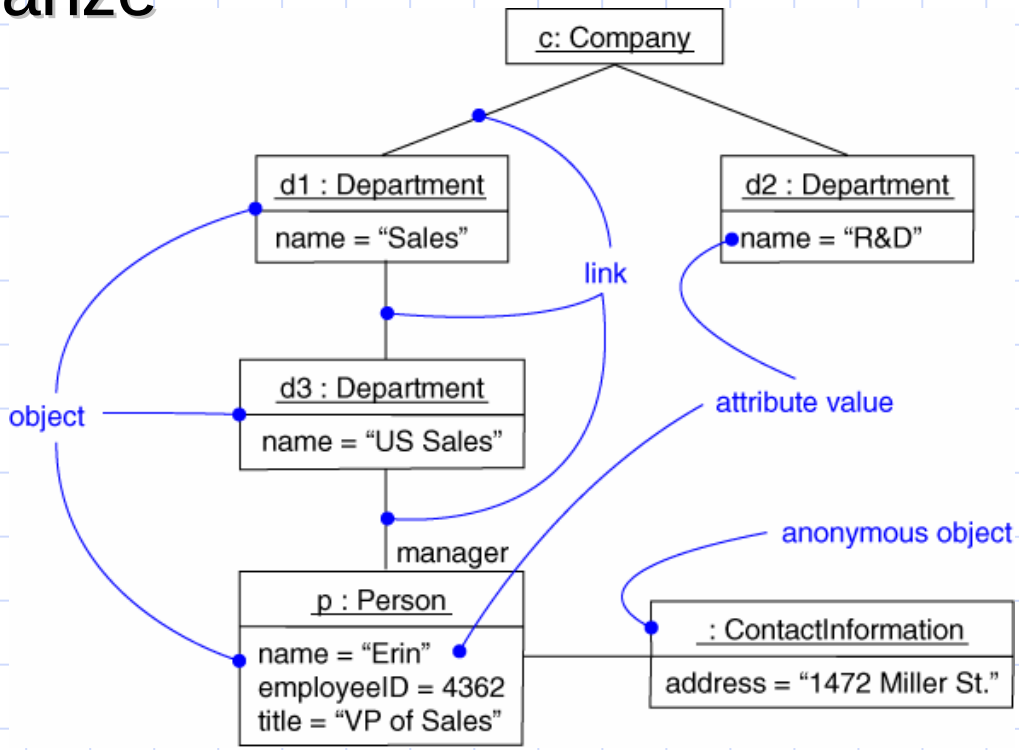
Class diagram

- Catturano il vocabolario del sistema



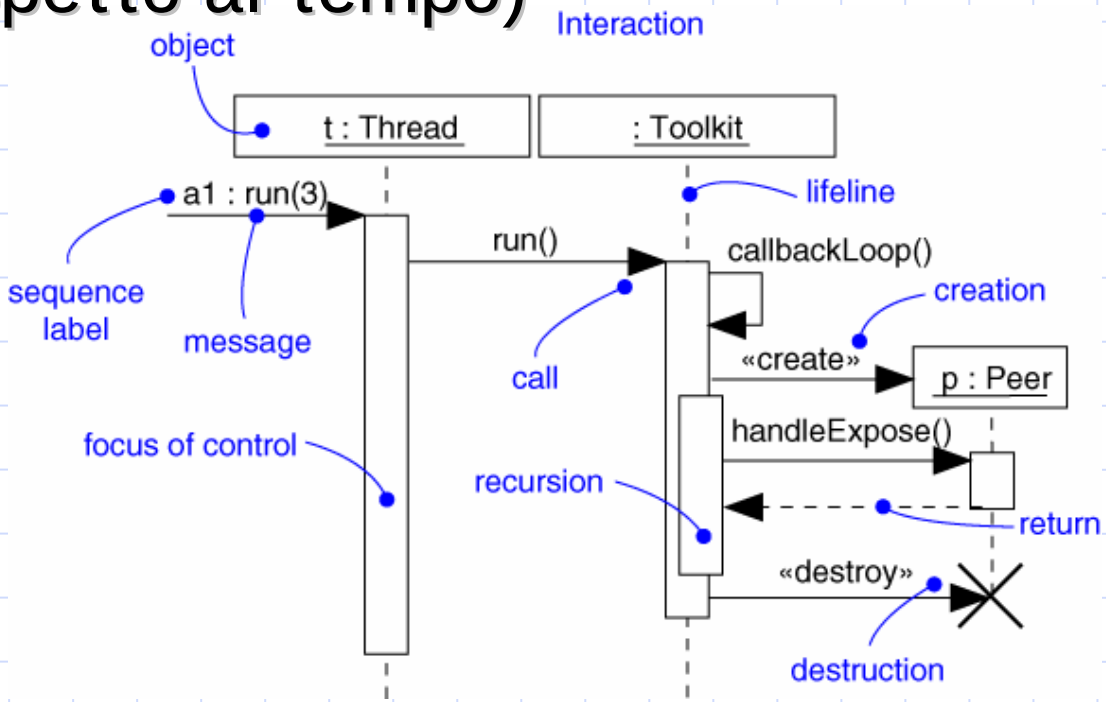
Object diagram

- Le istanze e i legami fra le istanze



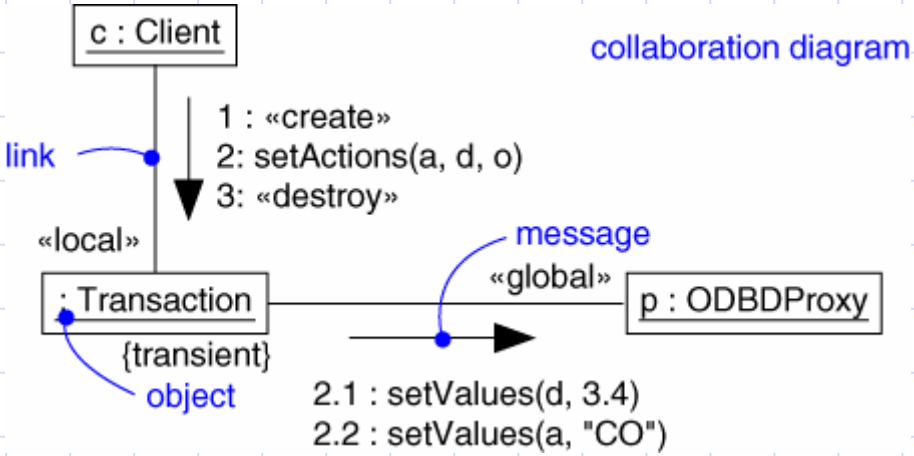
Sequence diagram

- Catturano il comportamento dinamico (rispetto al tempo)



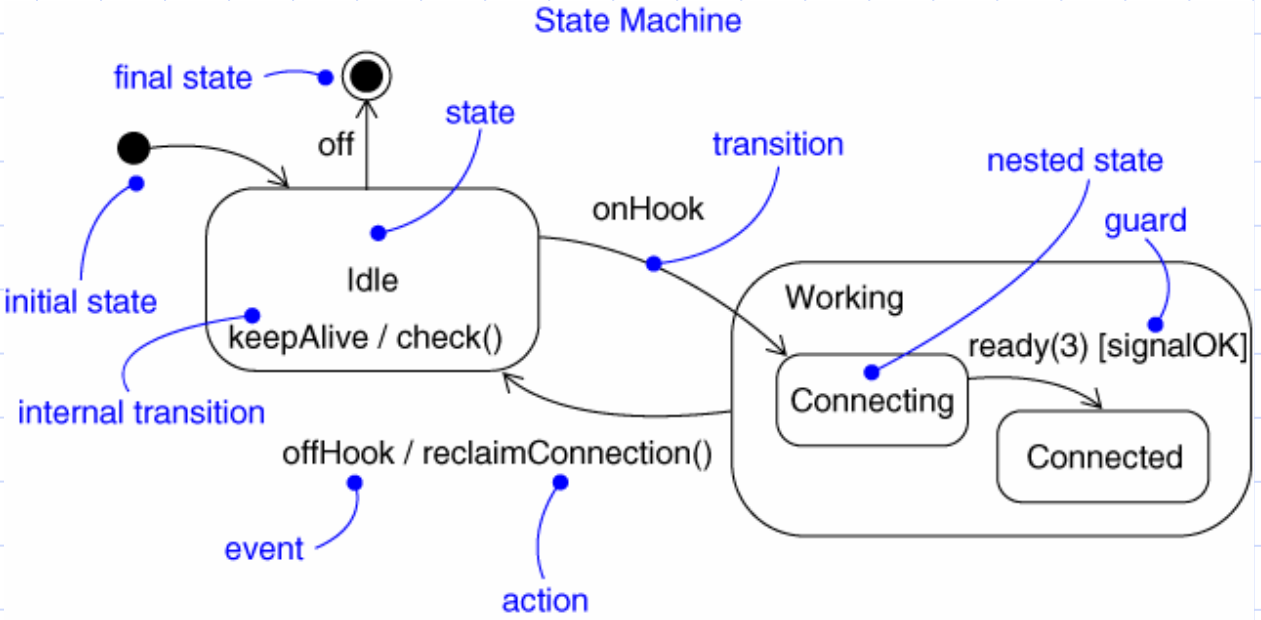
Collaboration diagram

- Catturano il comportamento dinamico (rispetto ai messaggi)



