Il processo di *ingegneria dei requisiti*

- Il processo di **ingegneria dei requisiti** (*requirements engineering*) varia in base al dominio applicativo, alle persone coinvolte ed all'organizzazione che sviluppa il sistema software
- Si può però individuare un insieme di attività generiche comuni a tutti i processi:
  - studio di fattibilità (*feasibility study*)
  - identificazione e analisi dei requisiti (*req. elicitation and analysis*)
  - specifica dei requisiti (*req. specification*)
  - convalida dei requisiti (*req. validation*)
  - gestione dei requisiti (*req. management*)

Studio di fattibilità

- Fase iniziale del processo di ingegneria dei requisiti
- Si basa su una descrizione sommaria del sistema software e delle necessità utente
- Le informazioni necessarie per lo studio di fattibilità vengono raccolte da colloqui con:
  - client manager
  - ingegneri del software con esperienza nello specifico *dominio applicativo*
  - esperti delle tecnologie da utilizzare
  - utenti finali del sistema

Report di fattibilità

- Lo studio di fattibilità produce come risultato un report che stabilisce l'opportunità o meno di procedere allo sviluppo del sistema software
- Domande tipiche dello studio di fattibilità:
  - In che termini il sistema software contribuisce al raggiungimento degli obiettivi strategici del cliente?
  - Può il sistema software essere sviluppato usando le tecnologie correnti e rispettando i vincoli di durata e costo complessivo?
  - Può il sistema software essere integrato con altri sistemi già in uso?

Attività di identificazione e analisi dei requisiti

- Il team di sviluppo incontra il cliente e gli utenti finali al fine di identificare l'insieme dei requisiti utente, dalla cui analisi si generano i requisiti di sistema (specifiche)
- L'identificazione dei requisiti può coinvolgere personale che copre vari ruoli sia all'interno dell'organizzazione del cliente che in altre organizzazioni o tra gli utenti finali
- Il termine *stakeholder* viene usato per identificare tutti coloro che hanno un interesse diretto o indiretto sui requisiti del sistema software da sviluppare

Identificazione e analisi dei requisiti

Task

- ***Comprensione del dominio***: l'analista deve acquisire conoscenze sul dominio applicativo (es. se il sistema software deve supportare il lavoro di un ufficio postale, l'analista deve comprendere il funzionamento di un ufficio postale)
- ***Raccolta dei requisiti***: mediante interazione con gli stakeholder si identificano i requisiti utente
- ***Classificazione***: l'insieme dei requisiti raccolti viene suddiviso in sotto-insiemi coerenti di requisiti
- ***Risoluzione dei conflitti***: eventuali contraddizioni e/o conflitti tra requisiti vanno identificati e risolti
- ***Assegnazione delle priorità***: mediante interazione con gli stakeholder, ad ogni requisito o sotto-insiemi di requisiti va assegnata una classe di priorità
- ***Verifica dei requisiti***: i requisiti vengono controllati per verificarne completezza e consistenza, in accordo a quanto richiesto dagli stakeholder

Identificazione e analisi dei requisiti (2)

- Tecniche di identificazione dei requisiti
  - Ethnography
  - Casi d'uso (basati su *scenari*)
  - Prototipazione
- Tecniche di analisi (e specifica) dei requisiti
  - **semi-formali**, basate su *modelli del sistema* e usate dai metodi di *analisi strutturata* o *analisi orientata agli oggetti*
  - **formali** (basate su Petri Net, FSM, Z, etc.)

Convalida dei requisiti

- La convalida dei requisiti è finalizzata ad accertare se il documento dei requisiti, ottenuto come risultato della fase di analisi, descrive realmente il sistema software che il cliente si aspetta
- La scoperta di errori in questa fase è fondamentale per evitare costosi *rework* in fasi più avanzate del ciclo di vita
- I *controlli* da effettuare includono:
  - validità
  - consistenza
  - completezza
  - realizzabilità
  - verificabilità

Tecniche di convalida dei requisiti

Le *tecniche di convalida* dei requisiti includono:

- revisioni informali
- revisioni formali
  - *walkthrough*
  - *ispezioni*
- prototipazione
- generazione dei test-case
- analisi di consistenza automatizzata (per requisiti formali)

Gestione dei requisiti

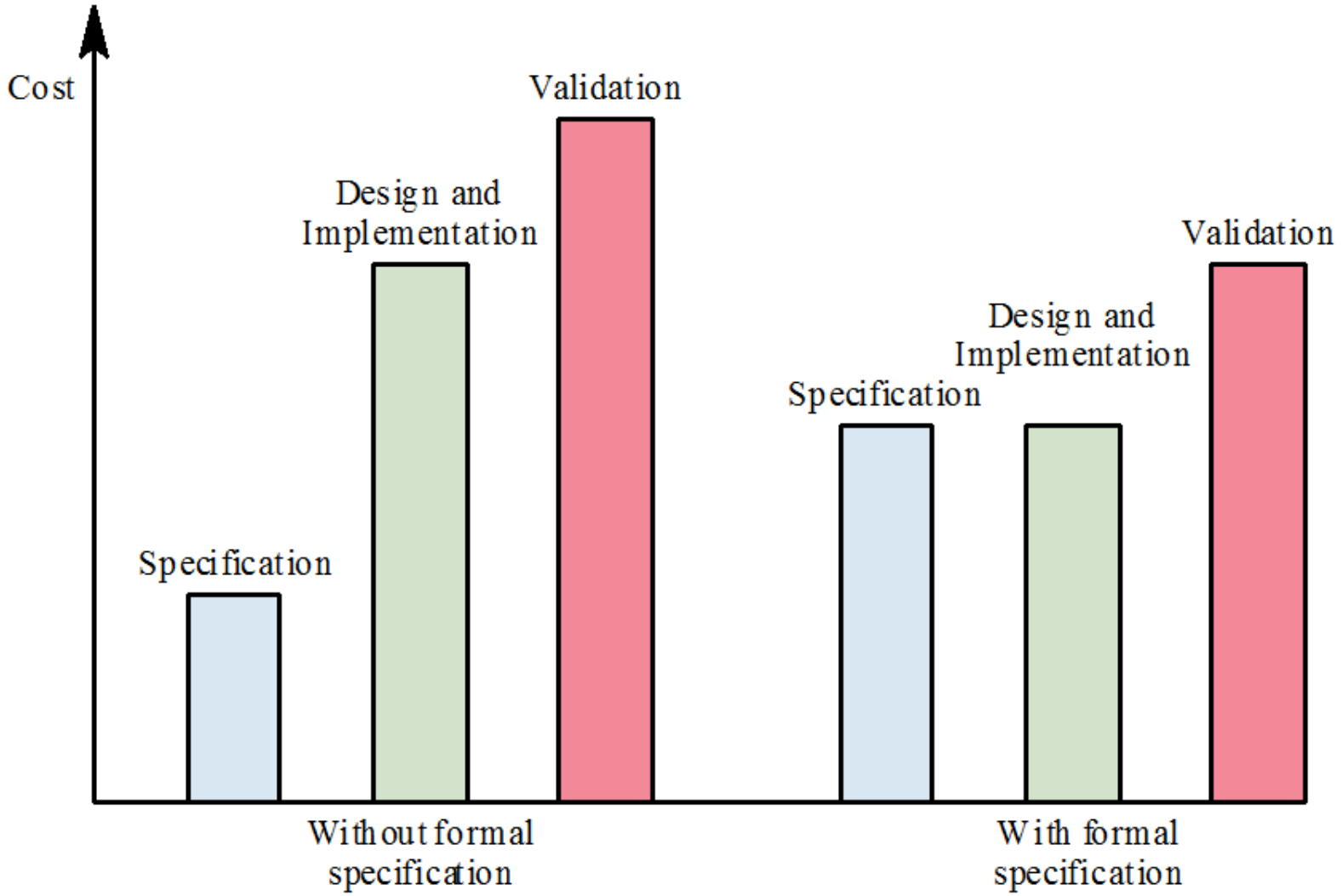
- Processo di identificazione e controllo delle modifiche subite dai requisiti di un sistema software lungo il ciclo di vita
- I requisiti di un sistema software possono essere classificati in termini della loro evoluzione come:
  - requisiti **stabili** (probabilità minima di essere modificati nel tempo)
  - requisiti **volatili** (probabilità elevata di essere modificati nel tempo):
    - *mutabili* (modifiche legate a cambiamenti nell'ambiente operativo)
    - *emergenti* (modifiche causate da una migliore comprensione del sistema software)
    - *consequenziali* (modifiche legate all'introduzione di sistemi informatici nel flusso di lavoro)
    - *di compatibilità* (modifiche legate a cambiamenti nei sistemi e nei processi aziendali)

