

# System models

---

rappresentare in fase di analisi le domande su cui  
stiamo per operare e rappresentare le caratteristiche  
(specifiche) del sistema SW che stiamo per costruire.  
E servono anche a rappresentare in fase di progettazione

- Abstract descriptions of systems whose requirements are being analysed

(Sommerville + Libro Baruzzo + Slide prof. Tasso)

# System modelling

---

- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers
- Different models present the system from different perspectives
  - External perspective showing the system's context or environment
  - Behavioural perspective showing the behaviour of the system
  - Structural perspective showing the system or data architecture
- Tecniche utilizzate sia in fase di Analisi che di Sintesi (Progettazione)

# Model types

---

- **Model** = abstract representation of the system being modelled.  
Abstraction obtained from different perspectives:
- **Data processing** model showing how the data is processed at different stages
- **Data Semantics** model showing structure and relationships of data
- **Composition** model showing how entities are composed of other entities
- **Architectural** model showing principal sub-systems
- **Classification** model showing how entities have common characteristics
- **Stimulus/response** model showing the system's reaction to events

# Tipologie di modelli esaminati

---

1. **Context models** → adottiamo una visione esterna (quindi l'ambiente esterno ed il contesto dell'applicazione)
  1. Diagrammi contesto → evidenziamo i **confini dei SISTEMI**
  2. Process models → evidenziamo i **confini dei PROCESSI**
2. **Behavioural models**
  1. Data processing models
  2. State machine
  3. Petri nets
3. **Semantic Models**
  1. E-R models
  2. Data Dictionary
4. **Object Models**
  1. Inheritance
  2. Aggregation
  3. Behavioural

→ modelli contestuali

# 1. Context models

---

Adottiamo visione esterna

# Systems (HW+SW+...) and their **environment**

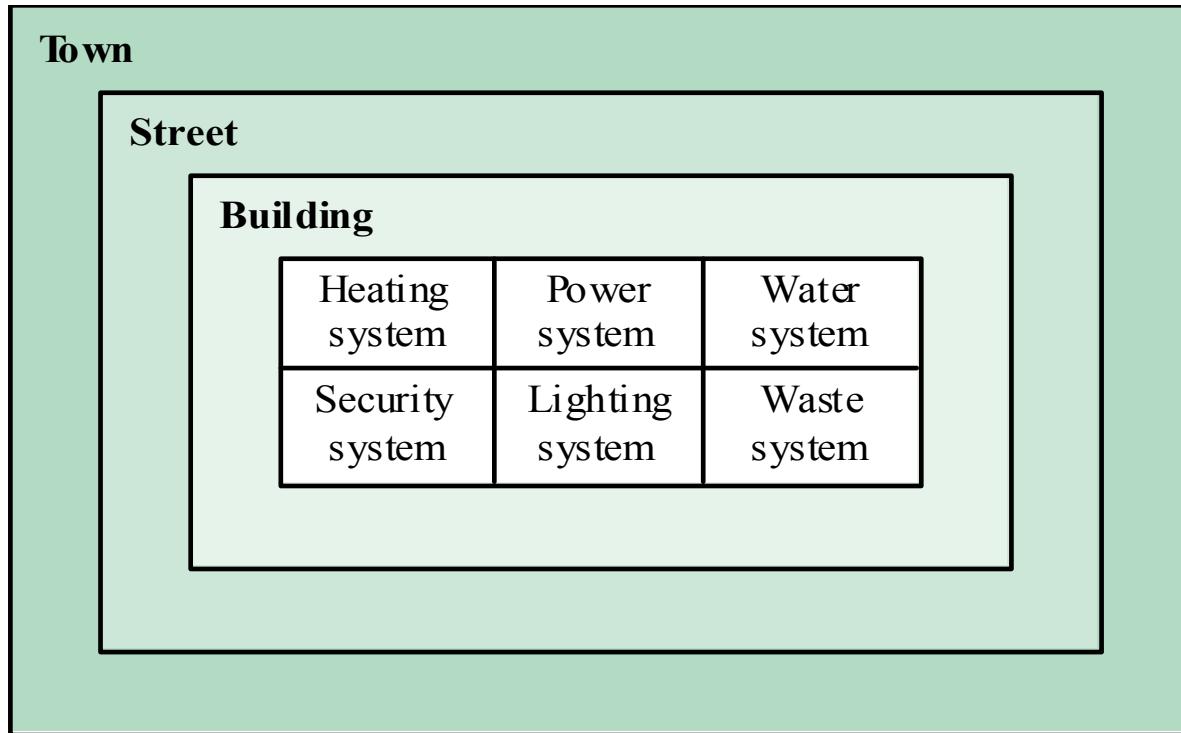
es. la casa che non va senza interazioni con il mondo esterno (es. gas, luce...). Lo stesso avviene per il SW (in che ambiente va inserito?)

- Systems are not independent but exist in an environment
- System's function may be to change its environment
- Environment affects the functioning of the system e.g. system may require electrical supply from its environment
- The organizational as well as the physical environment may be important
- For a **sw system**, the environment can include other SW systems, archives (DB), external devices, communication lines, various kinds of users, etc.

=> TUTTE LE ENITÀ (che fanno parte dell' "AMBIENTE ESTERNO")  
Sono quelle su cui NON ABBIAMO CONTROLLO

↳ Ci deve essere chiaro cosa è dentro al sistema e cosa è fuori al sistema.

# System hierarchies



# Environment of a SW system

---

- For a sw system, the environment can include other SW systems, archives (DB), external devices, communication lines, users, ecc.
- All the entities considered to be part of the environment are characterized by the fact that WE (as designers or as developers) **DO NOT HAVE ANY CONTROL ON THEM!!!**
- Quindi ci deve essere chiaro **cosa è ‘dentro’ al sistema, sotto il nostro controllo** e **cosa è ‘fuori’ dal sistema**, delle cui caratteristiche dobbiamo tenere conto e con il quale dobbiamo interagire  dobbiamo conoscere i confini

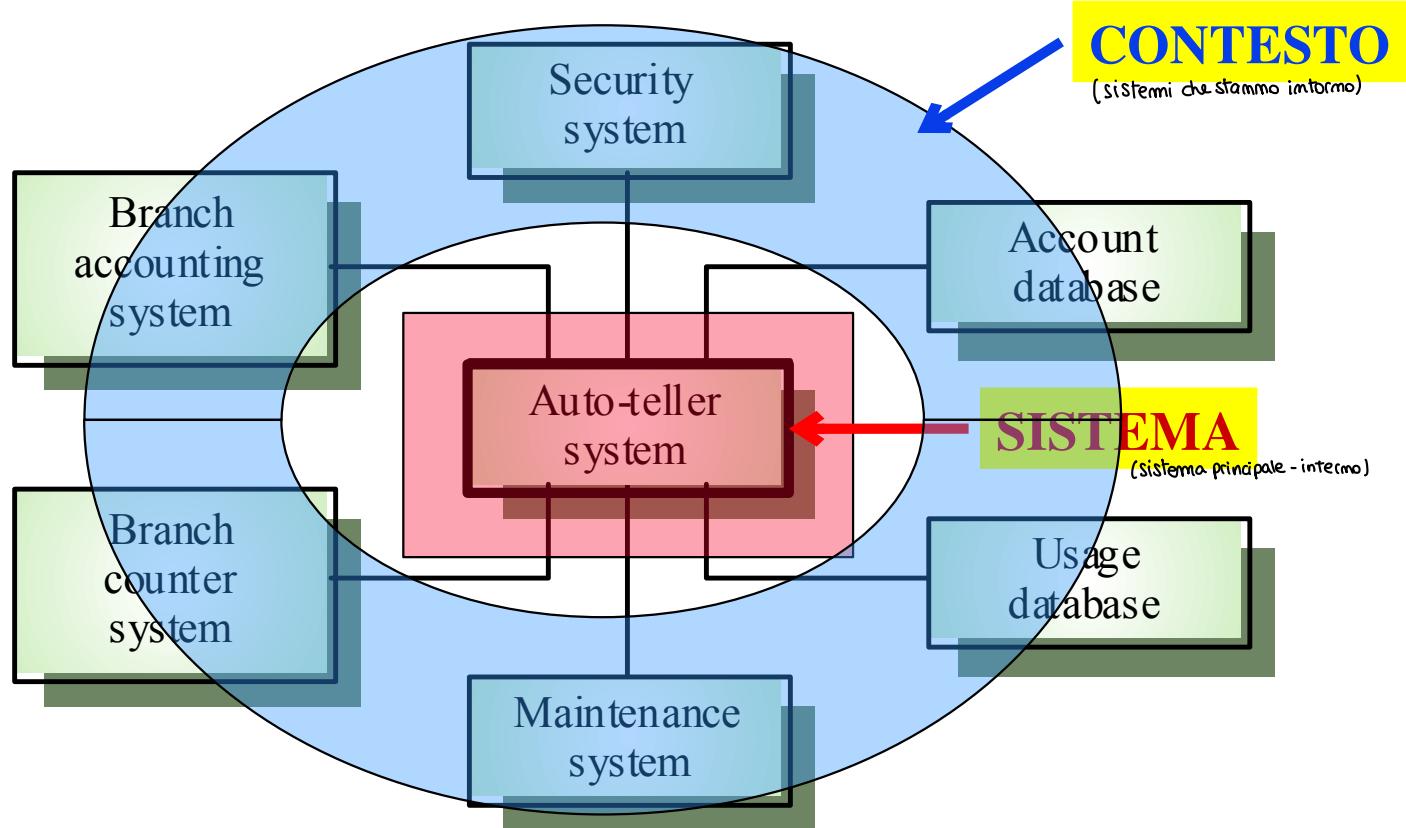
# Cos'è il Modello del Contesto

---

- Context models are used to illustrate the **boundaries** of a system and the **entities external to the system** and **interacting** with the system (such as users, various categories of users, other sw systems, sensors, actuators, archives and data bases, etc.)
- Social and organisational concerns may affect the decision on where to position system boundaries
- Architectural models show the system and its relationship with other systems

# The context of an ATM system

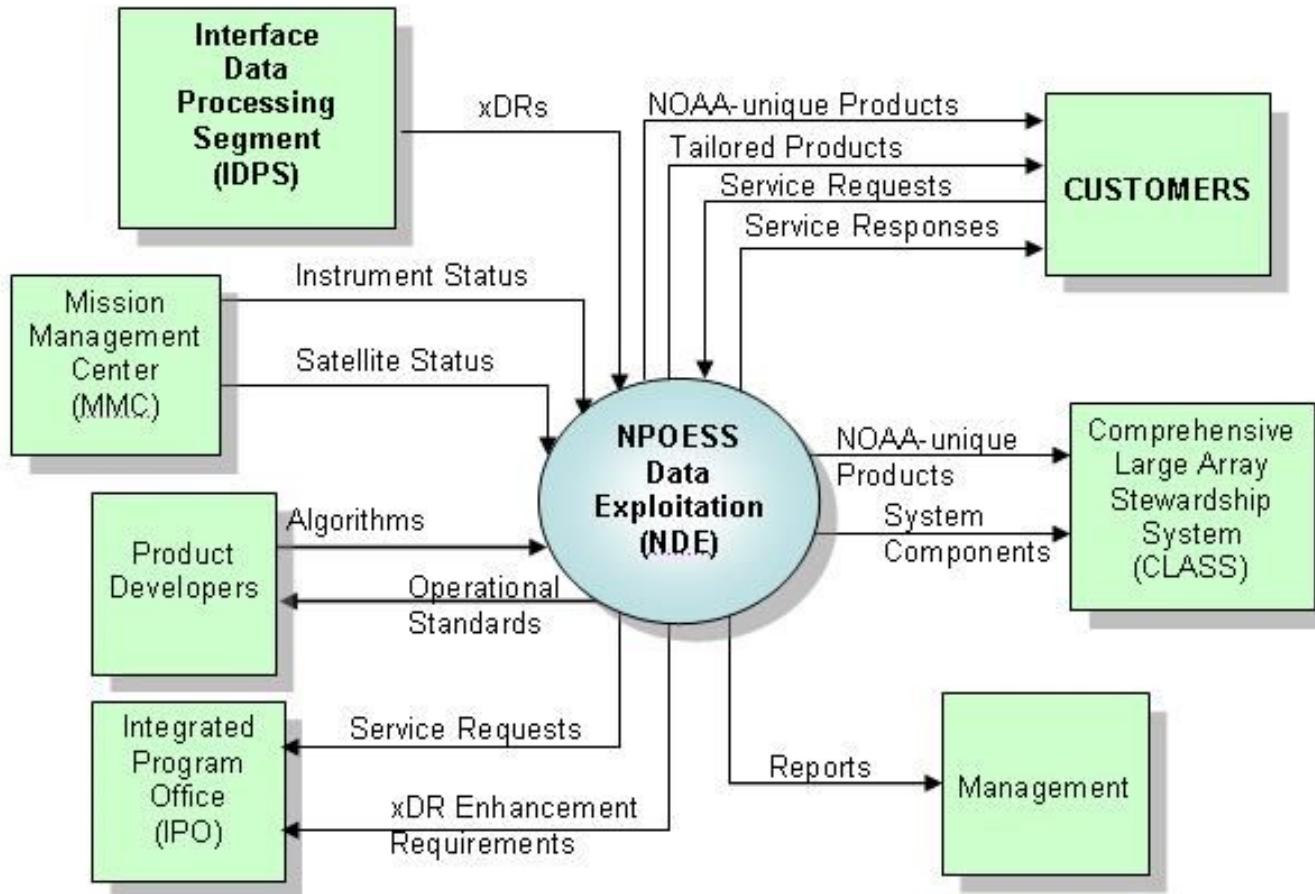
## (1.1 Diagramma di Contesto)



# Example: Data acquisition from satellites

- più dettagli con flussi dati e direzioni -

---



- 
- Dal primo tipo di diagramma di contesto che illustra i sistemi (il sistema sw considerato e i sistemi esterni, inclusi gli utenti, ...)
  - Si passa ad un secondo tipo di diagramma di contesto che riguarda i **PROCESSI**, mettendo in evidenza il/i processo/i eseguiti dal sistema sw e i **processi esterni** con cui il sistema interagisce.

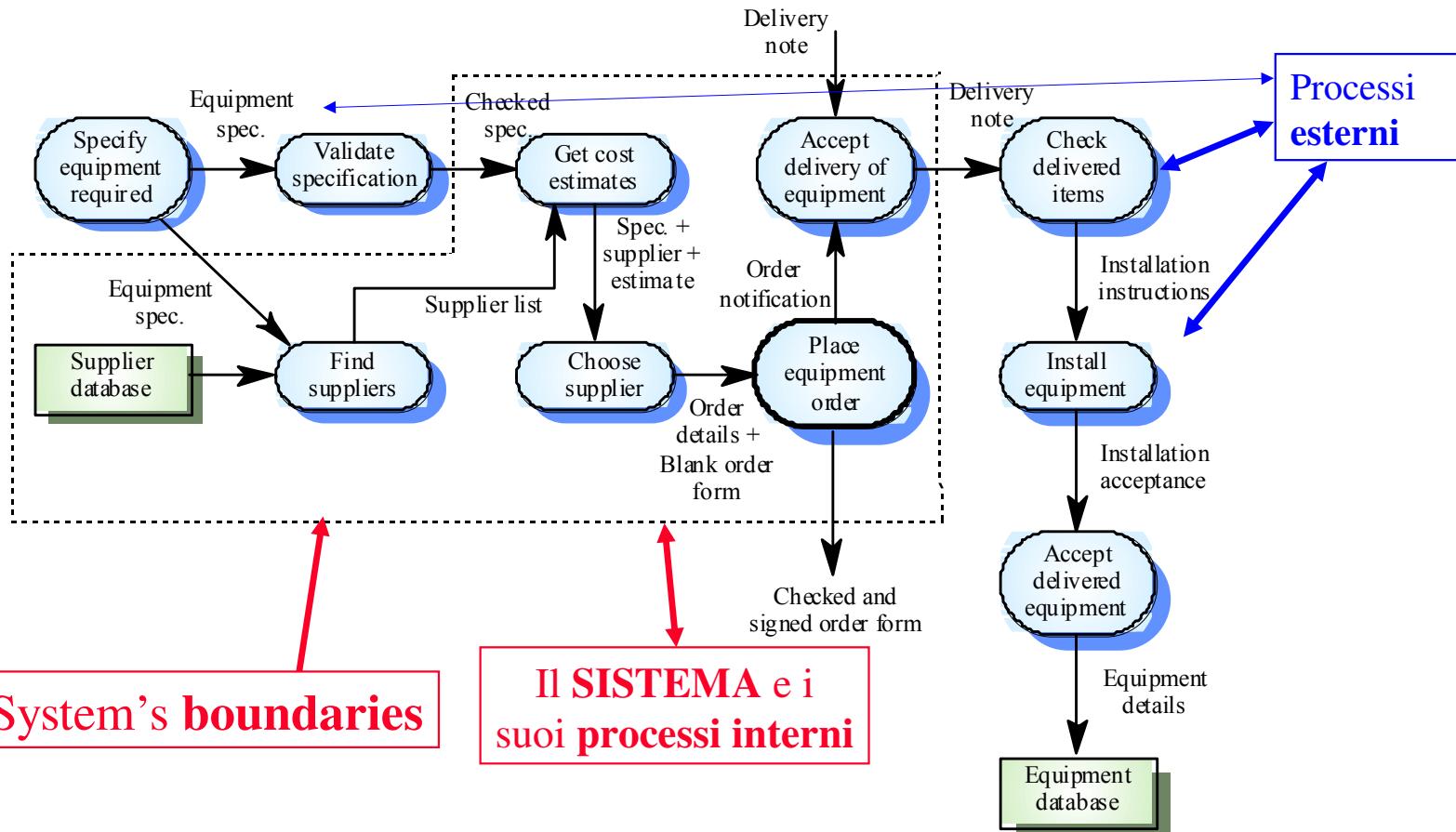
# 1.2 Process models

---

*[abbiamo già visto molti esempi di modelli dei processi software!!]*

- Process models show the overall process and the processes that are supported/external by/to the system
- Un Linguaggio molto diffuso per rappresentare modelli di processi è il **BPNM** (Business Process Modeling and Notation) – non trattato nel corso. Idem per gli Activity Diagram dell'UML – non trattati nel corso.

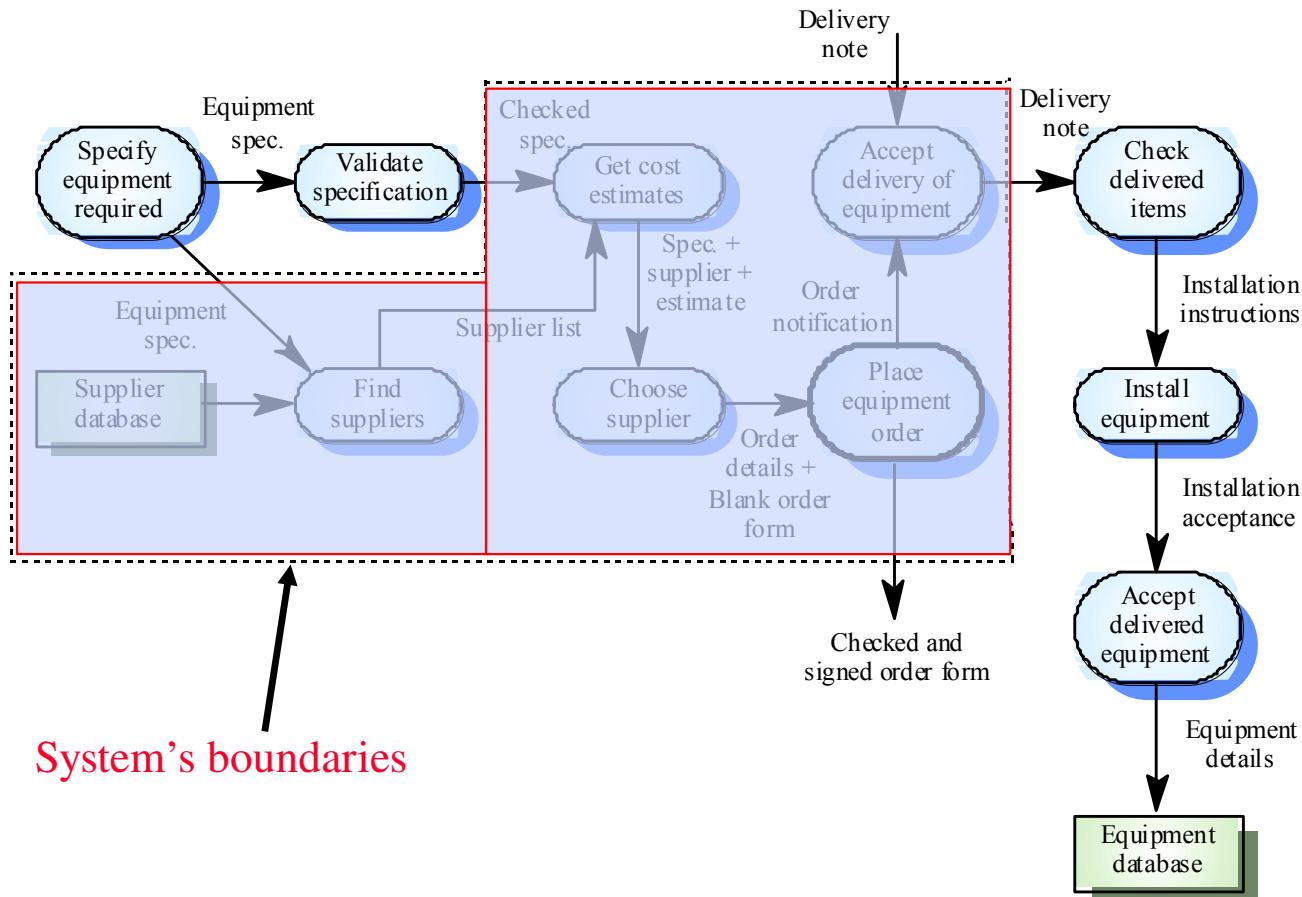
# Esempio di D. del Contesto in termini di processi: Equipment procurement process



System's boundaries

Il SISTEMA e i  
suoi processi interni

# Equipment procurement process



System's boundaries

# Tipologie di modelli esaminati

---

## 1. Context models

1. Diagrammi contesto
2. Process models

## 2. Behavioural models → modelli comportamentali, come funziona e si comporta il singolo sistema?

1. Data processing models → come vengono elaborati i dati lungo tutta esecuzione del sistema?
2. State machine → come risponde il sistema a determinati eventi?
3. Petri nets

## 3. Semantic Models

1. E-R models
2. Data Dictionary

## 4. Objec Models

1. Inheritance
2. Aggregation
3. Behavioural

→ comportamento del singolo sistema

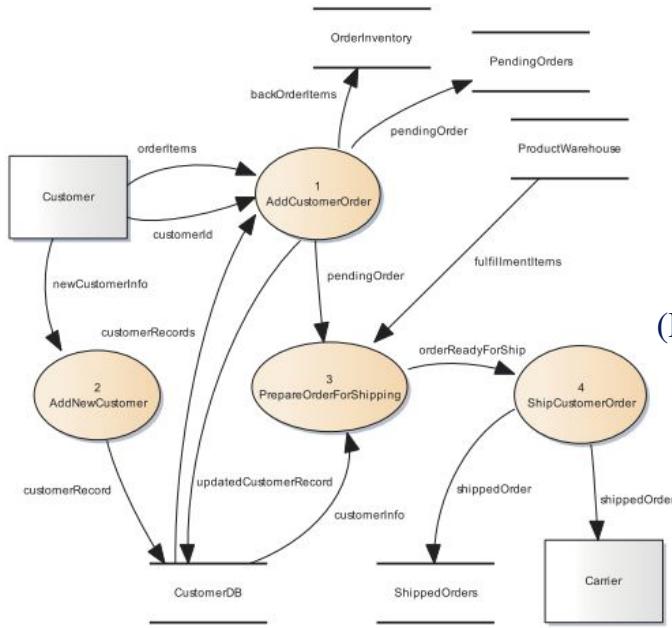
## 2. Behavioural models

---

- Behavioural models are used to **describe the overall behaviour** of a system
- Two types of behavioural model are shown here
  - **Data processing** models that show **how data is processed** as it moves through the system  **DFD**  
→ elaborazione dei dati (come i dati passo passo si trasformano)
  - **State machine** models that show **the systems response to events**  **RdP**  
→ es. Automi a stati finiti
- **Both** of these models **are required** for a description of the system's behaviour

# 2.1 Data-processing models

## DFD - Data flow diagrams



(Libro A. Baruzzo)

Testo di Riferimento per lo Studio:

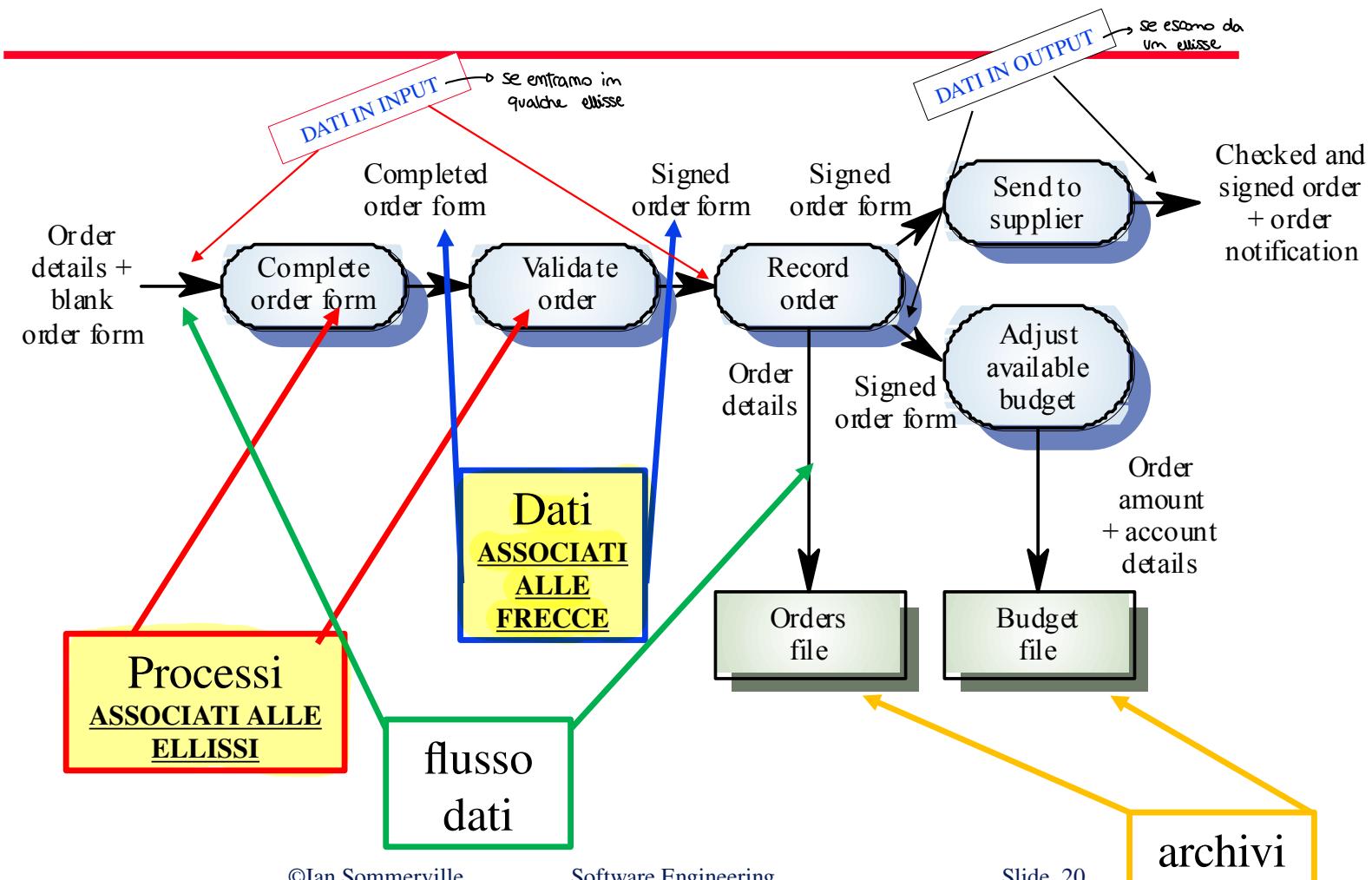
**A. Baruzzo, Analisi e Progettazione di Sistemi Software Industriali – Vol. 1, Create Space, 2017.**  
©Ian Sommerville Software Engineering

## 2.1 Data-processing models

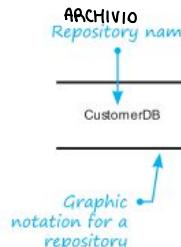
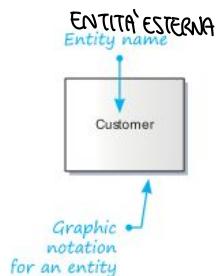
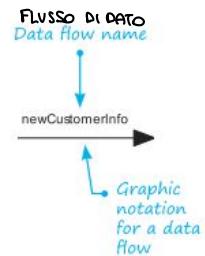
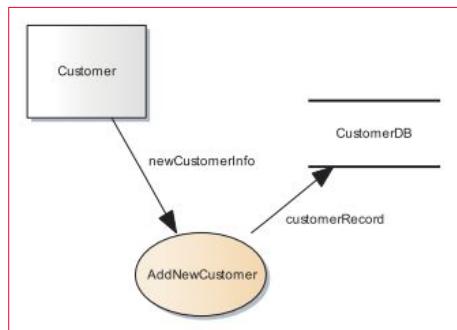
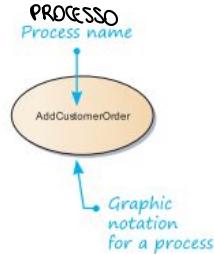
↳ RAPPRESENTANO dall' INIZIO alla FINE tutta l'elaborazione

- **Data flow diagrams** are used to model the system's data processing
  - **Visione puramente FUNZIONALE** (quali input e quali corrispondenti output)
    - ↳ passaggi della trasformaz. dei dati
    - ↳ quale è il processo che li trasforma in output?
- These show the **processing steps** (rappresentati da ellissi) **as data flows** (rappresentati da frecce che congiungono le ellissi) **through a system**
- Denominato **APPROCCIO STRUTTURATO**
- Intrinsic part of **many** analysis methods
- **Simple** and **intuitive** notation that customers can understand
- Show **end-to-end** processing of data
- L'Approccio Strutturato all'analisi e rappresentazione di sistemi è da considerarsi **complementare** all'approccio OO (che trattiamo con l'UML).