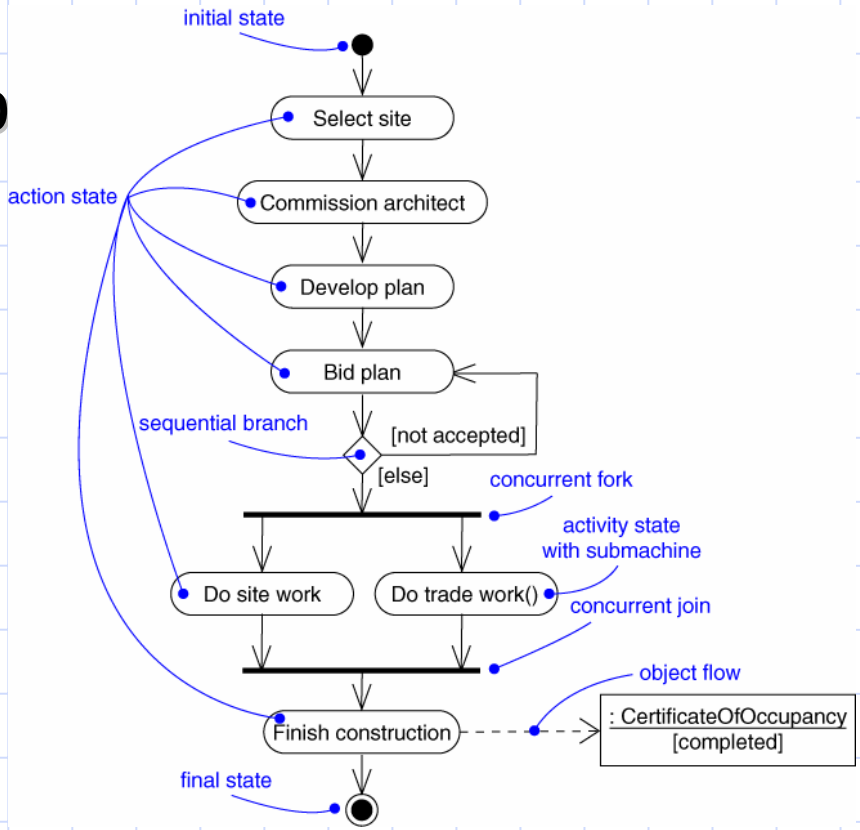
Statechart diagram

- Catturano il comportamento dinamico (rispetto agli eventi)



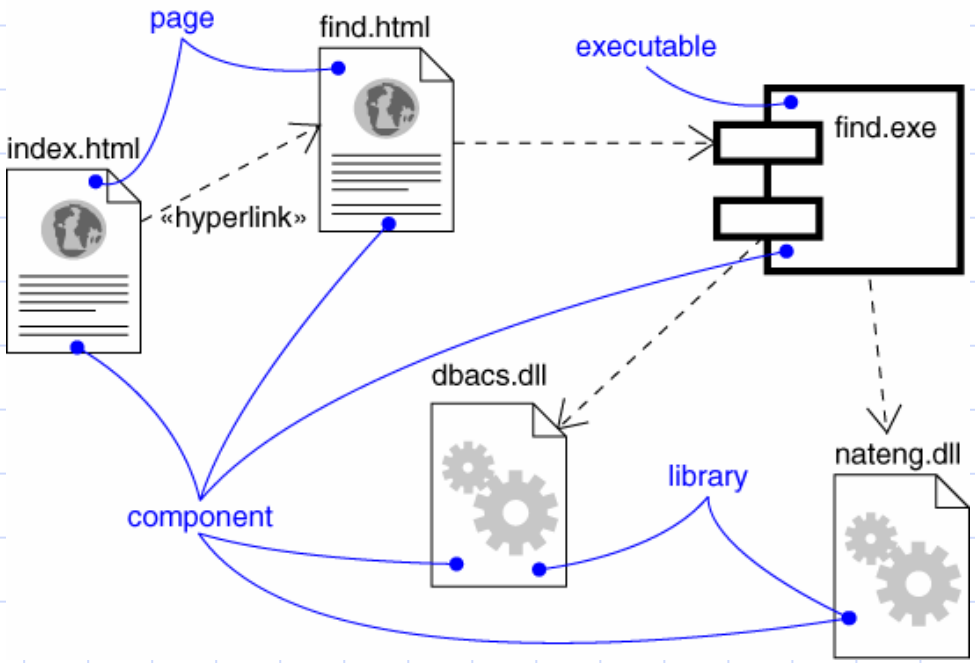
Activity diagram

- Catturano il comportamento dinamico (rispetto attività)



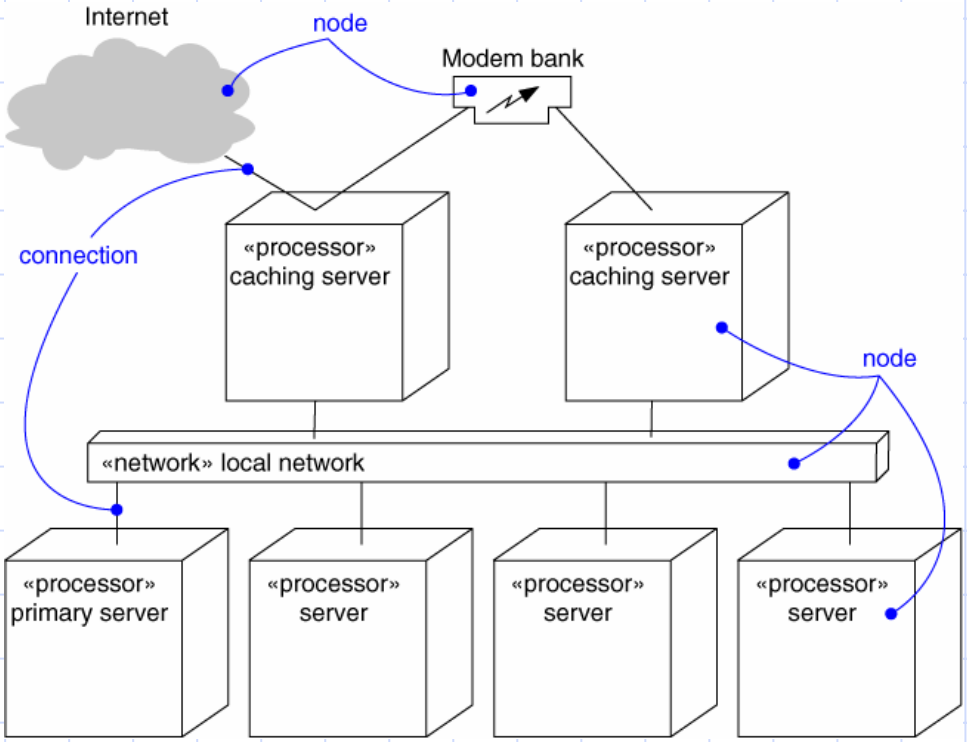
Component diagram

- Catturano la struttura fisica dell'implementazione



Deployment diagram

- Catturano la topologia del sistema



Rational Unified Process

UML e il processo di sviluppo

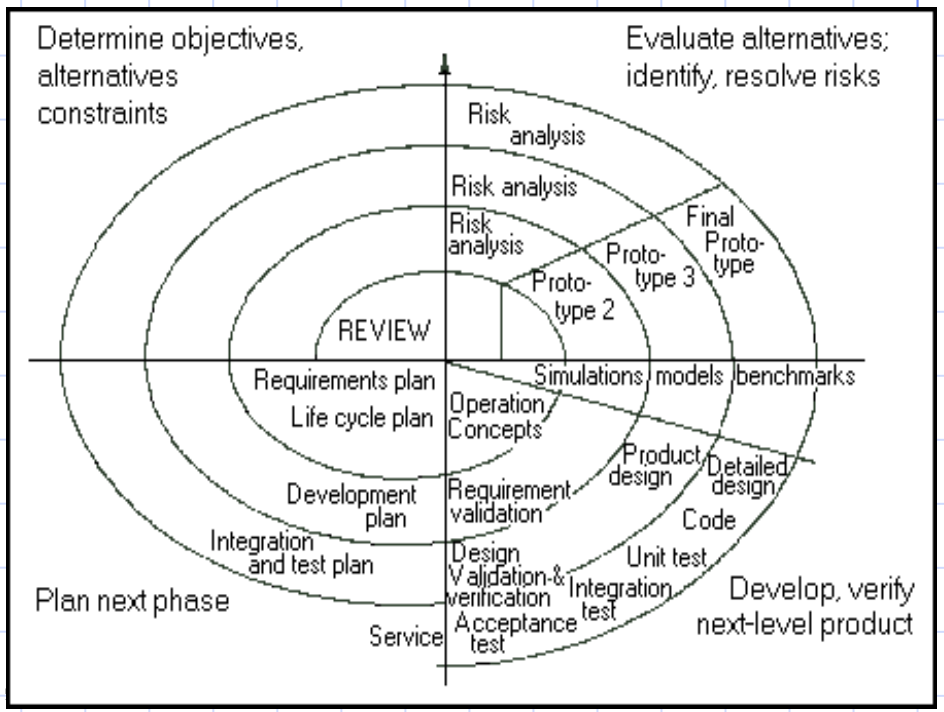
- UML è largamente indipendente dal processo
- Tuttavia, si è ritenuto opportuno proporre un modello di ciclo di vita particolarmente adatto allo sviluppo mediante UML

Caratteristiche del processo

- Iterativo
  - per assecondare la complessità dello sviluppo
  - e i cambiamenti nei requisiti
- Enfasi sui modelli più che sui documenti
  - più leggibili, più modificabili, più maneggiabili in modo automatico
- Centrato sull'architettura
  - sviluppo in parallelo
  - riuso e manutenibilità
- Guidato da use case
- Configurabile
- Incoraggia controllo della qualità e gestione del rischio

