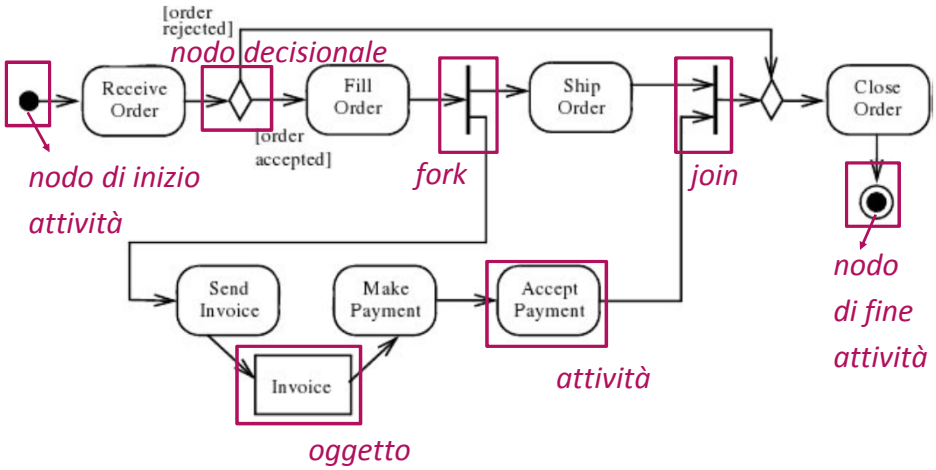
Laboratorio di
Ingegneria del Software
a.a. 2013-2014

LEZIONE 7 - STATE MACHINE DIAGRAM

Catia Trubiani
Gran Sasso Science Institute (GSSI), L'Aquila
catia.trubiani@gssi.infn.it

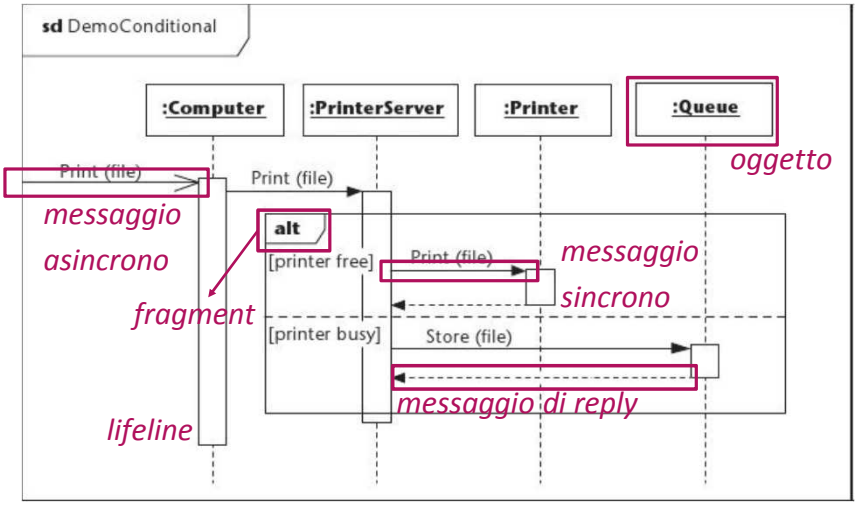
Riepilogo della lezione precedente

Cos'è un **Activity Diagram**? A cosa serve?!



Riepilogo della lezione precedente

Cos'è un **Sequence Diagram**? A cosa serve?!



Riepilogo della lezione precedente

Quali sono i **fragments** che possono essere definiti in un Sequence Diagram?! Qual'è la loro semantica??

alt: alternative

opt: opzionali

par: parallele

loop: ciclo

critical: sezione critica

neg: tracce invalide

Un po' di storia su state machines...

State Machine Diagram deriva dalle state charts proposte da David Harel intorno al 1980, e riadattate al mondo Object Oriented

Implementano anche alcuni aspetti delle macchine di Moore e Mealy, ovvero:

- macchina a stati e transizioni
- enfatizzano il flusso di controllo da stato a stato

Quando e perchè?