# ESEMPIO: Order processing DFD



# Data flow diagrams: NOTAZIONE

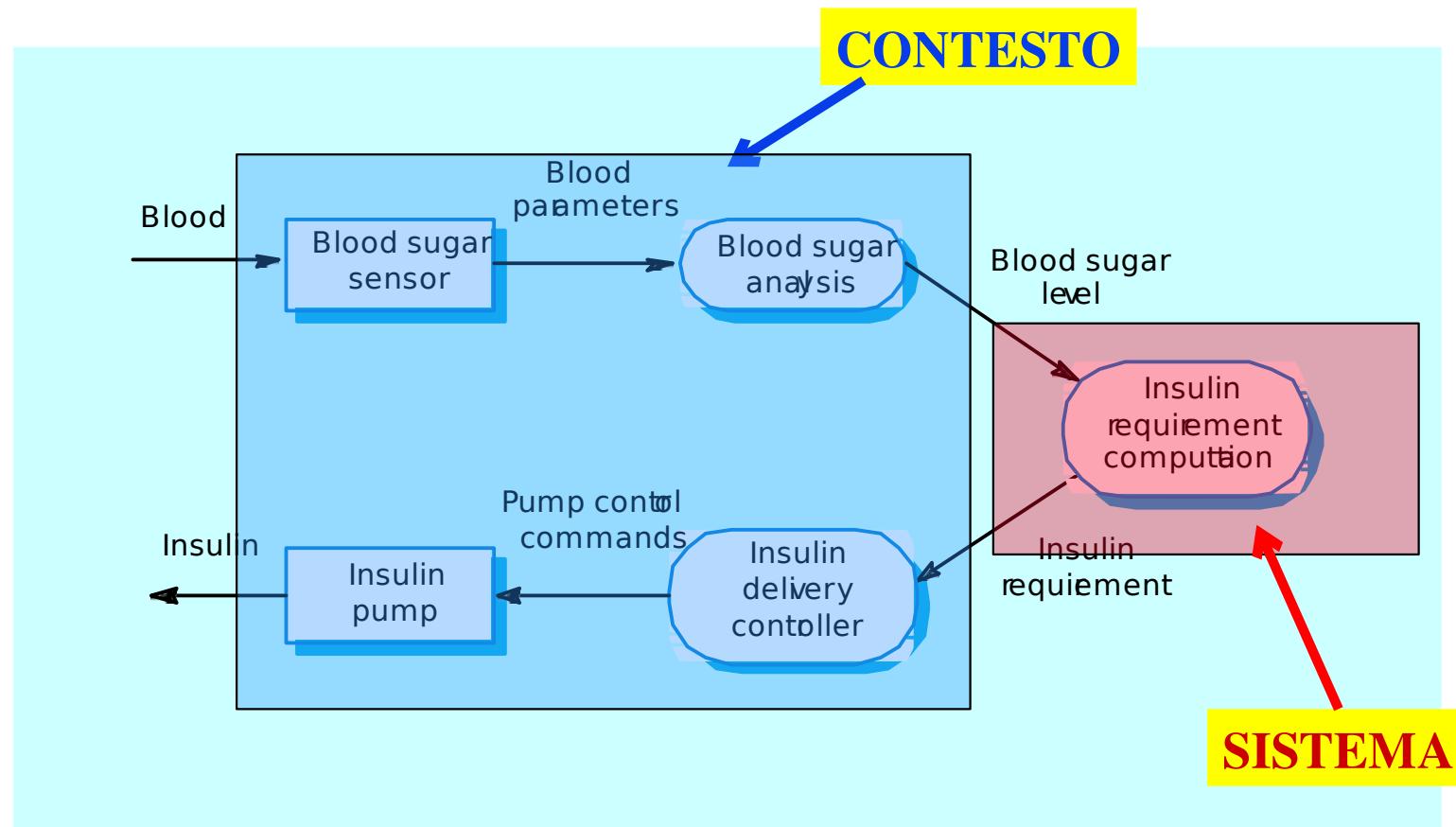


# Data flow diagrams

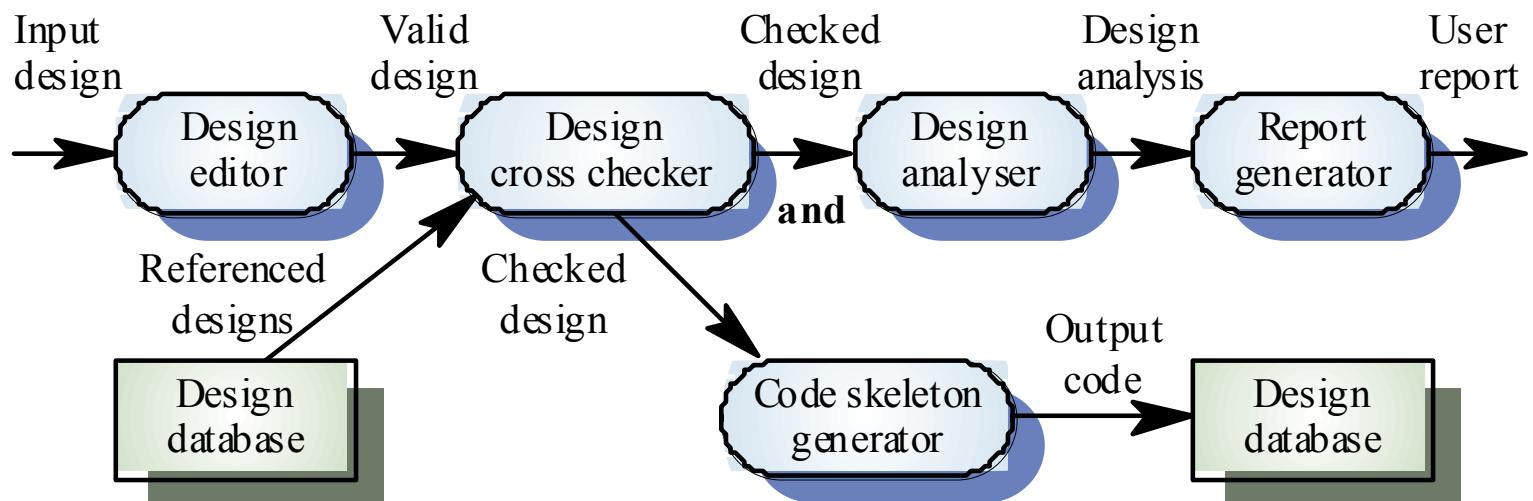
---

- DFDs model the system from a **functional perspective**
- Tracking and documenting how **ALL** the data associated with a process are processed and how **output** are produced: it allows understanding of how the system works
- Data flow diagrams may also be used in showing the **data exchange** between a system and **other systems in its environment/context**

# Insulin pump DFD



# Other example: CASE toolset DFD



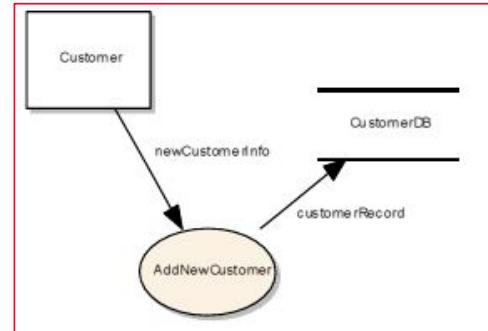
# DFD Guidelines and exercises

---

# DFD – una sintesi per lo studio

## - COME SCEGLIERE I NOMI DA ASSOCIARE A ELLISI E FRECCE -

- Identifica i **dati elaborati**
- Identifica i **processi di elaborazione/trasformazione**, specificandone i **dati in input** e i **dati prodotti in output**, ossia i **flussi** di dati in ingresso e in uscita.
- I **dati** vengono associati ai flussi (graficamente rappresentati dalle **frecce**, denominate anche *pipeline*) e **devono venir descritti mediante nomi che indichino 'dati'/staticità e non processi/dinamicità.** es. **FATTURA**
- I **processi** di trasformazione vengono associati alle **ellissi** e **devono venir descritti mediante nomi che indichino 'processi/attività/esecuzione di ...'/dinamicità e non dati/staticità.** es. **FATTURAZIONE**
- **Archivi** di dati possono venir indicati da **rettangoli** o da linee parallele orizzontali
- **Entità esterne** possono venir rappresentate alle estremità delle frecce entranti o uscenti dal diagramma, utilizzando rappresentazioni iconiche o rettangoli o semplicemente un nome che le denota.



# DFD a più LIVELLI GERARCHICI

- I DFD possono essere costruiti a più livelli gerarchici
- Un singolo processo caratterizzato da precisi Input e Output può essere scomposto nei suoi **sottoprocessi**
- GLOBALMENTE tali sottoprocessi **devono** avere gli **stessi Input e gli stessi Output** (eventualmente descritti ad un più alto livello di dettaglio)



## SCOMPOSIZIONE FUNZIONALE



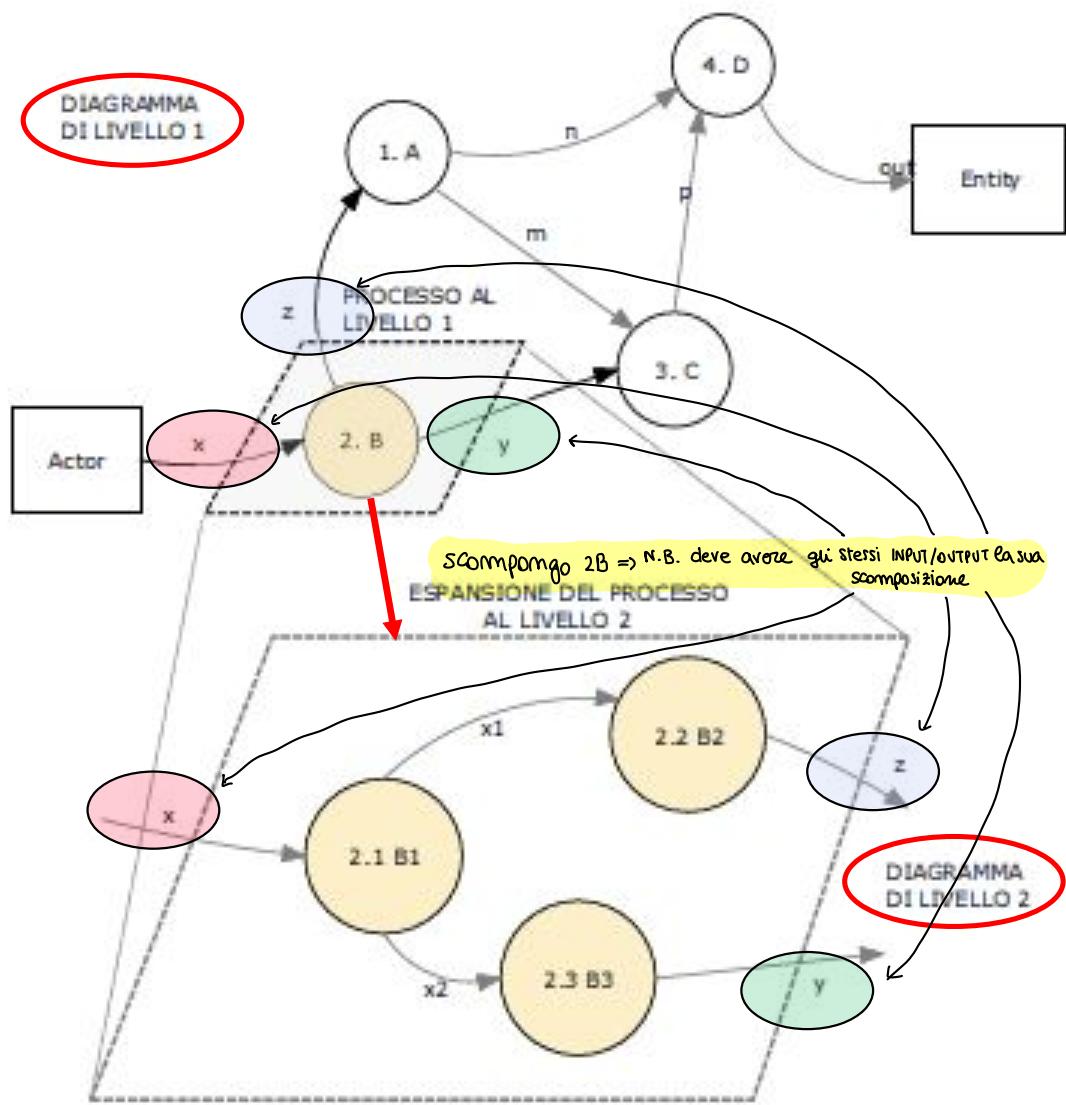
## REGOLA di RRISPONDENZA/CONTINUITÀ DEI FLUSSI

# Esempio: somposizione gerarchica

(Libro A. Baruzzo)

## Suggerimento:

Numerare i processi  
in modo gerarchico



©Ian Sommerville Software Engineering Figura 5.11: Diagramma di secondo livello in un modello DFD Slide 28

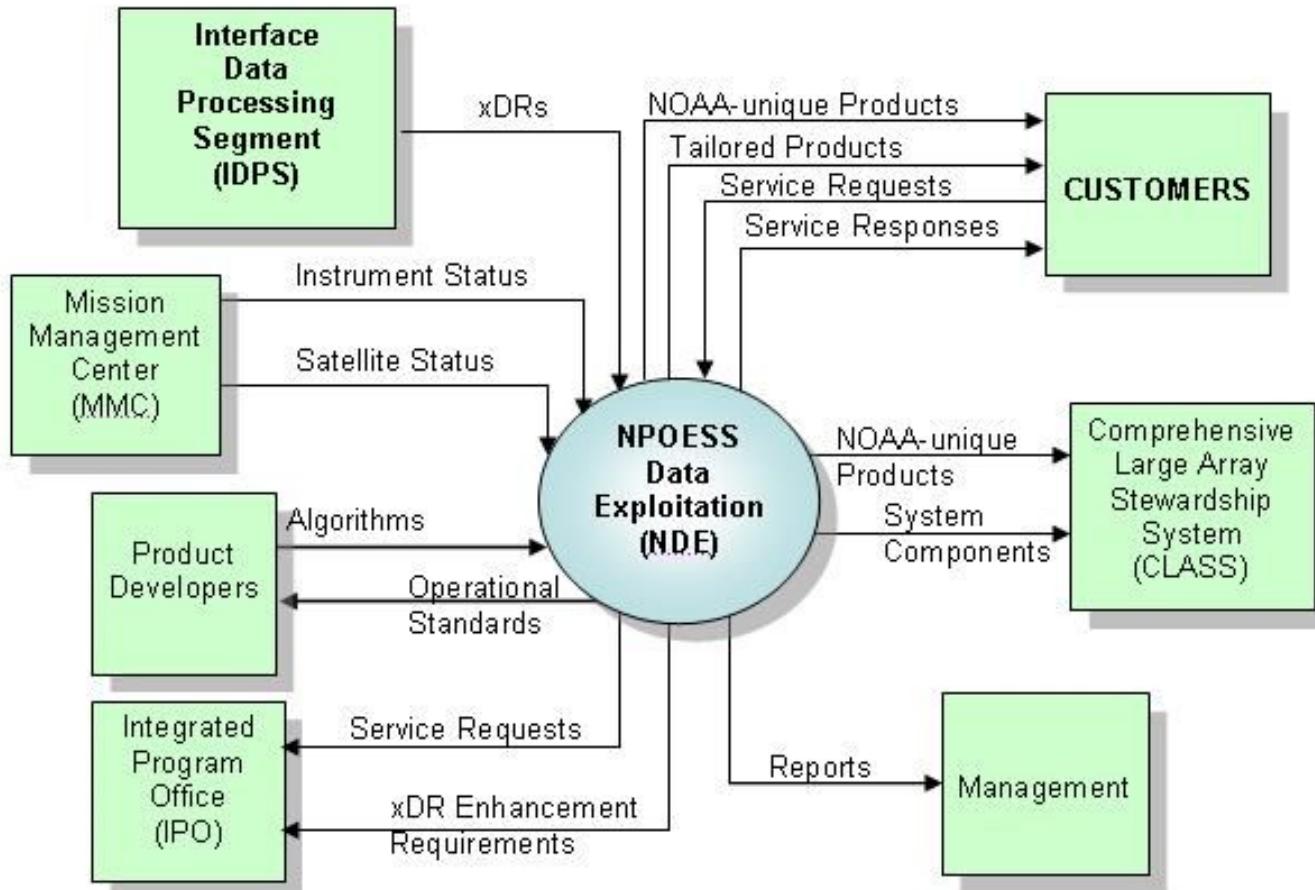
# Diagramma di contesto e DFD di livello 0

---

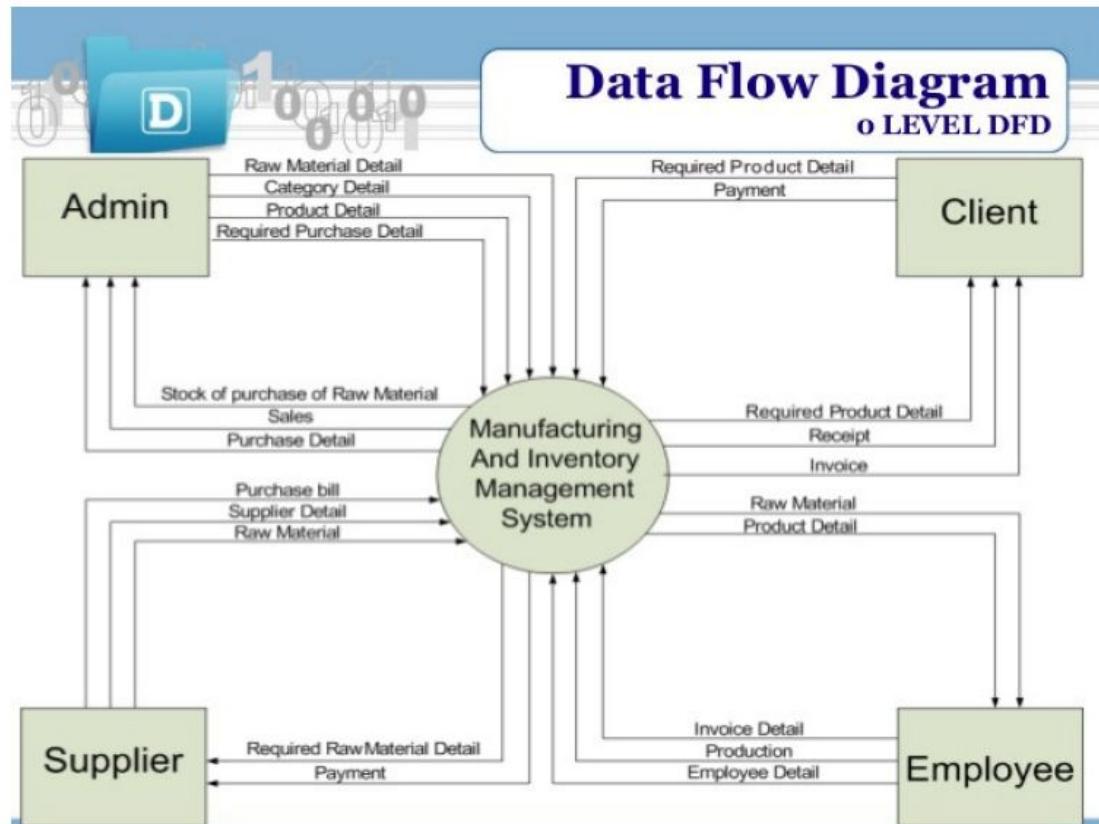
- Il diagramma di *livello 0* ‘corrisponde’ ad un Diagramma di Contesto
  - Nel **livello 0** il sistema è rappresentato con una singola ellisse e con i flussi di dati che vengono scambiati con entità *esterne* al sistema
- So, a context diagram is a data flow diagram, with only one massive central process that subsumes everything inside the scope of the system. It shows how the system will receive and send data flows to the external entities involved.

→ il sistema è un unico ellisse

# Example of a DFD as Context Diagram



# Example of a DFD – Level 0



# Ulteriori Linee Guida

## ASPETTI NON RAPPRESENTATI NEI DFD:

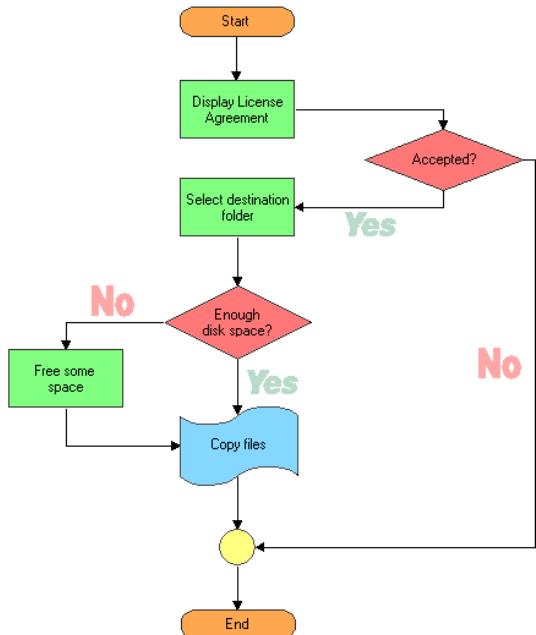
- IGNORARE condizioni e operazioni di **inizializzazione**, di **terminazione**, di gestione degli **errori**  cioè **modellare la situazione ‘stazionaria’/‘stabile’**, non il transitorio, l’inizializzazione, l’avvio, la terminazione, le eccezioni, ...)
- IGNORARE il flusso (la logica) di **controllo** e la sincronizzazione
- IGNORARE i **dati di controllo** (dettaglio molto più implementativo, quindi successivo)
- IGNORARE il **funzionamento interno** di un processo (considerarlo una **black box**), ma modellarne solo gli aspetti funzionali ed i relativi flussi I/O, eventualmente scomporlo a livello più basso
- **VERIFICARE** da ‘**dati**’ a ‘**risultati**’ e viceversa da ‘**risultati**’ a ‘**dati**’

# Data Flow Diagrams e Flow Chart

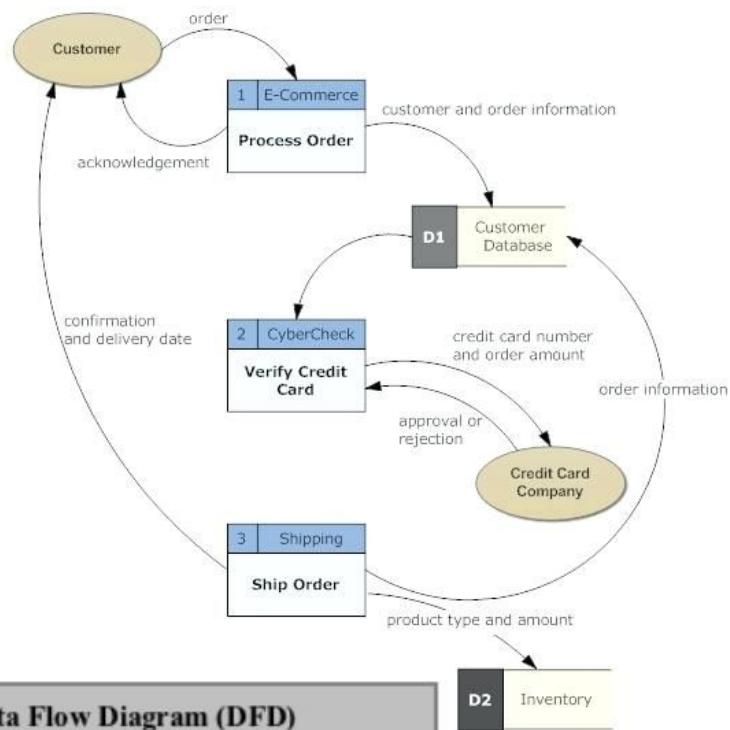
---

- **NON vanno confusi**  
↳ sequenza di trasformazioni dei dati
- I **DFD** descrivono le trasformazioni sui dati, dai dati di **input** ai **risultati di output**. Descrivono il flusso tra una trasformazione (funzione) e l'altra. Sono **FUNZIONALI**.
- I **Flow Chart (Diagrammi di Flusso)** illustrano il **controllo**, quali operazioni vengono fatte prima, quali dopo, e quali test servono per decidere fra flussi diversi. Ciascun blocco descrive un'operazione e tra un blocco ed il successivo non vengono 'passati' dei dati, bensì viene solo indicato quale operazione segue. **NON** sono funzionali.

## Software Installation Flowchart



## Data Flow Diagram - Online Order System



S.No	Flowchart	Data Flow Diagram (DFD)
1	Flow chart presents steps to complete a process	Data flow diagram presents the flow of data
2	Flow chart does not have any input from or output to external source	Data flow diagram describes the path of data from external source to internal store or vice versa.
3	The timing and sequence of the process is aptly shown by a flow chart	The processing of data is taking place in a particular order or several processes are taking simultaneously is not described by a data flow diagram.

Spesso noi programmatori abbiamo una visione mista: modelliamo i dati e descriviamo il flusso delle operazioni che il programma deve fare (es. con costrutti if/else).

Nei DFD non ci vanno if/else, solo le TRASFORMAZ. DEI DATI, non importa cosa fare PRIMA o cosa dopo

---

S.No	Flowchart	Data Flow Diagram (DFD)
1	Flow chart presents steps to complete a process	Data flow diagram presents the flow of data
2	Flow chart does not have any input from or output to external source	Data flow diagram describes the path of data from external source to internal store or vice versa.
3	The timing and sequence of the process is aptly shown by a flow chart	The processing of data is taking place in a particular order or several processes are taking simultaneously is not described by a data flow diagram.

# DFD: sincronizzazione e controllo



**CHE POSSIBILI SINCRONIZZAZIONI  
RAPPRESENTA?** cioè, chi fa cosa e quando?

Diverse possibili alternative: (di cosa vuole dirci il disegno) che però non si considerano nei DFD

## 1. Relazione **PRODUTTORE-CONSUMATORE**:

- I. A produce un dato che viene consumato da B
  - II. B attende il dato prodotto da A
  - III. A attende che B abbia terminato prima di produrre un nuovo dato
2. **A controlla B.** B usa il dato prodotto da A, più DI UNA VOLTA, senza ‘consumarlo’ (ad esempio B ha altri ingressi e l’uscita di A non è ‘consumata’). Se A produce un nuovo dato, allora B lo considera

→ ovvero usa delle sue istanze

# DFD: sincronizzazione e controllo

---

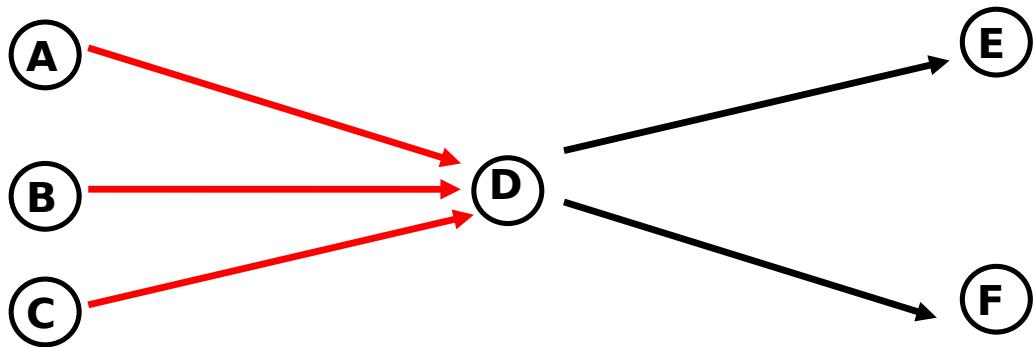


3. **A e B hanno VELOCITÀ DIVERSE!** Vengono collegati da una coda, per evitare perdite o duplicazioni di dati

**NB.** Il DFD non indica quale delle 3 alternative va considerata. Questa informazione non è inclusa nei DFD e va indicata in altro modo.

# DFD: uso dei dati

---

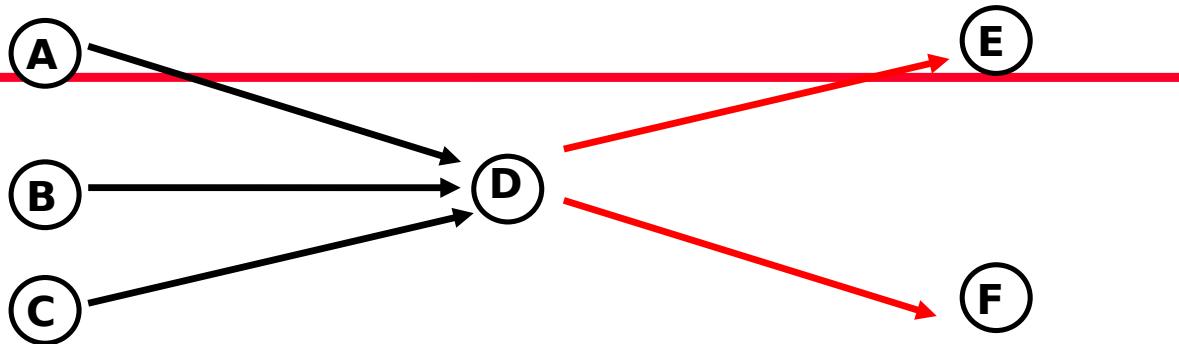


→ D si prende tutti i dati? certe volte me usa solo uno?  
NON CONTA nel dfd

Diverse possibili alternative:

- D è attivata dalla presenza di tutti i dati di ingresso, o
- E' sufficiente solo UNO dei dati in ingresso per l'attivazione di D, o ...
- Altre combinazioni più complesse....

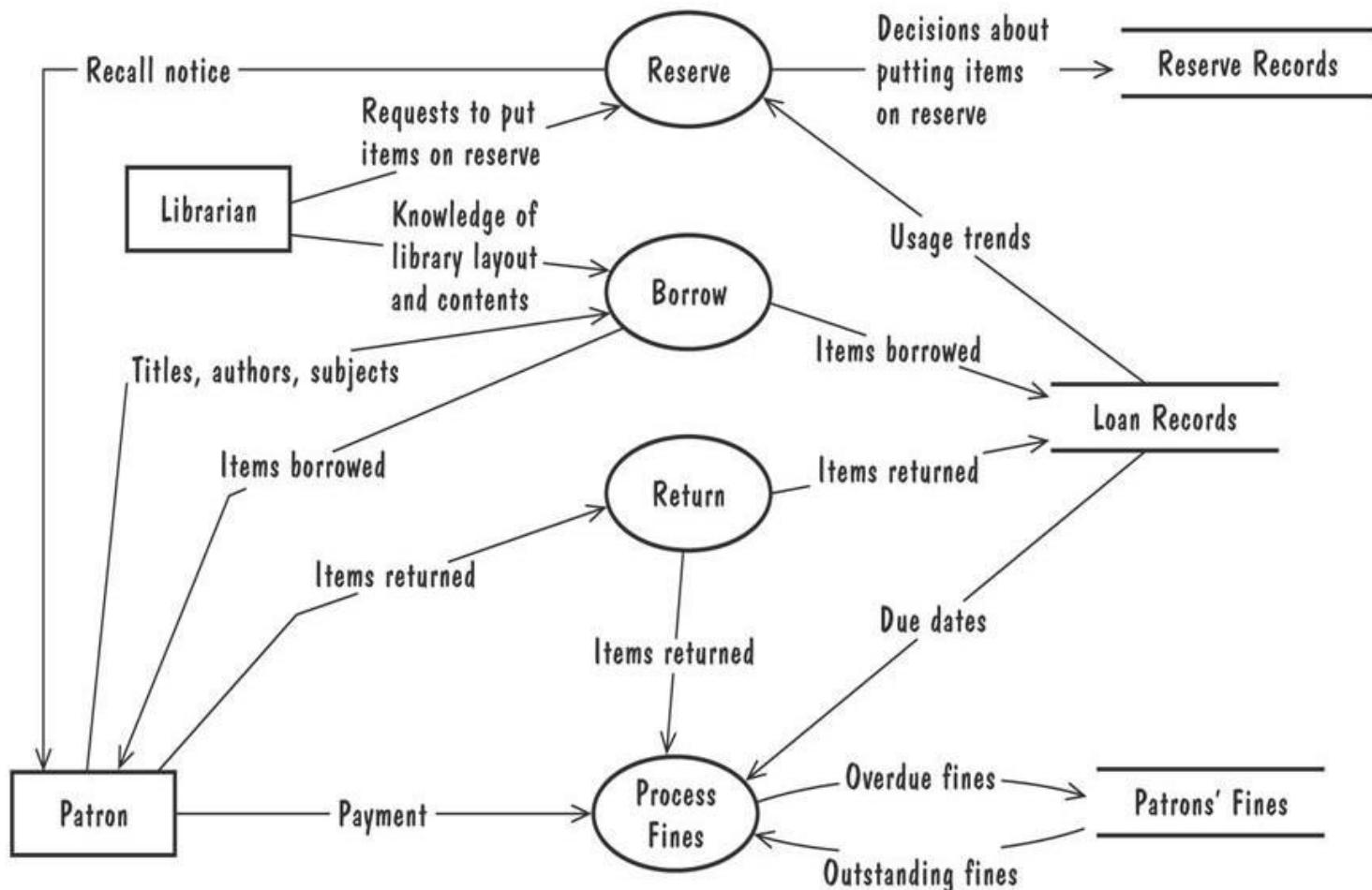
# DFD: uso dei dati / 2



- ☞ D invia un dato distinto a E ed F
  - ☞ D invia lo stesso dato a E **ed** F (spesso viene aggiunto l'**AND**)
  - ☞ Casi più complessi: ad esempio USCITE ALTERNATE a E e F, ....
- ☞ ...

