Modellazione discreta per attività

Simulazione & Logistica, I modulo
Lezione n. 5

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

Contenuti

- Modellazione per attività
- Simulatori per attività
- Il metodo delle tre fasi
- Un motore a tre fasi
- Esempio: l'impiegato sempre tormentato (per attività)

Obiettivo della modellazione

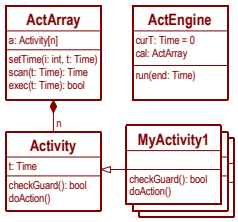
- Modellazione finalizzata
  - Modellare la realtà per simularla
  - Costruire simulatori corretti ed efficienti
  - Con costi di sviluppo possibilmente bassi
  - Producendo codice verificabile, manutenibile, ...
  - Sfruttando la conoscenza dell'architettura del simulatore
- Modellazione per eventi (classica)
  - Modelli compatti e simulatori efficienti (coesione)
  - Critica storica: codice poco manutenibile
- Metodi per attività: ridurre la coesione (frammentare)
- Perché i metodi per attività ci interessano?

Modellazione per attività

- Metodo in uso fin dagli anni '60
  - Proposto nel '62 (Buxton & Laski, prima di Markovitz!)
  - Adottato nelle prime versioni di CSL (Esso & IBM)
  - Antenato del metodo delle tre fasi (con cui è a volte confuso)
- Semplificare la logica del modello
  - Codice meno efficiente, ma più manutenibile
  - Relativamente agli strumenti di sviluppo di quel tempo
- Caratteristiche
  - Motore sequenziale
  - Attività semplici, "atomiche", (~ eventi × condizioni)
  - Gli eventi sono gli istanti a cui le attività avvengono

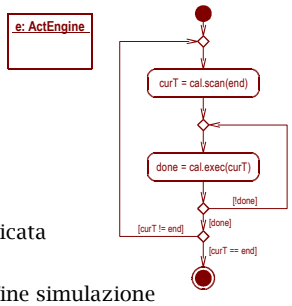
Motore ad attività, strutture dati

- Un vettore di attività
  - Per tutte le *n* attività del sistema
  - Definizione del tempo delle attività
  - Avanzamento del tempo
  - Esecuzione delle attività abilitate, con traccia dell'esecuzione
- Specializzazione
  - Specifica delle attività
  - Specializzazione di guardie e azioni
  - Eventuali dati associati all'attività
  - Una classe per ogni elemento del vettore



Motore ad attività, esecuzione

- Avanzamento del tempo
  - Il tempo corrente avanza al tempo della prima attività in calendario antecedente alla fine della simulazione
- Esecuzione delle azioni
  - Sono eseguite tutte le attività al tempo corrente con guardie verificate
  - Finché nessuna attività è verificata
- Terminazione
  - È stato raggiunto il tempo di fine simulazione



Metodi scan() e run()

- L'agenda è un array, l'ordinamento è arbitrario
  - ```
    Time ActArray::scan(Time end) {
      Time t = end;
      for (int i=0; i<n; i++) {
        t = min(t, a[i].time)
      }
      return t;
    }
    ```
  - ```
    void EvEngine::run(Time end) {
      while (curT < end) {
        done = false;
        while (!done) {
          done = cal.exec(curT);
        }
      }
    }
    ```


Metodo exec()

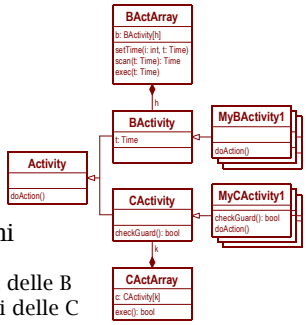
- Effetti di un'azione
  - Abilitare altre attività
  - Riprogrammandone il tempo (nel futuro, di solito)
  - Rendendone verificate le guardie (al tempo corrente)
- Esecuzione delle azioni associate alle attività
  - ```
    bool ActArray::exec(curT: Time) {
      bool r = true;
      for (int i=0; i<n; i++) {
        if (a[i].time == curT && a[i].checkGuard()) {
          a[i].doAction();
          r = false;
        }
      }
      return r;
    }
    ```


Il metodo delle tre fasi

- Inefficienza del motore ad attività
  - Valutazione del tempo e delle guardie eseguita più volte
  - È possibile ottimizzare qualcosa?
- Metodo e motore, prospettiva di analisi
  - Attività di tipo B
    - Bound, book-keeping
    - Sappiamo quando accadono, dipendono solo dal tempo
  - Attività di tipo C
    - Conditional, cooperative
    - Dipendono da condizioni (determinate da cooperazione)
  - Logica del modello più organizzata (frammentata)
  - Alcune attività si spezzano in una B e una C
  - Più naturale di quanto sembri
    ```

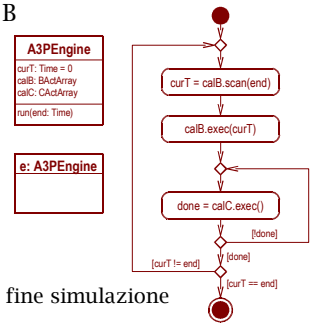
Motore a 3 fasi, strutture dati

- Due classi di attività
  - Metodo doAction() per tutte
  - Attività B, tempo
  - Attività C, guardia
- Tabelle separate
  - Attività diverse
  - Metodi diversi
- Specializzazione delle azioni
  - Non ci sono limitazioni
  - Riprogrammazione dei tempi delle B
  - Cambiamenti sulle condizioni delle C



Motore a 3 fasi, esecuzione

- Il tempo avanza alla prima B
- Esecuzione delle attività B
  - Al tempo corrente
  - Non hanno guardie
- Esecuzione delle attività C
  - Tutte le C le cui guardie sono verificate
  - Finché nessuna è verificata
- Terminazione
  - È stato raggiunto il tempo di fine simulazione



Considerazioni

- Affrontare la decomposizione del problema
  - Scambiando efficienza con manutenibilità del codice
  - Contando su tecnologie di programmazione diverse
- Decomporre la gestione degli eventi
  - Tipi di entità attive definite da classi
  - Tipi di eventi definiti da classi
  - Metodi doAction() e handle() specializzati
  - Code, dinamiche, di oggetti di tipo diverso
- Senza OO
  - Si perde il concetto di tipo
  - Si usano strutture statiche e uniformi
  - Tabelle per istruzioni di salto

Riferimenti

- M. Pidd,
  Computer Simulation in Management Science,
  Capp. 5 e 6, J. Wiley & Sons, 1998
- G. Gallo, *Note di Simulazione*, cap. 2