Vengono usate per modellare l'aspetto dinamico del comportamento di un sistema o parti di esso, in particolare:

- modellano il comportamento di oggetti reattivi
- modellano il comportamento di oggetti "stateful"
- sono anche usati per modellare il comportamenti di attori, use-case o metodi

Rappresentazione grafica

Graficamente sono rappresentate da una specie di grafo: nodi e archi

Concettualmente aggiungono alla nozione di grafo un po' di semantica, in particolare:

nodo: stato, attività, azioni

arco: transizioni, eventi, invocazione di operazioni

Concetto di -stato-

Lo **stato** rappresenta una situazione (nella vita di un oggetto) durante la quale delle condizioni vengono soddisfatte e delle azioni o attività possono essere eseguite

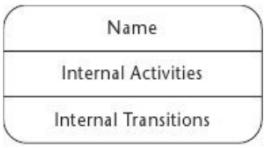
Può essere semplice oppure composto

Esempio: un aereo può trovarsi nei seguenti 5 stati:

» On, Off, TakingOff, Landing, Flying

Come rappresentare uno stato

Graficamente uno **stato** è rappresentato da un rettangolo con gli angoli smussati



Uno stato è composto da 3+1 parti

- Nome: stringa di testo
- Attività interne: azioni (interne) eseguite in risposta ad eventi o ad invocazioni di operazioni user-defined/standard
- Transizioni interne: cambiamenti di stato interni che avvengono al soddisfacimento di una condizione
- Decomposizione: solo per state machine gerarchiche

Attività interne

Sono caratterizzare dai seguenti concetti:

```
event-name args-list '/' action-expression
```

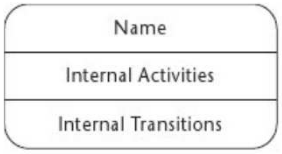
entry: attività eseguita nel momento in cui si entra in uno stato

exit: attività eseguita nel momento in cui si esce da uno stato

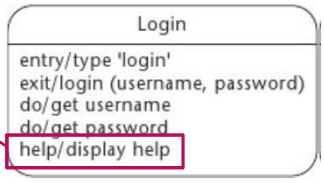
do: identifica un'attività in esecuzione mentre l'oggetto è nello stato della transizione

include: invocazione ad una submachine; l'azione contiene il nome della submachine invocata

Esempio di stato ed azioni



event/action attività definita dall'utente



Alcuni stati un po' speciali (1/3)

Lo **stato iniziale** modella il punto di inizio per l'esecuzione di una state machine, è uno stato un po' speciale dato che:

- è unico per ogni state machine
- non può avere transizioni entranti
- non ha associate attività interne
- non è possibile per un sistema permanere nello stato iniziale
- a dirla tutta è chiamato *Pseudostate* nella specifica di «UML», non è sottoclasse di *State*

Alcuni stati un po' speciali (2/3)

Lo **stato finale** modella il punto di fine per l'esecuzione di una state machine, è uno stato un po' speciale dato che:

- una state machine può avere più stati finali
- non può avere transazioni uscenti
- può avere associate azioni all'entrata nello stato
- a differenza dello stato iniziale, lo stato finale è uno stato a tutti gli effetti, nella specifica «UML» è sottoclasse di *State*

Alcuni stati un po' speciali (3/3)

Gli **stati composti** sono stati un po' speciali dato che:

- stato che ha un insieme di "sotto-stato"
- due categorie:
  - sotto-stati sequenziali
  - due o più sotto-stati concorrenti (regioni)

Gli stati composti includono stati iniziali e finali:

- ogni transizione entrante nello stato composto è una transizione nel sotto-stato iniziale
- ogni transizione entrante in un sotto-stato finale di un sotto-stato rappresenta il completamento dell'attività nello stato

Sotto-stati sequenziali

Spesso sono riferiti come *or-substate*

I sotto-stati composti possono presentarsi solo in modo sequenziale, lo stato composto può trovarsi in un unico sotto-stato per volta

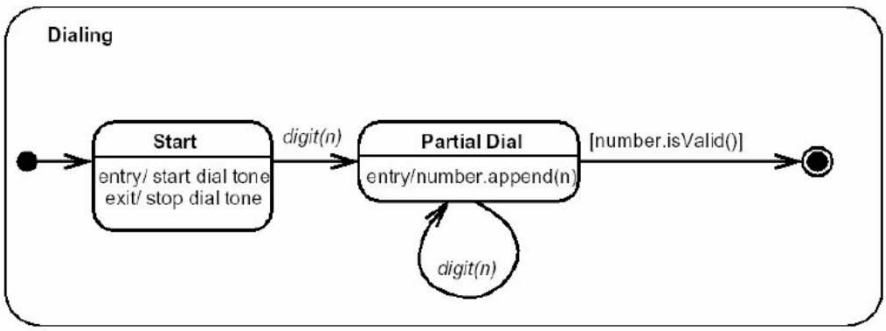
Vengono utilizzati quando uno stato complesso è composto da una serie di sotto-stati *mutuamente esclusivi* tra loro

Un esempio di sotto-stati sequenziali

Si vuole modellare il comportamento di un client VoIP, in particolare da specifica il client può essere in vari stati: Idle, Dialing, Offline, Call