**NB.** Il DFD non indica quale delle alternative va considerata. Questa informazione non è inclusa nei DFD e va indicata in altro modo.

# Esempio: Biblioteca



# DFD e diagrammi di conto

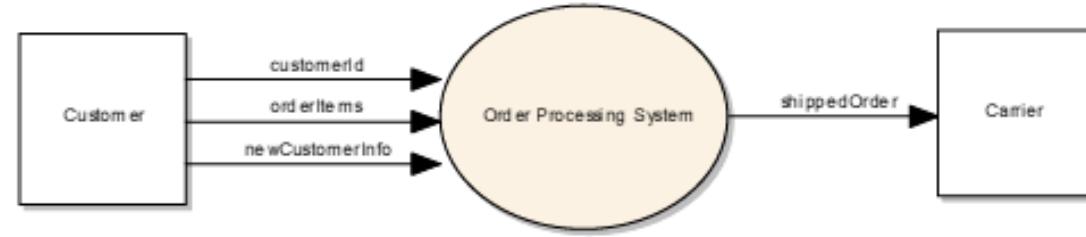


Figura 5.14: Diagramma di contesto per il sistema di gestione degli ordini online

(Libro A. Baruzzo)

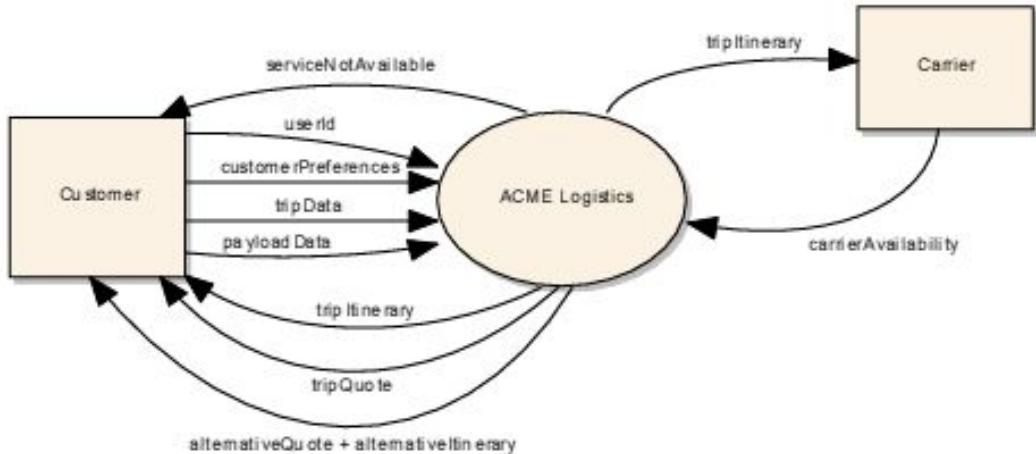
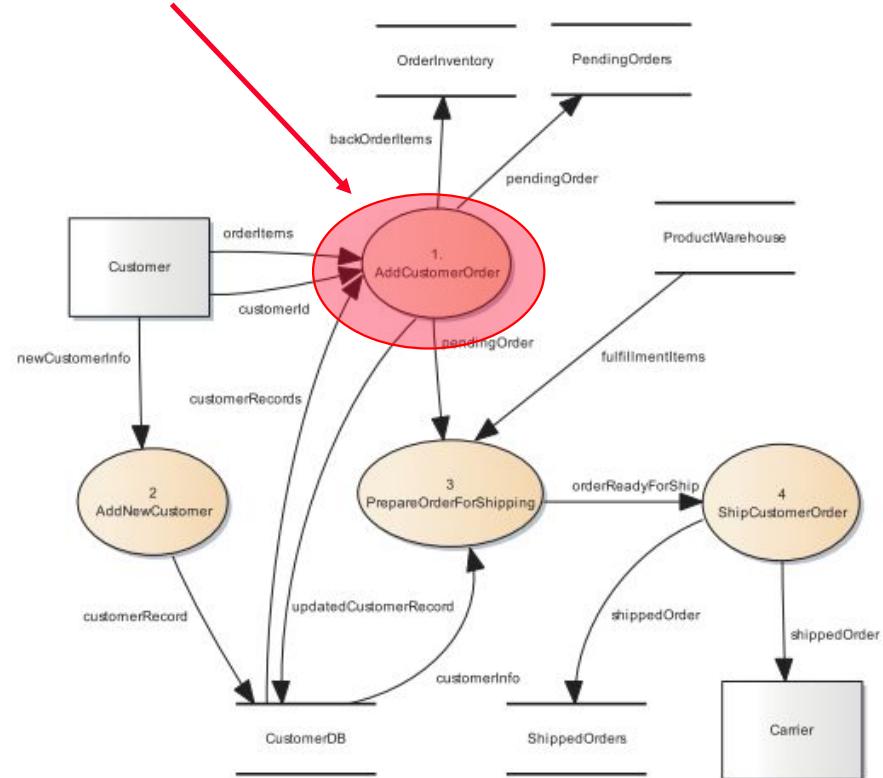


Figura 5.30: Diagramma di contesto per il sistema ACME Logistics

# Esempio: Gestione Ordini

Scomposizione di

è un SISTEMA di liv. 1



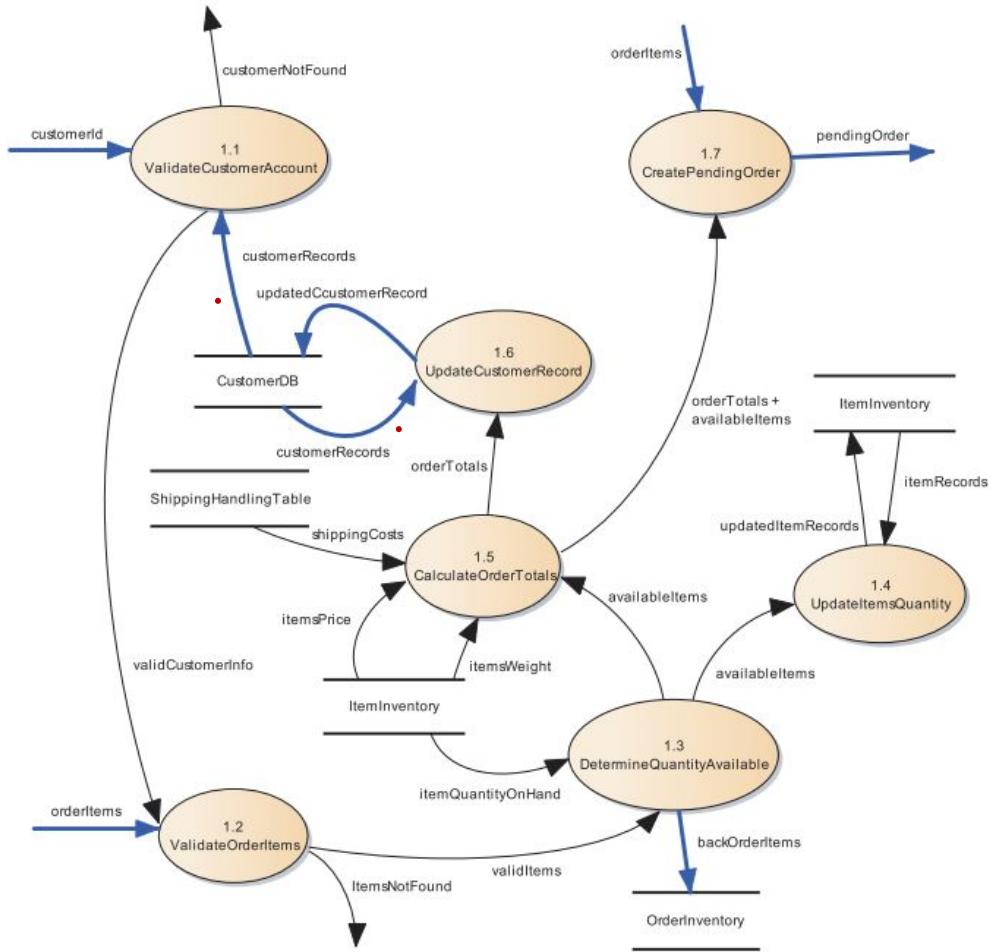
(Libro A. Baruzzo)

# Esempio: Gestione Ordini - 2

Diagramma di secondo livello per il  
processo di registrazione di un nuovo  
ordine

FIGURA 2.11

(Libro A. Baruzzo)



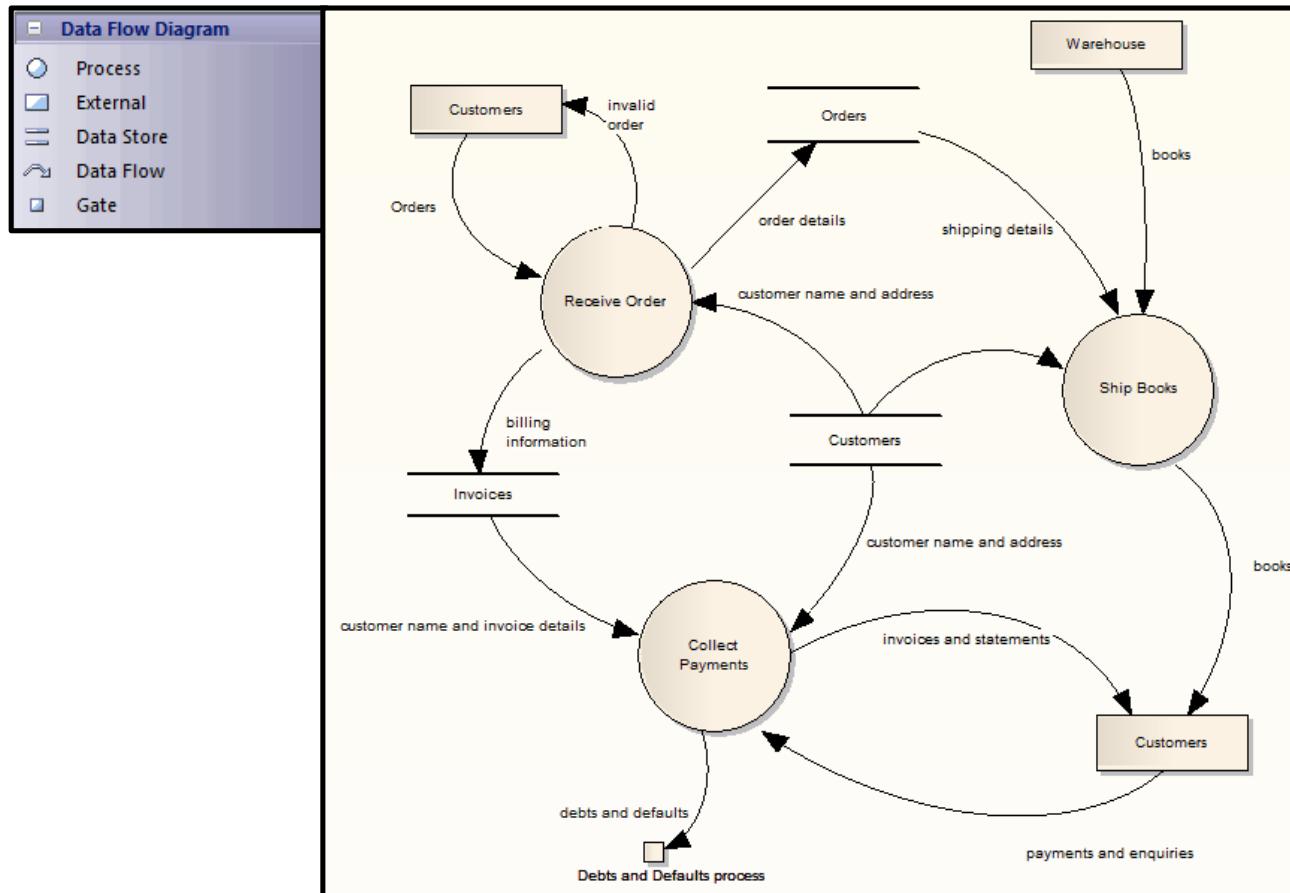
# Data Flow Diagram Toolbox di Enterprise Architect

---

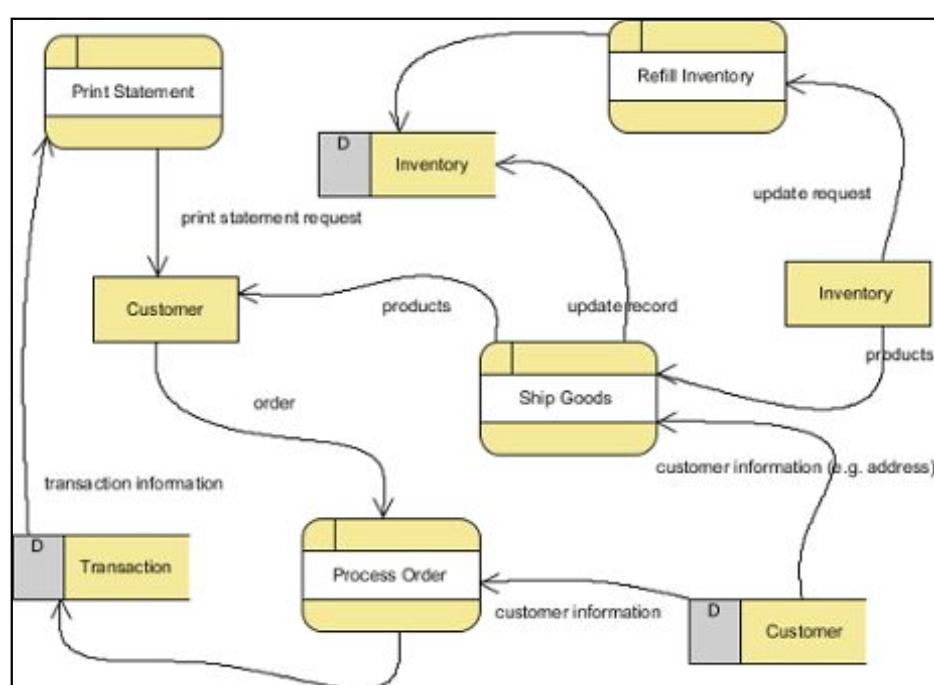
→ tool gratuito

- *Process* is a process or activity in which data is used or generated
- *External* represents an external source, user or depository of the data
- *Data Store* represents an internal physical or electronic repository of data, into and out of which data is stored and retrieved
- *Data Flow (connector)* represents how data flows through the system, in physical or electronic form
- *Gate* represents the termination point of incoming and outgoing messages on a lower level diagram (that is, messages to and from processes depicted elsewhere)

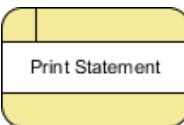
# Data Flow Diagram Toolbox di Enterprise Architect



# Visual Paradigm (Honk Hong)



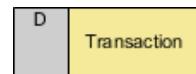
Process



External Entity



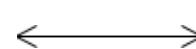
Data Store



Data Flow



Bidirectional Flow

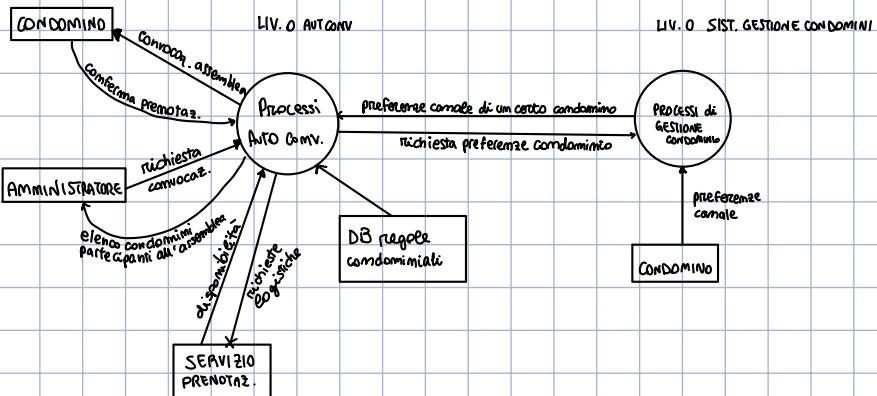
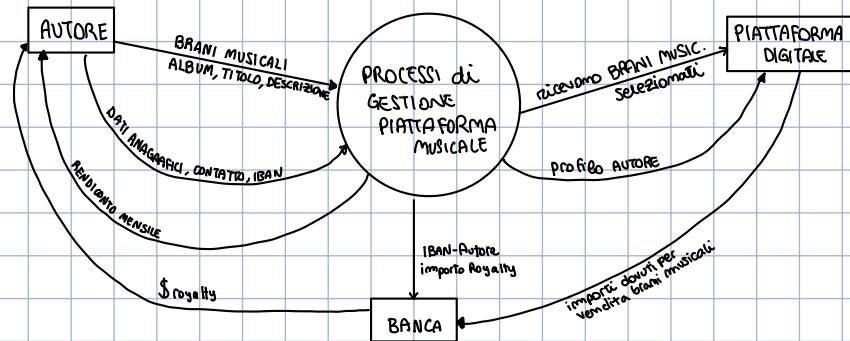


Es. 22.09.21

DFD liv. 0

↳ evidenzia cose esterne e

scambi di dati



quando faccio il livello 1 vado ad espanderne la bolla dentro

LIVELLO 1:



# Esercizi

---

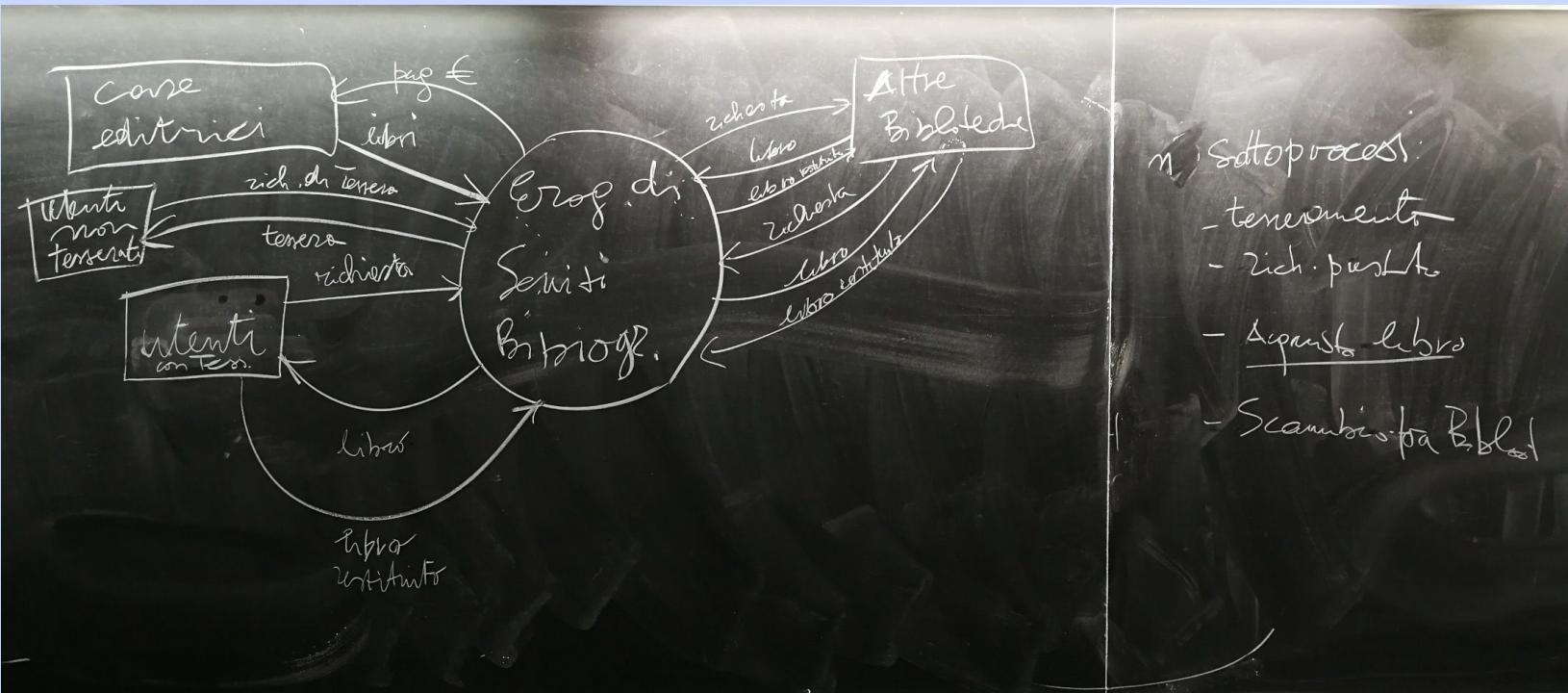
- DFD di Bancomat e suo contesto e di ‘prelievo da Bancomat’
- DFD di mail system (send e separatamente recieve)
- DFD di processi relativi a una Biblioteca
- Libro di A. Baruzzo per temi d'esame

# ESERCIZI

---

# Esercizio del Biblioteca (AA 17-18)

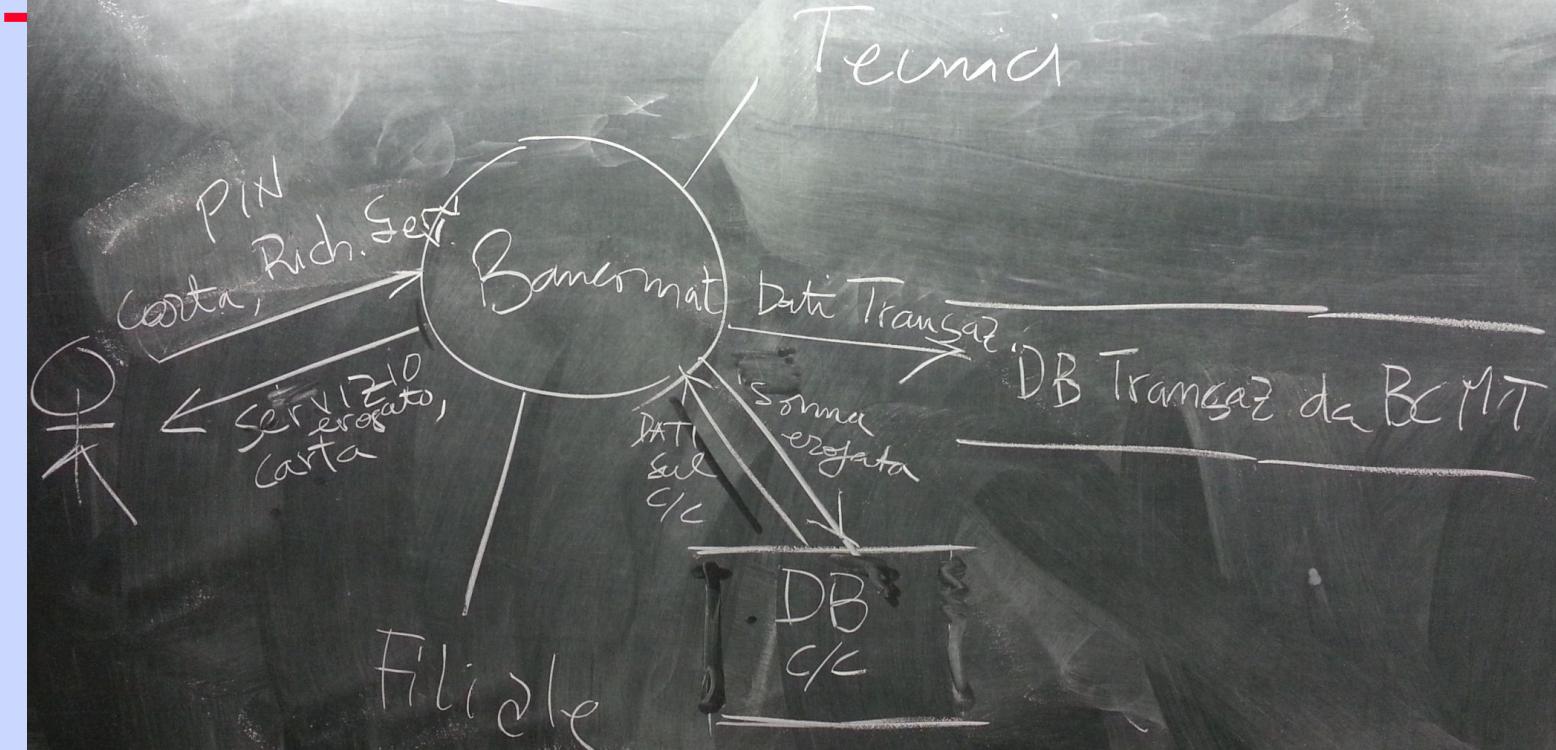
## 1^ versione diagramma CONTESTO



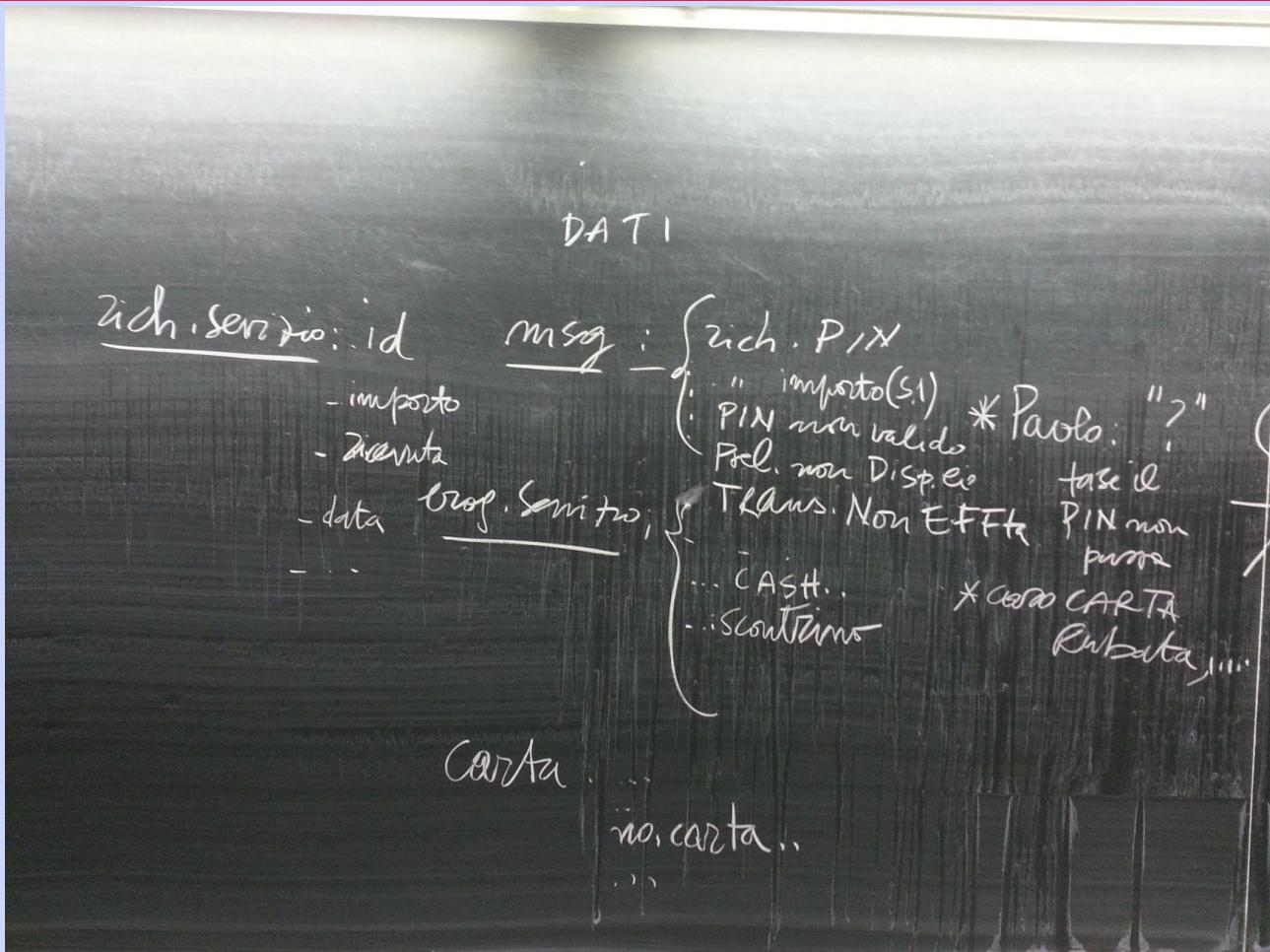
# Esercizio del Bancomat

---

# 1^ versione diagramma CONTESTO

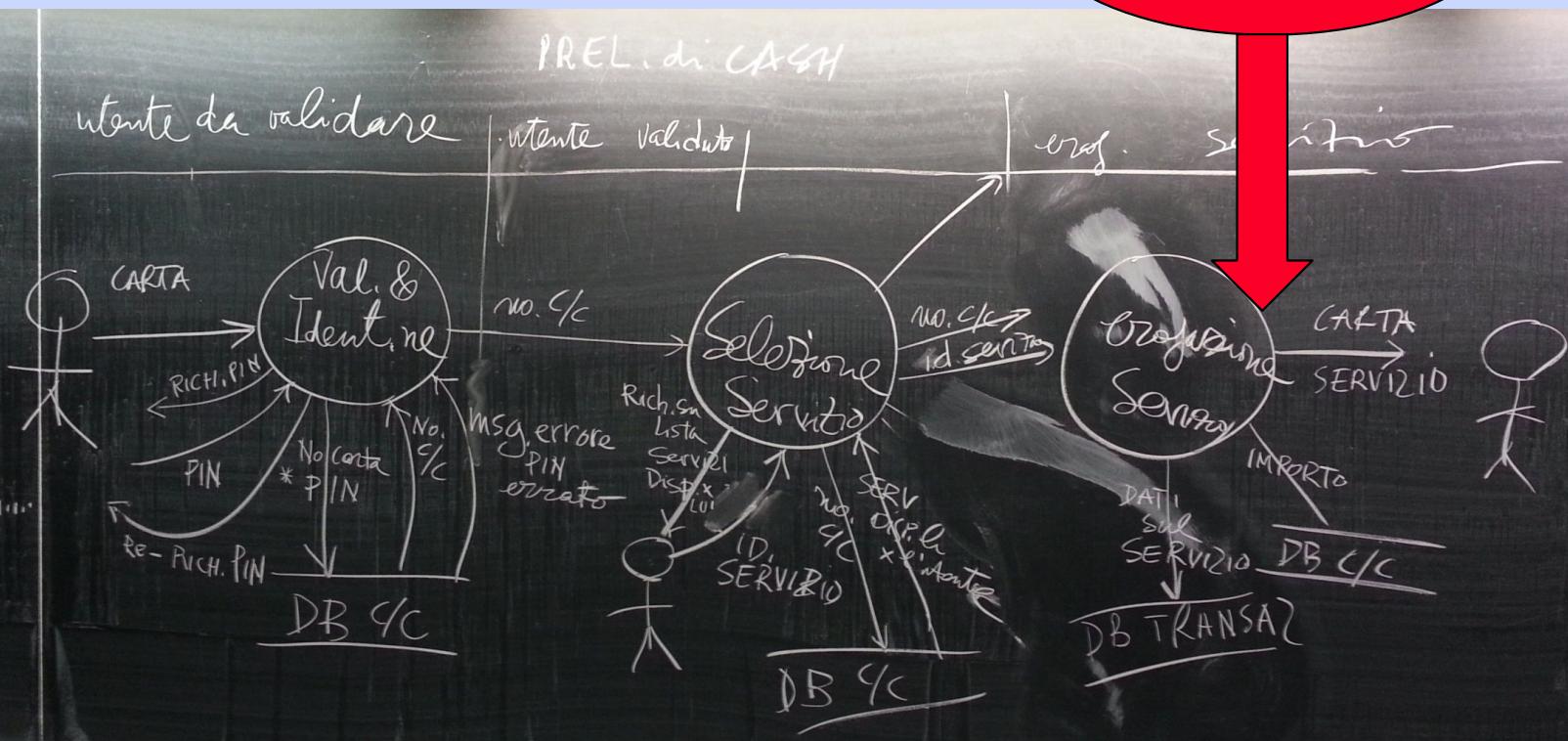


# Scomposizione dei DATI per i livelli più dettagliati

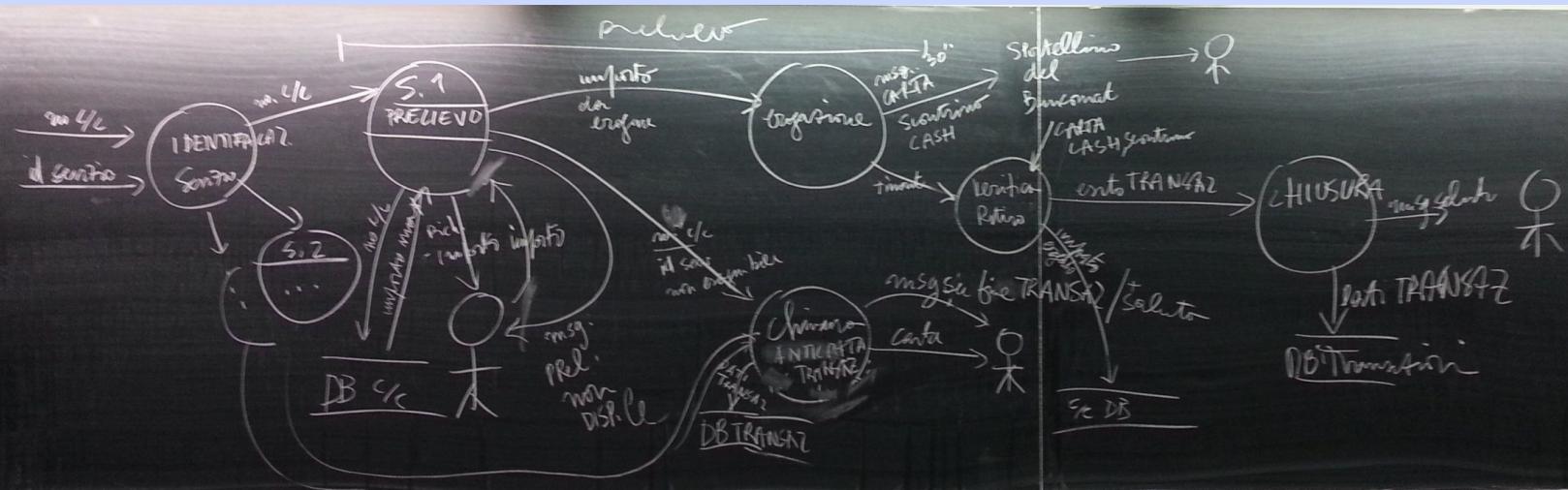


## II° Livello

+ Registrazioni  
e chiusura



# Bancomat: prelievo cash



# Esercizio d'esame

---

## • **Esame Scritto di Ingegneria del Software 1 - 24 Settembre 2013**

1. Si consideri un sistema di pubblicazione libri di una casa editrice che fornisce ai suoi clienti il servizio di Print On Demand. Il sistema prevede due tipi di utenti: gli autori e i clienti. Un autore per pubblicare un libro deve fornire i propri dati anagrafici e firmare elettronicamente un contratto con il quale cede i diritti di pubblicazione all'editore, a fronte di una royalty per ogni copia venduta. Solo a questo punto il sistema permette all'autore di caricare il file del proprio libro da pubblicare. Una volta caricato il file, questo viene aggiunto al catalogo dei libri in vendita. I clienti in qualunque momento possono consultare il catalogo, selezionare un libro, specificare il numero di copie da stampare, pagare e richiedere la spedizione al loro indirizzo. Anche i clienti devono essersi prima registrati, fornendo i loro dati anagrafici.