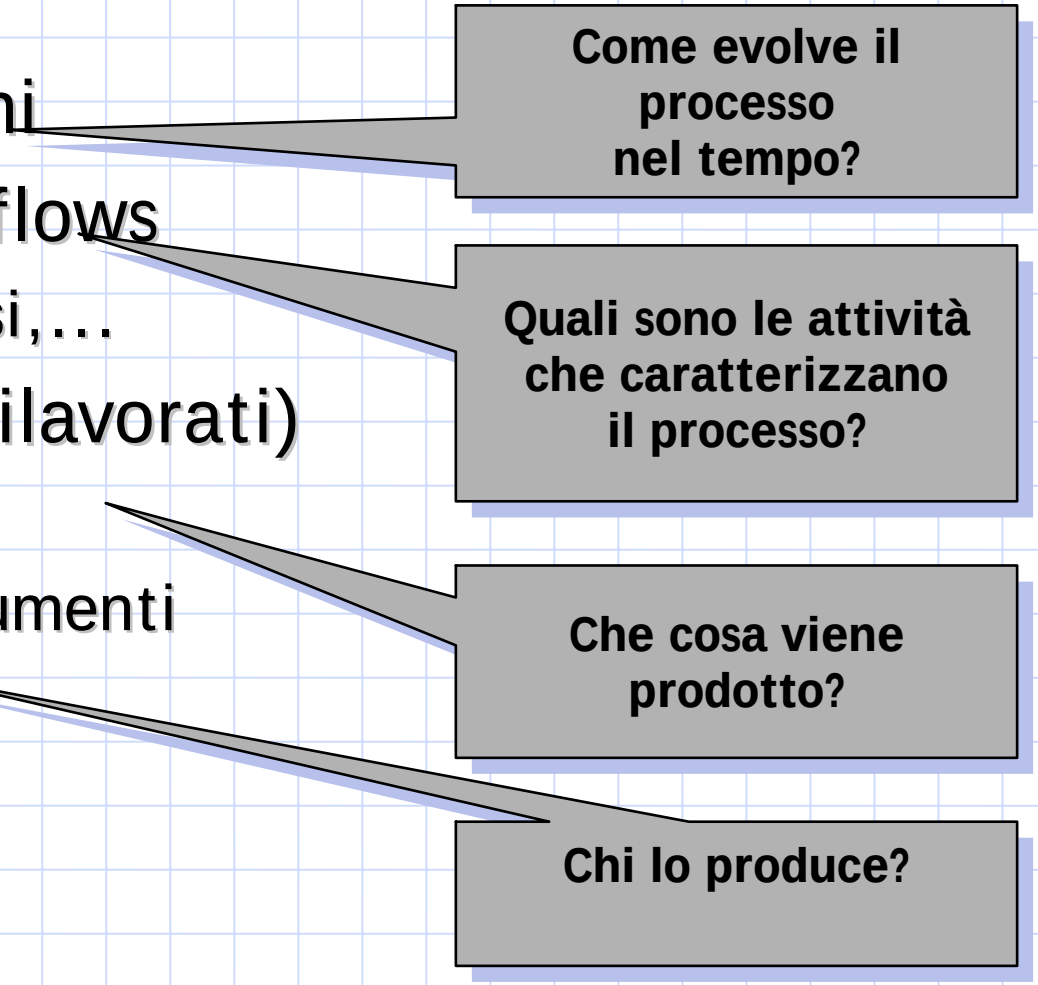
Unified Process: Ciclo di vita

- Il meta-processo di riferimento è il modello a spirale (Boehm):
  - È un meta-processo evolutivo
  - Riciclo come fondamento del modello
  - Ripetizione ciclica di task regions:
    - Es: planning,risk analysis,
    - sviluppo, validazione
  - Ogni spirale può avere struttura diversa (waterfall prototyping etc.)



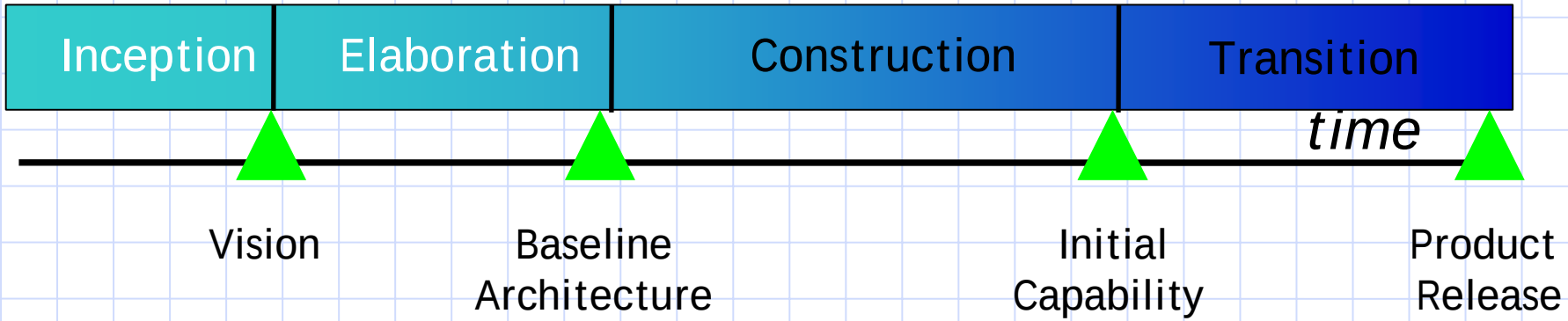
Concetti chiave

- Fasi, Iterazioni
- Process Workflows
  - Attività, passi,…
- Artifatti (semilavorati)
  - modelli
  - reports, documenti
- Ruoli

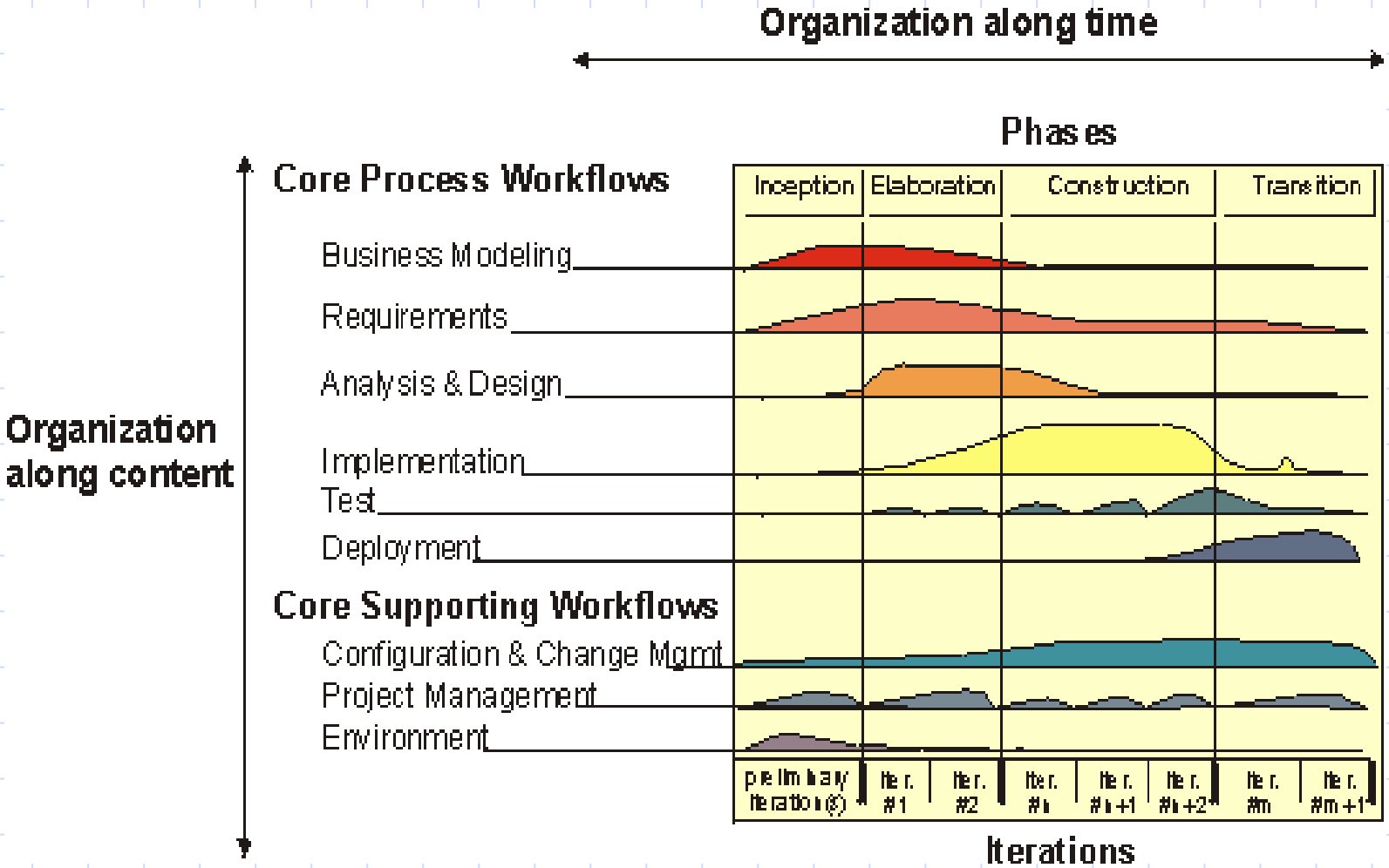


Fasi del progetto

- Fasi (intervalli di tempo tra milestone). Sono le macro-attività del progetto:
  - Inception: stabilire il "business case" e definire ambito e scopo del progetto
  - Elaboration: pianificare il progetto, specificare le "features" e definire le caratteristiche fondamentali dell'architettura
  - Construction
  - Transition: fornire il prodotto agli utenti