Gestione delle modifiche di requisiti

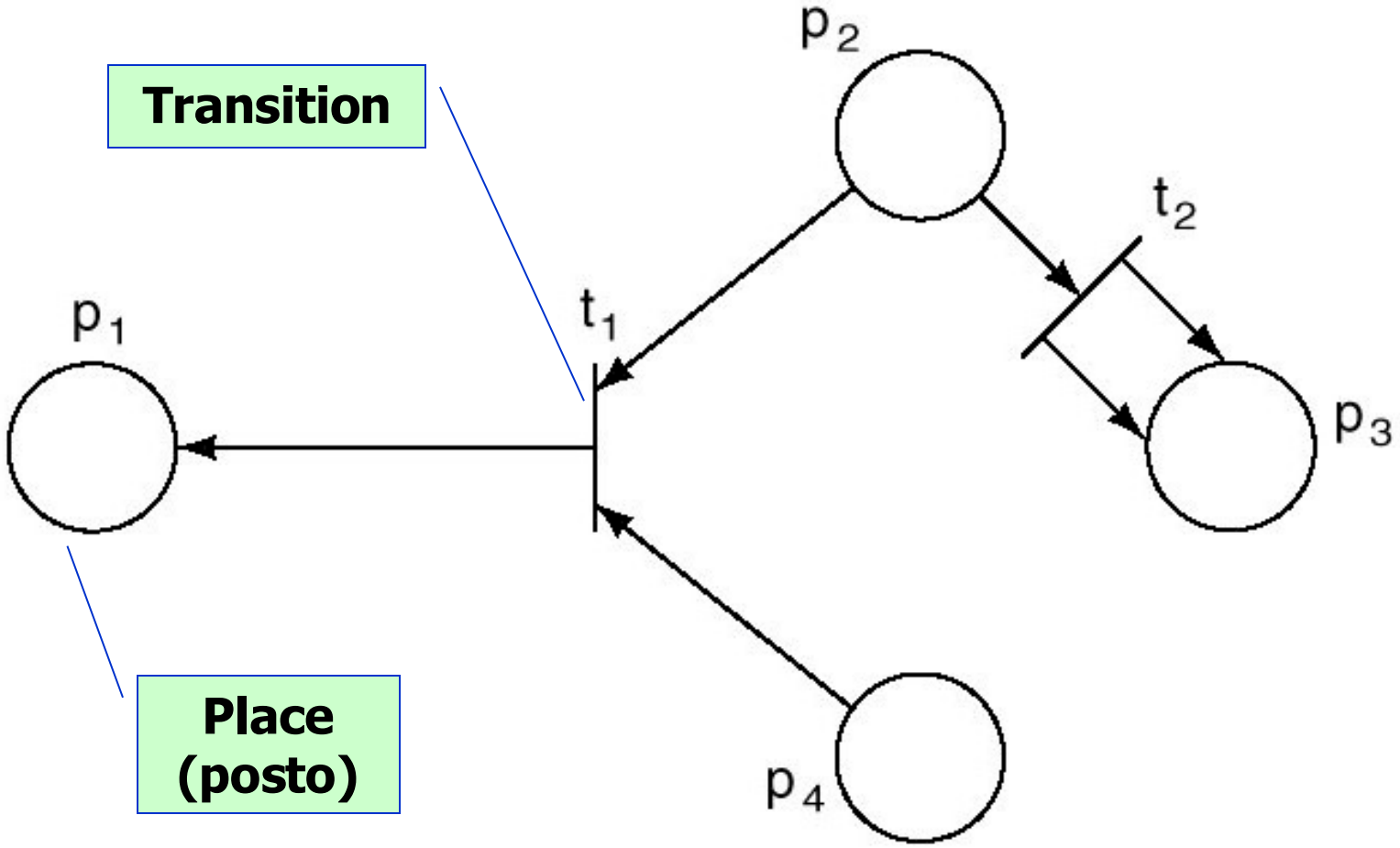
Le modifiche dei requisiti vanno opportunamente pianificate mediante:

- identificazione univoca dei requisiti
- gestione delle modifiche
  - analisi dei costi
  - analisi dell'impatto
  - analisi della realizzazione
- politiche di *tracciabilità* (relazioni tra requisiti e tra requisiti e progetto del sistema software)
- uso di tool CASE per il supporto alle modifiche