Quando il client si trova nello stato Dialing viene avviata la procedura di composizione del numero sulla rete. Viene quindi atteso il segnale di inizio digitazione e vengono fornite le cifre del numero che si vuole comporre

Un esempio di sotto-stati sequenziali



N.B. l'esempio è solo illustrativo.. non abbiamo ancora spiegato bene come sono caratterizzate le transizioni

Sotto-stati concorrenti

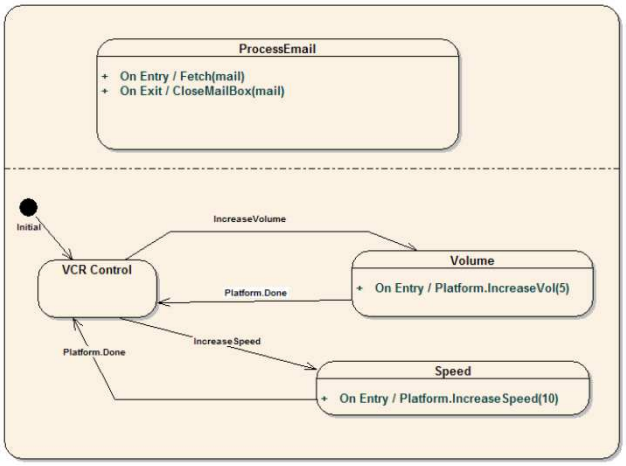
Spesso sono riferiti come *and-substate*

Un insieme di stati identifica la configurazione attuale dello stato composto; lo stato composto può trovarsi contemporaneamente in più sotto-stati

In pratica è costituito da un insieme di regioni concorrenti, ed ogni regione viene specificata una macchina a stati

Il completamento delle attività nello stato composto è dato dal compimento delle attività in tutte le regioni concorrenti

Un esempio di sotto-stati concorrenti



Esempio mail-reader tratto da “An Introduction to SCXML” (J. Barnett) @ http://www.voicexml.org/

Sub-machine

Sono semanticamente equivalenti agli stati composti

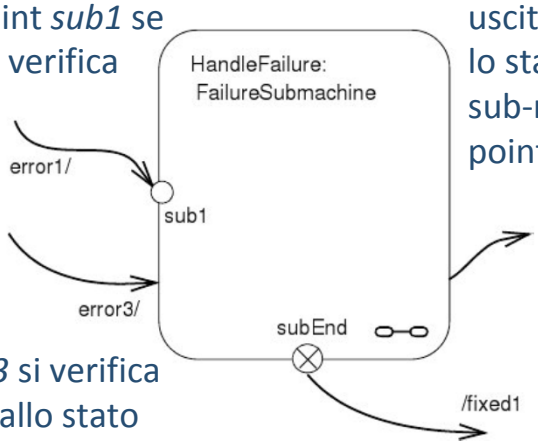
Si entra in una state machine tramite un entry point

Si esce da una state machine tramite un exit point

Una sub-machine può essere acceduta tramite un entry point o tramite il suo stato iniziale (che rappresenta l’entry point di default)

Un esempio di una sub-machine

entry point *sub1* se *error1* si verifica

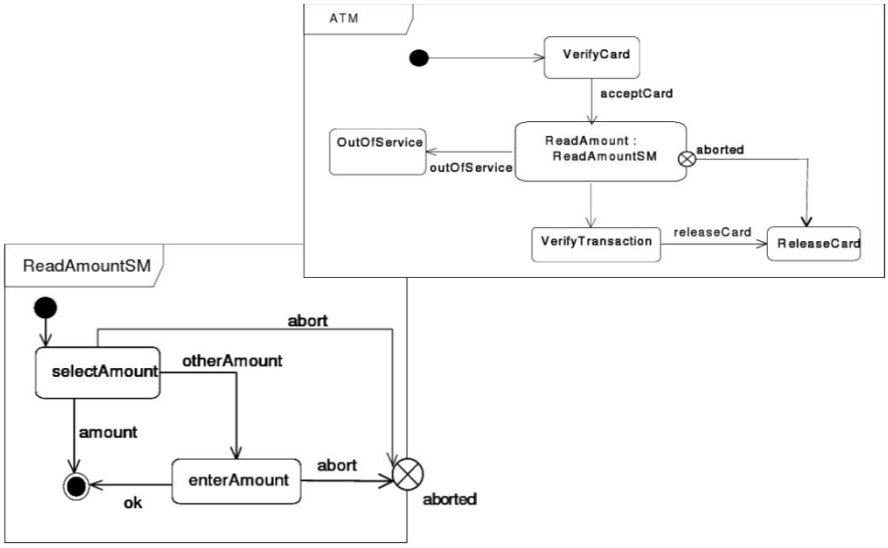


uscita attraverso lo stato finale della sub-machine (exit point)

se *error3* si verifica accesso allo stato iniziale della sub-machine (entry point)

subEnd ed esecuzione di fixed1

Un esempio di sub-machine



Il concetto di -transizione-

Una transizione modella la relazione tra due stati

Indica un cambiamento di stato da parte di un oggetto al verificarsi di un certo evento e di una certa condizione