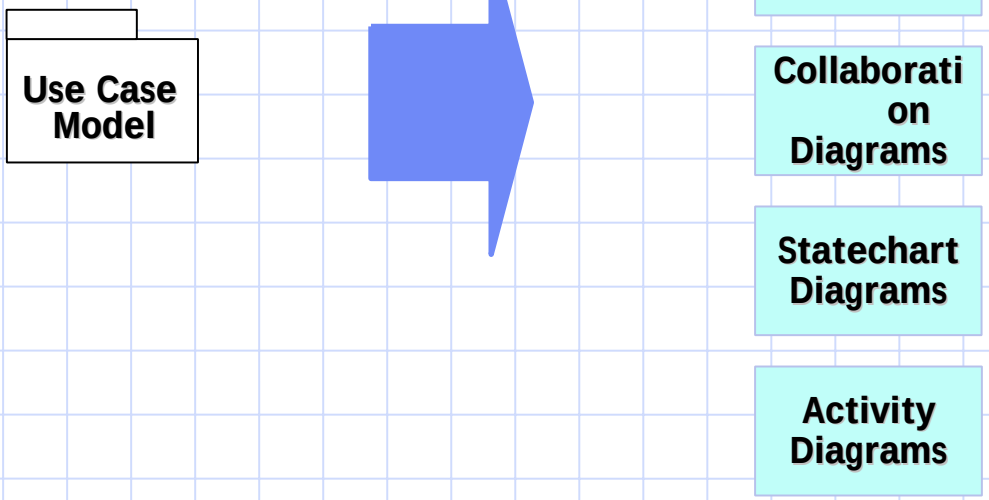


Iterazioni e attività

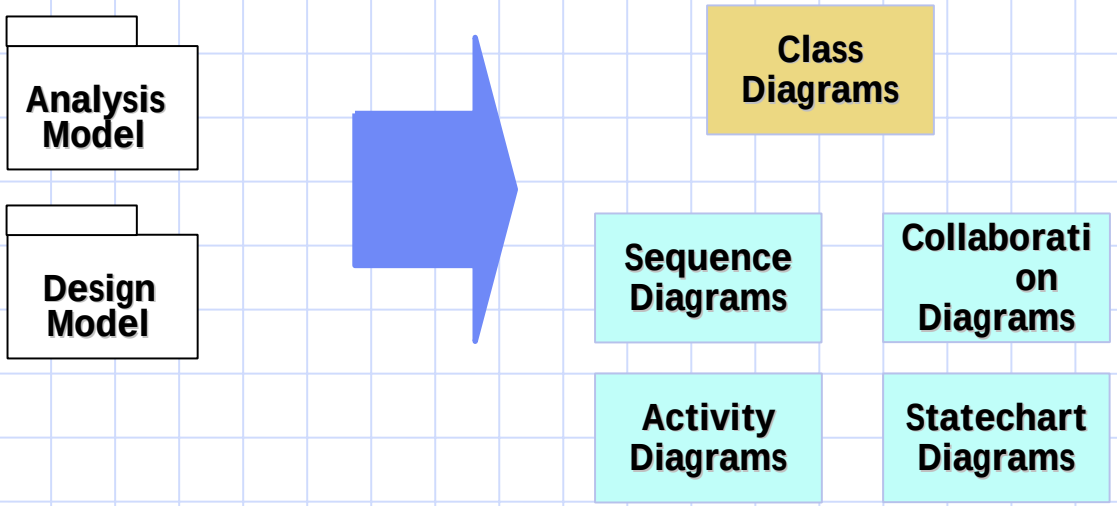


UML: Unified Modeling Language

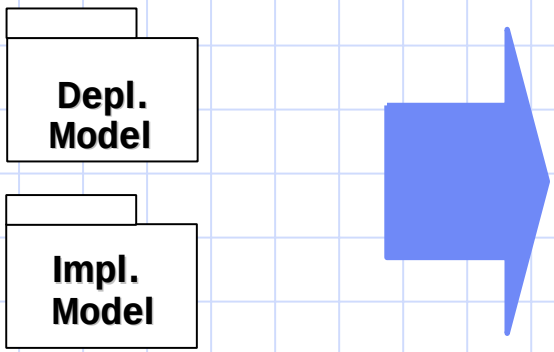
Use Case


Use Case Model
Use Case Diagrams
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams

Analisi e Progetto



Implementazione e Deployment



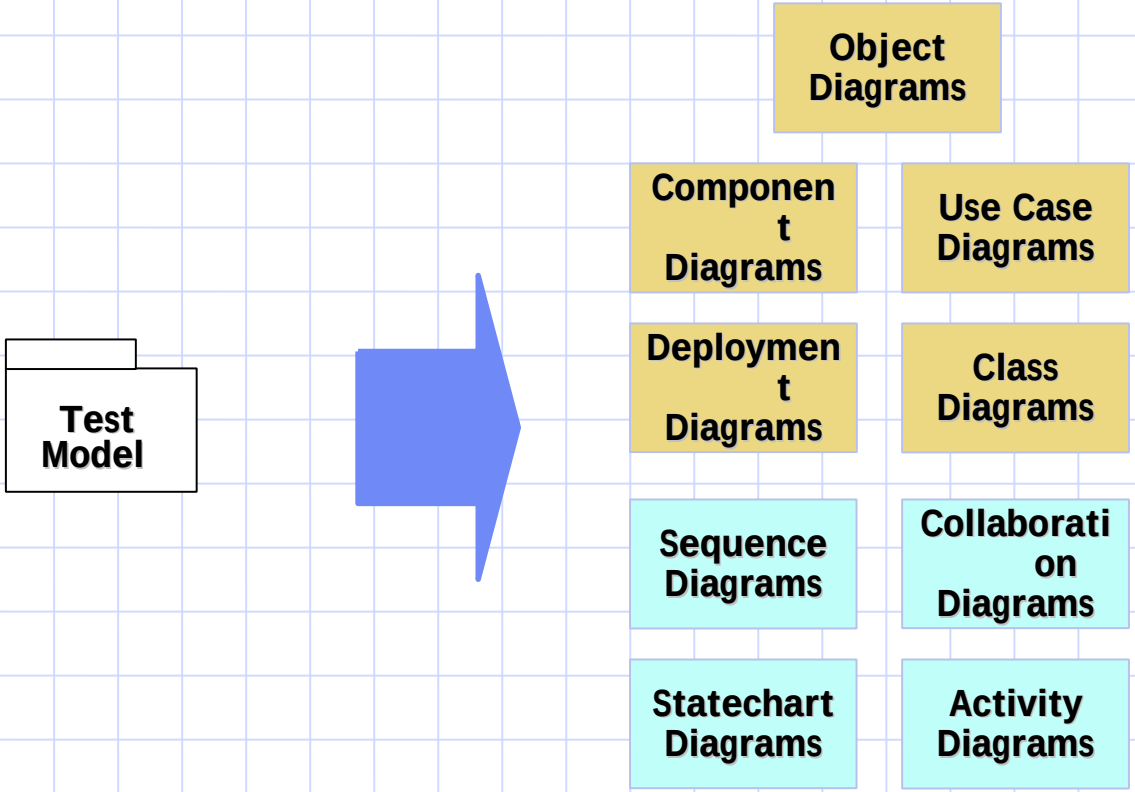
Component Diagrams

Deployment Diagrams

Sequence Diagrams

Collaboration Diagrams

Test


Test Model
Object Diagrams
Component Diagrams
Use Case Diagrams
Deployment Diagrams
Class Diagrams
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams

Alcuni suggerimenti pratici

Un uso "classico" e completo

- Use case diagrams
- Interaction diagrams
- Class diagrams (modello logico)
  - Statechart diagrams
  - Activity diagrams
- Class diagrams (modello fisico)
- Component diagrams
- Deployment diagrams

Sviluppo iterativo

Uso in ambienti legacy

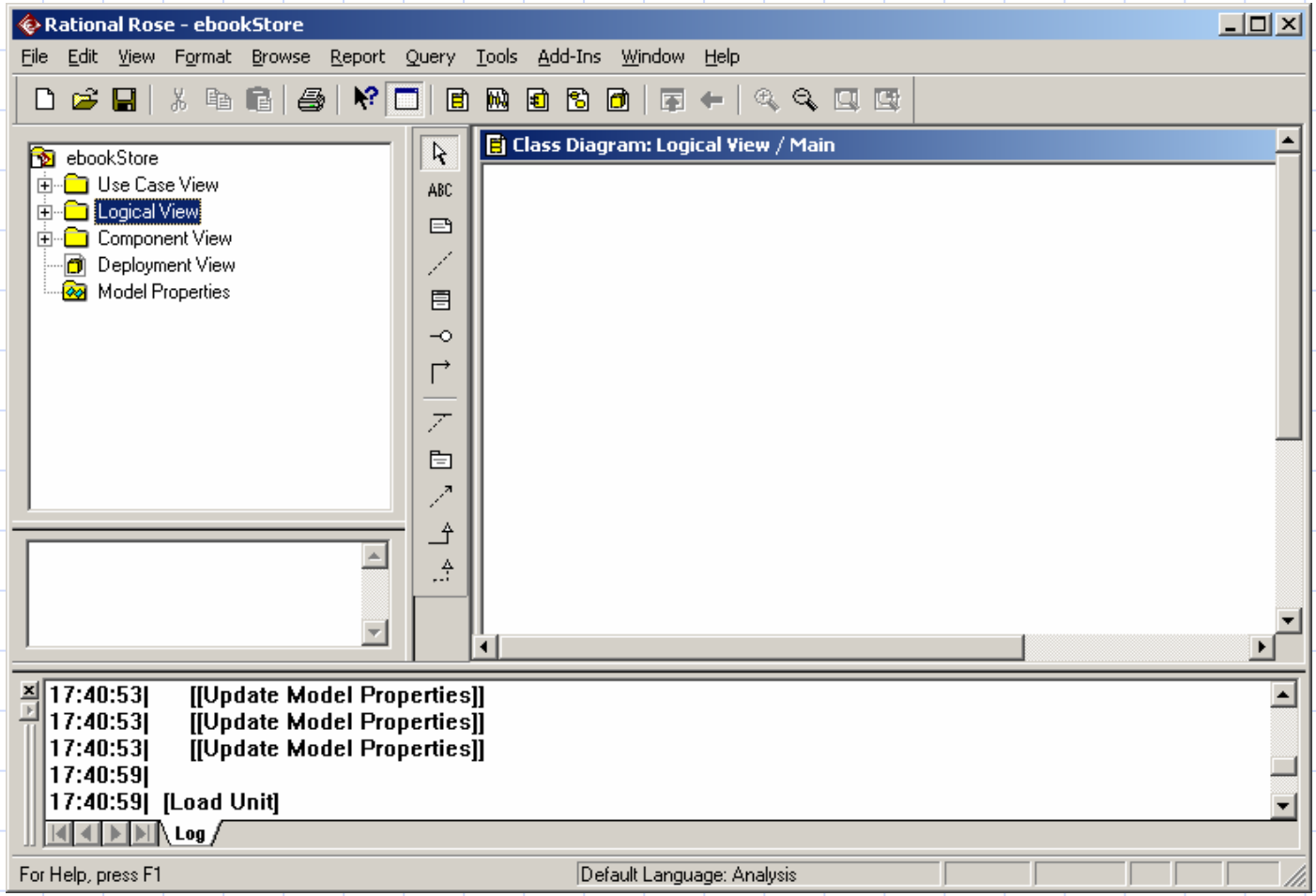
- Analisi
  - con metodologie ad oggetti
- Progetto
  - con metodologie ad oggetti
  - con metodologie funzionali classiche
- Implementazione
  - in C o linguaggio simile

Documentazione

Strumenti di supporto

- Usare UML con carte e matita non ha molto senso
- Esistono diversi tipi di strumenti
  - Semplici editor grafici “generici”
  - Semplici editor grafici con icone predefinite
  - Strumenti CASE
    - Fissi, relativi ad una versione particolare di UML
    - Personalizzabili

Rational Rose



UML: Unified Modeling Language

Bibliografia

Libri

- The Unified Modeling Language User Guide
  G. Booch, J. Rumbaugh e I. Jacobson
  Addison Wesley, 1998
- The Modeling Language Reference Manual
  J. Rumbaugh, I. Jacobson e G. Booch
  Addison Wesley, 1998
- The Unified Software Development Process
  I. Jacobson, G. Booch e J. Rumbaugh
  Addison Wesley, 1999
- UML Distilled
  M. Fowler e K. Scott
  Addison Wesley, 1999
- UML Explained
  K. Scott
  Addison Wesley, 2001

Libri (cont.)