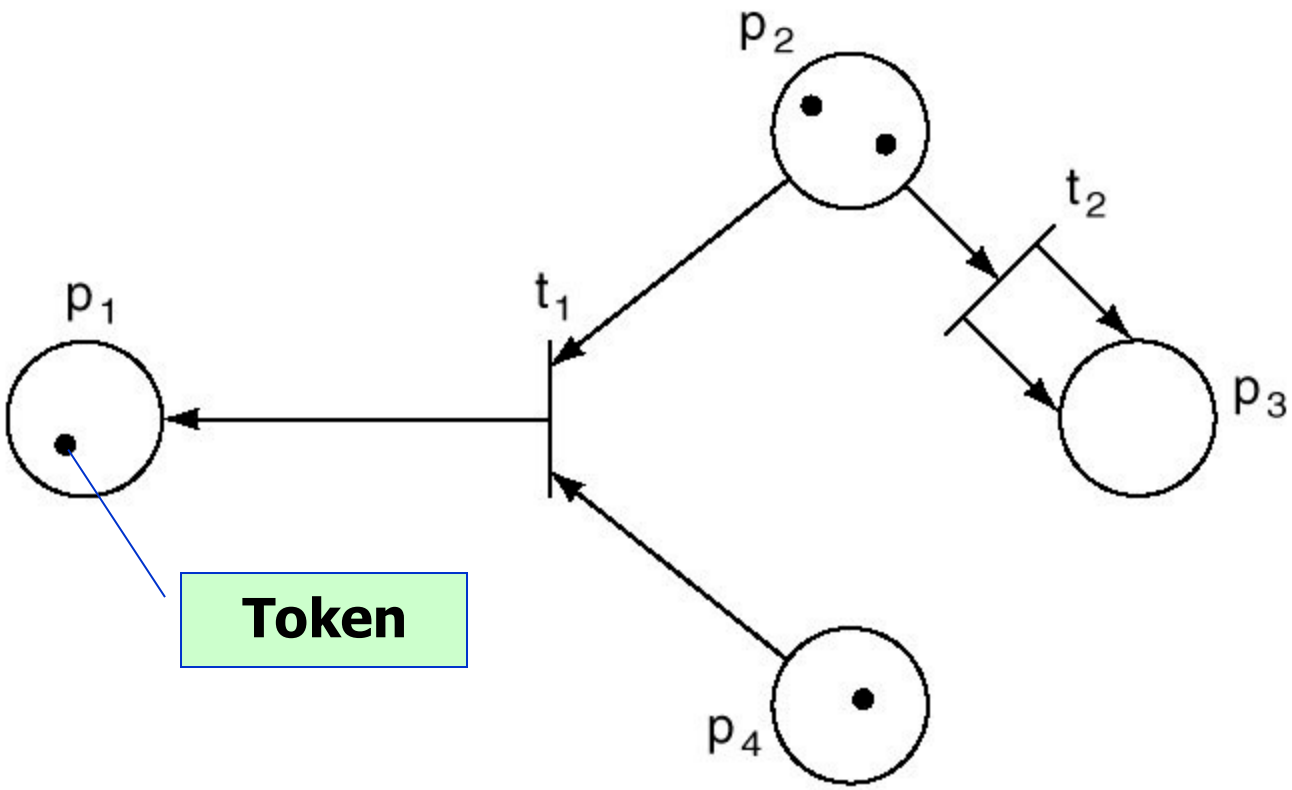
Specifiche formali vs. informali



Specifiche formali con Petri Net

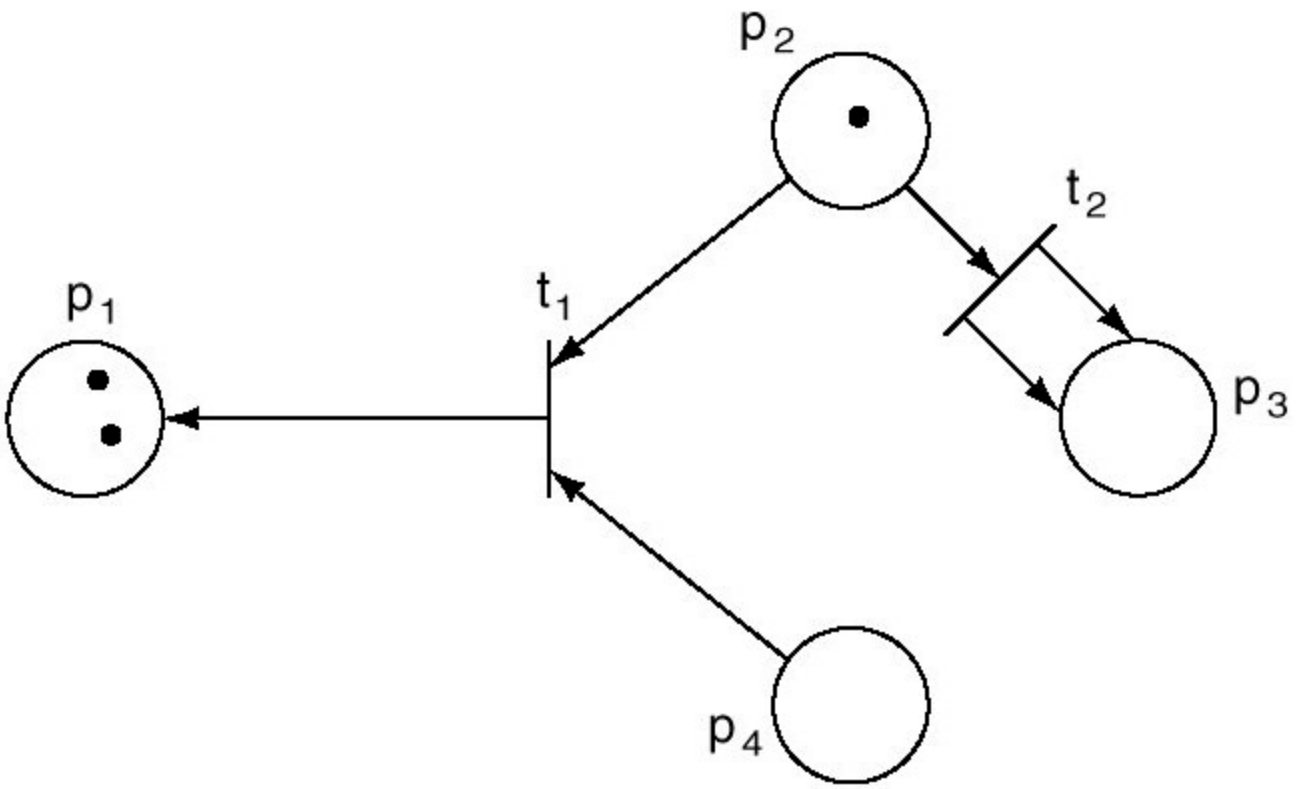


Marked Petri Net (1)



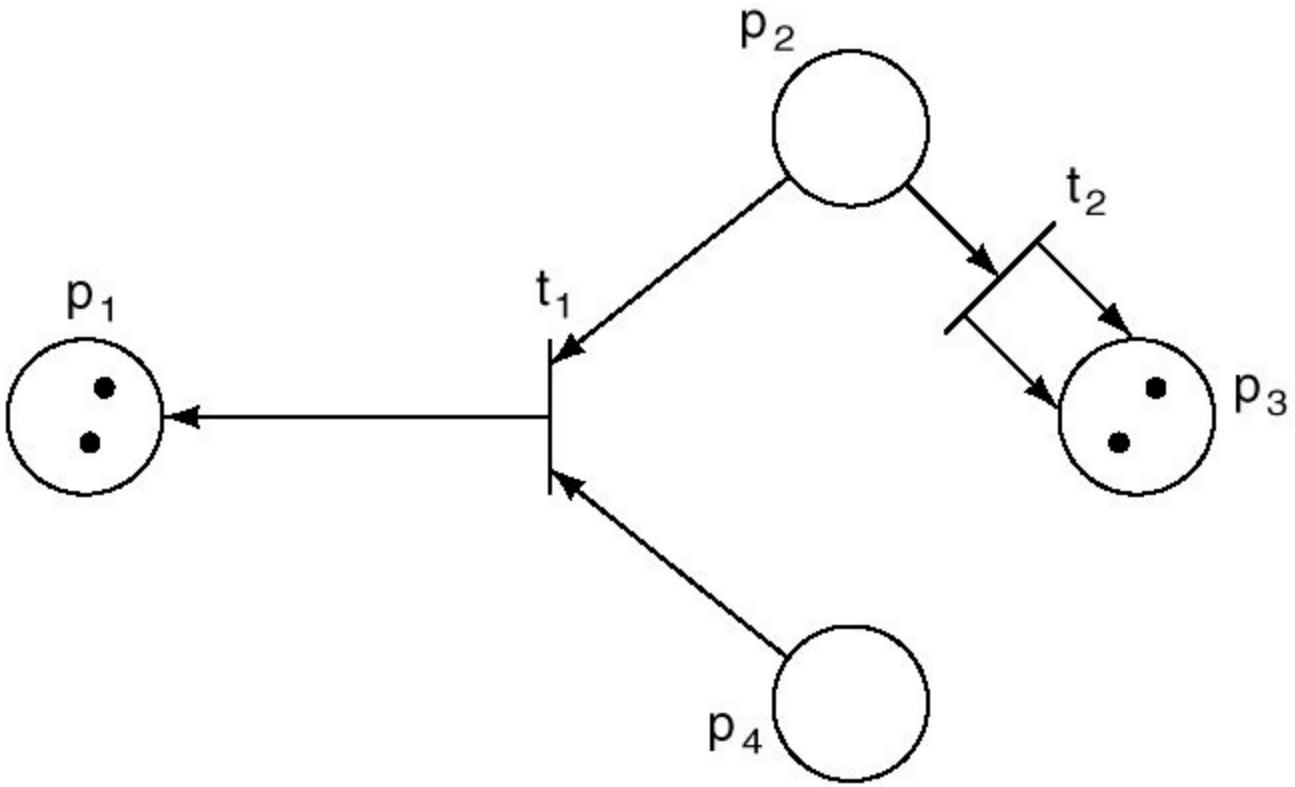
Marked Petri Net (2)