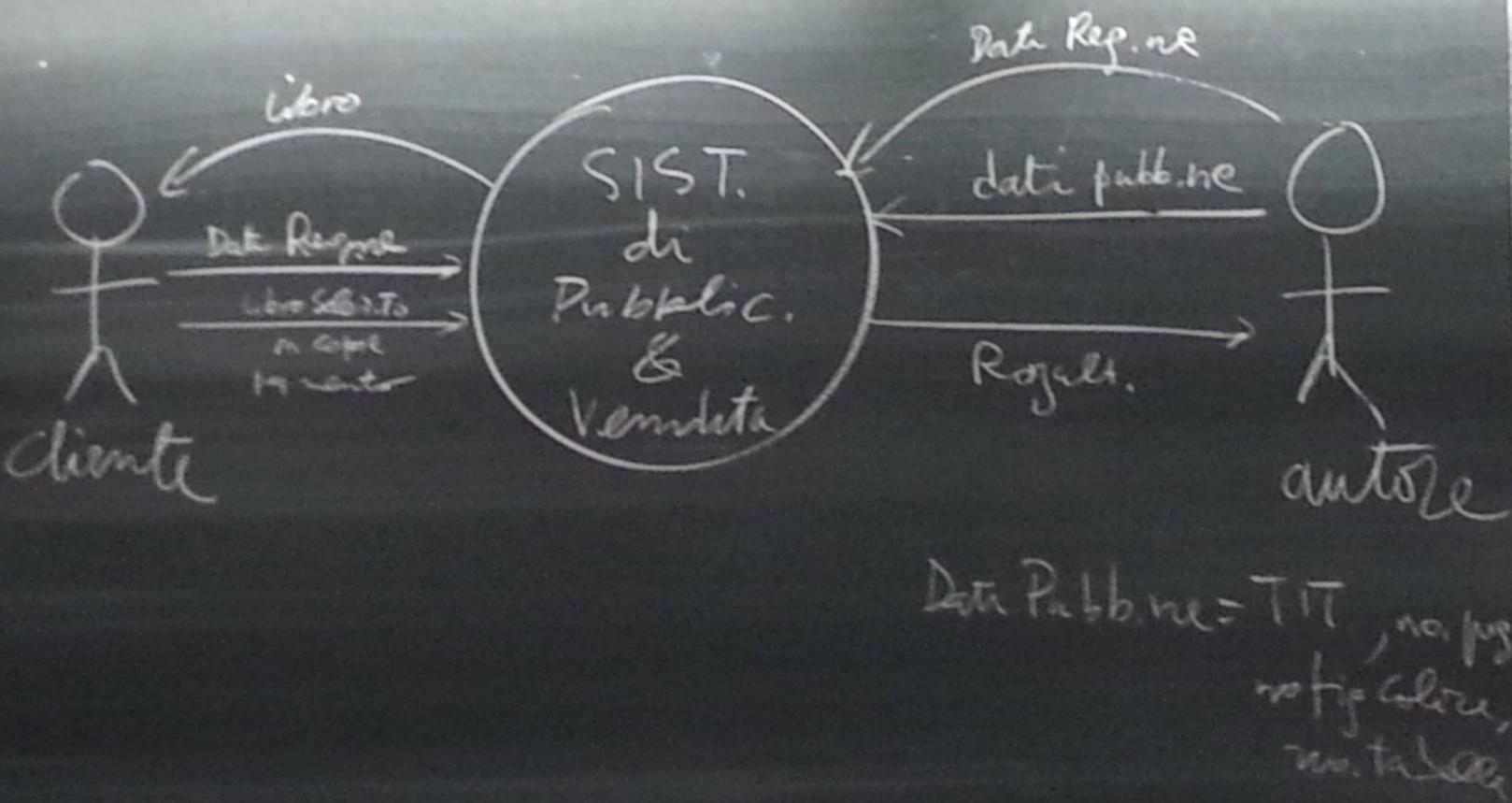
• Si modelli tale sistema in UML, fornendo (i) un diagramma dei casi d'uso, (ii) un diagramma delle classi che descrive il dominio del problema in esame e (iii) un diagramma di sequenza che illustra la procedura di registrazione di un cliente e di acquisto di un libro ricercato nel catalogo. (8)

2. Si consideri il sistema di Print On Demand dell'esercizio precedente. Lo si modelli questa volta con la tecnica dei DFD (Data-Flow Diagram), creando sia un diagramma di contesto (diagramma di livello 0), sia un diagramma DFD di primo livello (nel quale si mostrano i processi principali descritti nel testo). Quali differenze si notano rispetto alla soluzione documentata con la tecnica object-oriented in UML? (Suggerimento: si ragioni sull'espressività delle diverse notazioni (DFD e UML), indicando quali aspetti emergono meglio in un caso rispetto all'altro, e viceversa.) (8)

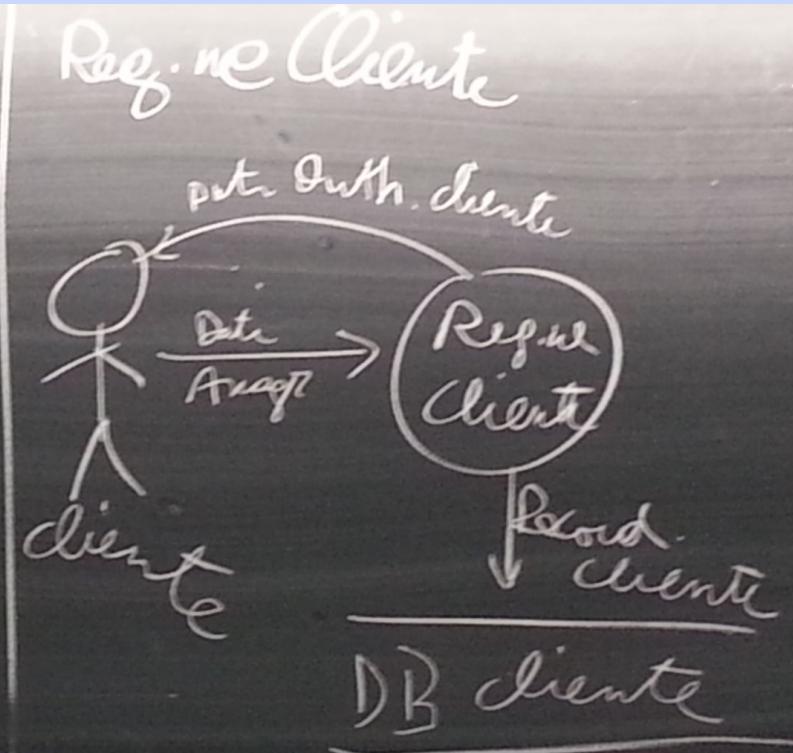
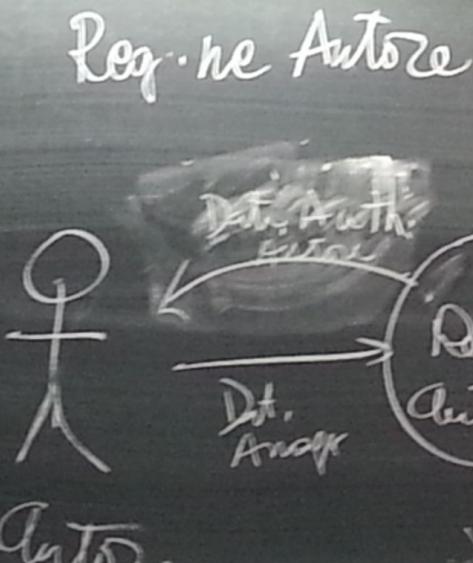
3. A. Si descriva il modello dei processi di sviluppo software denominato sviluppo incrementale. B. Se ne illustrino i vantaggi relativamente ad altri modelli. C. Come viene chiamato l'approccio allo sviluppo che si ottiene riducendo moltissimo la dimensione degli incrementi successivamente realizzati? (6)

4. A. In cosa si accomunano i due approcci al testing, denominati rispettivamente black-box e white-box testing? B. In cosa si differenziano? C. Su cosa si basa nei due diversi casi l'individuazione dei casi di test (test case)? (6)

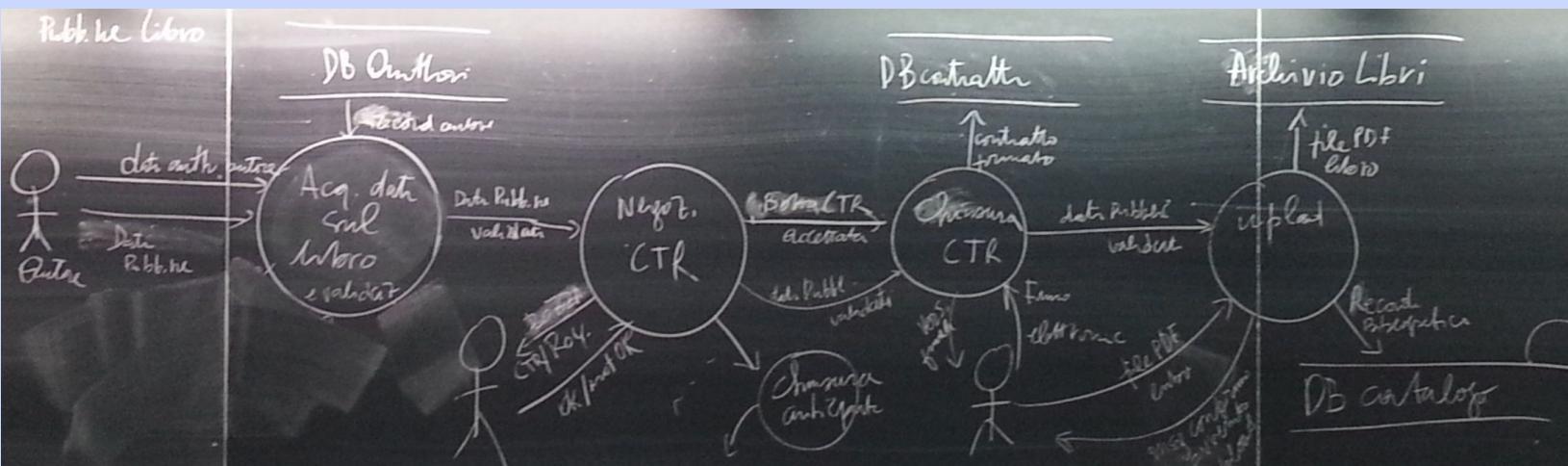
# DFD di livello 0 (Diagramma di contesto)



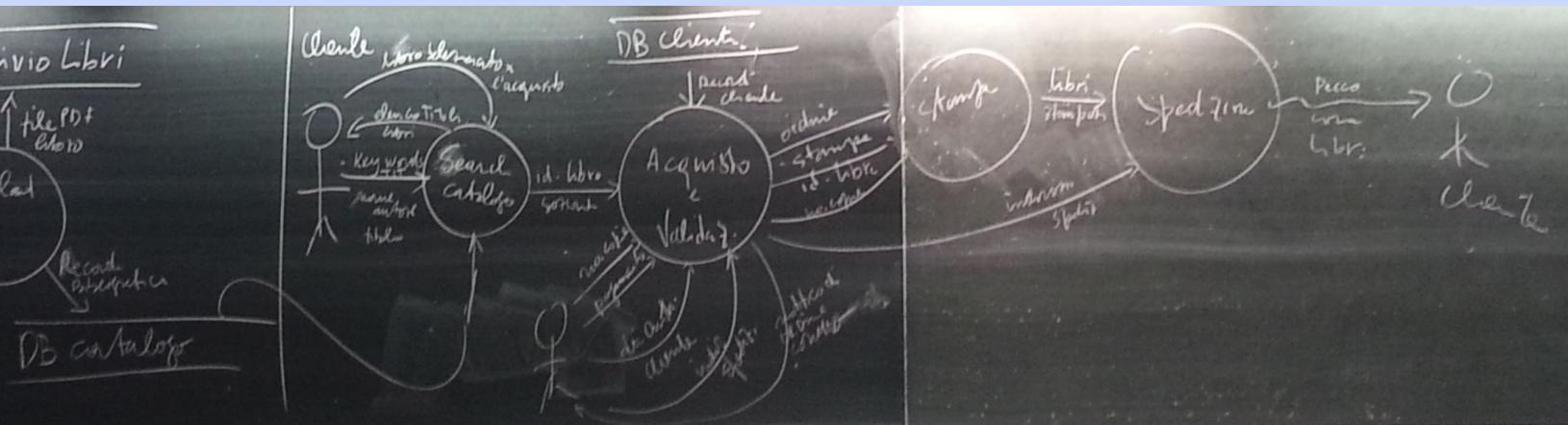
# DFD di livello 1



# DFD di livello 1 (cont.ne)



# DFD di livello 1 (cont. 2)



(manca la parte economico/amministrativa e il pagamento delle royalty)

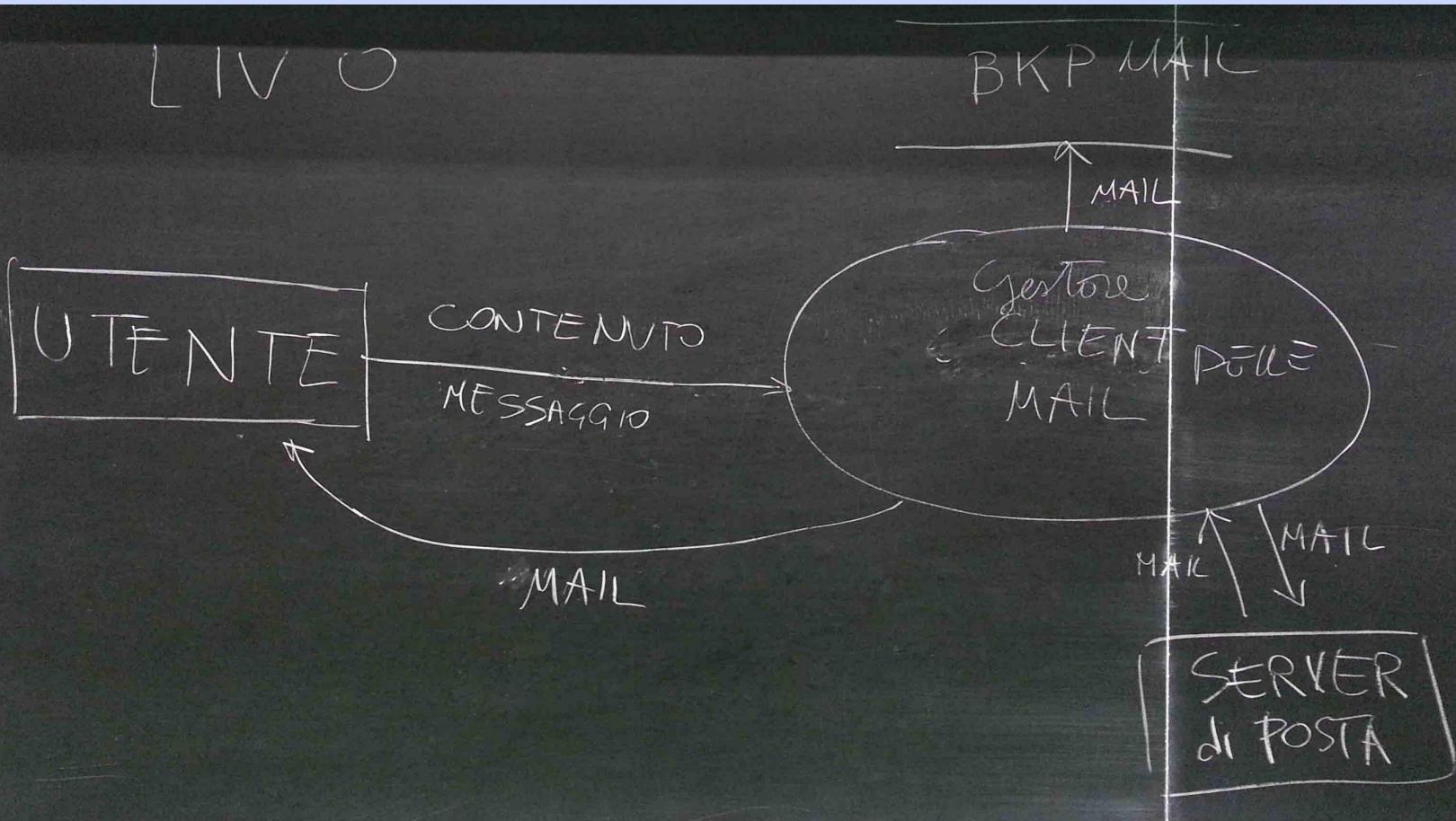
# Esercizio del Client di Posta

---

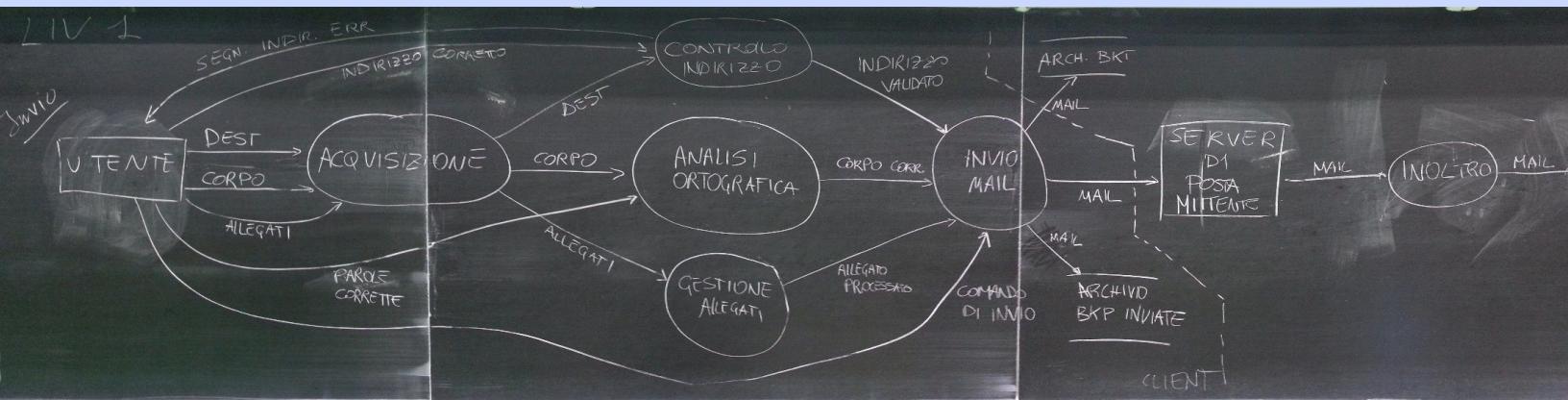
# Soluzione cooperativa con gli studenti AA 14-15

---

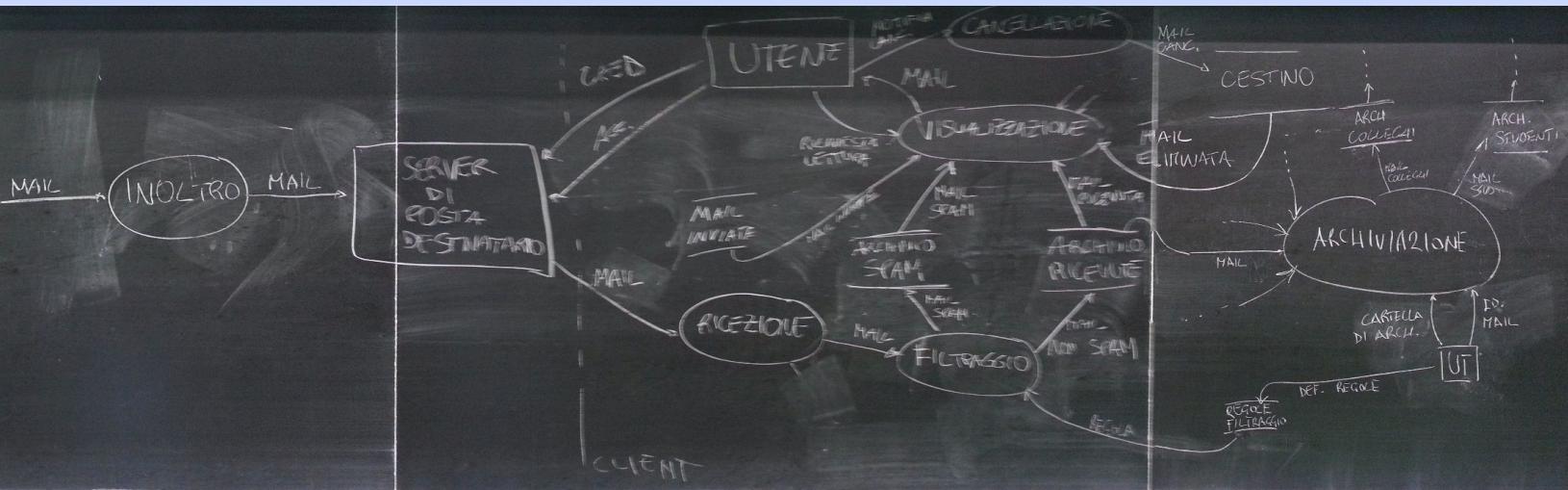
# Esercizio svolto in aula: Sistema di gestione della mail – Livello Zero (Diagramma di contesto)



# Mail/Liv. 1: Preparazione e invio della mail



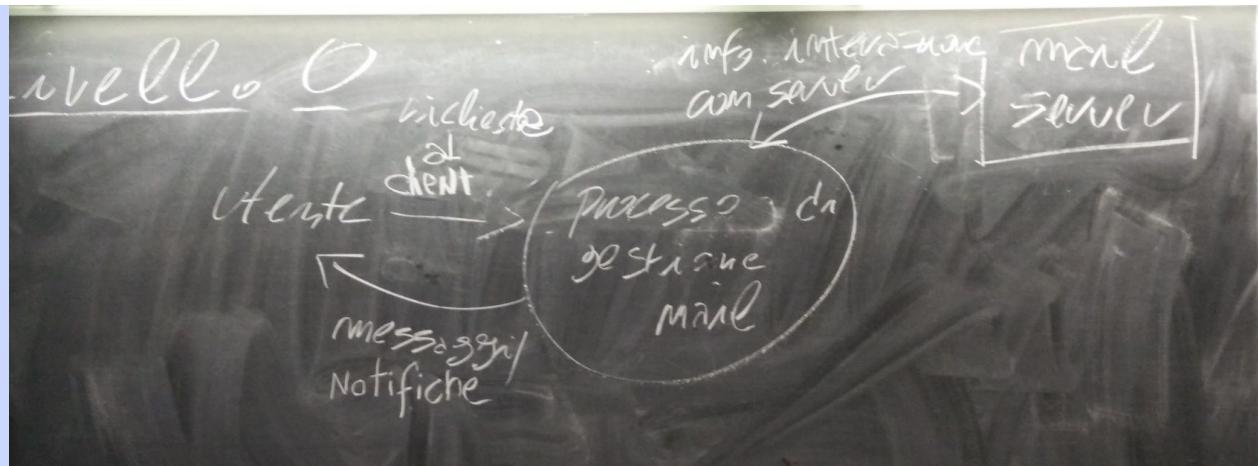
# Mail/Liv. 1: Lettura delle mail



# Soluzione cooperativa con gli studenti AA 16-17

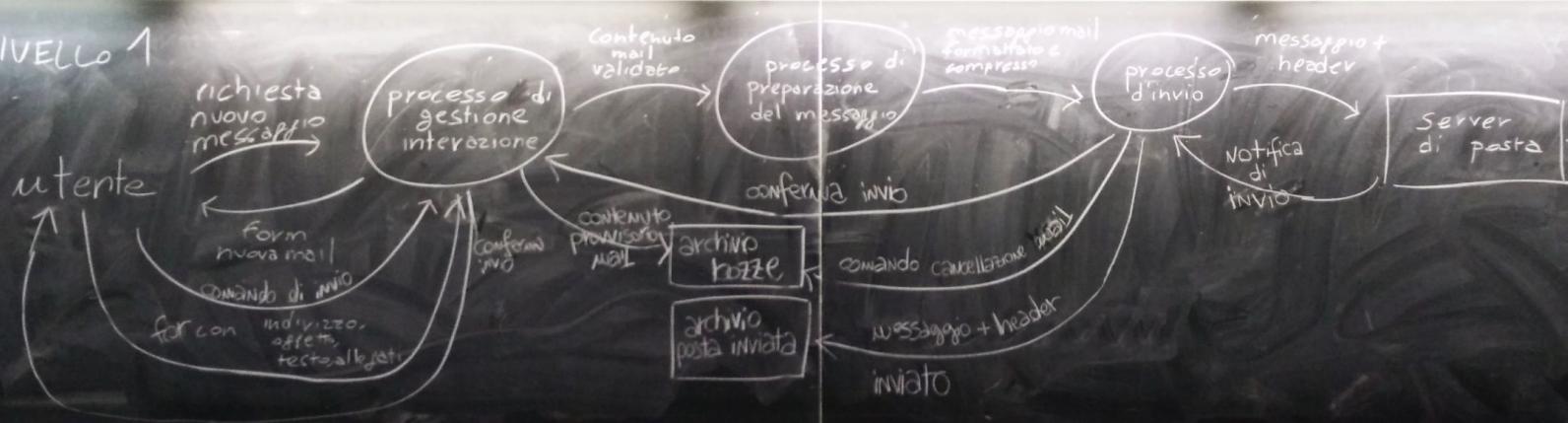
---

# Livello 0



# Livello 1 - SEND

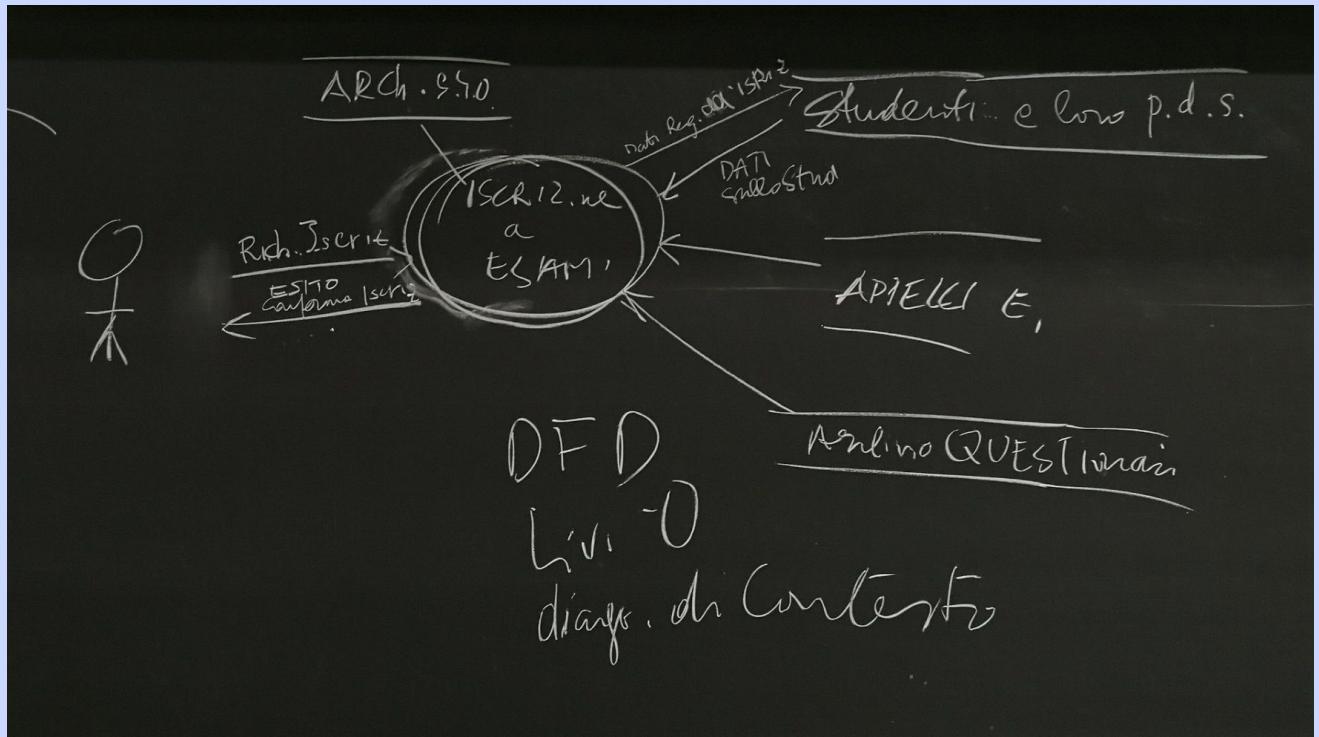
LIVELLO 1



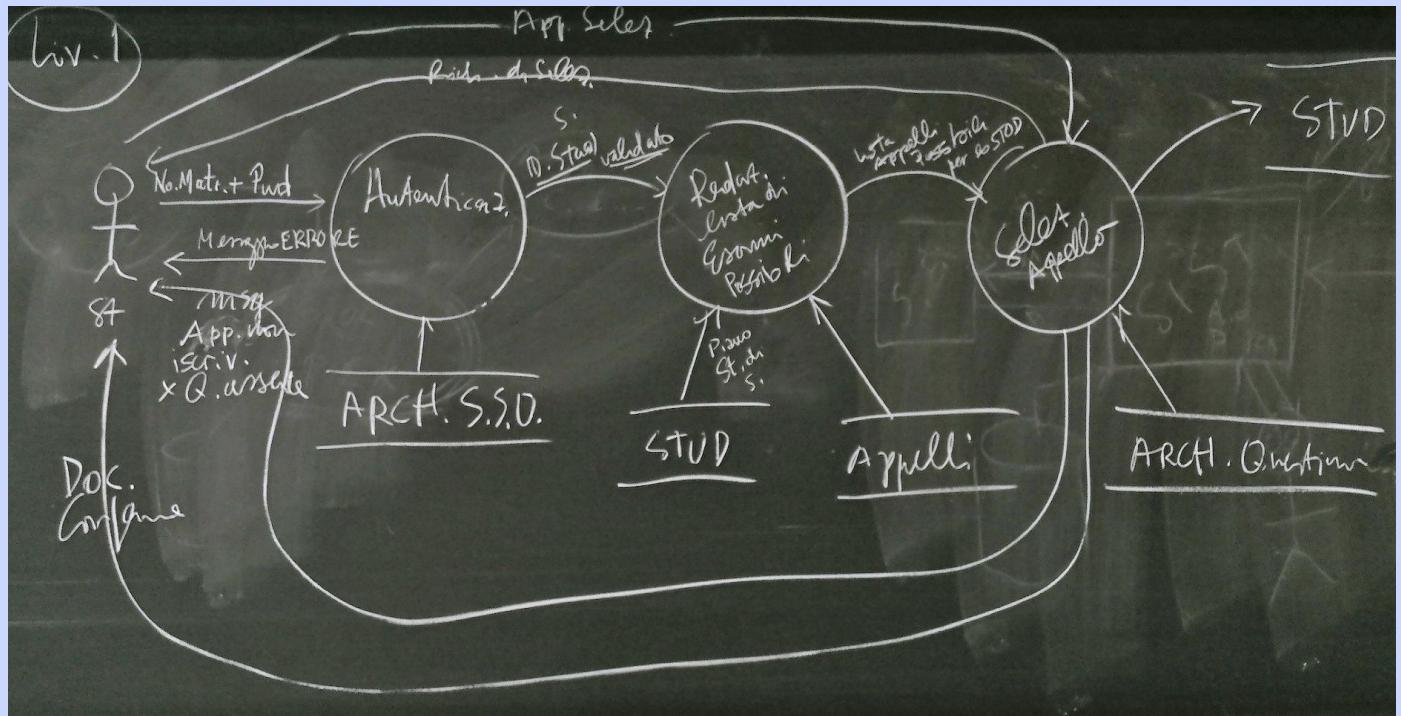
# Esercizio su un Sistema di Iscrizione agli Esami