- Visual Modeling with Rational Rose 2000 and UML
  T. Quatrani
  Addison Wesley, 1999
- Use Case Driven Object Modeling With Uml : A Practical Approach
  D. Rosenberg e K. Scott
- Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example
  D. Rosenberg e K. Scott
  Addison Wesley, 2001
- UML for Database Design
  E.J. Naiburg e R.A. Maksimchuk
  Addison Wesley, 2001
- Building Web Applications with UML
  J.Conallen
  Addison Wesley, 2001

I tool più famosi

- Embarcadero
- Rhapsody
- Together J
- Rational Rose
- Objecteering UML Modeler
- UML Suite
- Visio
- ArgoUML
- MetaEdit+

- http://www.objectsbydesign.com/tools/modeling

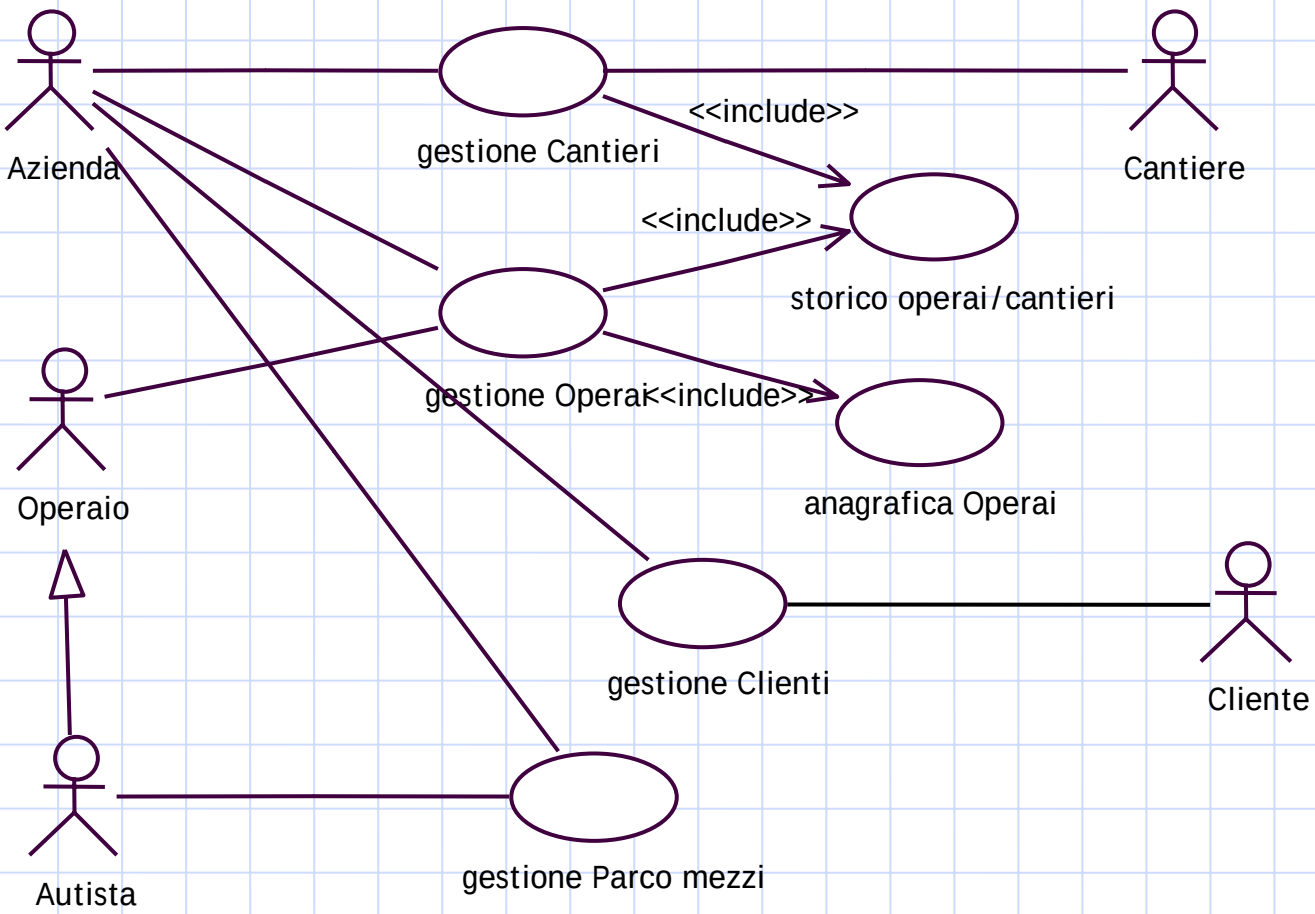
Siti web

- http://www.cetus-links.org
- http://www.rational.com
- http://www.omg.org
- http://www.klasse.nl/ocl/

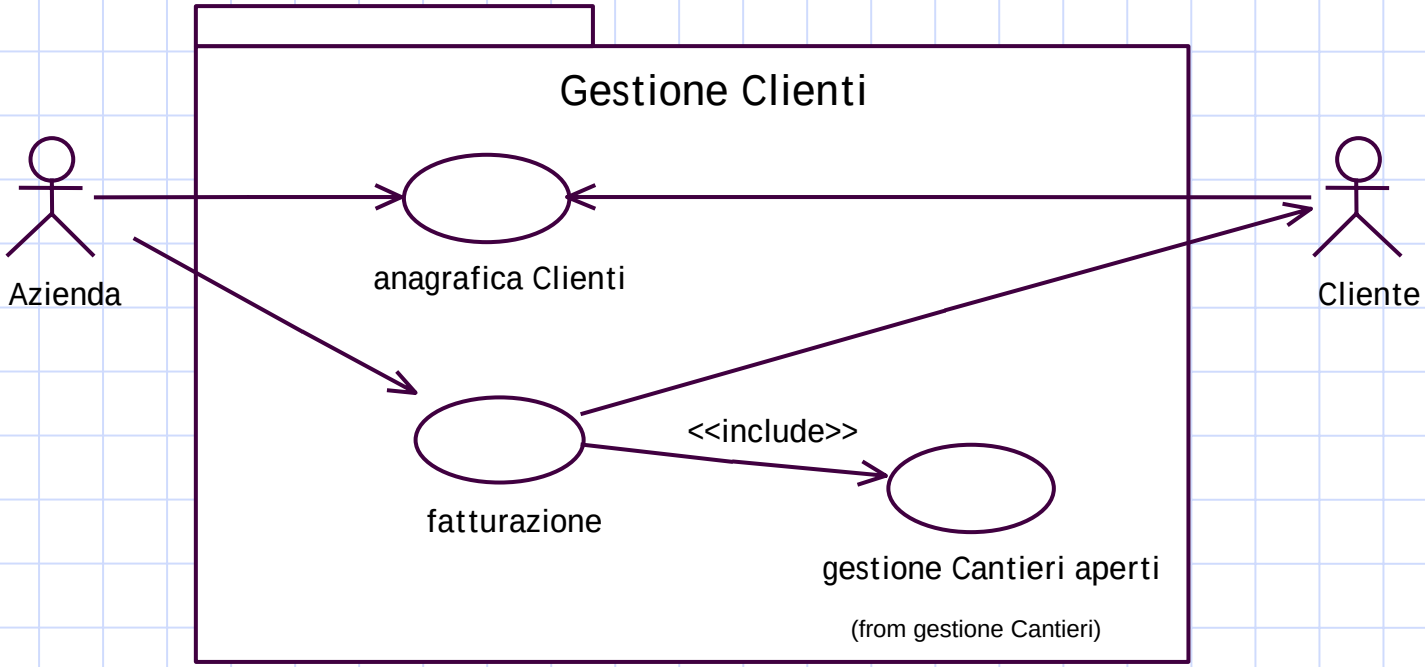
- "Using UML for Modeling a Distributed Java Application“
  K. Bergner, A. Rausch, and M. Sihling
  Technische Universität München, TUM-I9735, July 1997.
  http://www4.informatik.tu-muenchen.de/reports/TUM-I9735.html

