Modellazione discreta per eventi

Simulazione & Logistica, I modulo
Lezione n. 4

Corso di Laurea in Informatica Applicata
Università di Pisa, sede di La Spezia
A.a. 2008/09, I semestre

Contenuti

- Modellare per simulare
- Architettura di un simulatore
- Modellazione a eventi
- Simulatori a eventi
- Esempio: l'impiegato sempre tormentato (per eventi)

Dove siamo arrivati

- Cicli di attività: modelli lontani dalla specifica
- UML: specifica concreta di
  - entità e insiemi, come classi e oggetti
  - stati, delle entità attive
  - eventi, per definire le transizioni di stato
    - identificati
    - con guardie, definite sugli attributi degli oggetti
    - con azioni, definite da metodi degli oggetti
- Domande in sospeso (se vogliamo costruire un sim)
  - Chi gestisce lo scorrere del tempo?
  - Chi fa scattare gli eventi?
  - Sappiamo anche che vorremmo definire priorità fra eventi e ordinare l'esecuzione dei metodi handle()

Obiettivo della modellazione

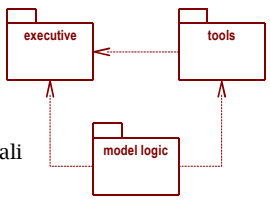
- Modellare la realtà per simularla
  - Cioè per costruirne un simulatore
  - Economico nella realizzazione ed efficiente nell'esecuzione
  - Corretto rispetto al modello, ma anche con qualità proprie del sw: verificabile, manutenibile, ...
- La modellazione è finalizzata
  - Ciò che è "proprio come nella realtà" ...
  - ... non sempre conviene riportarlo nel simulatore
- Conviene utilizzare tutto quanto si conosce
  - La modellazione è guidata
  - Metodi ispirati dall'architettura del simulatore
  - Quando si ha un martello in mano tutto sembra un chiodo

Modellare per simulare

- Sapere il "come" dei simulatori
  - Perché siamo informatici, prima di tutto
  - Per non fare assunzioni irrealizzabili o costose
  - Per sapere su cosa si può contare
  - Per arrivare a modelli convenientemente implementabili
- Metodi classici di modellazione
  - Basati su l'architettura del simulatore (sempre sequenziale)
  - Architetture diverse, caratteristiche diverse, metodi diversi
  - Modellazione per eventi (classica e con GS DSLibs)
  - Modellazione per attività
  - Metodo delle tre fasi
  - Modellazione per processi

Generica architettura di un sim

- Motore
  - Generale
  - Gestione della sequenza e del tempo
  - Gestione dell'agenda dei cambiamenti di stato
- Logica del modello
  - Particolare
  - Realizzata dal modellatore
- Strumenti
  - Gestione di ingressi & uscite
  - Generazione di dati pseudocasuali
  - Strumenti di registrazione dati
  - ...



Dentro il motore

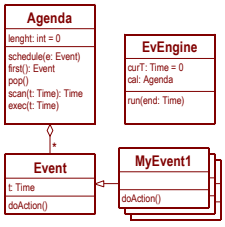
- Tempo
  - In una simulazione gli eventi avvengono al tempo corretto
  - Anche quando la simulazione non è in tempo reale
  - Anche quando gli eventi sono casuali
  - Un buon motivo per evitare parallelismo nel motore
- Meccanismi di agenda, previsti da tutti i motori
- Agenda
  - Programmare i cambiamenti di stato
  - Decidere quando è il momento di produrli
  - Preservando la sequenza
  - Preservando il tempo (reale o meno)
  - Preservando la contemporaneità (con esec. sequenziale)

Modellazione per eventi

- Metodo in uso fin dagli anni '60
  - Proposto nel '63 (Markowitz et al.)
  - Adottato nelle prime versioni di SIMSCRIPT
  - Largamente adottato fino agli anni '80
- Simulatori efficienti e semplici
  - Modelli compatti
  - Codice, in generale, non molto manutenibile (!)
- Caratteristiche
  - Motore sequenziale
  - Eventi: cambiamenti di stato a dati istanti di tempo
  - Azione: tutto ciò che succede in seguito a un evento

Motore a eventi, strutture OO

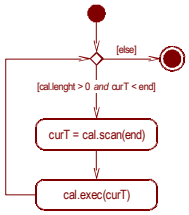
- Un agenda
  - Per tenere traccia degli eventi
  - Inserimento
  - Ricerca di eventi da far scattare
  - Esecuzione delle azioni
- Specializzazione
  - Definizione dei tipi di eventi
  - Come classi da definire nella modellazione del sistema
  - Specializzazione dell'azione
  - Eventuale specializzazione di dati associati all'evento



Motore a eventi, esecuzione

- Ricerca degli eventi
  - Il tempo corrente avanza al tempo dei primi eventi in agenda purché antecedenti alla fine della simulazione
- Esecuzione delle azioni
  - Tutti gli eventi al tempo corrente scattano e quindi sono rimossi
- Ciclo infinito
  - Ci sono eventi in agenda
  - Non è stato raggiunto il tempo di fine simulazione

e: EvEngine



Metodi scan(), exec() e run()

- L'agenda è una coda ordinata sul tempo (e non solo)

```
Time Agenda::scan(Time end) {
  if (lenght > 0) return min(end, first().t);
  else return end;
}
void Agenda::exec(Time curT) {
  while (lenght > 0 && first().t == curT) {
    first().doAction();
    pop();
  }
}
void EvEngine::run(Time end) {
  while (curT < end && cal.lenght > 0) {
    curT = cal.scan(end);
    cal.exec(curT);
  }
}
```


Problemi di manutenzione

- Caratteristiche del motore a eventi
  - Efficiente, semplice ed estremamamente compatto
  - Però il codice della logica del modello è poco manutenibile
  - Perché?
- Un passo indietro
  - Cicli di attività: diagramma degli stati del sistema
  - Linguaggi di programmazione non orientati agli oggetti
- Il metodo doAction()
  - Produce i cambiamenti di stato per tutto il sistema
  - Il codice "tocca" tutte le entità
  - Cambiare un'entità implica modificare codice condiviso
  - Difficoltà di intervento e possibilità di errori

Riferimenti

- M. Pidd,
  Computer Simulation in Management Science, Cap. 6,
  J. Wiley & Sons, 1998
- G. Gallo, *Note di Simulazione*, cap. 2
- GS_SDSLibs 2.00b, *http://genesim.sourceforge.net*