- after firing transition $\mathbf{t_1}$

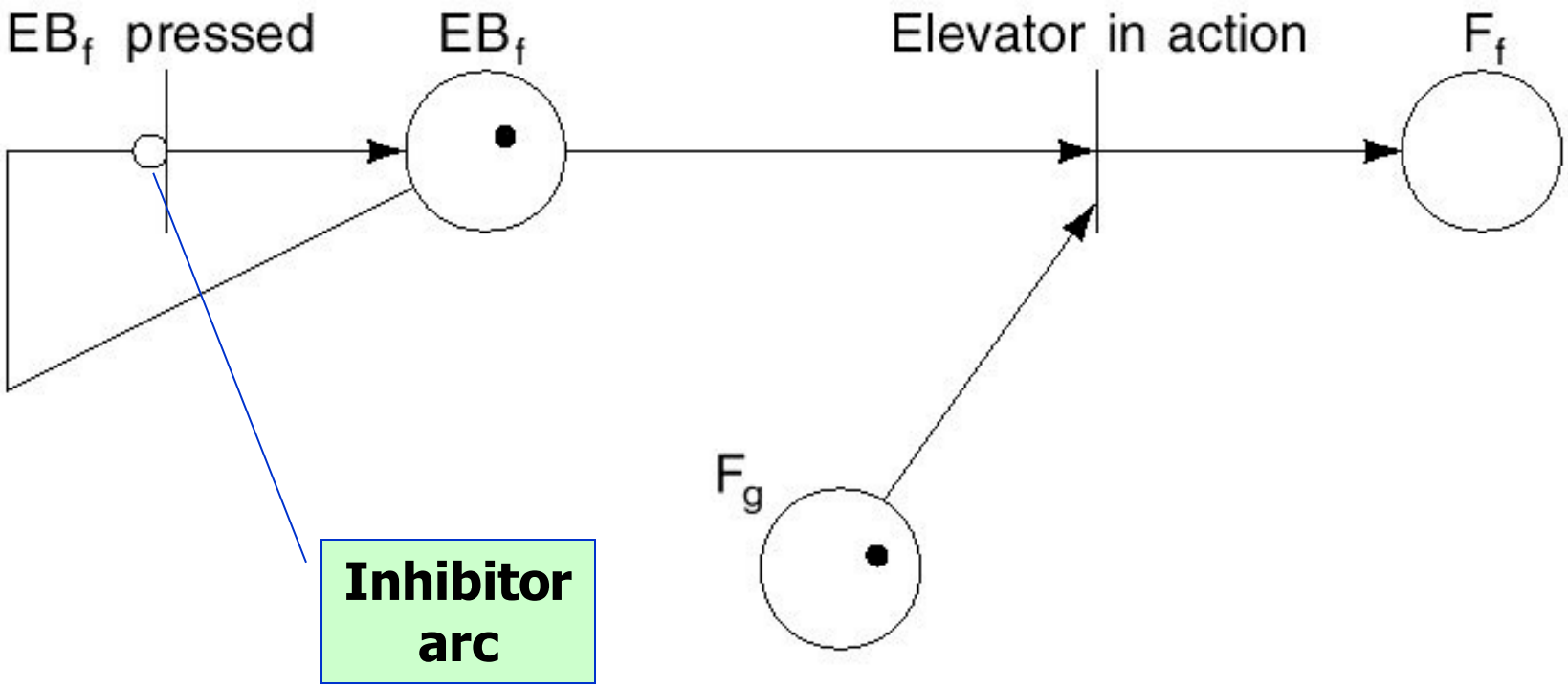


Marked Petri Net (3)

- after firing transition $\mathbf{t_2}$



Petri Net: esempio



Specifiche formali con Finite State Machine (FSM): esempio

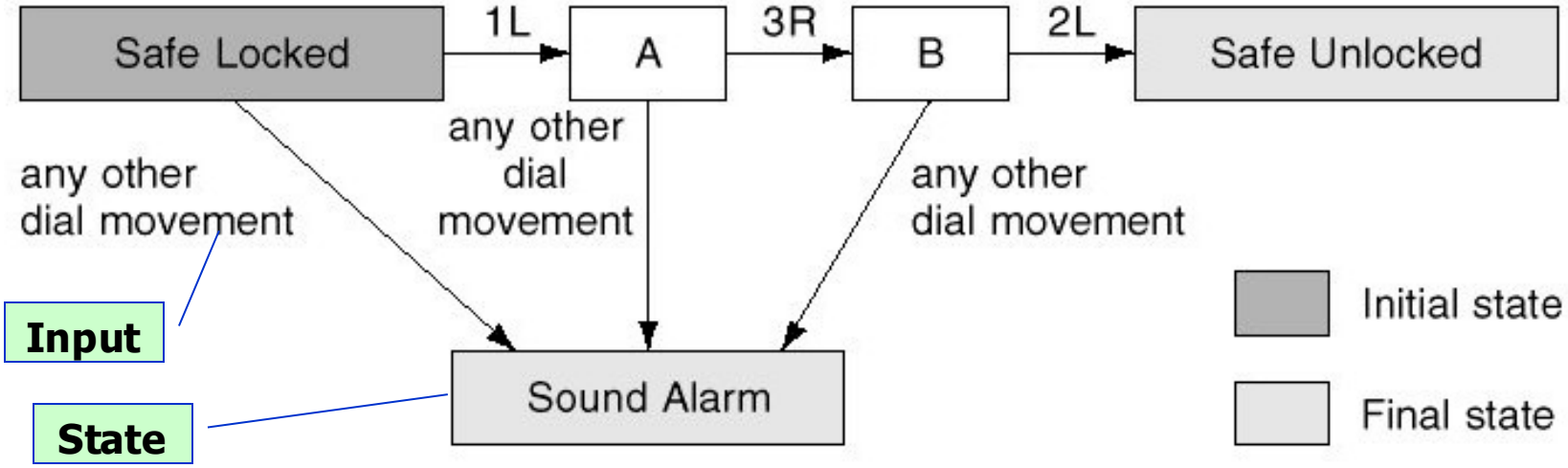
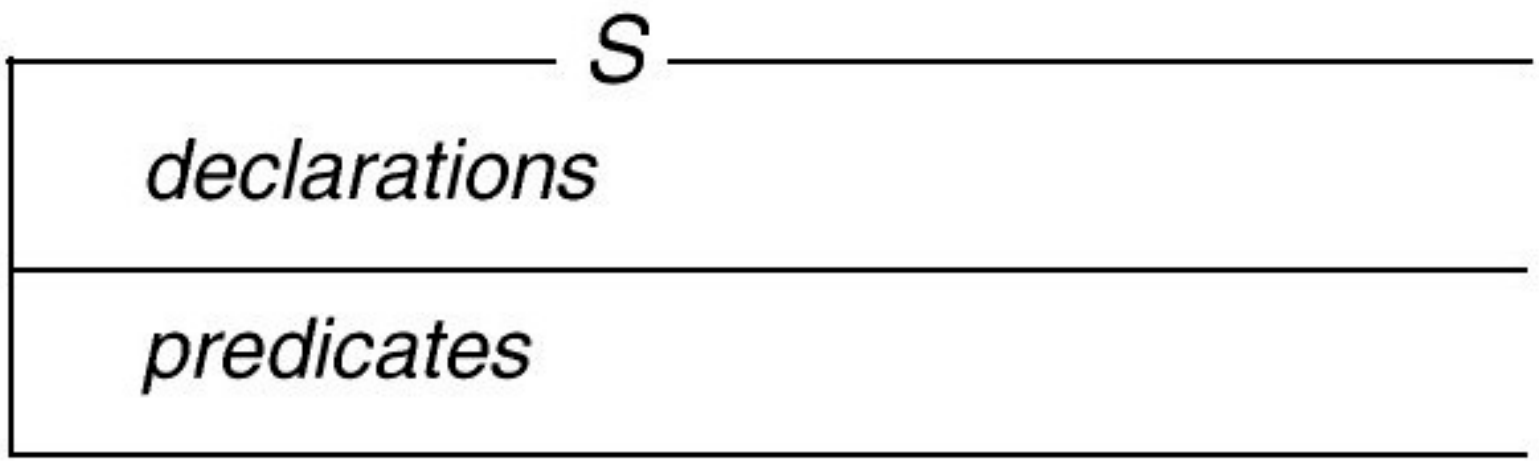