Soluzioni esercizi

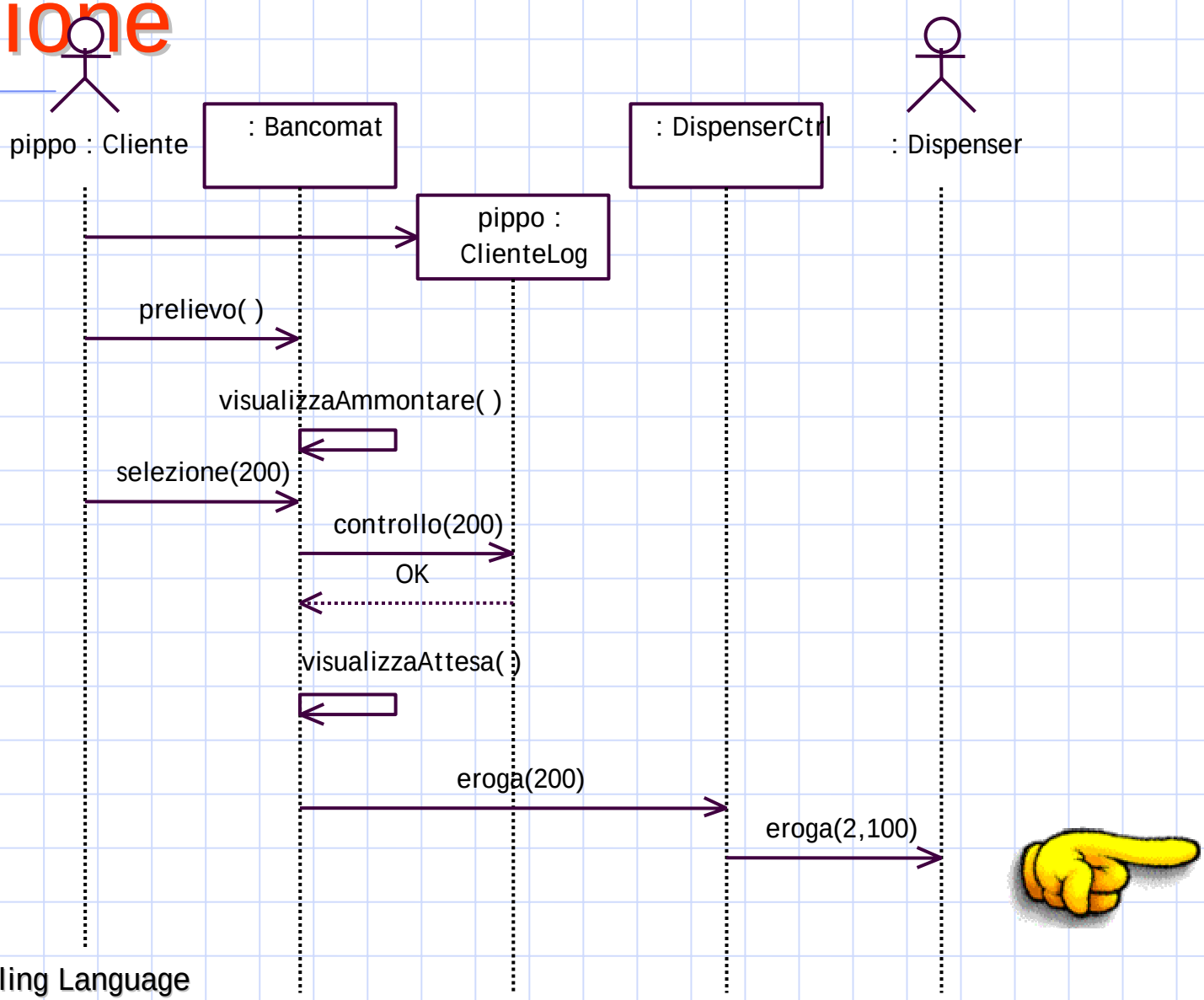
Soluzione


Azienda
gestione Cantieri
<<include>>
Cantiere
<<include>>
storico operai/cantieri
gestione Operai
<<include>>
anagrafica Operai
Operaio
gestione Clienti
Cliente
Autista
gestione Parco mezzi

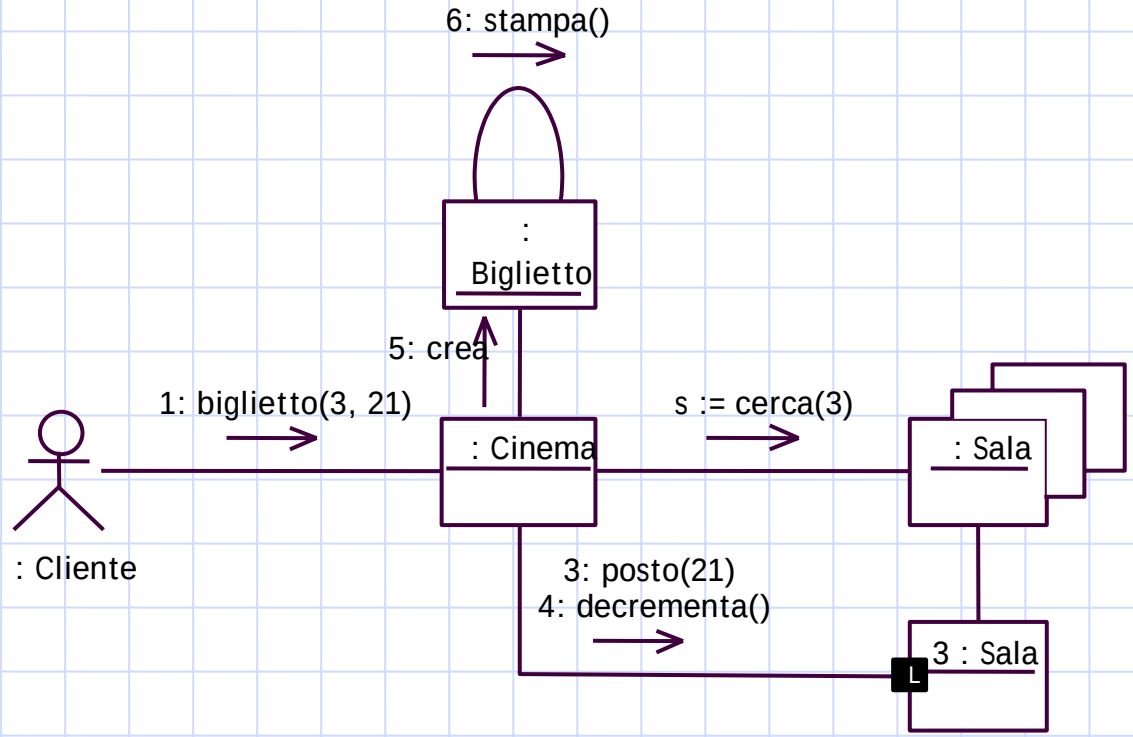
Soluzione
(Gestione Clienti)