---

# DFD Liv. 0 (Diagr. di Contesto)



# DFD di livello 1



# Esercizio Controllo Ortografico et al.

---

Si consideri un sistema per il controllo ortografico e grammaticale (da inserire in un sistema di videoscrittura). Il sistema ortografico riceve in input l'indicazione di una lingua da utilizzare, un testo (formato da parole) ed il relativo dizionario. Il sistema produce in output: nel caso in cui sono stati trovati degli errori, la lista delle parole errate ed una lista di suggerimenti per ciascun termine errato; nel caso in cui le parole sono corrette, emette un messaggio che specifica che il controllo è terminato correttamente. In quest'ultimo caso, viene attivato il sistema per il controllo grammaticale: questo prende in input anche una serie di regole grammaticali e segnala potenziali errori e relativi suggerimenti. Si mostrino specificamente un diagramma DFD che illustri il funzionamento del sistema ed uno o più structure chart, ossia i diagrammi che illustrano la decomposizione strutturale (structural decomposition) del sistema software.

# Tipologie di modelli esaminati

---

## 1. Context models

1. Diagrammi contesto
2. Process models

## 2. Behavioural models

1. Data processing models (DFD)
2. **State machine** → reazioni del sistema in base a degli eventi
3. **Petri nets**

## 3. Semantic Models

1. E-R models
2. Data Dictionary

## 4. Objec Models

1. Inheritance
2. Aggregation
3. Behavioural

## 2.2 State machine models

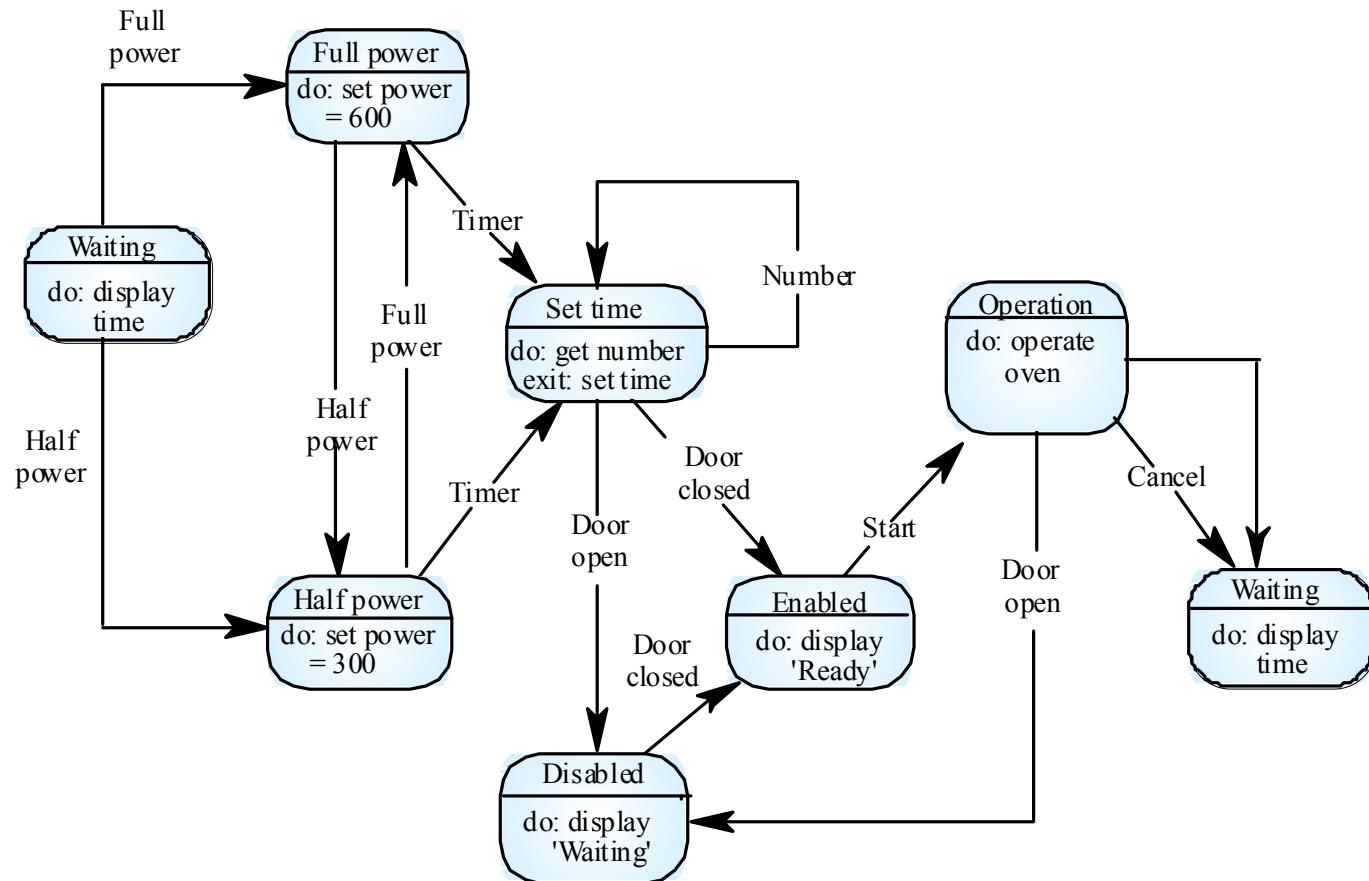
---

- These model the behaviour of the system in **response to external and internal events** → passa da uno STATO all'altro, cambiamenti di stato
- They show the system's **responses to stimuli** so are often used for modelling real-time systems
- State machine models show **system states as nodes** and **events as arcs** between these nodes\*. When an event occurs, the system moves from one state to another
- **Statecharts** are an integral part of the **UML**  
↳ diagrammi di stato fanno parte degli UML

---

(\*) NB.: LA GRAFICA è **SIMILE** AI DFD, MA LA SEMANTICA DEI SEGNI GRAFICI E QUINDI Ciò CHE SI RAPPRESENTA è **DEL TUTTO DIVERSA!!!!**

# Microwave oven model



→ documento che descrive stato per stato che cosa succede

# Microwave oven state description

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows 'Half power'.
Full power	The oven power is set to 600 watts. The display shows 'Full power'.
Set time	The cooking time is set to the user's input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for 5 seconds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.

# Microwave oven stimuli

---

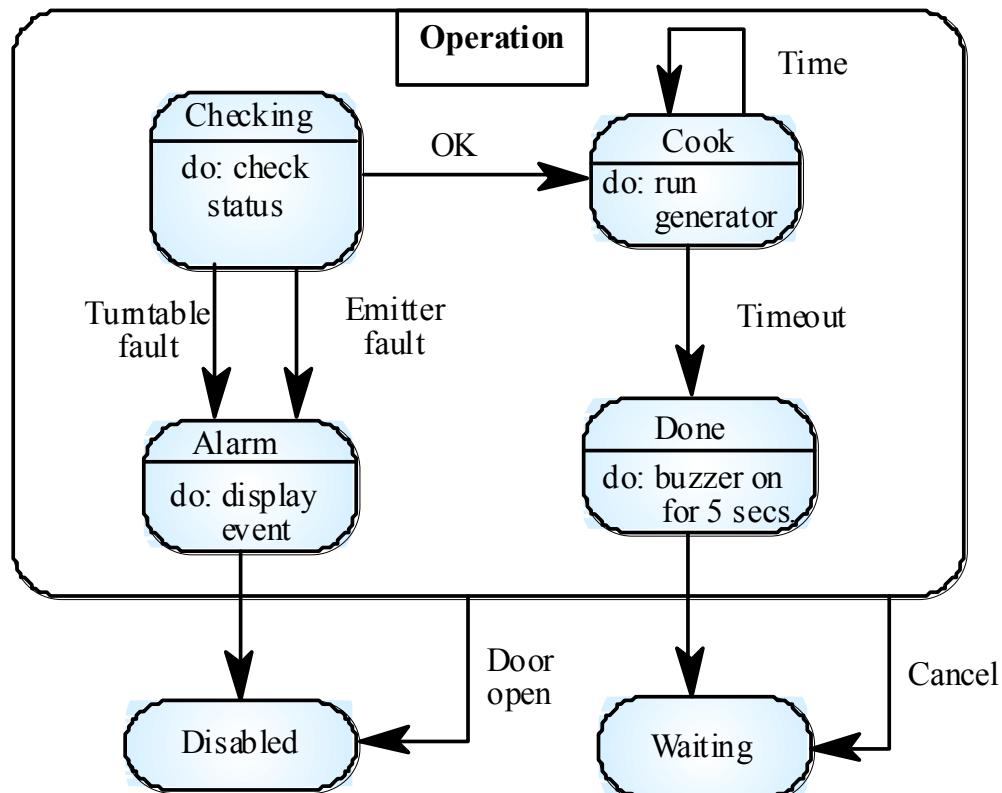
Stimulus	Description
Half power	The user has pressed the half power button
Full power	The user has pressed the full power button
Timer	The user has pressed one of the timer buttons
Number	The user has pressed a numeric key
Door open	The oven door switch is not closed
Door closed	The oven door switch is closed
Start	The user has pressed the start button
Cancel	The user has pressed the cancel button

# UML Statecharts

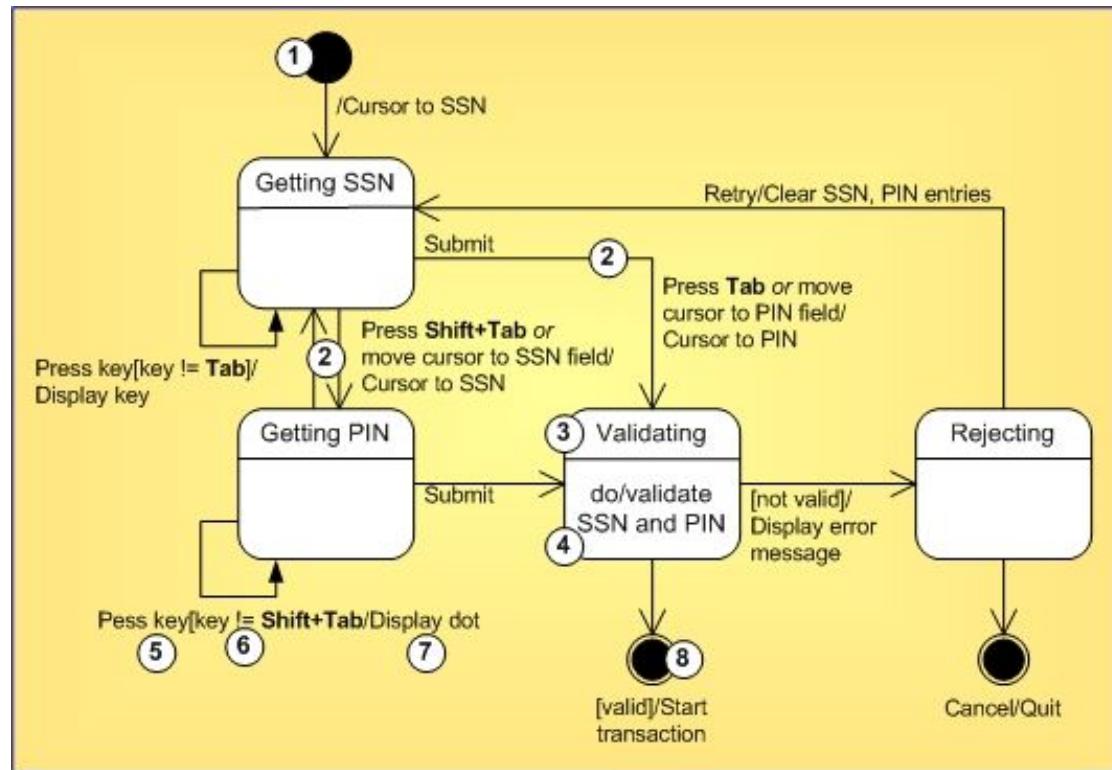
---

- Allow the **decomposition** of a model into **sub-models** (see following slide)
- A brief description of the actions is included following the ‘do’ in each state
- Can be complemented by tables describing the states and the stimuli

# Microwave oven operation



# UML 1.5



① Initial State

② Transition

③ State

④ Action

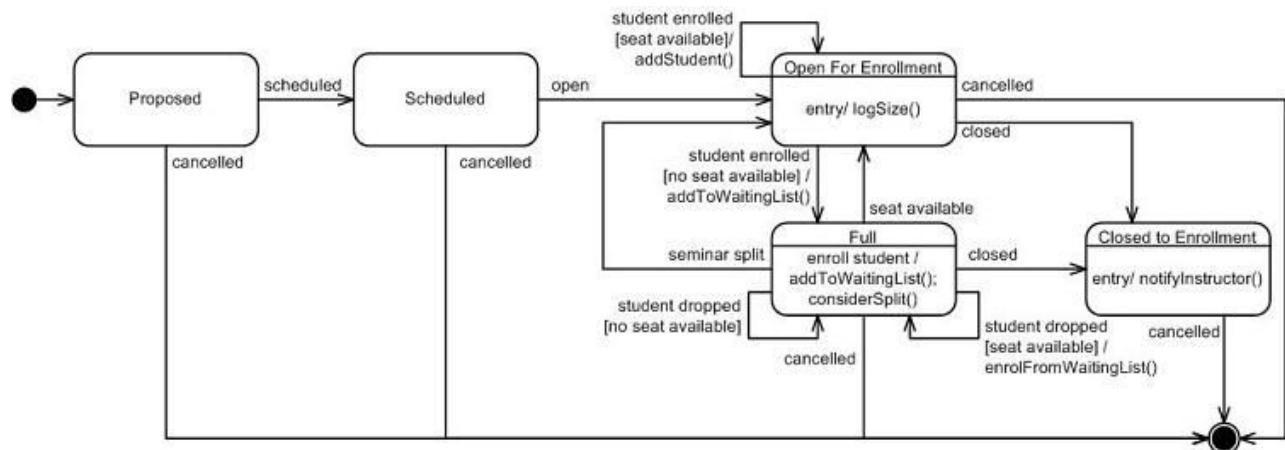
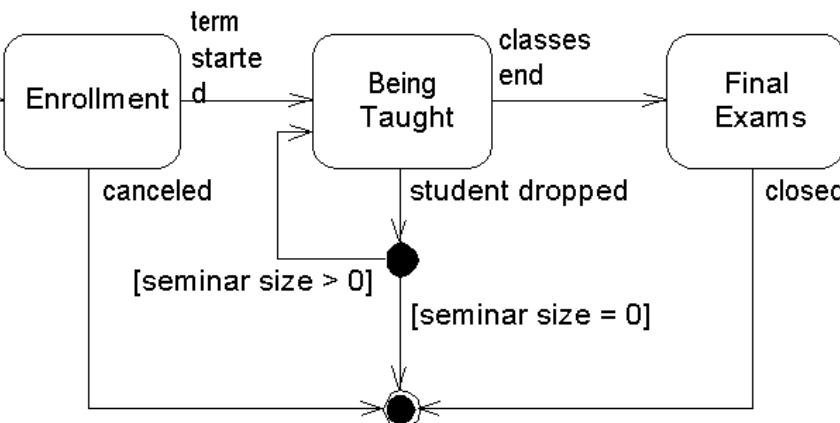
⑤ Event

⑥ Guard

⑦ Activity

⑧ Final State

# UML 2 Machine State Diagrams



# Limitazioni delle rappresentazioni basate su automi a stati finiti

- Lo **stato del Sistema** è rappresentato in modo **unitario**, anche nel caso di sistemi complessi, ossia costituiti da:
  - molte parti che interagiscono fra loro:
  - diversi processi (eventualmente concorrenti) fondamentalmente indipendenti fra loro, che però nell'ambito del funzionamento generale del Sistema devono sincronizzarsi in vari modi

→ *moi sistemi più complessi, es. quello del micro-ondi, come faccio a sapere, ad es. mello stato SetTime se arrivo da Full-power o Half-power? Se dovesse codificare uno stato per ogni possibile combinaz. sarebbe inefficiente... ⇒ RETI DI PETRI*
- (✉ alla domanda: qual'è ora lo stato del Sistema, si risponde indicando un unico nodo del diagramma ✉ non risulta chiaro in che stato si trovano alter parti del sistema – es. Full power o Half power?)
- Se con i vari stati si desidera rappresentare tutte le situazioni in cui si possono trovare le varie parti del Sistema si rischia una 'esplosione combinatoria' del no. degli stati!!!

→ ovvero in ogni istante di esecuz.  
sappiamo descrivere in che stato  
siamo

## 2.3 Reti di Petri [C.A. Petri, 1962]

---

- Introduzione alle Reti Posti-Transizioni

# Obiettivi

---

- Capire le definizioni fondamentali
- Capire la ‘sintassi’ delle Reti Posti-Transizioni
- Capire le caratteristiche principali
- Esaminare alcuni esempi
- Saper costruire una Rete di Petri per semplici situazioni applicative (**una delle domande d'esame**)

# Idee principali

→ utili per capire a che punto siamo arrivati del sistema

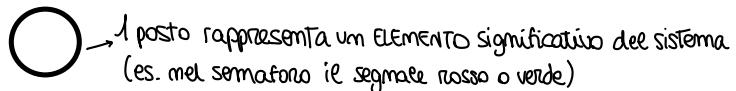
- Servono a rappresentare la **dinamica** di un sistema in cui risulta complesso utilizzare gli automi, ed in particolare **un unico** stato ‘globale’ [Negli automi a s.f., la situazione specifica in cui si trova l’intero sistema viene rappresentata da un’unica **SINGOLA** entità astratta, i.e. lo stato]
- Nelle RdP, lo stato viene rappresentato suddiviso/distribuito nelle sue **componenti significative**, ognuna delle quali rappresentata da un ‘**posto**’
- La dinamica del sistema viene descritta con ‘granularità fine’, attraverso **passaggi di stato (transizioni) riferiti solo ad una parte del sistema (singoli posti o insiemi di posti)**

mentre negli automi passo da uno stato all’altro (che però riguardano la config. di tutto il sistema), qui passo da una config. all’altra di solo una posta (mostro solo quello che cambia)
- In tal modo risulta possibile rappresentare più processi indipendenti tra loro, che però possono caratterizzarsi per varie forme di sincronizzazione

# Definizioni Fondamentali

- Una RdP è composta da 4 elementi:

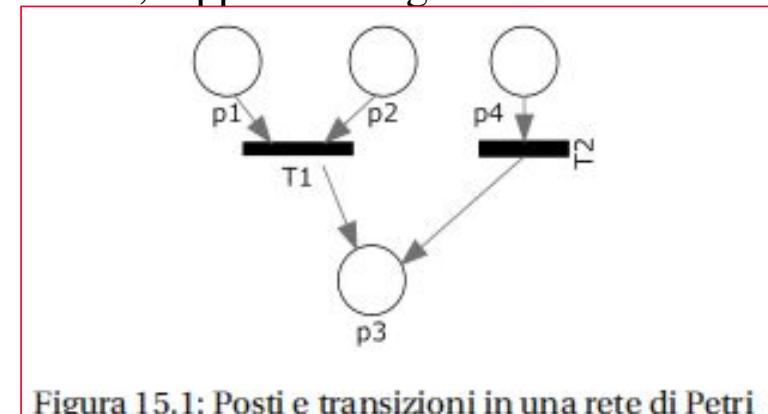
1. Un **insieme di POSTI P**, rappresentati graficamente da un cerchietto:



2. Un **insieme di TRANSIZIONI T**, rappresentati graficamente da una barretta:



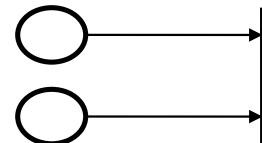
rappresentano le transizioni da uno stato all'altro del sistema



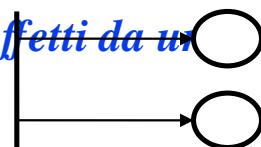
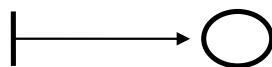
# Definizioni Fondamentali (2)

3. Una FUNZIONE DI INPUT I ( $I: P^n \rightarrow T$ ) che associa una collezione di posti  $I(t_j)$ , chiamati ‘posti di input’, ad una transizione  $t_j$ . I è rappresentata graficamente da un arco orientato da uno o più posti ad una transizione:

*Quali aspetti dello stato (posti) abilitano una transizione.*



4. Una FUNZIONE DI OUTPUT O ( $O: T \rightarrow P^n$ ) che mappa una transizione  $t_j$  in una collezione di posti  $O(t_j)$  chiamati ‘posti di output’. O è rappresentata graficamente da un arco orientato da una transizione a uno o più posti: *Quali aspetti dello stato (posti) sono affetti da un’transizione.*

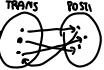


# Fare attenzione a....

---

- Nomi
- Sintassi grafica

# Definizioni Fondamentali (3)

- Graficamente, una RdP è quindi rappresentabile da un **grafo bipartito**.  $\Rightarrow$  
- Un'altra primitiva delle RdP è il **token**, rappresentato graficamente con un punto (interno ad un posto).
- Una **marcatura m** è un assegnamento di token sui posti della rete: zero, uno, o **più** token per ciascun posto della rete.  $\rightarrow$  es. per il lavaggio auto ho un POSTO che mi dice quante auto sono in ATTESA: il counter si tiene con i token, i.e. due macchine in coda.  $\rightarrow$  2 token  
e ogni volta che arriva una nuova macchina aggiungo un token  
tre " "  $\rightarrow$  3 token
- Serve a rappresentare lo **stato corrente** del sistema.

# Significato intuitivo

- I **posti** rappresentano **stati parziali**.
- La **marcatura** permette di identificare lo **stato corrente del sistema**, che risulta quindi ‘**distribuito**’, scomposto in più stati parziali
- Le **funzioni di input e di output** associati ad una specifica transizione rappresentano rispettivamente i **posti da cui è possibile attivare** la transizione ed i **posti a cui si transita** al completamento della transizione.
- I **token** sono astrazioni interpretabili in **diversi** modi in base alla specifica situazione: possono identificare la quantità di risorse presenti in un posto, oppure se una condizione (associata al posto) è o meno verificata (in un determinato momento), contatori, il numero di volte in cui un posto è stato usato, oppure ...

# Regole di Evoluzione

---

- L'**evoluzione** di una RdP è condizionata dal numero e dalla distribuzione di token nella rete. L'evoluzione è caratterizzata dallo **scatto delle transizioni** (che rappresenta un passaggio da uno stato del sistema ad uno stato successivo).
- Una **transizione** scatta ‘**prelevando**’/’**consumando**’ (‘**facendo sparire**’) dei token da **tutti** i suoi posti di **input**, e ‘**mettendone**’ (‘**facendo apparire**’) altri (**non necessariamente in numero uguale**) in **tutti** i suoi posti di **output**. (**N.B. Non confondere con ragionamenti funzionali**)
- L'evoluzione basata sugli ‘scatti’ ne fa un approccio *event-driven*

# Significato intuitivo

---

- L'evoluzione della rete rappresenta l'evoluzione del sistema, da una situazione ad una successiva, ciascuna modellata da più stati parziali (uno stato globale distribuito).

# Regole di Evoluzione: quando una transizione può scattare

---

- Lo scatto della transizione avviene se si verificano le seguenti condizioni:
  - Una transizione è abilitata allo scatto se tutti i posti in input contengono token /da ~~far transitare sugli archi~~ ‘prelevare’/’prendere’. Allo scatto verranno prelevati token dai posti in input alla transizione stessa.
  - Una sola transizione per volta può scattare; ciò per evitare che due transizioni abilitate prelevino lo stesso token. La transizione che scatta è scelta in modo **non deterministico**, ovvero fra tutte le possibili non si sa quale effettivamente scatterà : ne scatterà una a caso (!).

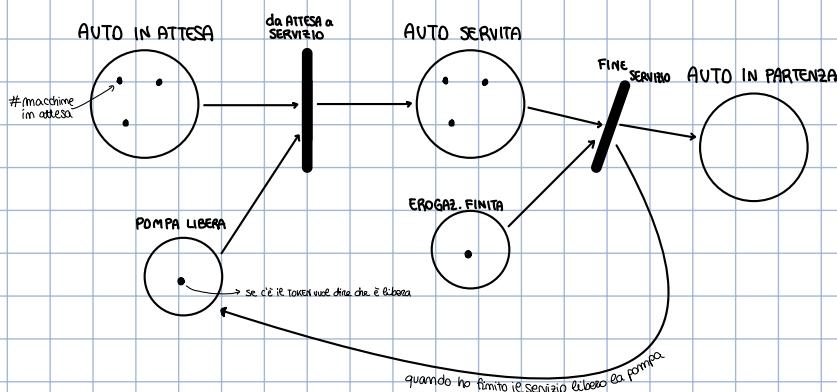
↳ Se possono scattare due transizioni, se me sceglie una a caso e viene eseguita

# Peso degli archi

---

- In una visione più generale, un arco può avere associato un **peso** (no. **intero positivo**).
- Tale numero, se associato ad un arco che connette un posto di input ad una transizione, specifica il **numero minimo di token** necessari affinché il posto in esame possa dare il proprio contributo ad abilitare la transizione.
- Il **peso** specifica anche **quanti TOKEN** ‘saranno tolti’ dal posto da cui si diparte l’arco verso la transizione che scatta.
- Viceversa, se il peso si riferisce ad un nodo di output, il numero specifica **quanti token verranno messi nel** posto una volta scattata la transizione.

## esempio dell'autolavaggio

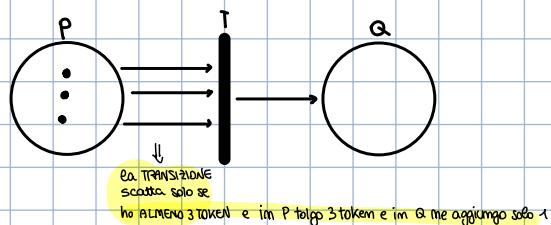


Dato che da auto in partenza non tolgo i token, mi dice quante auto ho fatto in giornata

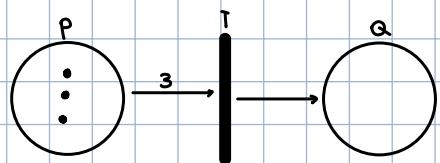
→ La TRANSIZIONE può scattare quando OGNI POSTO in INPUT ha almeno un TOKEN.

e in qualunque momento io posso sapere quale è lo stato del sistema, es. da quello sopra: ho 3 auto in attesa, la pompa è libera e ho 3 auto servite

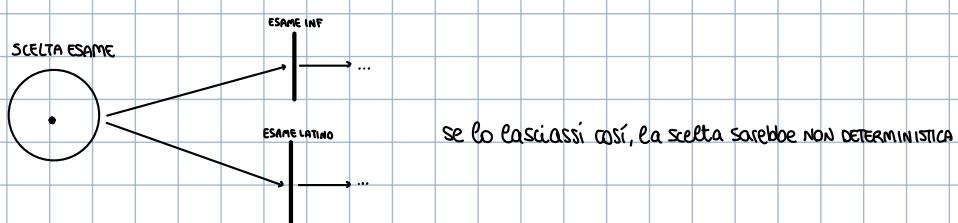
es. concettuale



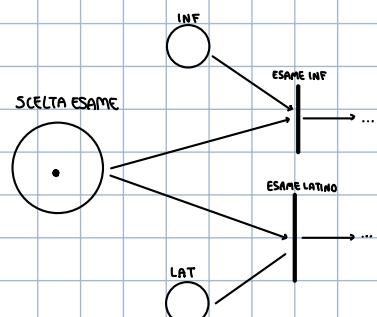
⇒ visto con i PESI =>



es.



RISOLVO con:



e così in base a dove è il TOKEN (LAT o INF) la scelta è deterministica.