Table of Next States

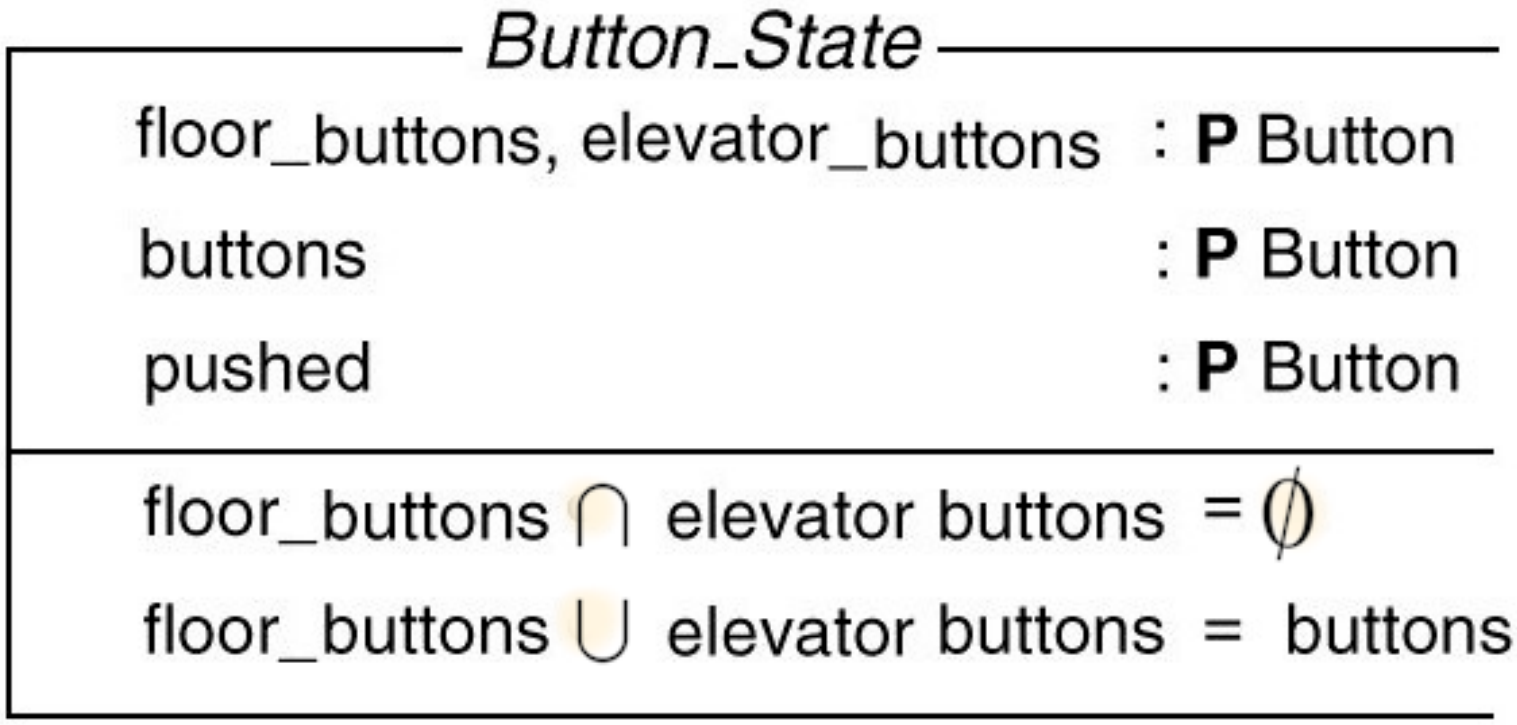
Dial movement \ Current state	Safe locked	A	B
1L	A	Sound Alarm	Sound Alarm
1R	Sound Alarm	Sound Alarm	Sound Alarm
2L	Sound Alarm	Sound Alarm	Safe Unlocked
2R	Sound Alarm	Sound Alarm	Sound Alarm
3L	Sound Alarm	Sound Alarm	Sound Alarm
3R	Sound Alarm	B	Sound Alarm

Specifiche formali con linguaggio Z

- Consiste di un set di schemi
- Ogni schema Z ha il seguente formato:



Linguaggio Z
esempio di specifica di stato



Abstract Initial State

```
Button_init := [Button_State' | pushed' = ∅]
```

Linguaggio Z
esempio di specifica di operazione

Push_Button

Δ *Button_State*

button?: Button

$$(button? \in buttons) \land$$
$$(((button? \notin pushed) \land (pushed' = pushed \cup \{button?\})) \lor$$
$$((button? \in pushed) \land (pushed' = pushed)))$$

Spec. semi-formali: modelli del sistema

- Per **modello del sistema** si intende una rappresentazione ***astratta*** del sistema che facilita la comprensione delle proprietà del sistema e delle sue caratteristiche di funzionamento, prima che il sistema venga costruito
- L'uso di modelli dei sistemi software è formalizzato all'interno di metodi di analisi dei requisiti (*specifica*) del software che fanno uso di *tecniche semi-formali*
- I metodi di analisi dei requisiti software sono di due tipi:
  - metodi di analisi **strutturata** (o **procedurale**)
  - metodi di analisi **orientata agli oggetti**
- Per descrivere completamente un sistema è necessario costruire vari modelli che rappresentino il sistema da vari punti di vista (*informazioni*, *funzioni* e *comportamento dinamico*)