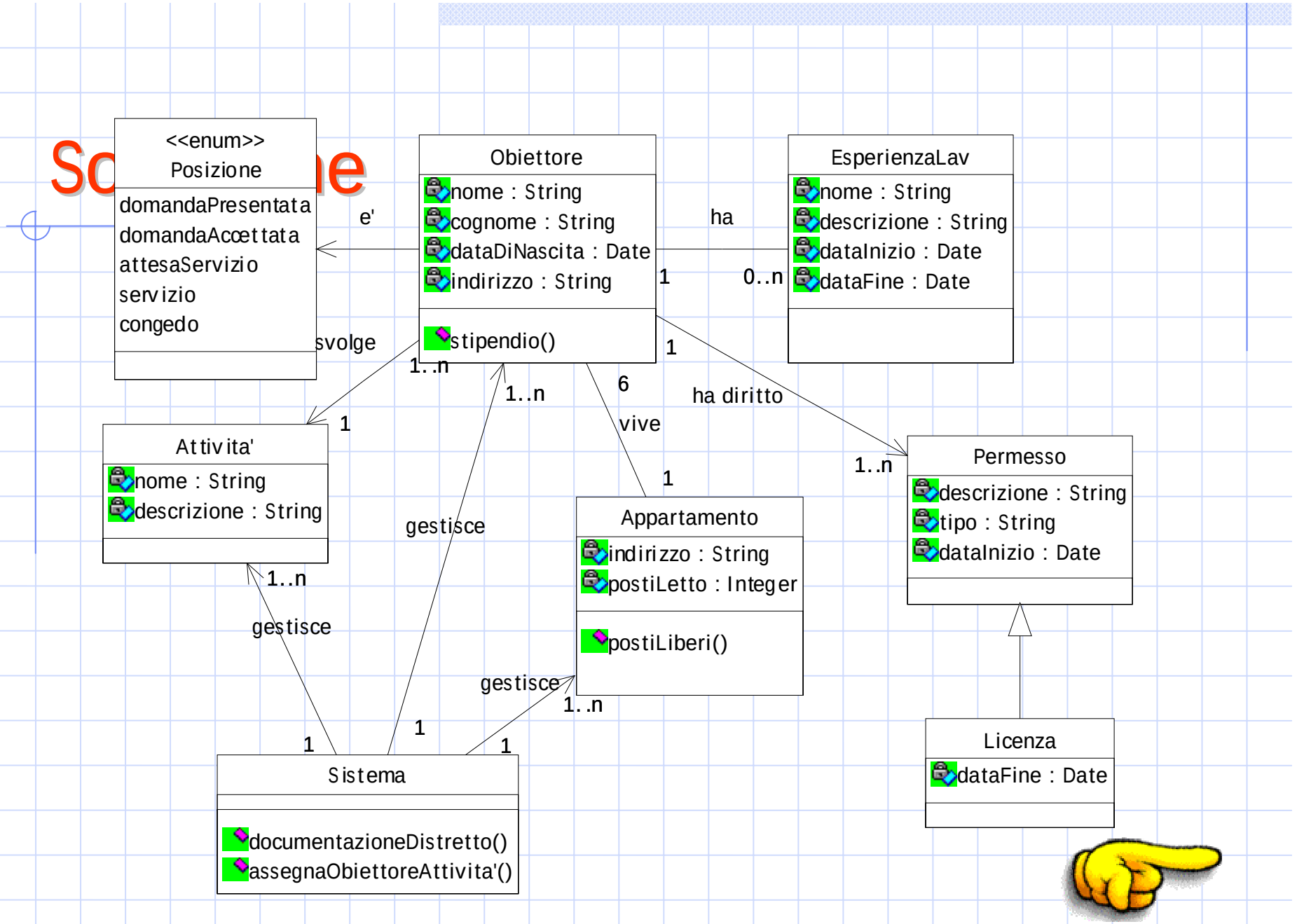


Soluzione

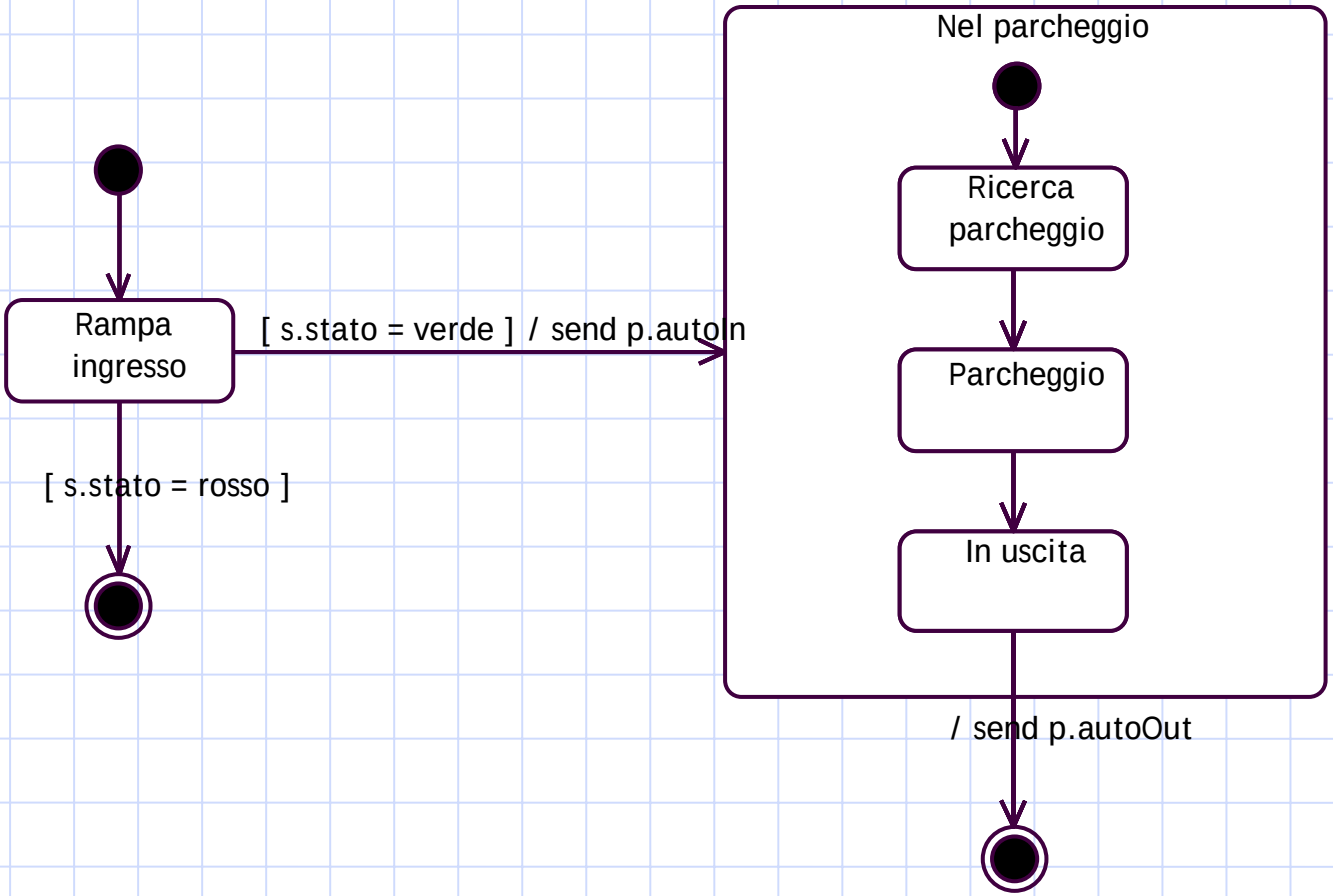

pippo : Cliente
: Bancomat
pippo : ClienteLog
: DispenserCtrl
: Dispenser
prelievo( )
visualizzaAmmontare( )
selezione(200)
controllo(200)
OK
visualizzaAttesa( )
eroga(200)
eroga(2,100)

Soluzione

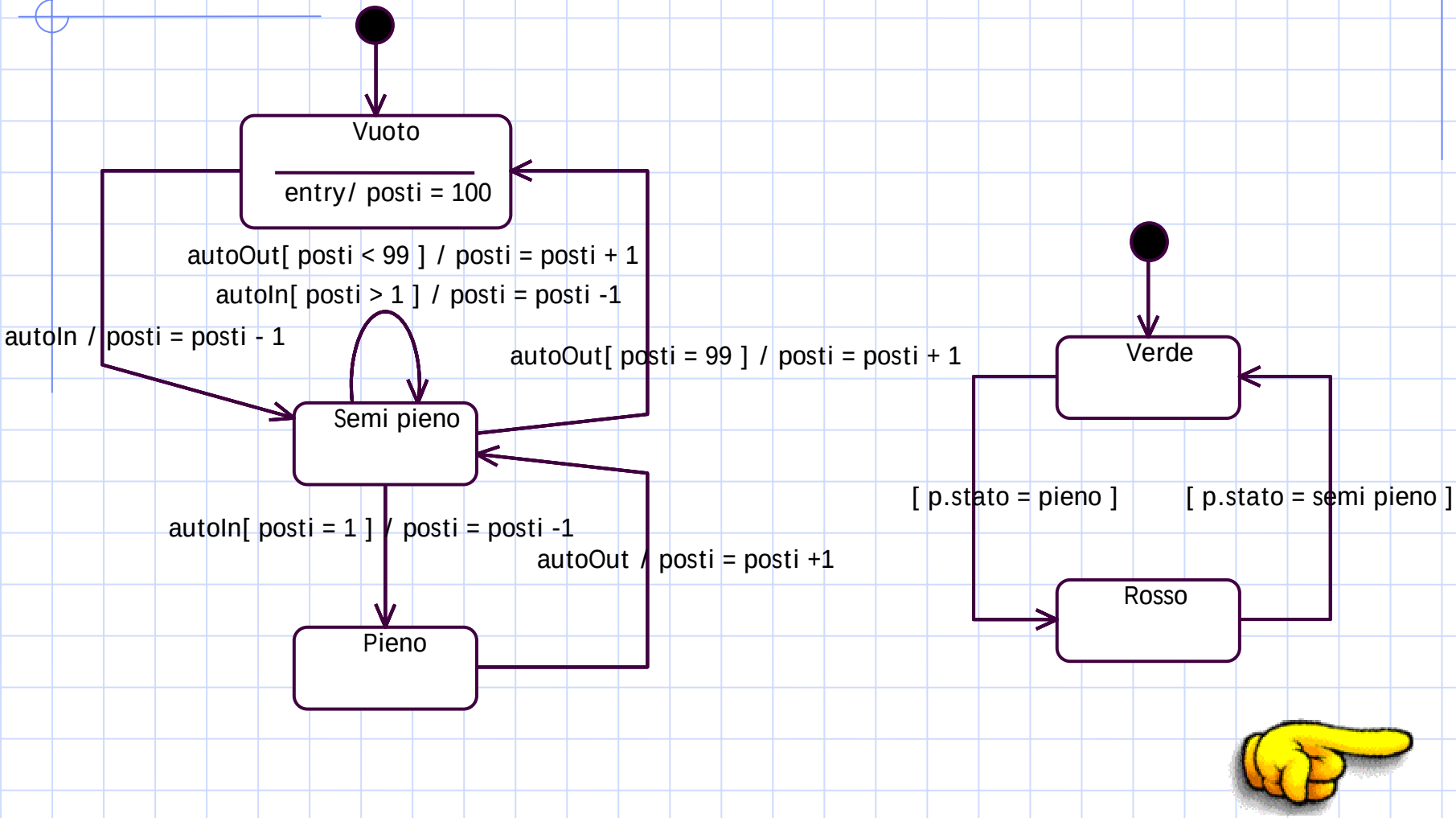

6: stampa()
: Biglietto
5: crea
1: biglietto(3, 21)
s := cerca(3)
: Cliente
: Cinema
: Sala
3: posto(21)
4: decrementa()
3 : Sala
L


<<enum>>
Posizione
domandaPresentata
domandaAccettata
attesaServizio
servizio
congedo
e'
Obiettore
nome : String
cognome : String
dataDiNascita : Date
indirizzo : String
stipendio()
ha
1
0..n
EsperienzaLav
nome : String
descrizione : String
dataInizio : Date
dataFine : Date
svolge
1..n
1
6
ha diritto
1..n
Attivita'
nome : String
descrizione : String
1..n
gestisce
vive
1
Appartamento
indirizzo : String
postiLetto : Integer
postiLiberi()
Permesso
descrizione : String
tipo : String
dataInizio : Date
1..n
gestisce
gestisce
1..n
1
1
1
Sistema
documentazioneDistretto()
assegnaObiettoreAttivita'()
Licenza
dataFine : Date

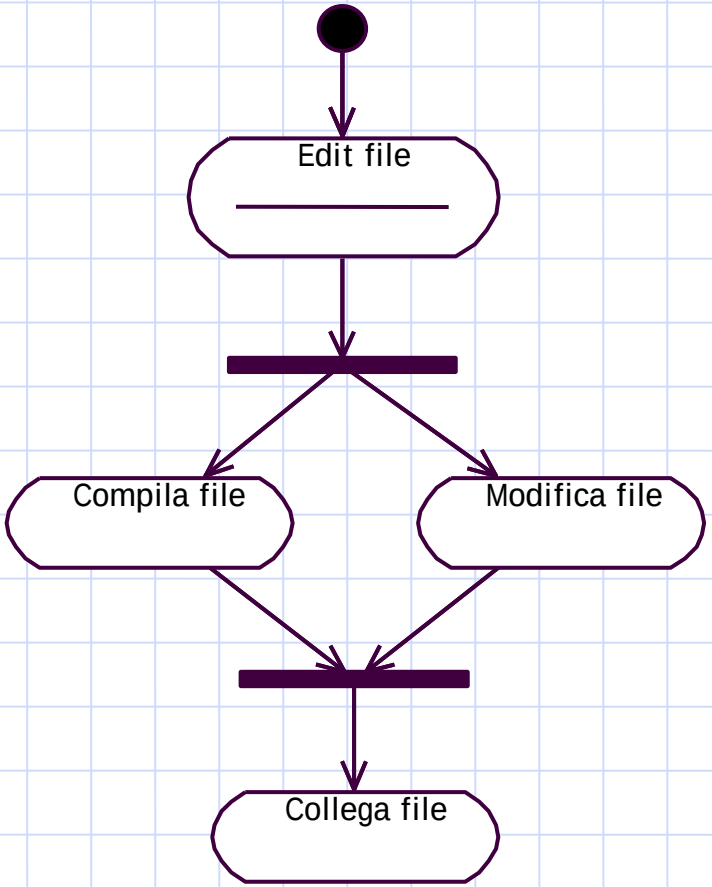
Soluzione
(Automobile)