# Osservazioni generali - 1

---

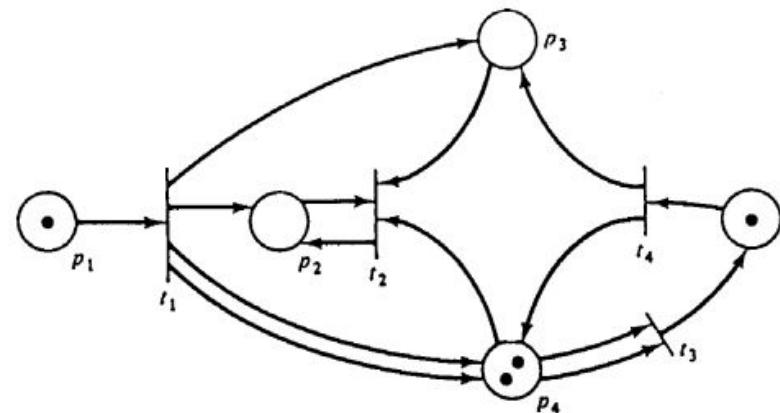
- Lo stato corrente del sistema è rappresentato dalla marcatura, e quindi è distribuito nei posti della rete.
- La rete permette di rappresentare **processi concorrenti** che si debbano sincronizzare. La sincronizzazione è rappresentata da una transizione che ha due (o più) posti in input, corrispondenti ai due (o più) processi che si desidera sincronizzare (cioè il completamento di entrambi è condizione per proseguire).
- La **sincronizzazione con eventi esterni** (ad esempio un input dell'utente, un evento, una selezione, ecc.), avviene dedicando all'evento un posto in cui il token/i token viene/vengono depositato/i dall'esterno, al verificarsi dell'evento stesso.

# Osservazioni generali - 2

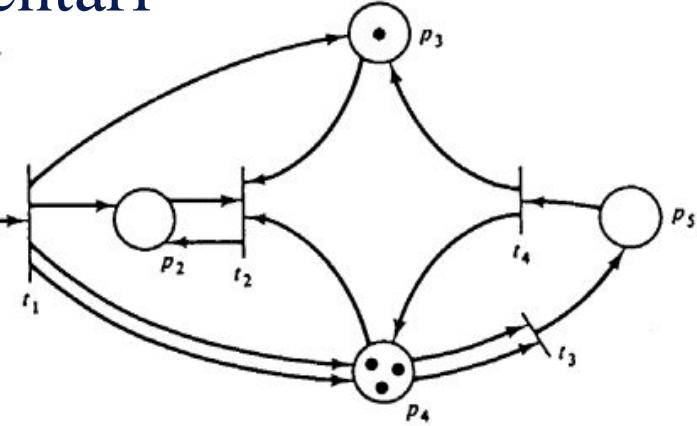
---

- **Non** sono rappresentati **Dati che fluiscono**,
- **Né** strutture di controllo, scelte condizionali, ecc.  
(queste sono realizzate mediante opportune configurazioni di posti e transizioni – vedi poi)
- **Non** valgono ‘**principi di conservazione**’: il **numero totale di token prelevati** (nei posti in entrata) dallo **scatto di una transizione** non è legato al **numero di token messi** (in posti in uscita) **in seguito al medesimo scatto**

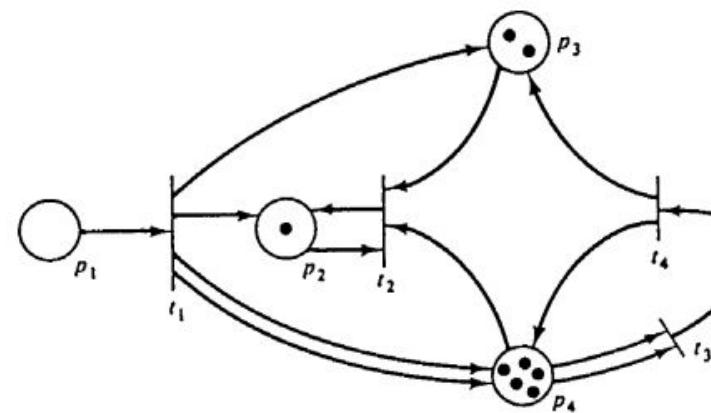
# Esercizi elementari



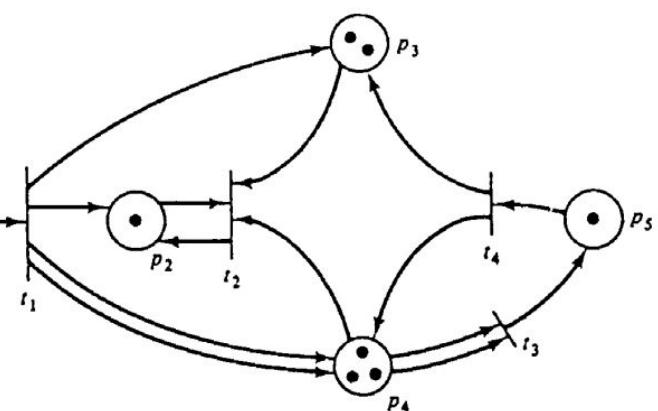
1. Quali le transizioni abilitate allo scatto?



2. Quale è scattata?



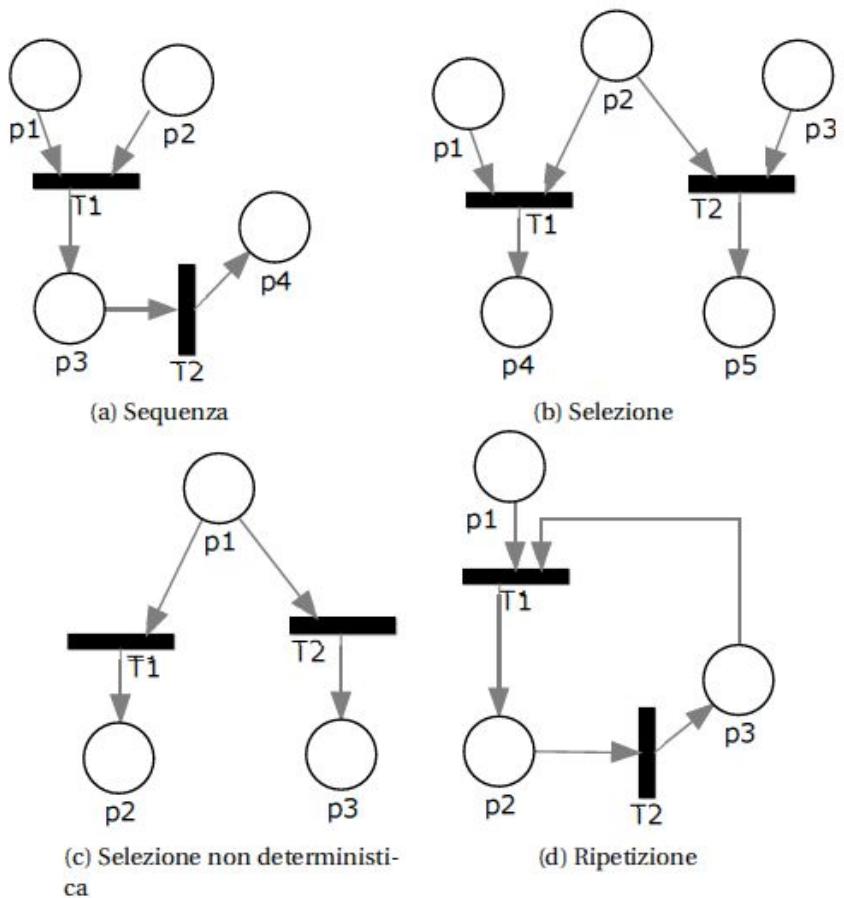
3. Quale è scattata?



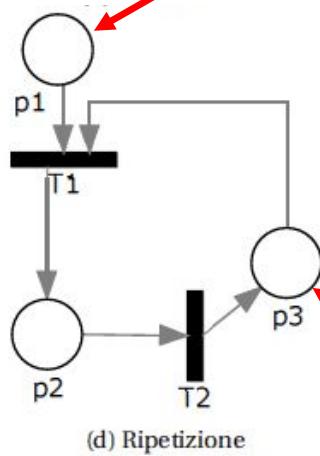
4. Quale è scattata?

# Esempio - costrutti di controllo

- Alcune configurazioni elementari di RdP che rappresentano **costrutti di controllo elementari e tipici**



# Esempio di ciclo



Qui vanno inseriti  $n$  token, con  $n$  che rappresenta il no. di ripetizioni del ciclo

Qui va inserito un token per far partire il ciclo

# Def. di RdP estratta da un articolo

---

“... Carl Adam Petri defined Petri Nets as a graphic mathematical model for describing information flow in 1962. This model proved versatile in visualizing and analyzing the behavior of asynchronous, concurrent systems. Later research led to the direct application of Petri Nets in automata theory. The Petri Nets can model the relations between events, resources, and system states particularly well. **A Petri Net is a bipartite graph with two classes of nodes: places and transitions. The number of places and transitions are finite and nonzero. Directed arcs connect nodes. Arcs either connect a transition to a place or a place to a transition. Arcs can have an associated integer weight. State variables are represented by places. Events are represented by transitions. Places contain tokens. The global state space is defined by the marking of the Petri Net. A marking is a vector expressing the number of tokens in each place. The firing of a transition changes the marking of the Petri Net and the state of the system. The system is nondeterministic in that any enabled transition can fire. ...”**

[da: MENGXIA ZHU and RICHARD R. BROOKS, Comparison of Petri Net and Finite State Machine Discrete Event Control of Distributed Surveillance Networks, *International Journal of Distributed Sensor Networks*, 5: 480–501, 2009.]

# Esercizio

---

- C'è un treno che percorre un binario circolare, dove sono posti tre semafori (verde/rosso): X, Y, Z. Il binario viene così suddiviso in tre tronconi, a, b, c.
- Il treno avanza non appena il semaforo diventa verde.
- I semafori diventano verdi in ordine casuale, ma uno solo alla volta e quando il treno si trova in uno stato stabile (treno fermo).
- Un semaforo da verde diventa rosso non appena il treno gli passa accanto.

# Soluzione

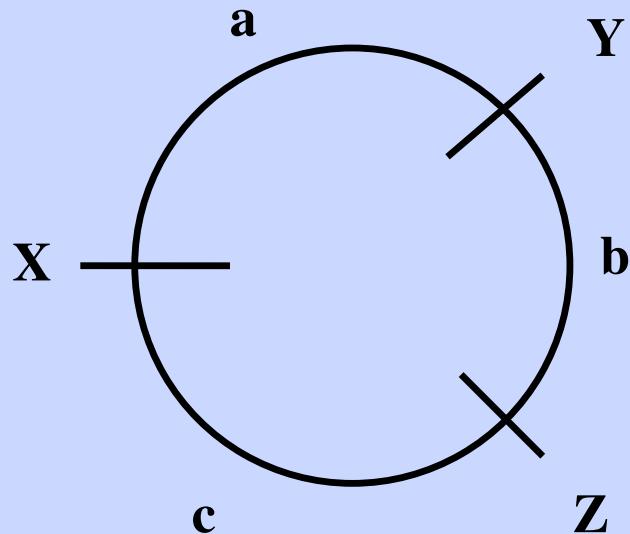
---

Utilizziamo:

- Un posto per ciascun troncone, per indicare (se in esso è presente un token) la presenza del treno nel relativo troncone
- Un posto per ciascun semaforo, per indicare (se in esso è presente un token) che il semaforo è ‘verde’, altrimenti il semaforo è ‘rosso’. ‘Diventa verde’ è un evento esterno, su cui ci si sincronizza
- Una transizione per rappresentare il passaggio del treno al troncone successivo, che può scattare solo se c’è il treno nel troncone precedente e se il semaforo diventa ‘verde’.

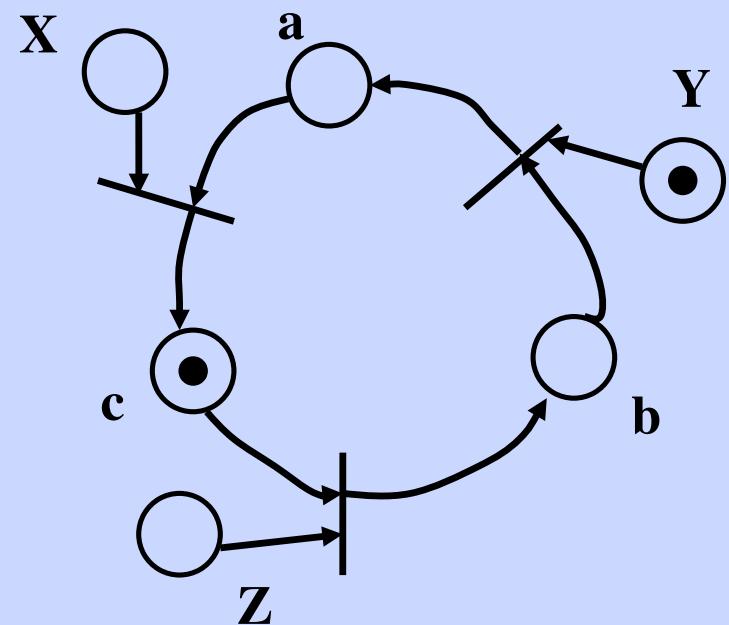
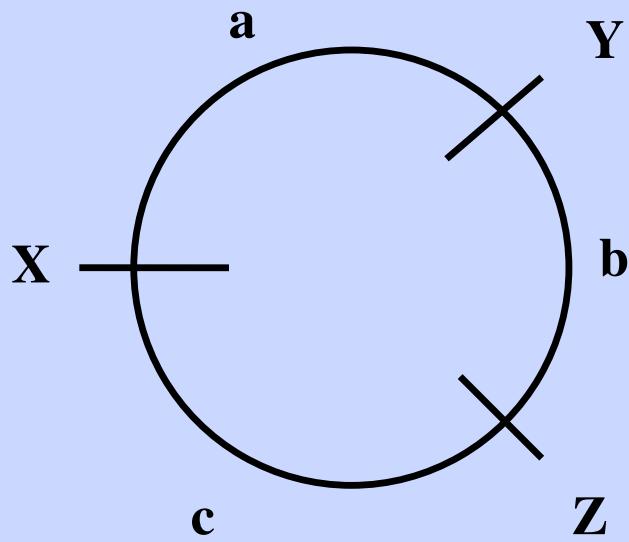
# Soluzione (2)

---



## Esempio (3)

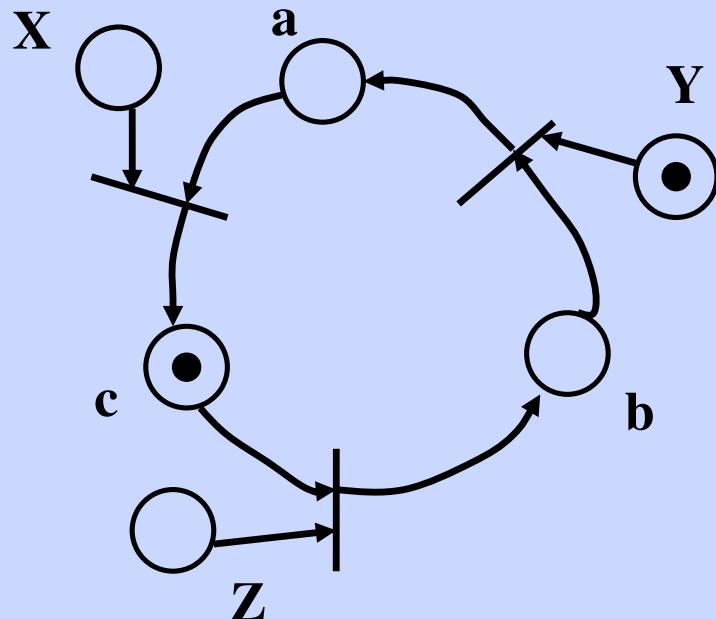
---



# Esempio di evoluzione - 1

---

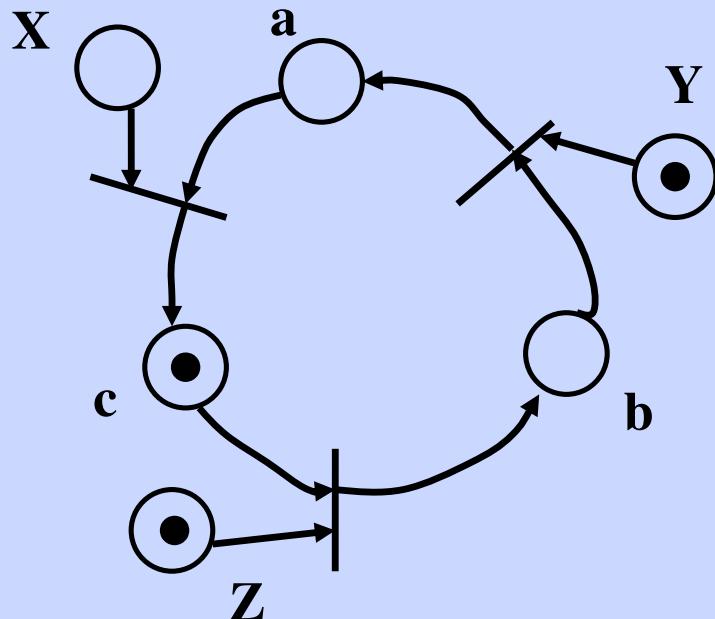
Il treno è in c e il semaforo Y è verde



# Esempio di evoluzione - 2

---

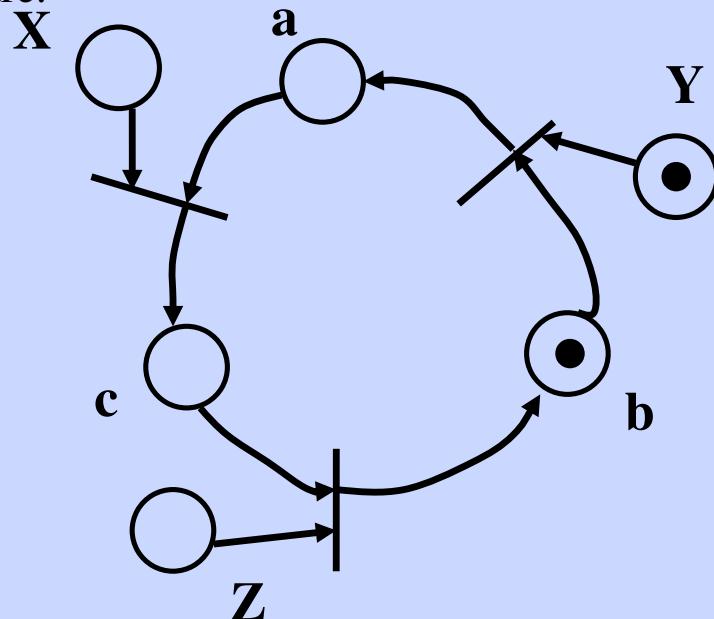
Il semaforo Z diventa verde  il treno può avanzare.



# Esempio di evoluzione - 3

---

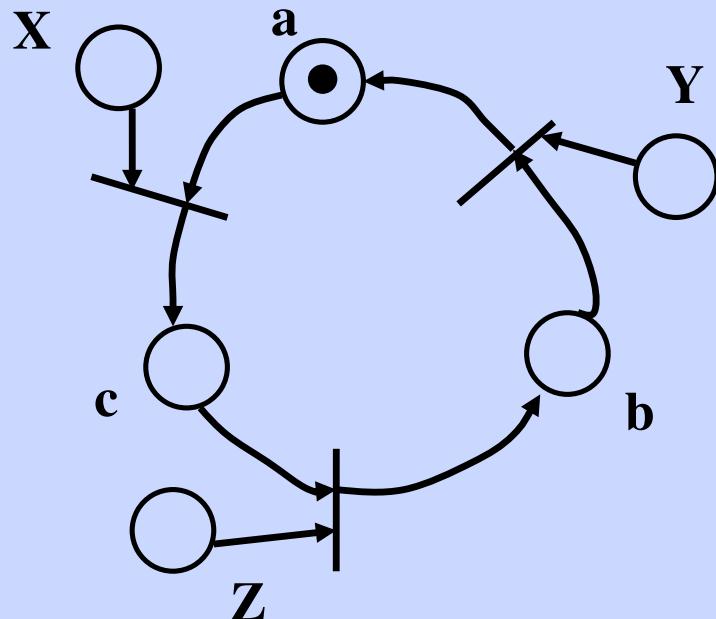
Il treno si trova in b, il semaforo Z diventa rosso ed essendo il semaforo Y già verde, il treno può avanzare.



# Esempio di evoluzione – 3....

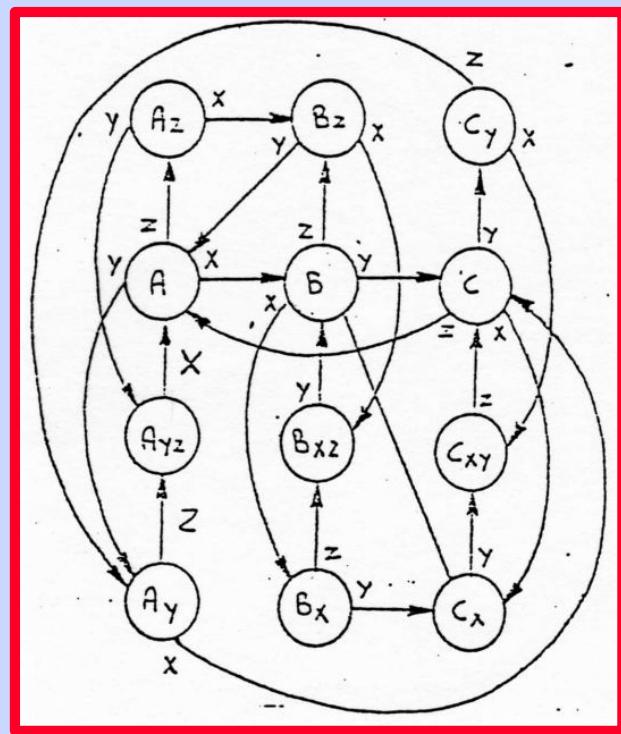
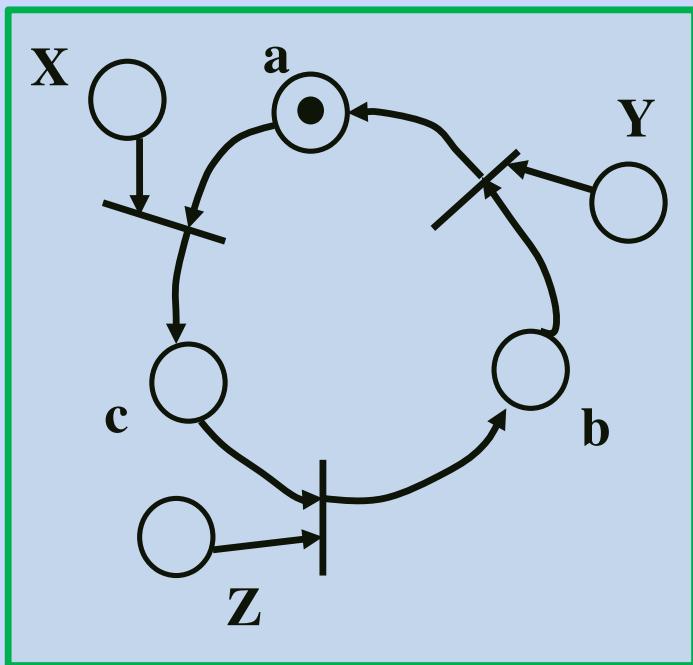
---

Il treno si trova in a, tutti i semafori sono rossi...

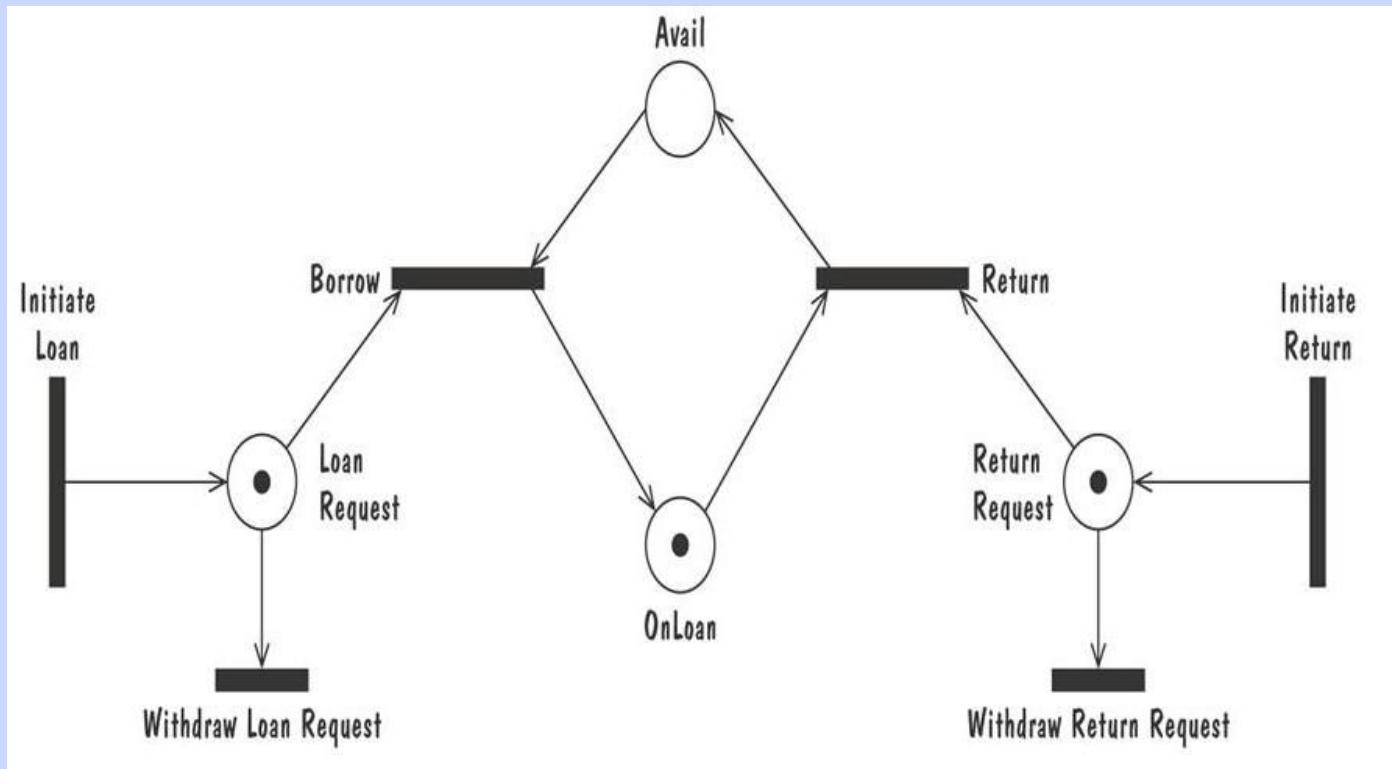


# Esempio di Automa a stati finiti per lo stesso problema

**DA NOTARE LA MAGGIOR COMPLESSITÀ**



# Example: book loan



# Altri esercizi

---

1. Si consideri il seguente frammento di programma:

```
...
T;
case A of
  A1: T1;
  A2: T2;
  A3: T3;
  else T4
end;
T5;
if E1 then goto L;
T6;
L: T7;
...
```

Lo si traduca in una Rete di Petri, associando le istruzioni  $T_i$  alle transizioni. (8)

# Esercizio precedente in Versione JAVA-like con il break

---

- In Java l'esercizio precedente potrebbe essere così modificato:

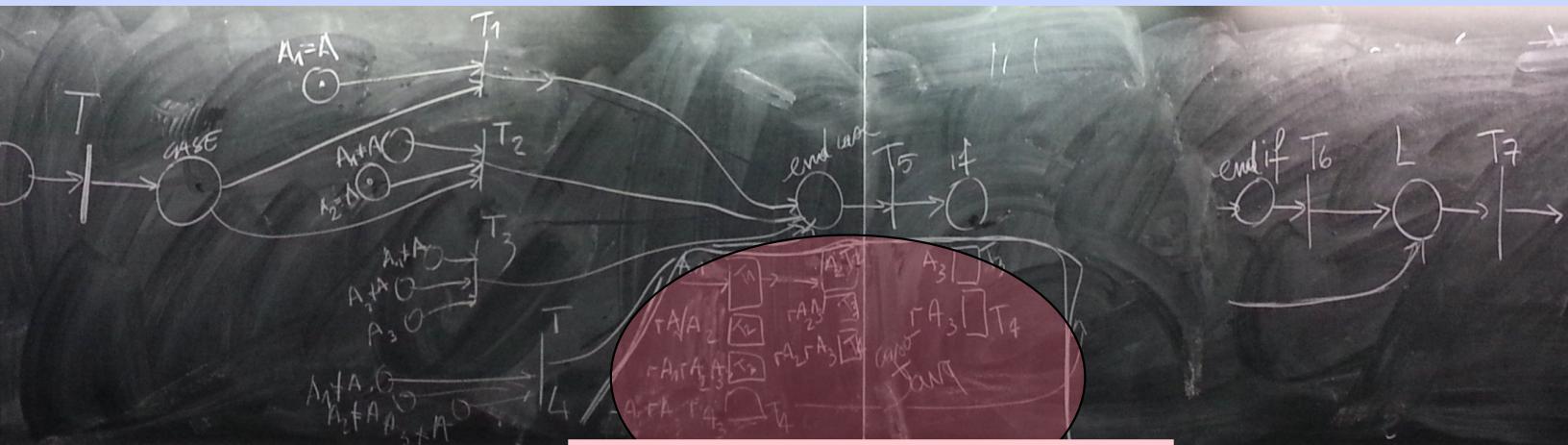
...