Tipi di modelli del sistema

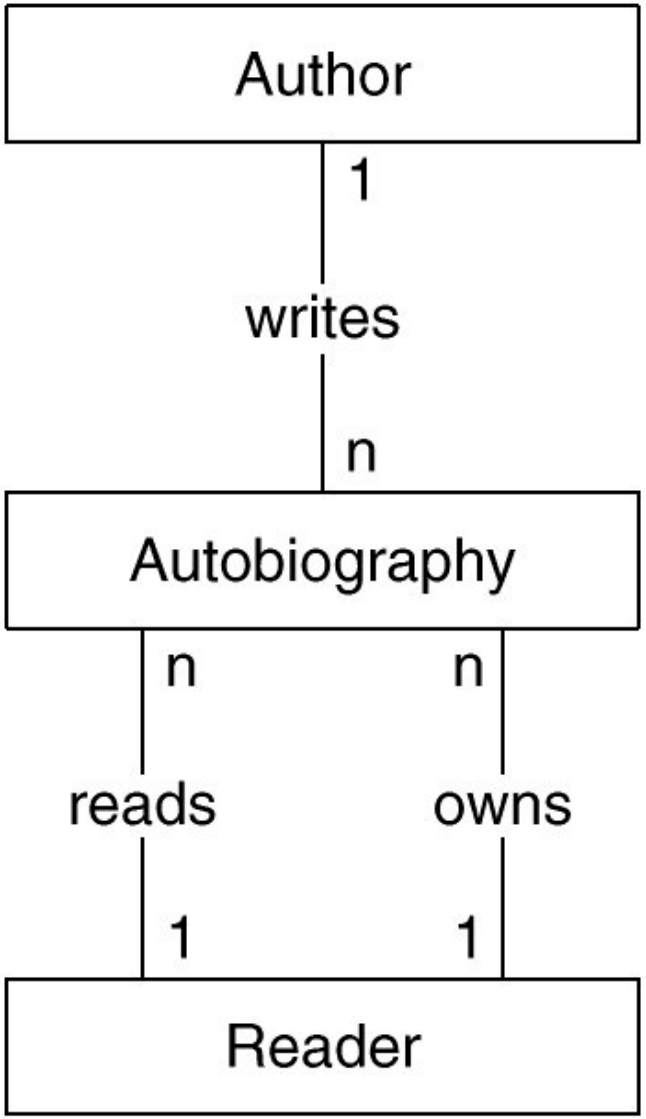
Per descrivere la specifica semi-formale di un sistema software si usano 3 tipi di modelli:

1. **modello dei dati**: rappresenta gli aspetti statici e strutturali relativi ai dati (*data requirements*)
   - *ERD (not UML)*
   - *class diagram (UML)*
2. **modello comportamentale**: rappresenta gli aspetti funzionali del sistema (*functional requirements*)
   - *data flow diagram (not UML)*
   - *use case diagram (UML)*
   - *activity diagram (UML)*
   - *interaction diagram (UML)*
3. **modello dinamico**: rappresenta gli aspetti di "controllo“ e di come le funzioni del modello comportamentale modificano i dati introdotti nel modello dei dati
   - *state diagram (UML)*

Entity Relationship Diagram (ERD)

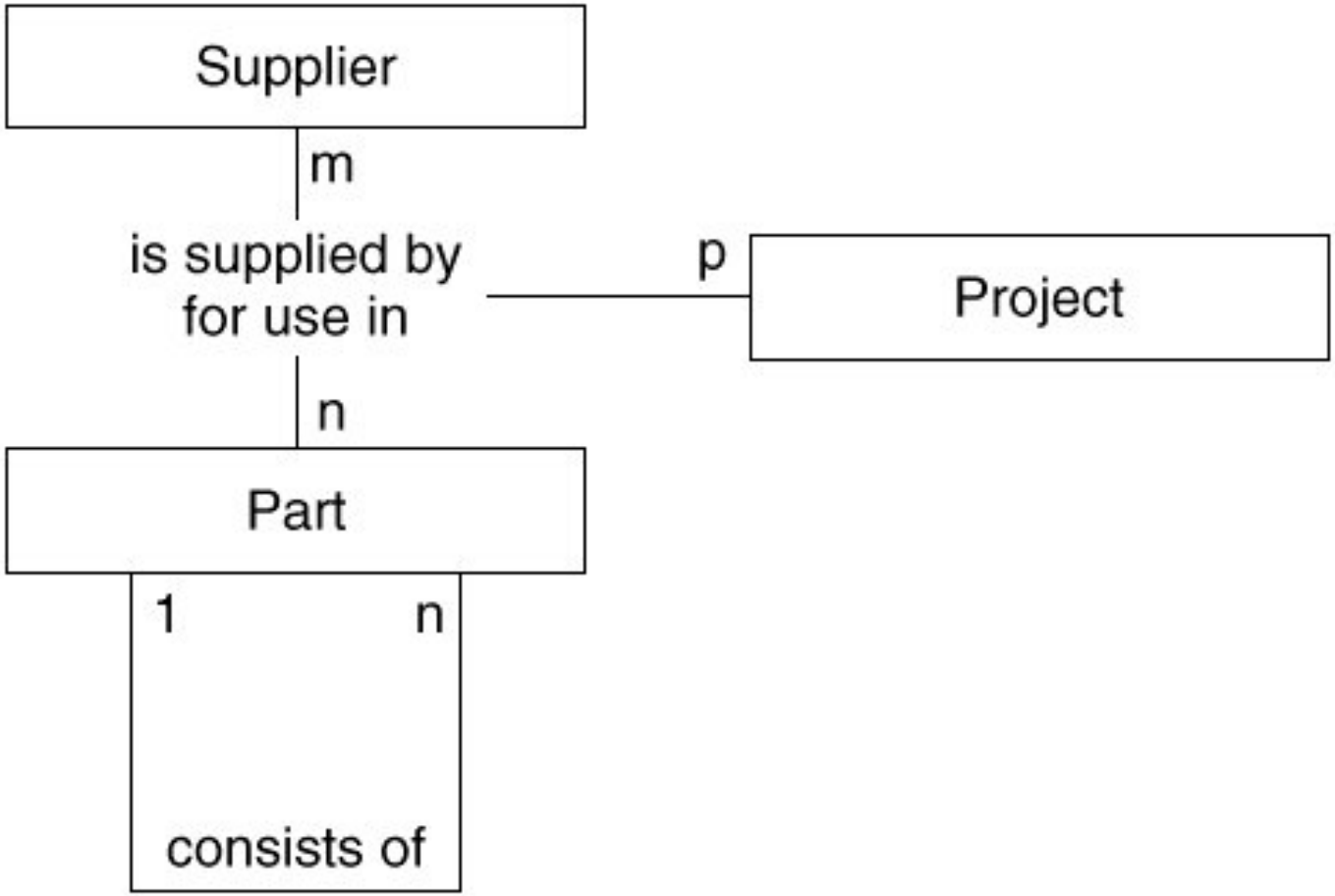
relazioni

uno-a-molti



ERD

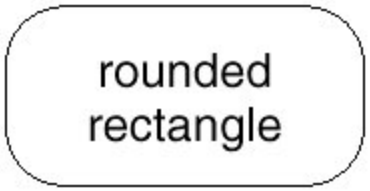
relazioni molti-a-molti



Data Flow Diagram (DFD)