Graficamente è una freccia con un'etichetta che esce dallo stato di partenza ed entra nello stato di destinazione a seguito della transizione

Rappresentazione delle transizioni

Una transizione è una 5-tupla composta da:

1- stato sorgente

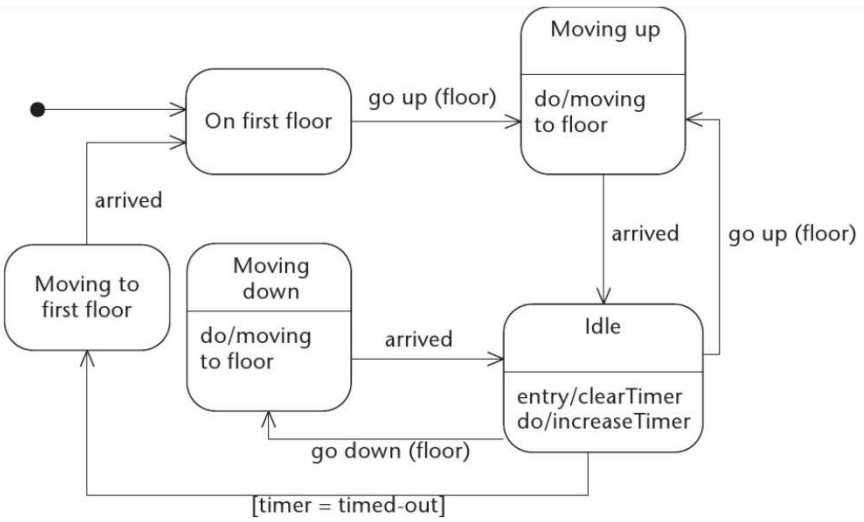
2- evento scatenante

3- condizione di guardia

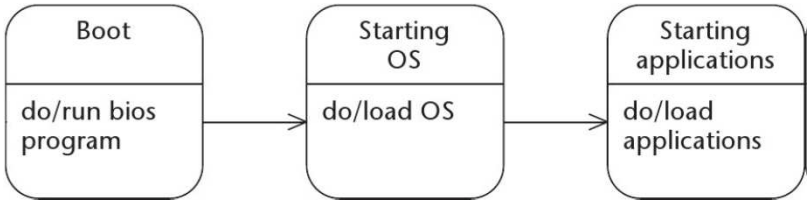
4- effetto: azione || generatore di eventi

5- stato destinatario

Un esempio di transizioni



Caso particolare per le transizioni



Quando nelle transizioni di stato non sono specificati eventi, una transizione occorre quando sono state eseguite le attività interne allo stato di partenza

Comportamento di state machines

Il comportamento delle classi può essere specificato:

- come un algoritmo
- esplicitamente con una state machine
- in entrambi i modi

Difficoltà di mapping tra state machines e O.O.

- rappresentazione e gestione dei segnali/eventi
- i modelli sono ambigui senza un trasformatore associato

Una possibile soluzione

Le transizioni di stato di una classe avvengono solo attraverso le operazioni pubbliche esportate dalla classe

Una State Machine SM può implementare il comportamento di una classe C se ogni transizione di SM corrisponde ad una operazione pubblica di C e viceversa

Le classi che devono ricevere/gestire un segnale/evento devono esportare una operazione corrispondente che riceve il segnale come argomento

Si possono implementare eventi e segnali come classi (con propri attributi ed operazioni)

Un esempio di state machine

Si vuole modellare il comportamento di un orologio digitale. L'orologio:

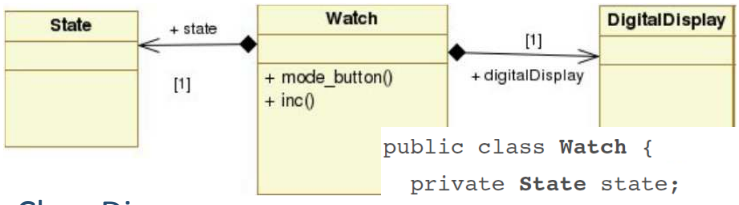
- si compone di una unità dedicata al controllo del display