Parcheggio/Semaforo

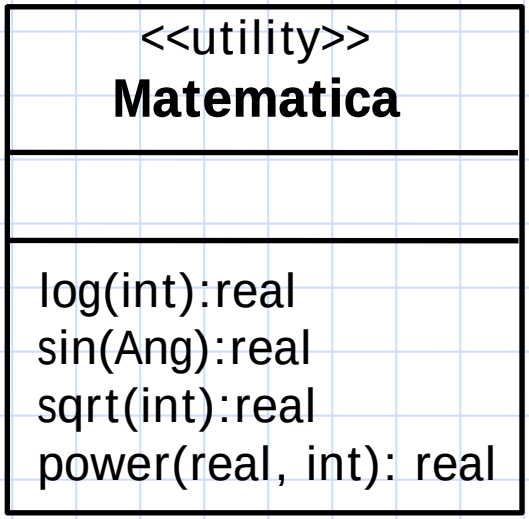


Soluzione



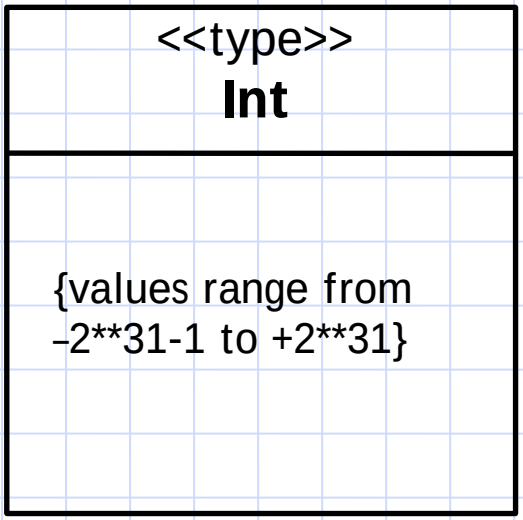
Classi Utility

- Retaggio dei linguaggi procedurali e del C++

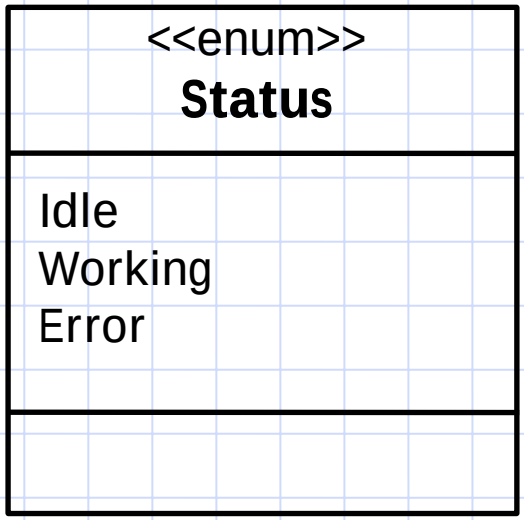


Tipi primitivi

- Tipi

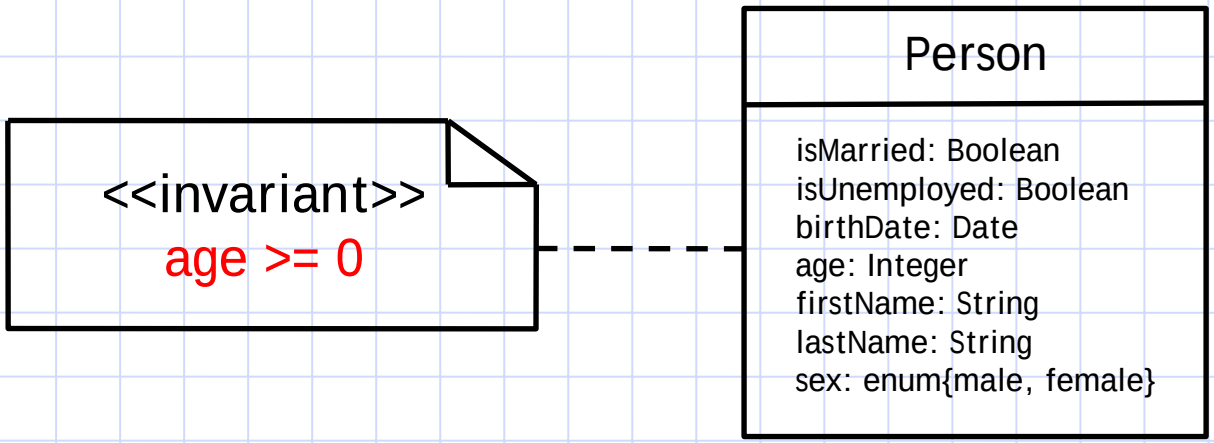


- Enumerativi



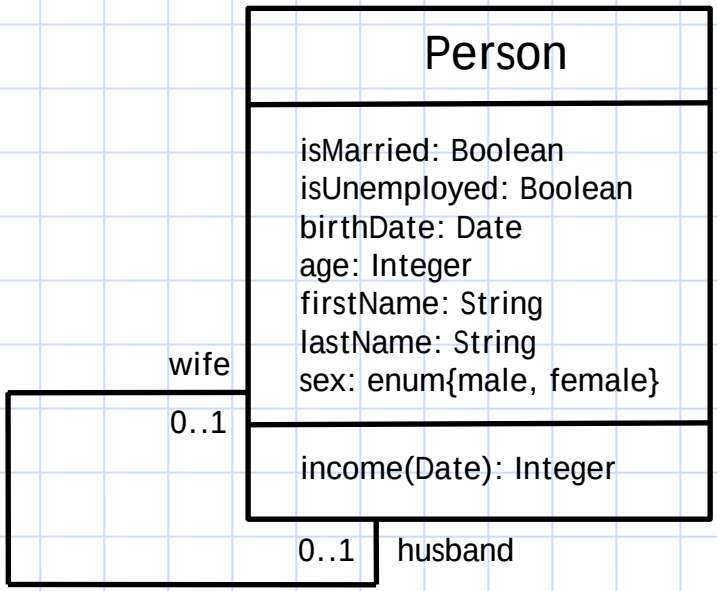
OCL
(Cenni)

- I vincoli si possono esprimere
  - Nei commenti
  - In un documento separato



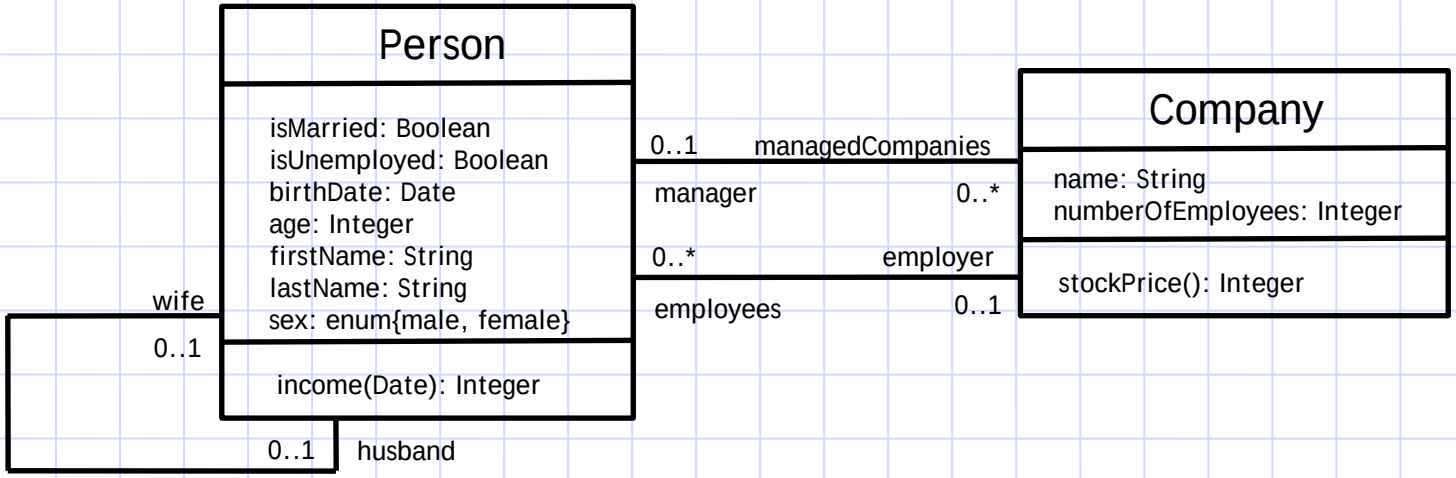
OCL
(Classi e attributi)

- Attributi
  - Context Person
    inv: self.age >= 0
  - Context Person
    inv: age >= 0



- Il context indica a cosa si applica l'espressione
  - Self è implicito (come this in C++)

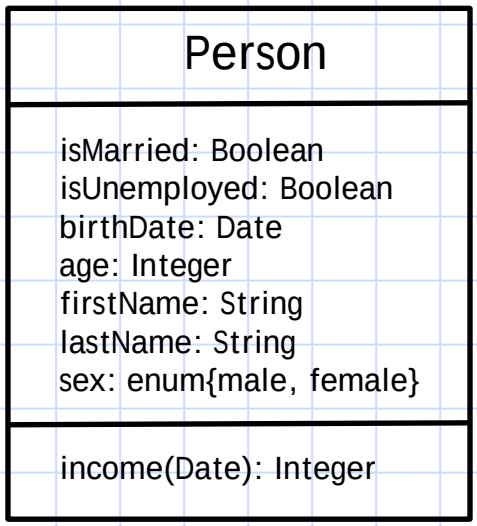
OCL
(Associazioni)



- Ogni associazione costituisce un riferimento "navigabile"
  - Il punto di partenza è dato dal context
  - Si possono usare i nomi dei ruoli che compaiono nel modello
    - Es. employees, manager
  - Context Company
    inv: self.manager.income() >= 0
    inv: self.manager.wife.age >= 18
    inv: self.manager->size...

-> quando la proprietà (attributo) è visto come un set

OCL
(Operazioni)



- Context Person::income(date: Date): Integer
  pre: isUnemployed = false
  post: result = (date.days*100)
- Le postcondizioni possono riferirsi a:
  - Result per indicare il valore restituito dal metodo
  - v@pre per indicare il valore della variabile v all'atto della chiamata

Operazioni sulle collezioni

- Collect
  - Restituisce il sottoinsieme degli elementi per cui expr è vera
- Select
  - Restituisce il sottoinsieme degli elementi per cui expression è vera
- ForAll
  - Restituisce vero se expr è vera per tutti gli elementi della collezione, falso altrimenti
- Exists
  - Restituisce vero se expr è vera per almeno un elemento della collezione, falso altrimenti

Altre operazioni

- isEmpty
  - vero se la collezione non ha elementi
- notEmpty
  - vero se la collezione ha almeno un elemento
- Size
  - numero di elementi contenuti nella collezione
- count(elem)
  - numero di occorrenze dell'elemento dato nella collezione
- includes(elem)
  - vero se elem appartiene alla collezione
- excludes(elem)
  - vero se elem non appartiene alla collezione

Alcuni esempi

- Context Person:
  inv: self.wife->notEmpty implies
  self.wife.age >= 18 and
  self.husband->notEmpty implies
  self.husband.age >= 18
- Context Company:
  inv: self.employees->size <= 50
- Context Person:
  inv: self.wife.sex = #woman and
  self.husband.sex = #man
- Context Company::hireEmployee(p: Person)
  pre: not employee->includes(p)
  post: employees->includes(p) and
  stockprice() = stockprice@pre() + 10