*T;*

```
switch (A) {  
    A1: T1; break;  
    A2: T2; break;  
    A3: T3; break;  
    default: T4;  
        break;  
}  
  
T5;  
if E1 then goto L;  
  
T6  
  
L: T7;  
  
...
```

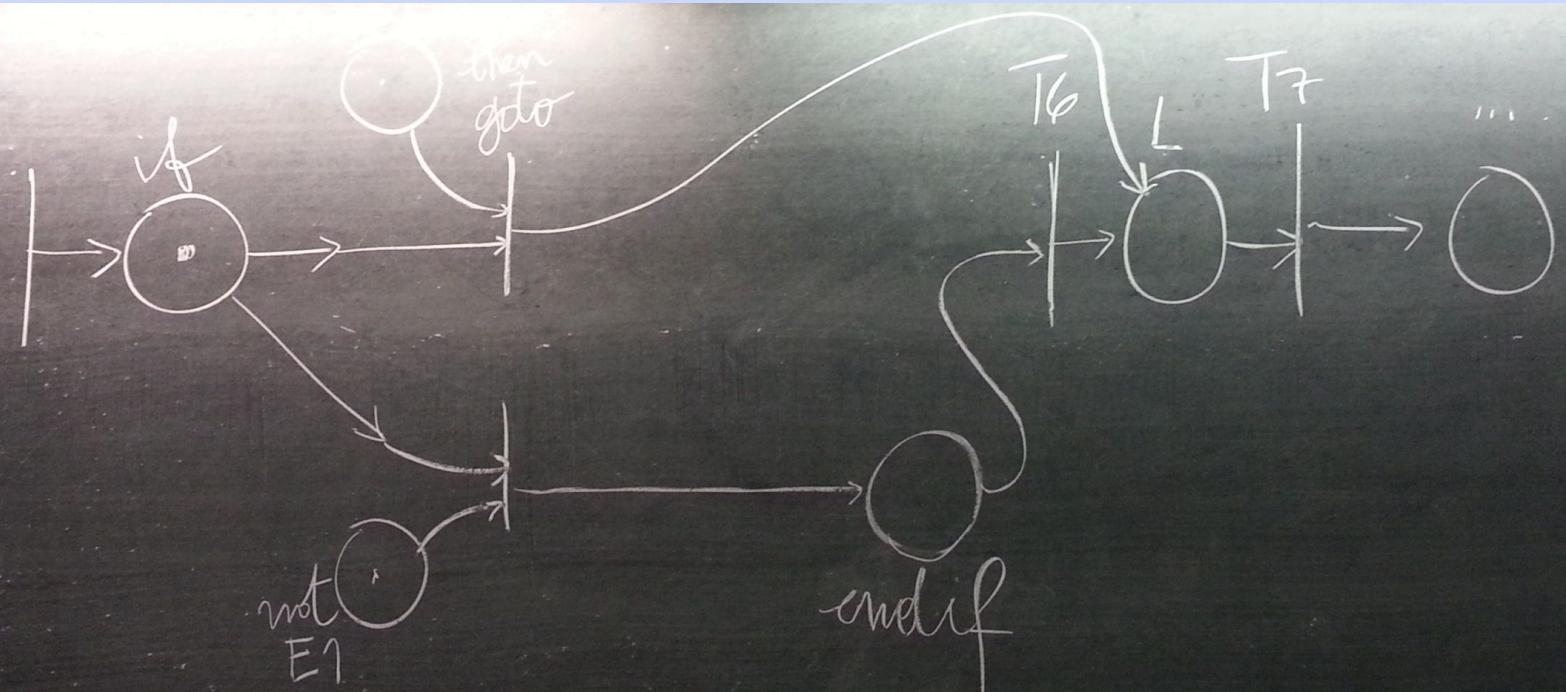
# Prima Parte (if escluso)

Assunzione sul Case. Viene verificato se  $A_1 = A$ , se sì, si esegue  $T_1$ , altrimenti se è verificato  $A_2 = A$ , si esegue  $T_2$ , altrimenti se  $A_3 = A$  si esegue  $T_3$ , altrimenti si esegue  $T_4$



Caso dello switch del JAVA

# Seconda Parte: if e Goto



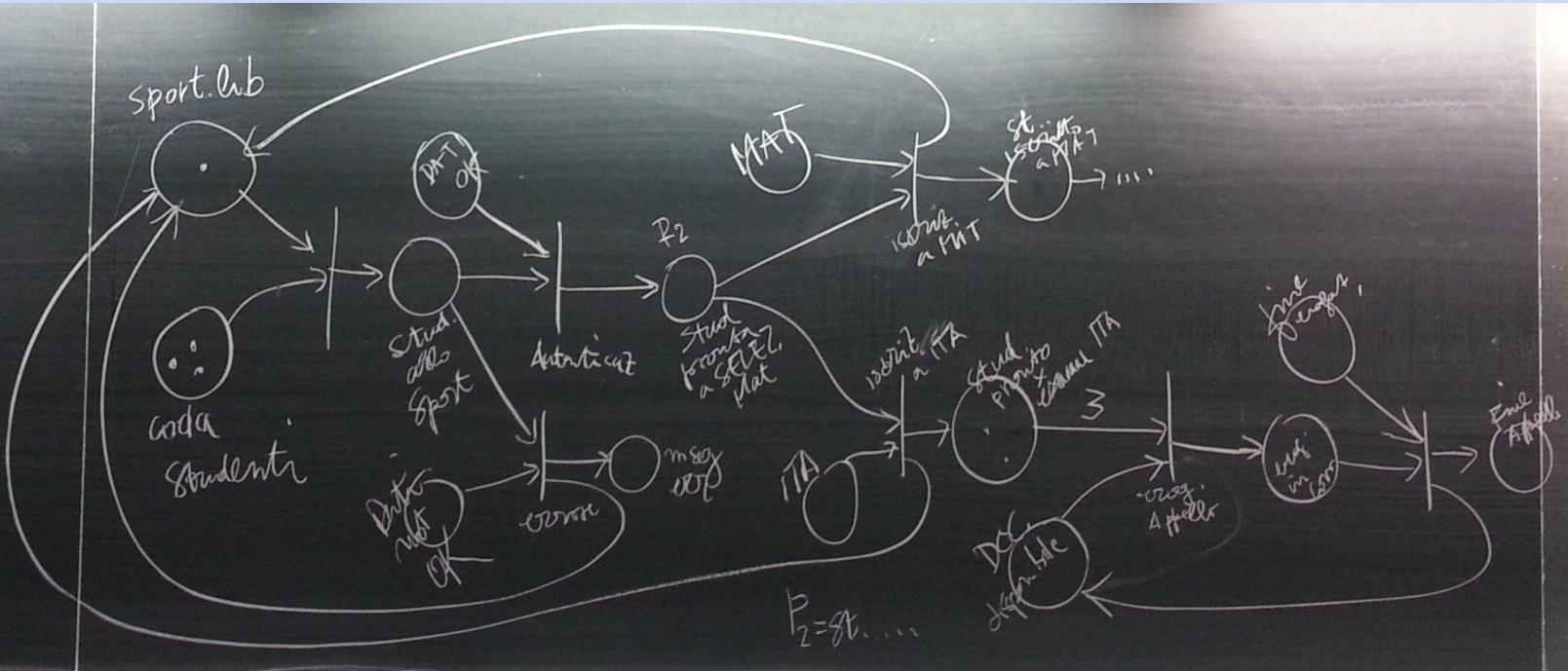
# Esercizio FACOLTA'

---

Una Facoltà intende automatizzare le modalità di iscrizione agli esami orali. In particolare si vuole sviluppare un sistema software collegato ad uno sportello automatico, al quale gli studenti possano accedere (uno alla volta), fornire opportuni identificativi, selezionare la materia su cui desiderano effettuare l'esame, ed iscriversi. Gli appelli non hanno date stabilite, bensì il docente di una data materia può accedere al sistema e se ci sono almeno 3 studenti iscritti, deve erogare un appello per 3 studenti. Gli eventuali altri studenti iscritti verranno esaminati successivamente. Si progetti una Rete di Petri che descrive il controllo di tale sistema. (8)

# Esercizio Sportello Automatico

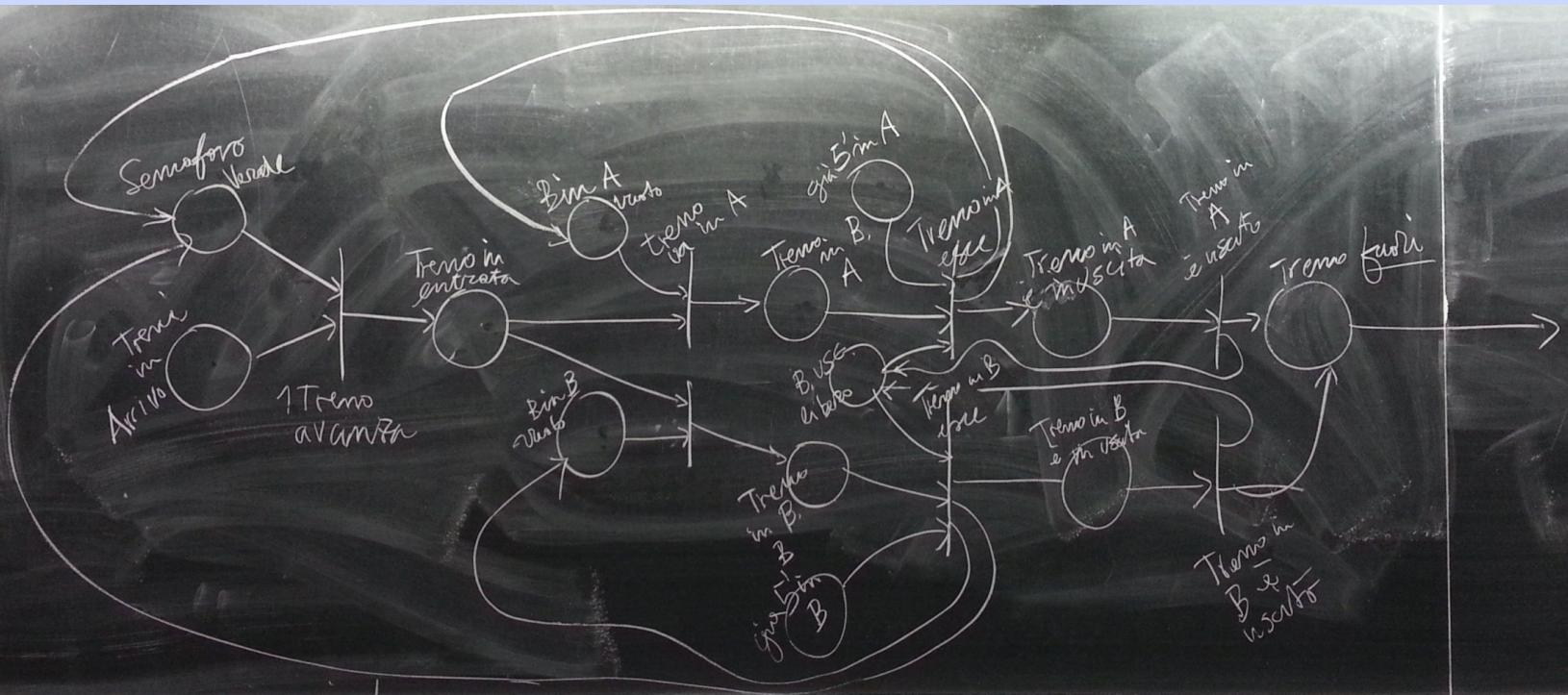
## Iscrizione Esami FACOLTA'



# Esercizio TRENO

**Una linea ferroviaria costituita da un solo binario è percorsa dai treni in una sola direzione. All'interno delle stazioni, il binario si sdoppia in due binari interni che possono ospitare al massimo un treno ciascuno. All'ingresso della stazione c'è un semaforo. Se questo è rosso (cioè se in stazione i binari sono già occupati), i treni in arrivo si accodano. Il semaforo è verde se almeno uno dei due binari interni è libero. All'accensione del verde i treni entrano in stazione e vengono indirizzati sul binario libero (se entrambi sono liberi, la scelta è casuale). La ripartenza di un treno è abilitata dopo 5' di fermata, ma il treno procede solo se i binari in uscita sono liberi, ossia nessun altro treno è già in fase di uscita, nel qual caso attende che il treno sia già fuori dalla stazione. Si rappresenti il sistema descritto mediante una rete di Petri. (8)**

# Esercizio TRENO



# Altro Esercizio TRENO

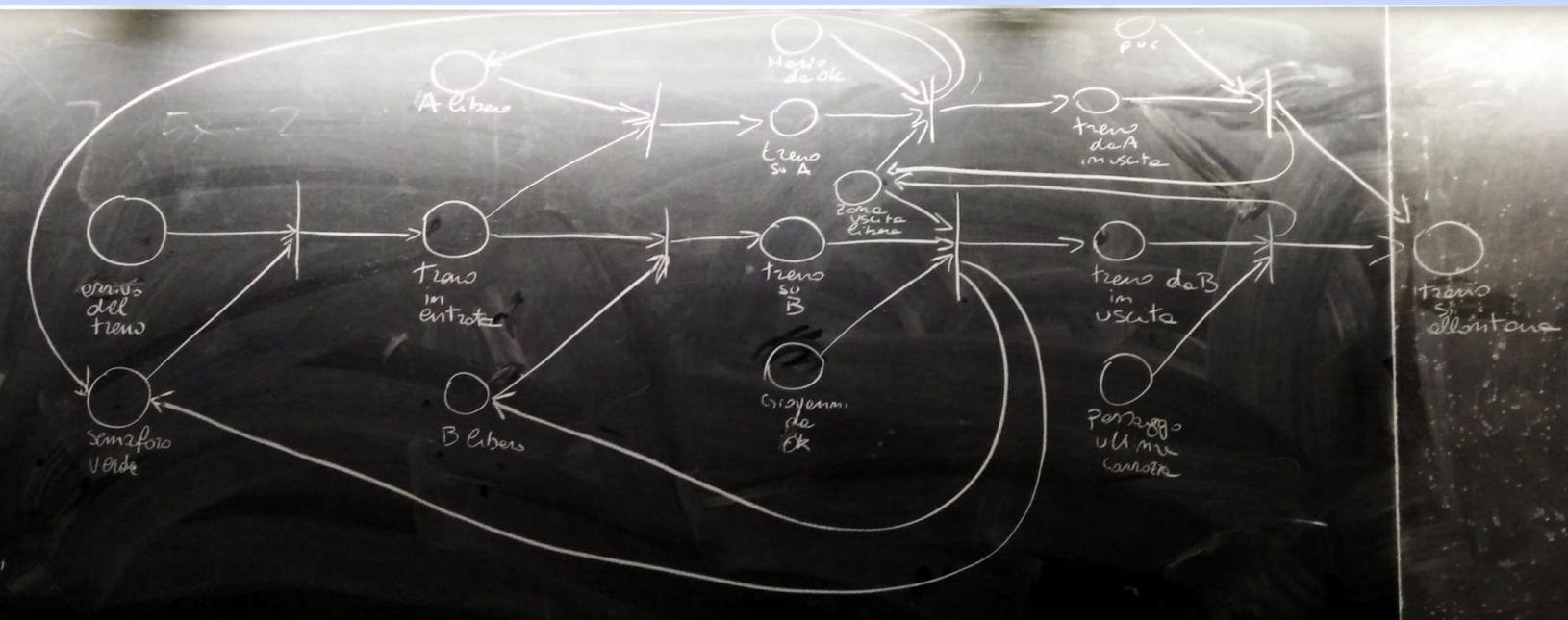
Una linea ferroviaria costituita da un solo binario è percorsa dai treni in una sola direzione. All'interno delle stazioni, il binario si sdoppia in due binari (A e B), che possono ospitare al massimo un treno ciascuno. All'ingresso della stazione, c'è un semaforo: se questo è rosso (cioè se in stazione ci sono già i binari occupati), i treni in arrivo si accodano. Il semaforo è verde se è libero uno qualunque dei due binari della stazione o entrambi. All'accensione del verde, un treno in arrivo entra in stazione, indifferentemente su A o B (se liberi!). La partenza del treno sul binario A è decisa dal capostazione Mario, mentre quella del treno sul binario B da Giovanni \*

Si progetti una Rete di Petri che modella tale situazione. (8)

(\*) Il binario di uscita può ospitare un treno alla volta.

# Soluzione in aula

(14.11.2016)



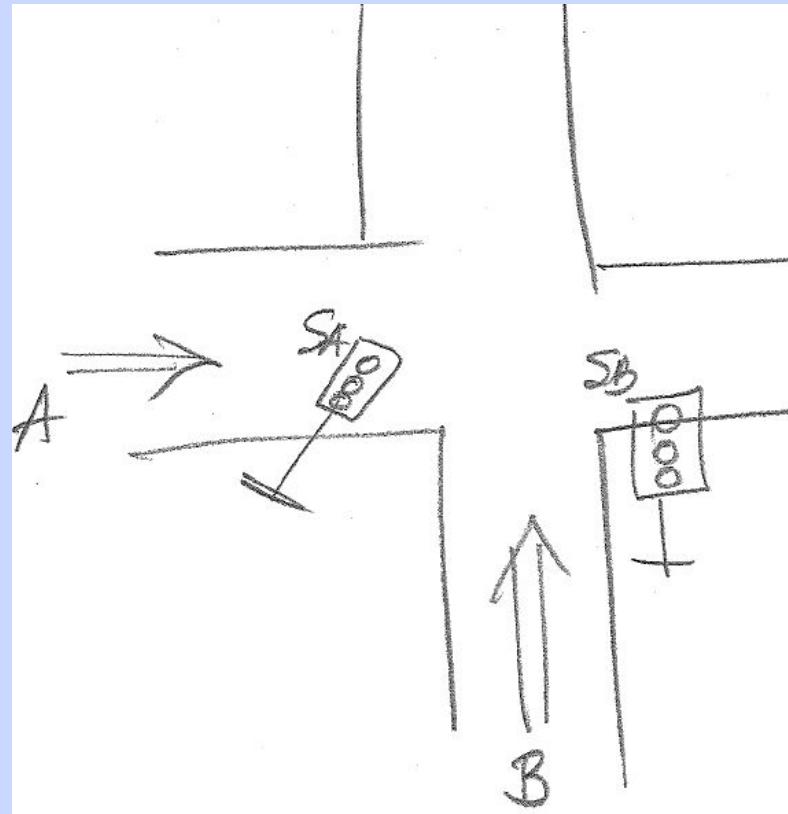
Requisito aggiunto rispetto al testo:  
«Il binario di uscita può ospitare un treno alla volta.»

# Altro Esercizio: INCROCIO con 2 semafori

Ad un incrocio (organizzato come rappresentato in figura) al semaforo SA rimane verde per 2', poi giallo per 10'' e poi Rosso per 1'10''.

Nel frattempo SB è rosso quando SA è verde o giallo e rimane giallo anch'esso per 10''.

I passaggi al rosso di un semaforo sono sincronizzati con i passaggi al verde dell'altro semaforo.



# Requisiti di sincronizzazione

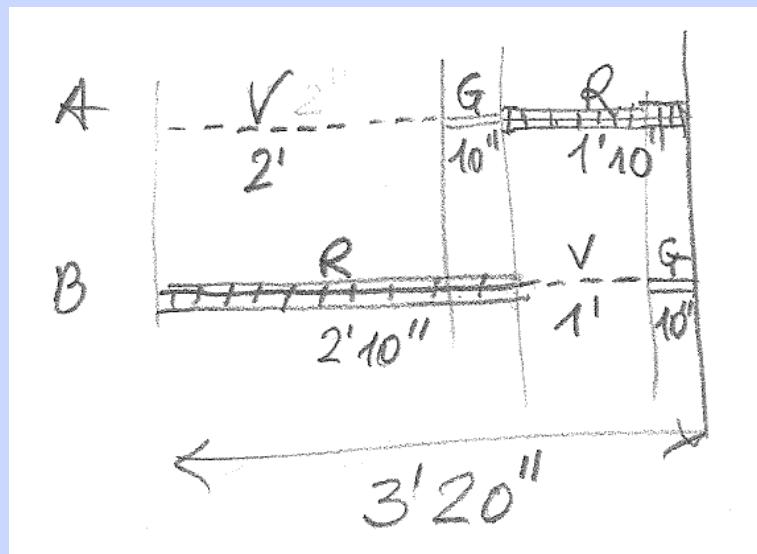
---

Quando SA passa  
da Giallo,  
allora SB passa  
da Rosso a Verde

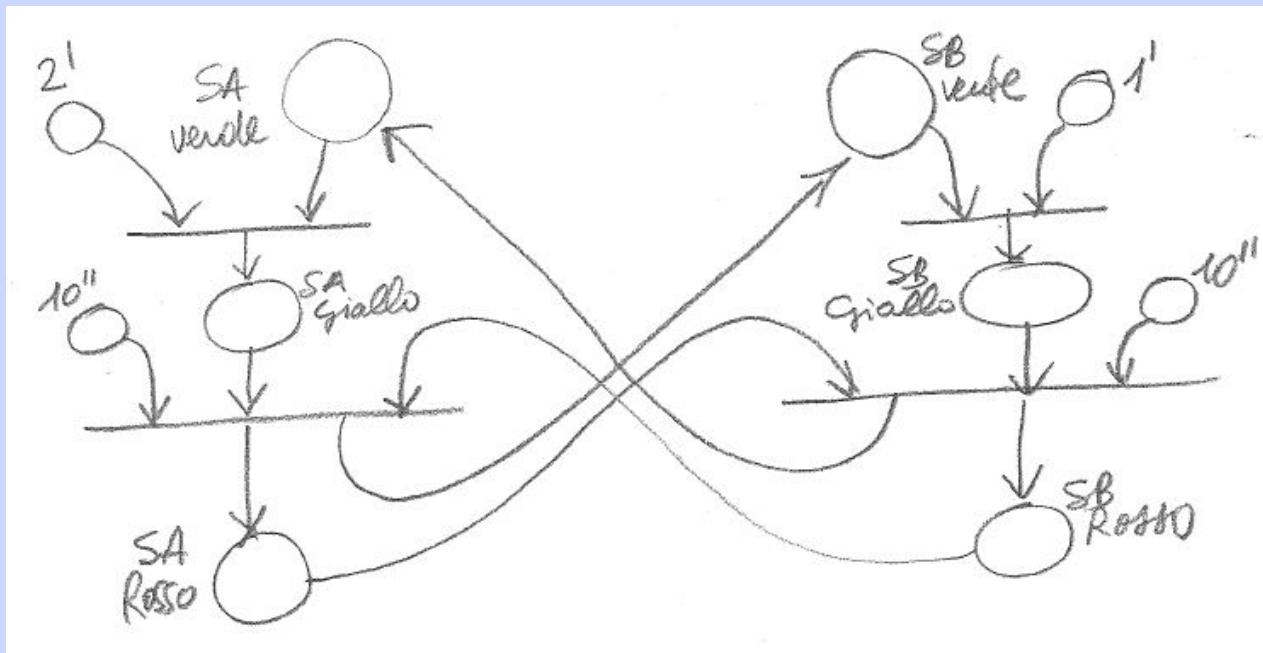
Quando SB passa  
da Giallo a Rosso,  
allora SA passa  
da Rosso a Verde

In altri termini, il  
passaggio al VERDE deve  
avvenire solo se l'altro  
semaforo è ROSSO!!

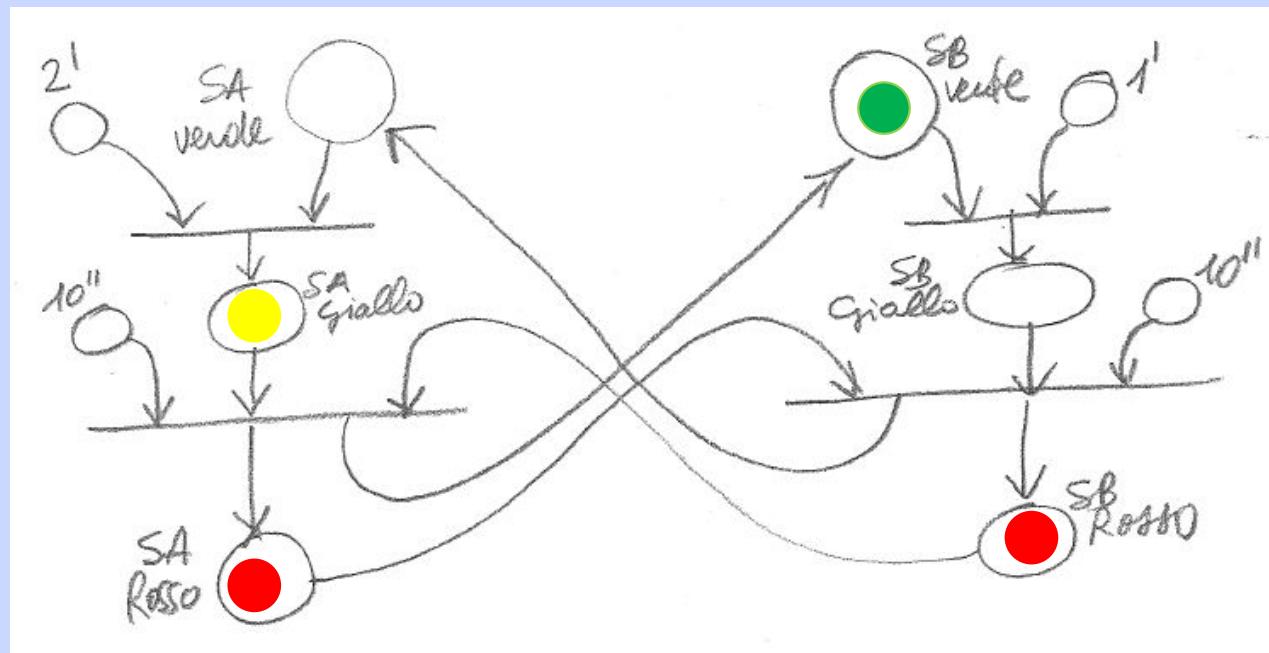
# Analisi delle tempistiche



# Soluzione con sincronizzazione sul passaggio da Giallo a Rosso

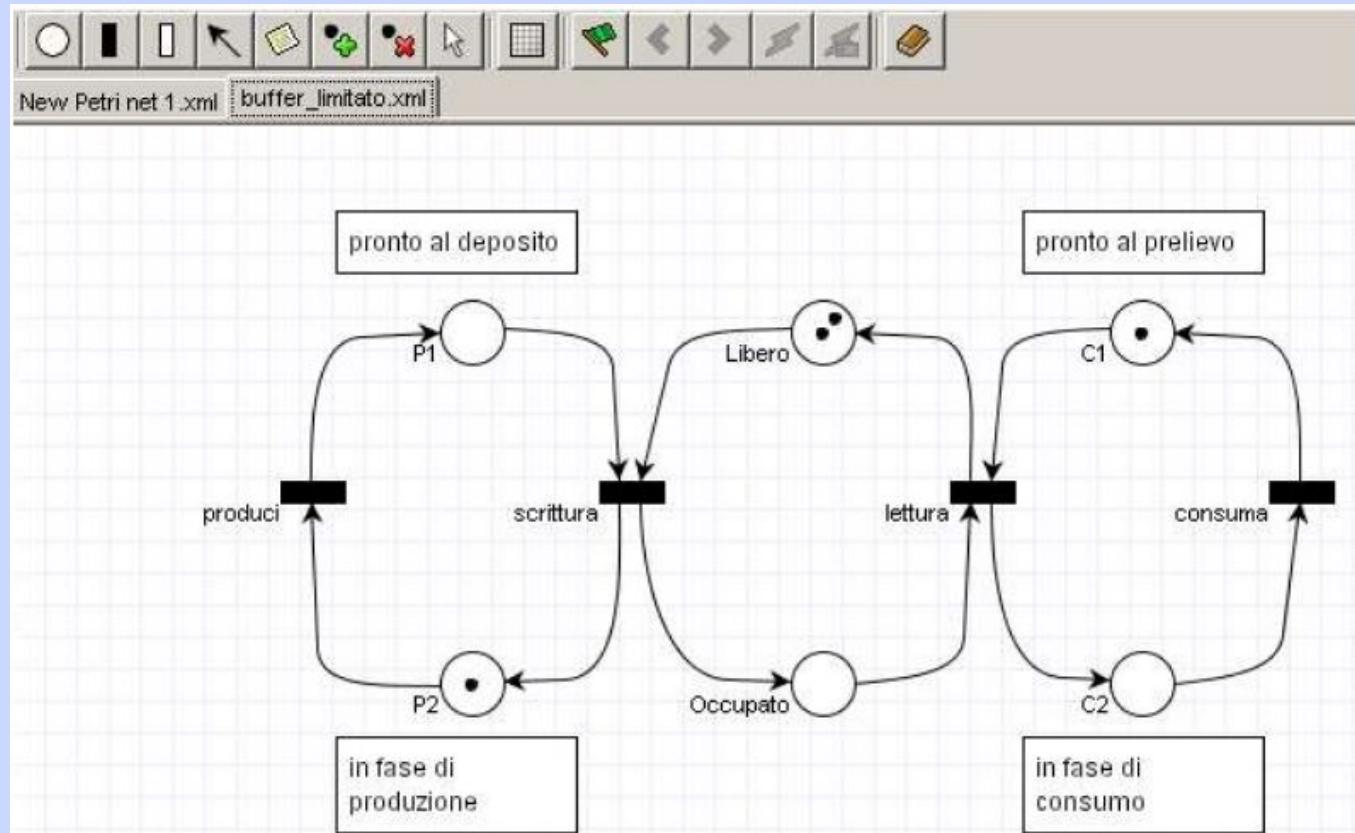


# Passaggio a Rosso di A e contemporaneo (stessa transizione) passaggio da Rosso a Verde di B

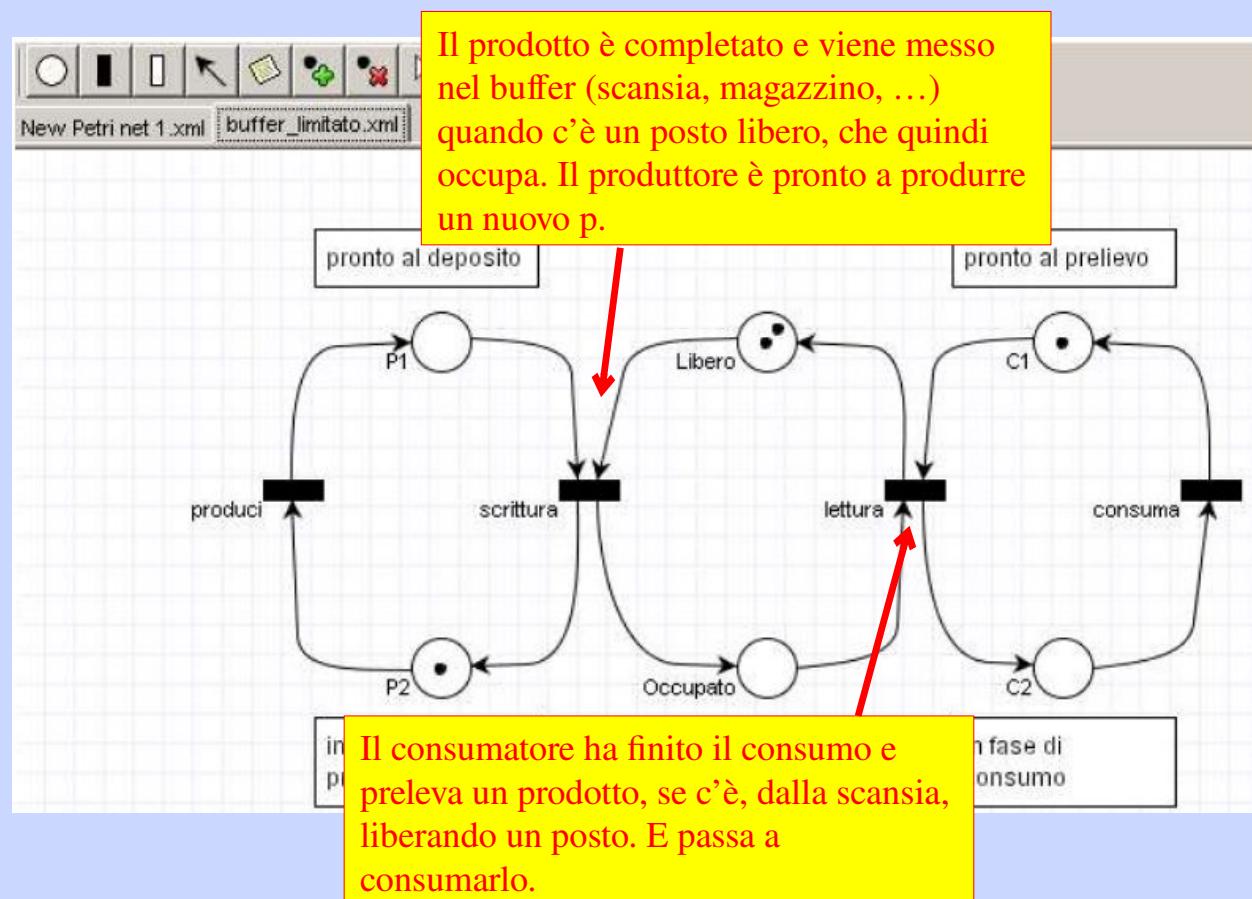


(analogamente quando B passa da Giallo a Rosso)