- ha diverse modalità di funzionamento (e.g. regolazione ora, regolazione minuti, display orario)

- permette di scegliere la modalità di funzionamento

- permette di ricevere speciali tipi di input (e.g. tick per incremento dell'ora)

Un esempio di modellazione statica



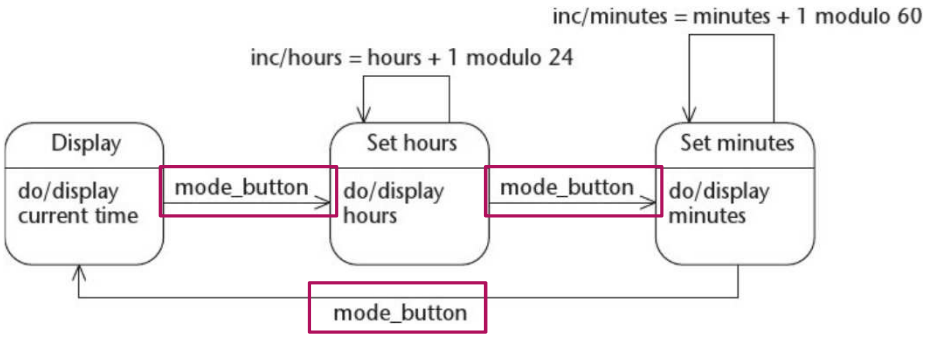
Class Diagram

e codice java che

lo rappresenta

```
public class Watch {
  private State state;
  private DigitalDisplay LCD;
  public Watch (){
    state = new State();
    LCD = new DigitalDisplay();
    state.value = State.Display;
    LCD.display_time();
  }
  public void mode_button(){ ... }
  public void inc(){ ... }
}
```

Un esempio di modellazione dinamica



State Machine Diagram e codice java che lo rappresenta

```
public void mode_button(){
  switch (state.value){
    case … … :
      … …
    break;
    … …
    case … … :
      … …
    break;
  }
```

```
public void inc() {
  switch (state.value){
    case … … :
      … …
    break;
    … …
    case … … :
      … …
    break;
    }
  }
```


Map in Java: mode_button

```
public void mode_button(){
  switch (state.value){
    case State.Display:
       LCD.display_time();
      state.value=State.Set_hours;
    break;
    case State.Set_Hours:
      LCD.display_hours();
      state.value=State.Set_minutes;
    break;
    case State.Set_minutes:
      LCD.display_minutes();
      state.value = State.Display;
    break;
  }
}
```

State machine diagram… proviamoci!

Modellare mediante uno state machine diagram il comportamento di un distibutore automatico.

In particolare il focus è sui seguenti stati:

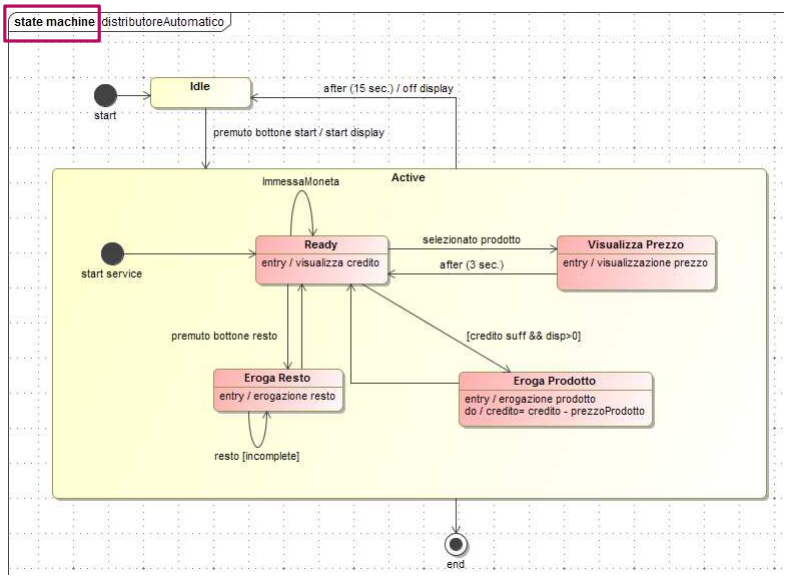
- Ready: il distributore è in attesa della selezione del prodotto da parte dell'utente

- Visualizza prezzo: il distributore visualizza il prezzo del prodotto

- Eroga resto: il distributore eroga il resto

- Eroga prodotto: il distributore eroga il prodotto

Una possibile soluzione



Questions?

catia.trubiani@gssi.infn.it