Source or destination of data

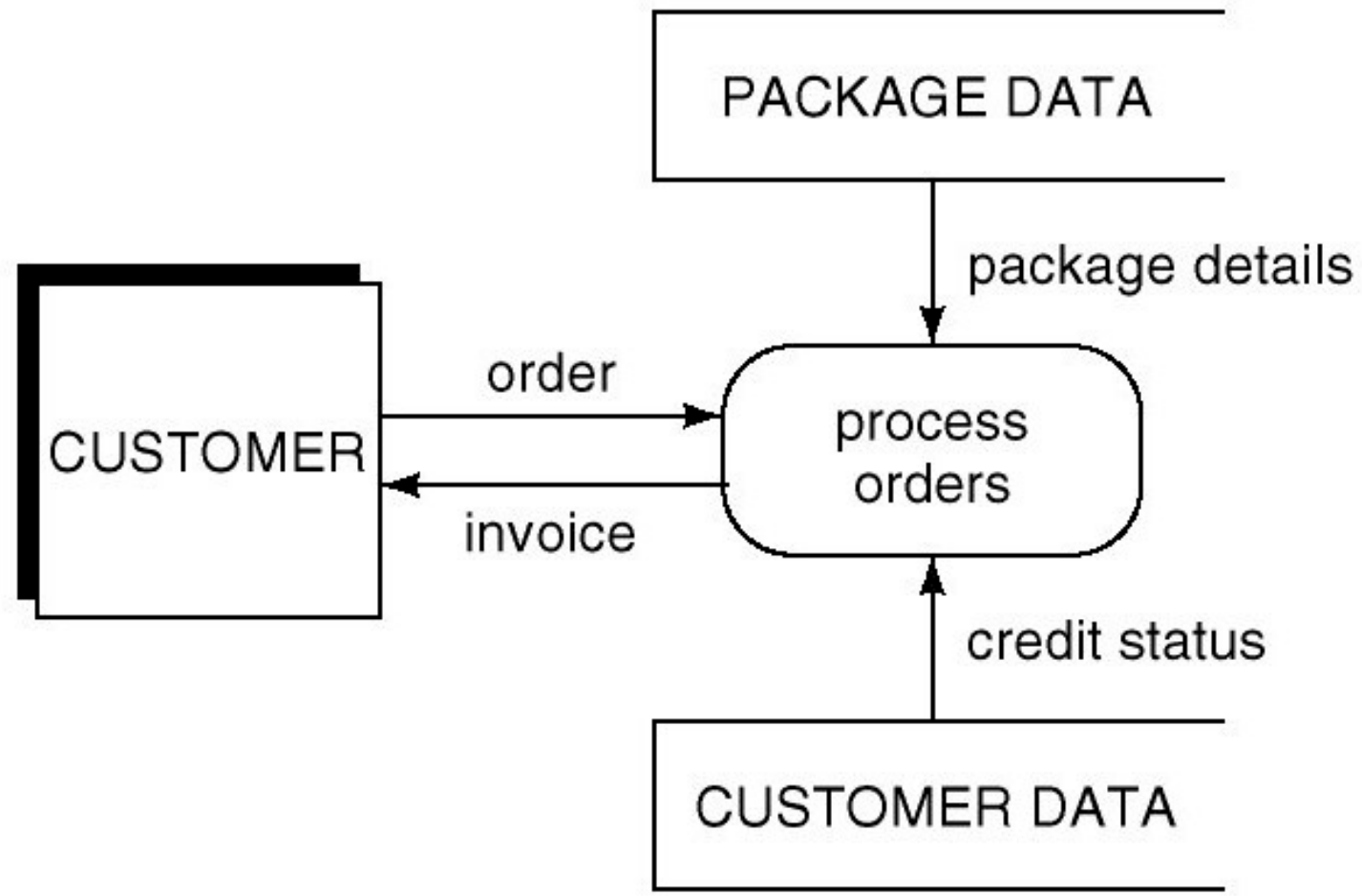
Flow of data



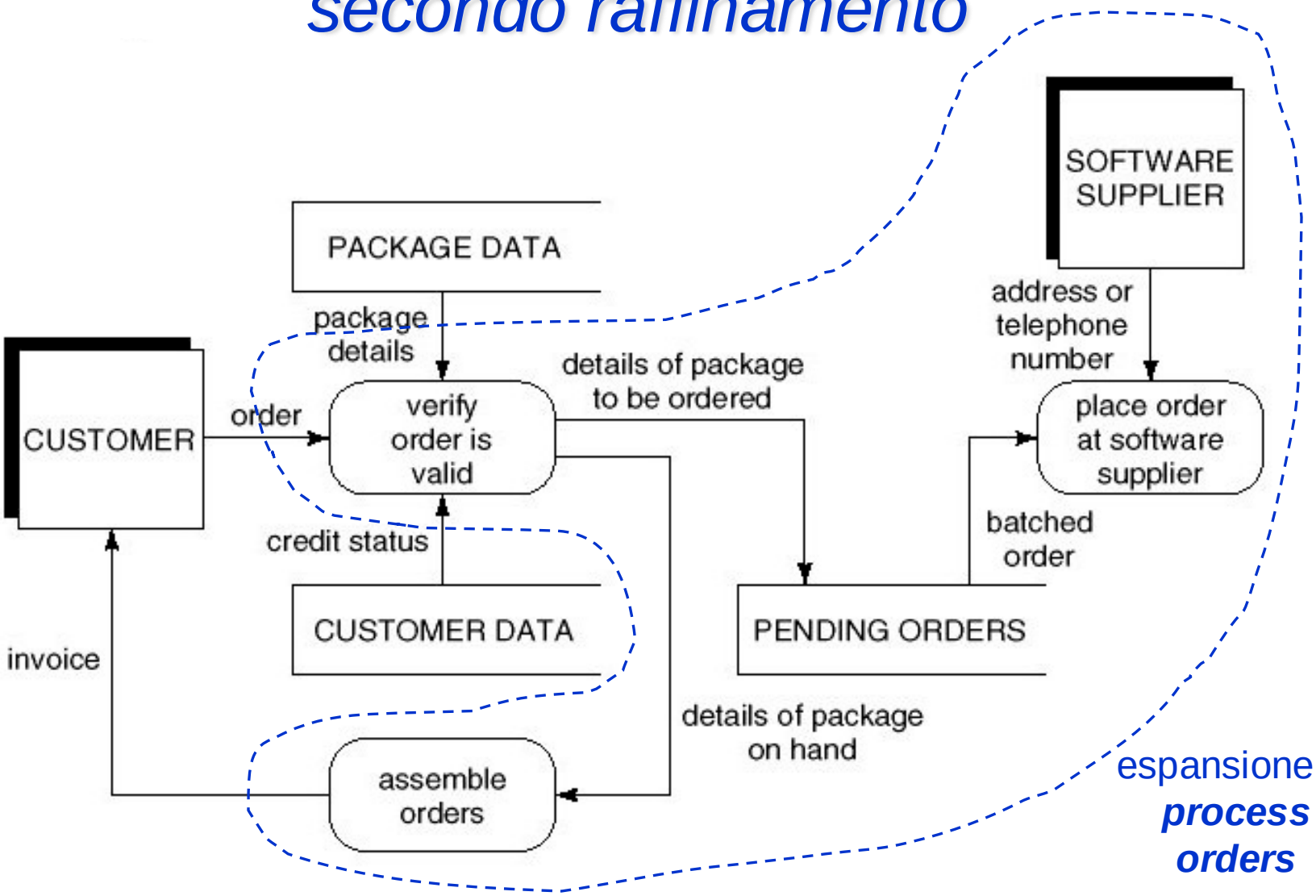
Process which transforms a flow of data

Store of data

Esempio di DFD
primo raffinamento



Esempio di DFD
secondo raffinamento



Structured System Analysis (SSA)
(metodo di analisi strutturata)