# Esempio: Produttore/Consumatore collegati da un buffer



# Esempio: Produttore/Consumatore collegati da un buffer - 2



Esercizio 1 - Si intende sviluppare un sistema di gestione di sessioni di esercitazioni on-line. All'interno di tale sistema gli utenti-istruttori possono programmare delle esercitazioni a cui possono partecipare determinati utenti-allievi. L'utente-istruttore è tenuto a specificare, per ogni sessione, data, ora, durata prevista e allievi invitati all'esercitazione. Ciascun utente-allievo ha a disposizione un calendario su cui sono segnate le esercitazioni a cui è tenuto a partecipare, il sistema inoltre genera delle notifiche push agli utenti-allievi quando vengono programmate nuove esercitazioni a cui sono invitati. All'ora fissata dall'utente-istruttore, gli utenti-allievi sono tenuti ad unirsi all'esercitazione on-line, gli utenti che non eseguono l'accesso entro 15' dall'inizio della sessione sono considerati assenti. Presenze e assenze vengono registrate dal sistema. Ad ogni esercitazione è associata, per ogni utente-allievo presente, una valutazione compilata dall'utente-istruttore al termine della sessione. Si descrivano, utilizzando il linguaggio UML, i casi d'uso principali del sistema, si stenda un diagramma delle classi e si dettagli mediante un diagramma di sequenza il processo di creazione di un'esercitazione. (8)

**Esercizio 2 - Si modelli attraverso il formalismo delle Reti di Petri il funzionamento di un sistema di gestione di una lavorazione industriale. Man mano che le commesse arrivano, vengono messe in attesa di un operatore umano che le valuta una per una ed inserisce il pezzo associato alla commessa nella coda del macchinario adatto. L'impianto di produzione in esame ha 3 operatori. Ciascun macchinario può lavorare un pezzo alla volta e l'impianto di produzione considerato consiste in tre linee di lavorazione distinte ciascuna con due macchine gemelle. I pezzi lavorati vengono inviati alla (unica) linea di imballaggio dove una macchina dedicata li imballa a gruppi di 4 pezzi. (8)**

Esercizio 3 - A. Quali sono i principali elementi che caratterizzano i Metodi Agili? B. Come si può descrivere l'Extreme Programming in termini di Incremental development durante il Corso indicato anche con termini quali 'Sviluppo basato su incrementi successivi', 'Consegna incrementale' o 'Incremental delivery'? (6)

Esercizio 4 - A. Si illustrino le due principali tecniche di verifica del software. B. Cos'è e a cosa serve il path testing? C. Cos'è la complessità ciclotomica e come può essere utile per il path testing? (6)

## Esercizio 2

Si modelli tramite il formalismo delle *Reti di Petri* un'istanza semplificata del sistema descritto nell'esercizio precedente. In tale scenario, la struttura di ricerca dispone di tre ricercatori e di due macchine remote. Il project manager, quando insorgono nuove esigenze, assegna le relative attività ai ricercatori, in particolare ciascuna attività ad un qualsiasi ricercatore libero (non impegnato nello svolgimento di precedenti attività assegnate a lui). Il ricercatore cui è assegnata un'attività accede al pannello di controllo mediante il quale avvia una sessione di elaborazione su una qualsiasi delle due macchine, purché libera (non impegnata in altre sessioni di elaborazione). Alla fine della elaborazione il ricercatore segnala la conclusione della stessa e carica all'interno del sistema *eventuali* risultati, rimanendo quindi libero per ricevere altre assegnazioni dal project manager. (8)

# Tipologie di modelli esaminati

---

## 1. Context models

1. Diagrammi contesto
2. Process models

## 2. Behavioural models

1. Data processing models
2. State machine
3. Petri nets

## 3. Semantic Models → semantica dei dati, caratteristiche delle varie entità e rappresentare le relazioni

1. E-R models
2. Data Dictionary → lista dei nomi utilizzati all'interno del sistema, GLOSSARIO

## 4. Objec Models

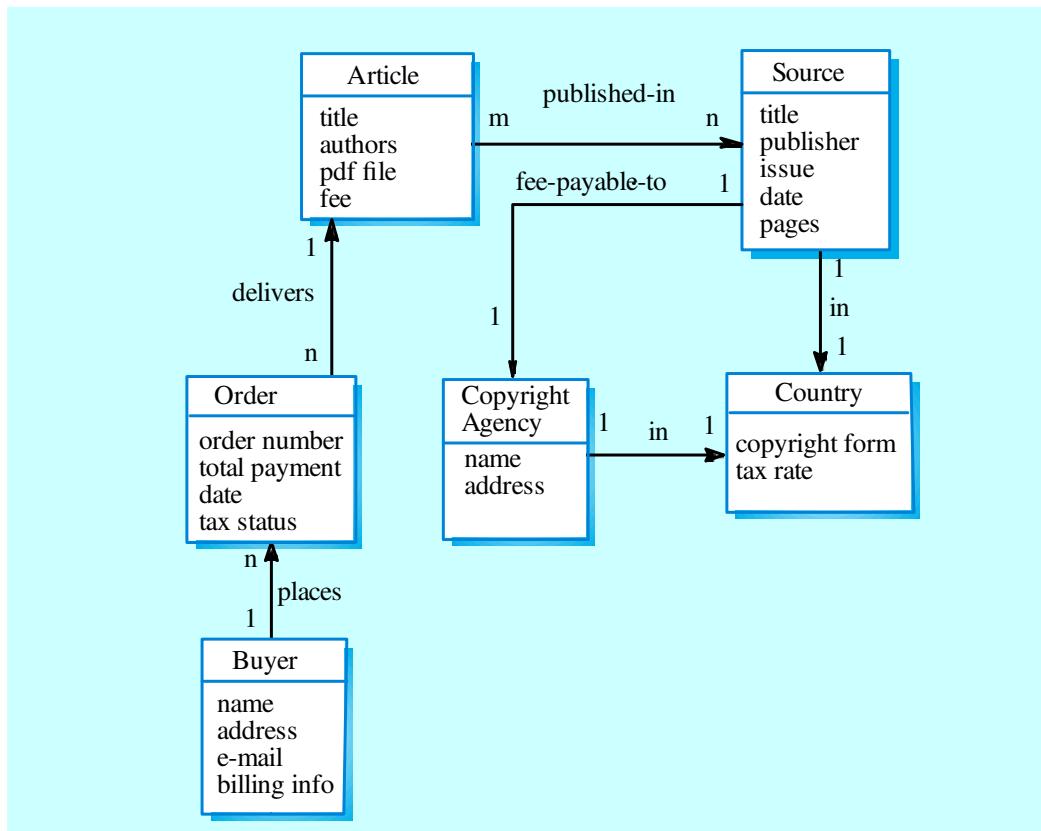
1. Inheritance
2. Aggregation
3. Behavioural

# 3. Semantic data models

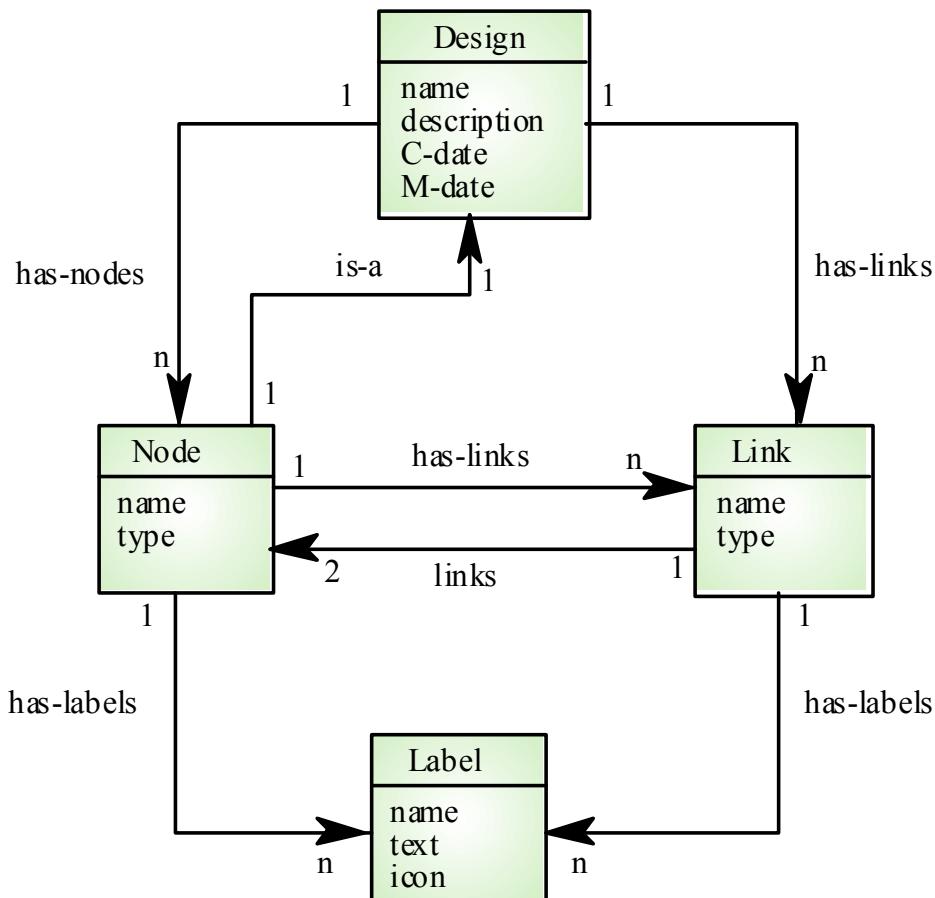
---

- Used to describe the logical structure of data processed by the system
- **3.1 Entity-relation-attribute model** sets out the entities in the system, the relationships between these entities, and the entity attributes
- Widely used in database design. Can readily be implemented using relational databases
- No specific notation provided in the **UML** but objects and **associations** ( + comments) can be used

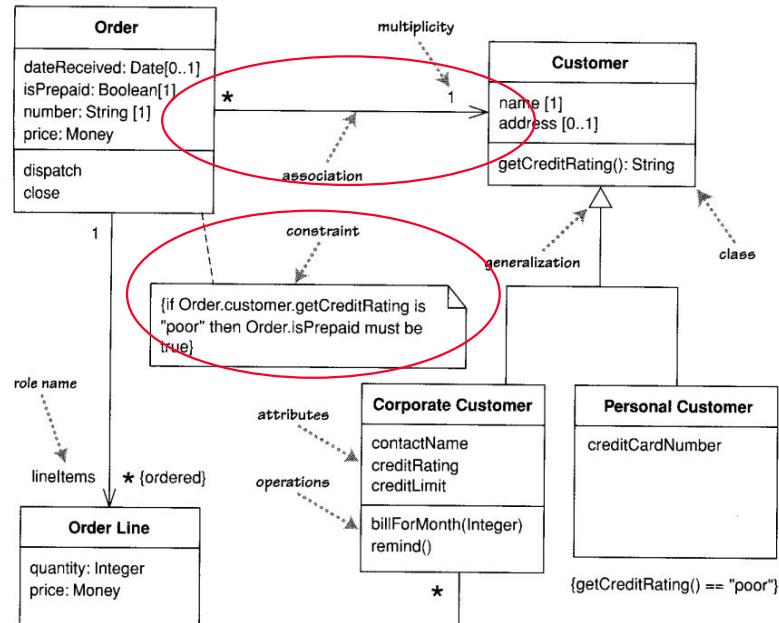
# Library semantic model



# Software design semantic model



# UML associations



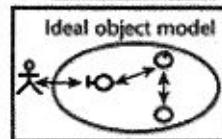
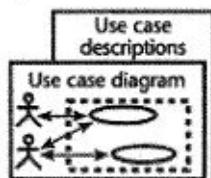
# Diagrammi UML

## BPR workflow diagrams

Workflow diagrams are a kind of activity diagram used to define an entire process.

## The UML use case diagrams

Use cases define generic processes the system must be able to handle. Descriptions define generic scenarios.



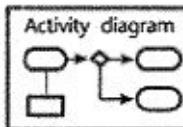
Jacobson's ideal object model defines three types of classes according to their overall function.

## CRC cards

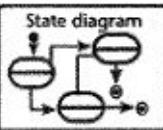
CRC cards provide users and developers with an informal way to identify classes, attributes, and messages by working through scenarios.

## The UML state diagrams

Activity diagrams show all the activities that occur as the values of an object change. Activity diagrams are used to capture workflows or decision sequences.

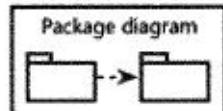


A state diagram shows all of the values (states) that the attribute of an object can take as messages (events) are processed. State diagrams are only prepared for classes whose instances are very dynamic.

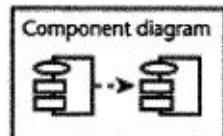


## The UML implementation diagrams

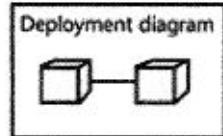
Implementation diagrams show design and architectural decisions.



Package diagrams show the logical division of classes into modules.



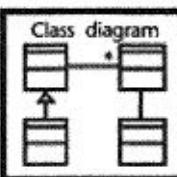
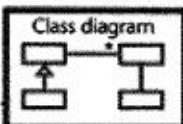
Component diagrams show the actual software modules in the final system. These are often the same as the package diagrams.



Deployment diagrams show the actual platforms (nodes) and network links used by the system.

## The UML static structure diagrams

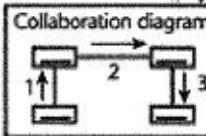
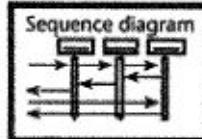
The class diagram describes classes, their associations with other classes (responsibilities), and inheritance relationships. More complex class diagrams describe class attributes and operation names.



Object diagrams are used to explore specific problems with specific classes.

## The UML interaction diagrams

Sequence diagrams show the flow of messages (events) between objects. In effect they provide a formal way to specify a scenario.



Collaboration diagrams are a combination of object and sequence diagrams. They show the flow of events between objects.

## 3.2 Data dictionaries

---

- Data dictionaries are **lists of all of the names** used in the system models. Descriptions of the **entities, relationships, and attributes** are also included
- Advantages
  - Support **name management** and avoid duplication
  - Store of organisational knowledge linking analysis, design and implementation
- Many CASE workbenches support data dictionaries

# Data dictionary entries

## Esempio

Name	Description	Type	Date
Article	Details of the published article that may be ordered by people using LIBSYS.	Entity	30.12.2002
authors	The names of the authors of the article who may be due a share of the fee.	Attribute	30.12.2002
Buyer	The person or organisation that orders a copy of the article.	Entity	30.12.2002
fee-payable-to	A 1:1 relationship between Article and the Copyright Agency who should be paid the copyright fee.	Relation	29.12.2002
Address (Buyer)	The address of the buyer. This is used to any paper billing information that is required.	Attribute	31.12.2002

# Data dictionary entries

---

Name	Description	Type	Date
has-labels	1:N relation between entities of type Node or Link and entities of type Label.	Relation	5.10.1998
Label	Holds structured or unstructured information about nodes or links. Labels are represented by an icon (which can be a transparent box) and associated text.	Entity	8.12.1998
Link	A 1:1 relation between design entities represented as nodes. Links are typed and may be named.	Relation	8.12.1998
name (label)	Each label has a name which identifies the type of label. The name must be unique within the set of label types used in a design.	Attribute	8.12.1998
name (node)	Each node has a name which must be unique within a design. The name may be up to 64 characters long.	Attribute	15.11.1998

# Tipologie di modelli esaminati

---

## 1. Context models

1. Diagrammi contesto
2. Process models

## 2. Behavioural models

1. Data processing models
2. State machine
3. Petri nets

## 3. Semantic Models

1. E-R models
2. Data Dictionary

## 4. Objec Models

1. Inheritance
2. Aggregation
3. Behavioural

# Modelli ad oggetti

---

Parte svolta da A. Baruzzo

Per l'esame, riferirsi alle lezioni e al libro del Dr. Baruzzo.

# Notazioni per la rappresentazione dei sistemi

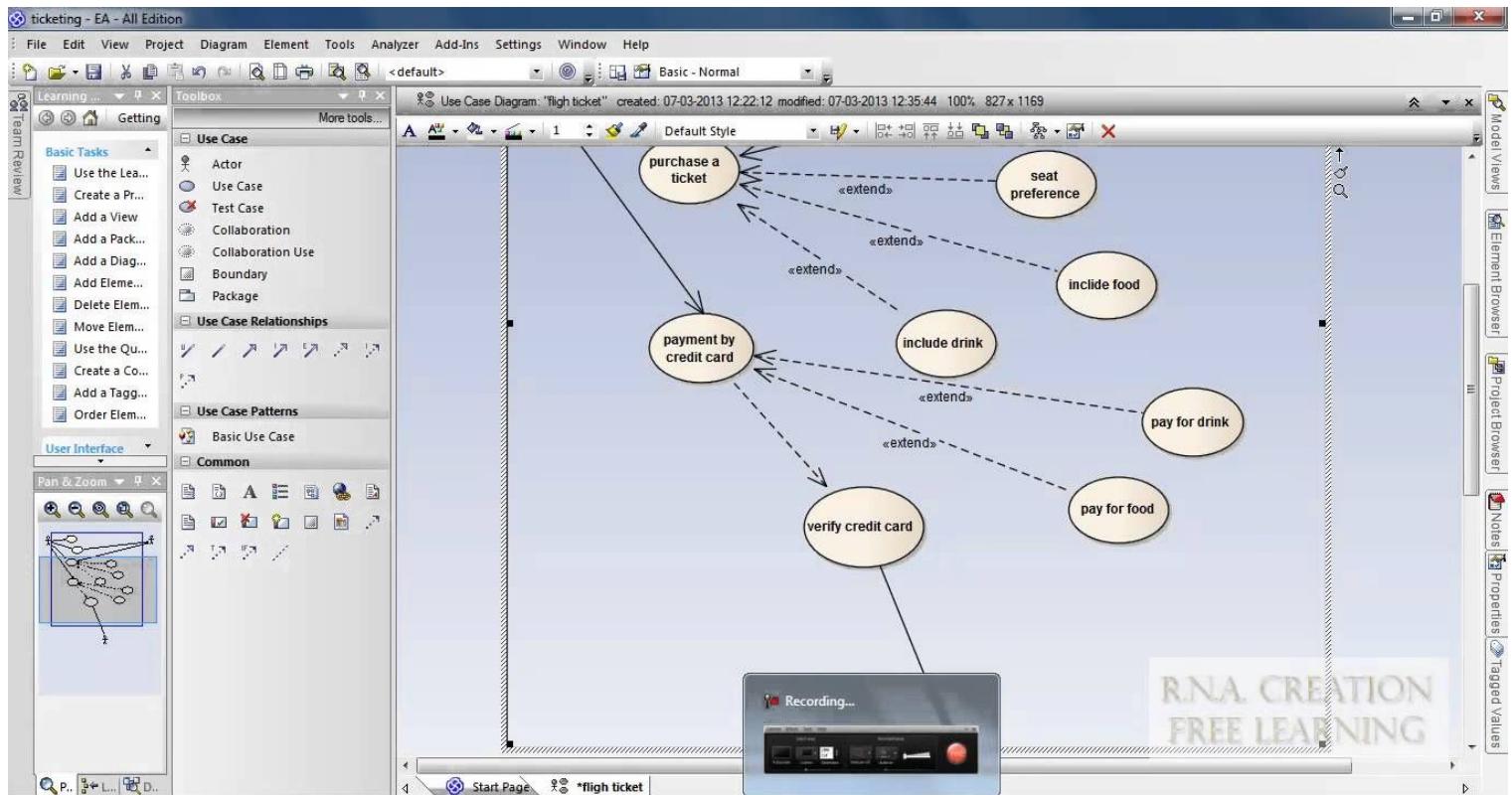
---

- 1 CONTESTO
  - 1.1 Diagrammi di Contesto (box e frecce/linee)
  - 1.2 Modelli dei processi (Contesto dei Processi, ellissi e frecce)
- 2 COMPORTAMENTALI
  - 2.1 Data Processing Models – DFD (ellissi e frecce)
  - 2.2 State Machine (UML Statecharts)
  - 2.3 Reti di Petri (nodi, frecce, barrette)
- 3 SEMANTIC Data Models
  - 3.1 Entity-Relationships Models (UML objects + associations)
  - 3.2 Data Dictionary (elenchi, tavole)
- 4 OBJECT MODELS (UML)
  - 4.1 Inheritance Models (UML object classes + associations + generalisations)
  - 4.2 Aggregation Models (UML object aggregation)
  - 4.3 Object Behaviour Models (UML sequence diagrams, activity & collaboration diagrams)

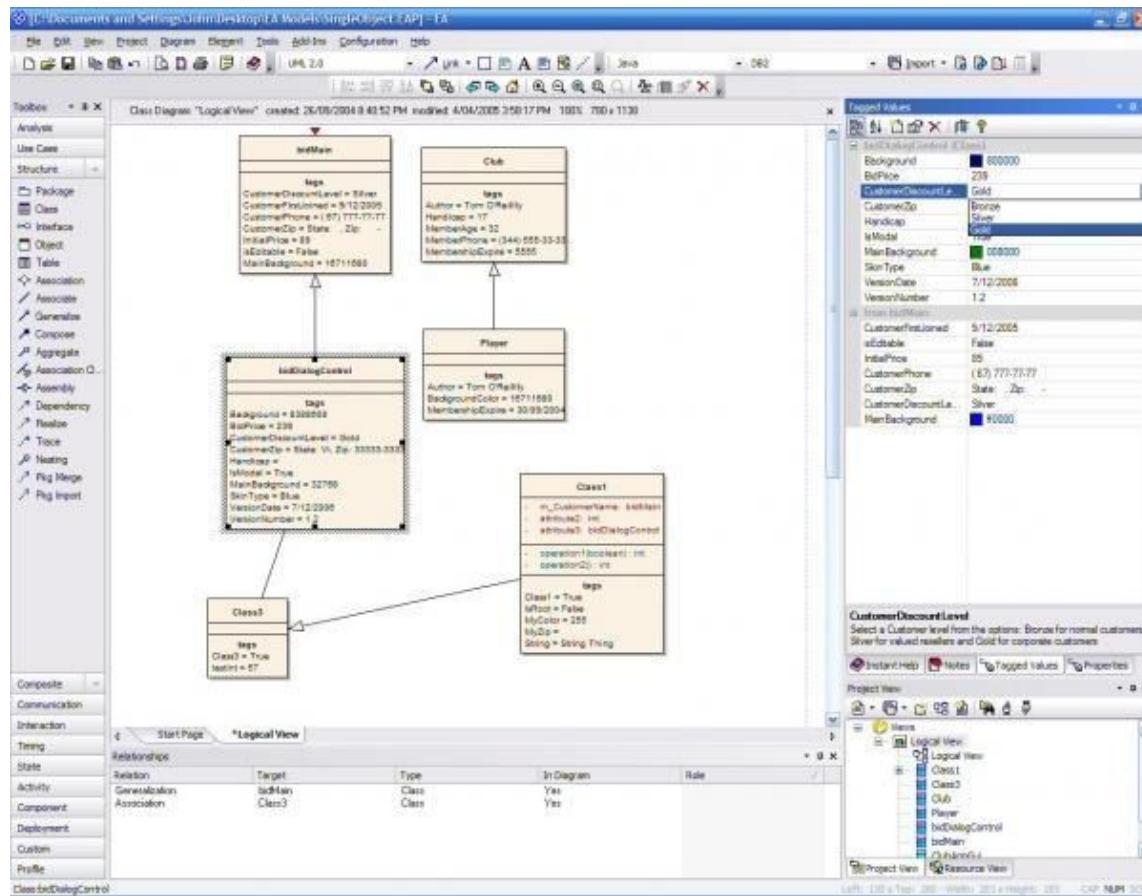
# CASE Tool a supporto dell'analisi e del design SW

---

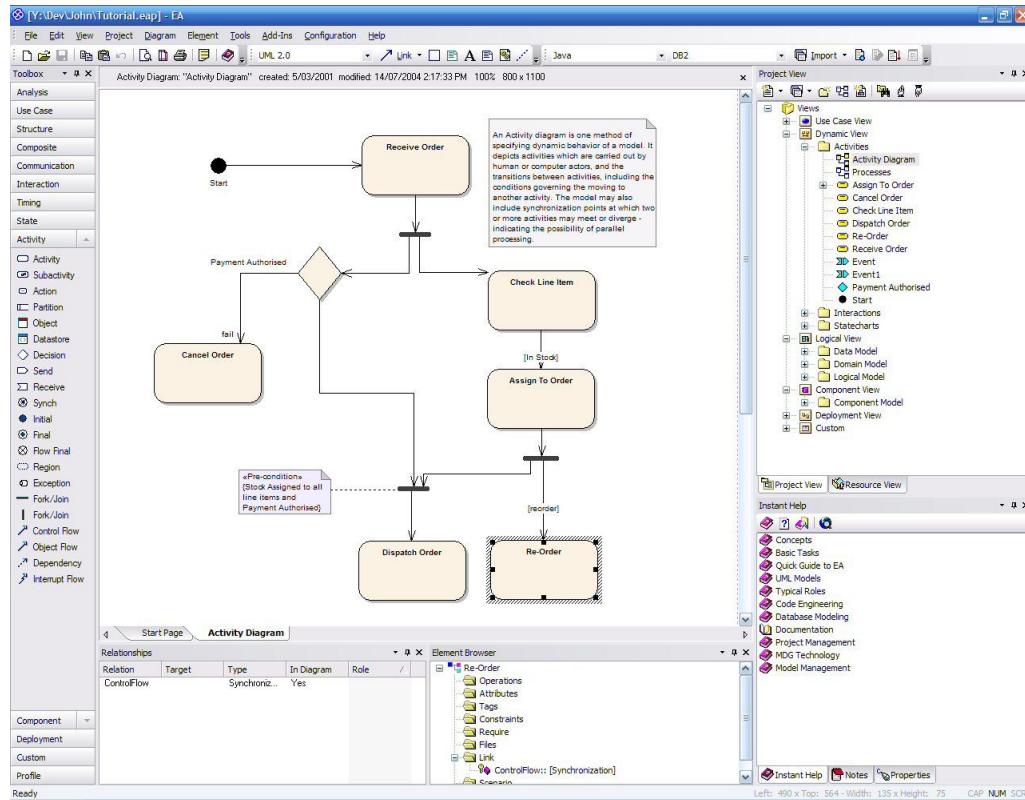
# Use Case Diagram



# Class Diagram



# Activity Diagram

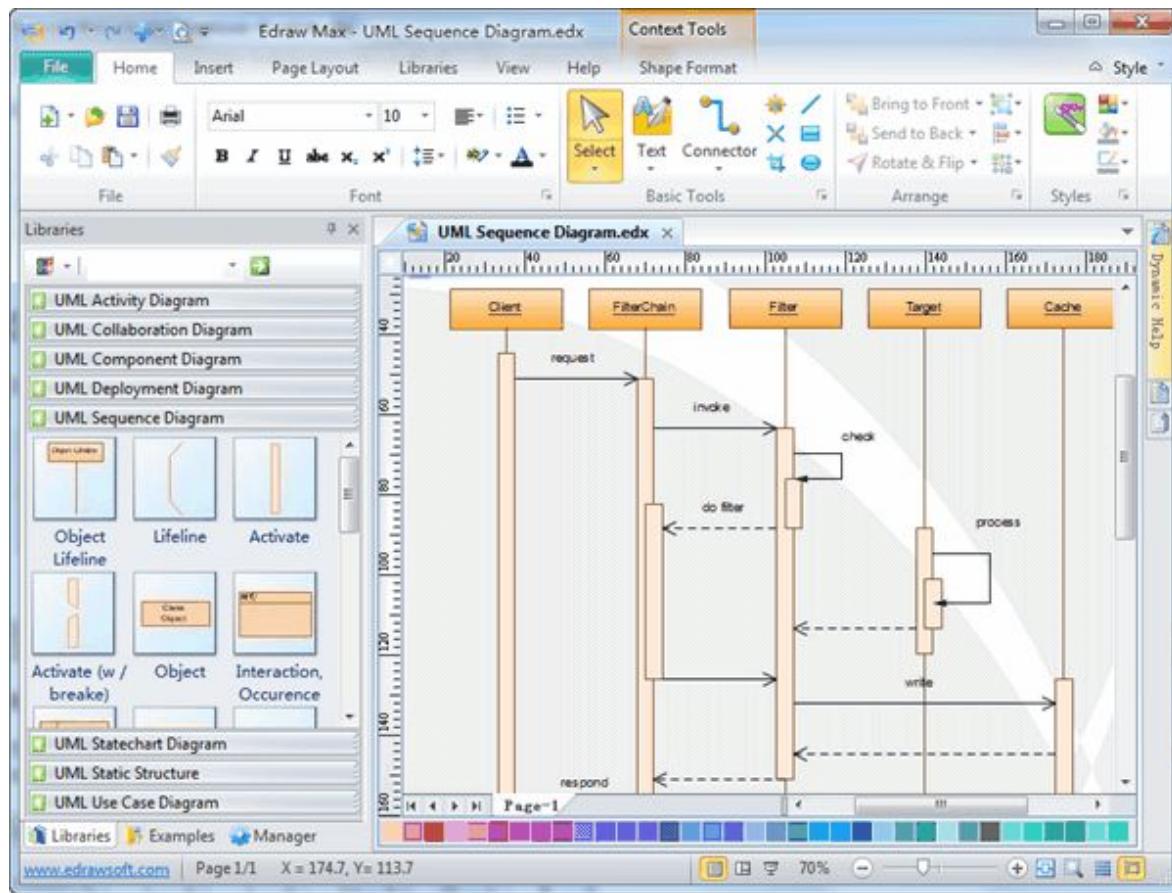


# Edraw (open source)

---

# Edraw (open source)

# Sequence Diagram



## Set 6 - Cosa ricordare: concetti, motivazioni, conseguenze, relazioni fra concetti, ecc.

- A cosa serve modellare, quali le diverse prospettive di modellizzazione
- Tipi di modelli

---
- Modelli del contesto: diagrammi del contesto e modelli dei processi del contesto
- Modelli comportamentali che rappresentano l'elaborazione dei dati. DFD, dati processi, flussi di dati, archivi, entità esterne. Terminologia dei dati e dei processi. Regole per la scomposizione di un processo in sottoprocessi di maggior dettaglio. Cosa non si rappresenta in un DFD. Differenze dai Data Flow Diagram. Esercizi
- Modelli comportamentali che rappresentano l'evoluzione mediante stati. Automi a stati finiti, limitazioni.
- (Cenni): Modelli per la rappresentazione della semantica dei dati. UML con associazioni (e commenti). E-R models (vedi corso DB). Data dictionary.
- Object model: (vedi lez. Dr. Baruzzo): varie tipologie di diagrammi. Ragionamento classificatorio, eredità.

## Set 6 - Cosa ricordare: concetti, motivazioni, conseguenze, relazioni fra concetti, ecc.

- Reti di Petri (RdP) posti-transizioni. Posti, transizioni, funzione di input, funzione di output, terminologia e sintassi, token, marcatura e stato corrente. Rappresentazione distribuita dello stato del sistema, evoluzione rappresentata mediante transizioni che hanno effetto sulle varie parti dello stato (posti). Regole di evoluzione. Condizioni per lo scatto di una transizione. Sincronizzazione di processi concorrenti, sincronizzazione con eventi esterni. Aspetti da non considerare in una RdP: trasformazioni I/O, principi di conservazione. Strutture di controllo rappresentate mediante specifiche semplici configurazioni di RdP.
- Esempi ed esercizi

## Esercizio 2

Si modelli tramite il formalismo delle *Reti di Petri* un'istanza semplificata del sistema descritto nell'esercizio precedente. In tale scenario, la struttura di ricerca dispone di tre ricercatori e di due macchine remote. Il project manager, quando insorgono nuove esigenze, assegna le relative attività ai ricercatori, in particolare ciascuna attività ad un qualsiasi ricercatore libero (non impegnato nello svolgimento di precedenti attività assegnate a lui). Il ricercatore cui è assegnata un'attività accede al pannello di controllo mediante il quale avvia una sessione di elaborazione su una qualsiasi delle due macchine, purché libera (non impegnata in altre sessioni di elaborazione). Alla fine della elaborazione il ricercatore segnala la conclusione della stessa e carica all'interno del sistema *eventuali* risultati, rimanendo quindi libero per ricevere altre assegnazioni dal project manager. (8)