- Metodo introdotto da Gane and Sarson (1979)
- E' costituito da *9 step*
- Basato sul concetto di step-wise refinement
- Altri metodi di analisi strutturata:
  - DeMarco (1978)
  - Yourdon and Constantine (1979)

SSA – Step 1
Draw the DFD

- Use the requirements document (or the prototype) to:
  - Identify data flows
  - Identify source and destinations of data (where data flows starts and ends, respectively)
  - Identify processes that transform data
- Refine the DFD by adding new flows of data or by adding details to existing data flows

SSA – Step 2

Decide what Sections to Computerize and How

- Use cost-benefits analysis to decide which sections of the DFD to automate
- Decide how to computerize:
  - Batch operations
  - On-line processing
- Example:
  - Automate *order placement* in batch
  - Automate *order validation* online
- The next 3 steps are the stepwise refinement of data flows, processes and data stores

SSA – Step 3

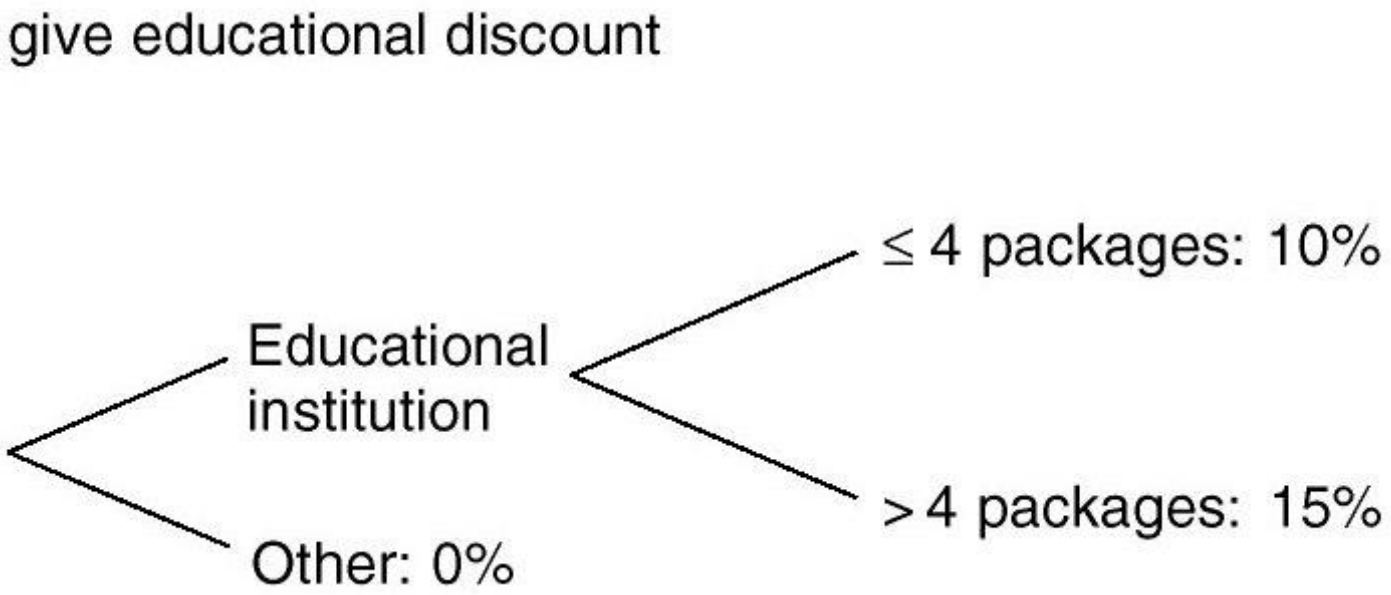
Determine the details of the data flows

- Decide what data items must go into the various data flows
- Example: the data flow *order* can be refined as follows:
  - *order_identification*
  - *customer_details*
  - *package_details*
- Then, refine each flow stepwise:
  - *order_identification* is a 12-digit integer
  - *customer_details* consists of *customer_name*, *customer_address*, etc.

SSA – Step 4

Define the logic of processes

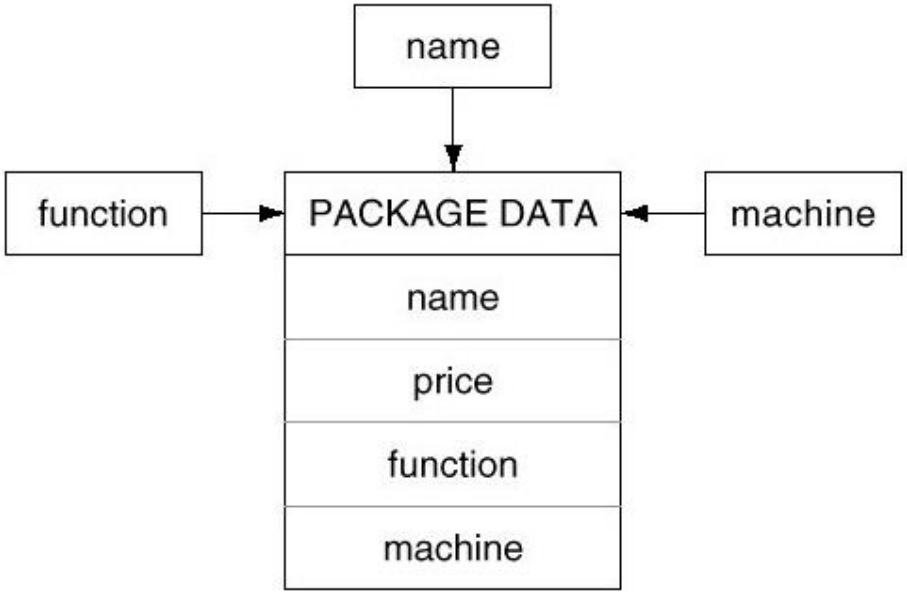
- Example: build the decision tree for a *give_educational_discount* process



SSA – Step 5

Determine the data stores

- Define the exact contents of each store and its representation (specific format in a given programming language)
- Define the level of access by use of *data-immediate-access diagram* (*DIAD*)
- Example:



SSA – Step 6

Define the physical resource

- Examples:
  - For each file, specify: file name, organization (sequential, indexed, etc.), storage medium, records, down to the field level
  - If a DBMS is to be used, then the relevant information for each table is specified

SSA – Step 7

Determine the Input/Output specifications

- The input forms must be specified (components and layout)
- The output screens must similarly be determined
- The printed output also must be specified (estimated length and details)

SSA – Step 8

Determine the sizing

- Compute:
  - volume of input (daily or hourly)
  - frequency of each printed report and its deadline
  - size and number of records that are to pass between the CPU and mass storage
  - size of each file

SSA – Step 9

Determine the hardware requirements

- From sizing information specified at step 8, determine:
  - Mass storage requirements
  - Mass storage requirements for backup
  - Characteristics of user terminals
  - Charateristics of output devices
  - Adequacy of existing hardware
  - Costs of hardware to be purchased

SSA Output

- Step 9 is the last step of SSA
- After approval by the client, the resulting specification document is handed to the design team, and the software process continues
- Drawbacks:
  - SSA cannot be used to determine response times
  - CPU size and timing cannot be determined with any degree of accuracy