Modellazione discreta, progetto didattico

Simulazione & Logistica, I modulo
Esercitazione n. 7

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

Contenuti

- Bene e male, in generale
- Entità e insiemi
- Problemi interessanti, eventi e priorità
- Meccanismi della soluzione, diagrammi
- Metodi notevoli

Bene e male, in generale

- UML e metodo di modellazione
- Bene (quasi tutti)
  - Diagrammi delle classi e degli oggetti
  - Diagrammi degli stati (insomma)
- Male (non tutti)
  - Transizioni senza eventi
  - Transizioni su condizioni
  - Stati come attività (tempo nullo)
  - Eventi generati da non si sa chi
  - Meccanismi alieni per la propagazione degli eventi
- Organizzazione del testo e discussione dei problemi

Entità e insiemi

- BaseCC
  - Specializzata in BaseCCM, BaseCCS
  - 1 istanza
- ScanArea
  - *n* istanze, statiche, fissate
- Ship
  - *k* istanze, dinamiche, variabili
- Code
  - *n* o 1 istanze di GS_Queue
- ShipGen
- Obiettivo: Ship e ScanArea indipendenti dalle politiche

Problemi interessanti

- Attitudine e doveri del bravo modellatore
  - Non inseguire "piccoli casi" particolari, almeno non subito
  - Però identificarli e discuterli
- Problema
  - Valutazione delle politiche in caso di eventi contemporanei
  - Navi e aree di scansione non possono essere autonome
- Casi particolari
  - Coda multipla, arrivi e disponibilità contemporanei
  - Coda singola, disponibilità contemporanee
  - Coda singola, valutazione contemporanea dell'anticipo
  - Arrivi e reincodamenti
  - Convenienza delle scelte

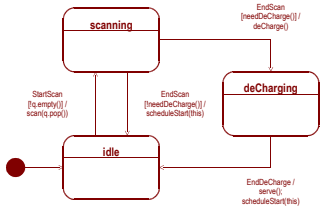
Eventi e priorità

GenShip	800 000	statica
EndDeCharge	700 000	statica
EndScan	600 000 + d (n) 500 000 + d (e)	dinamica
ReCheck	500 000 – m/10	dinamica
ShipArrival	400 000 – m/10	dinamica
StartScan	200 000 + d (n) 100 000 + d (e)	dinamica
StartService	000 000	statica

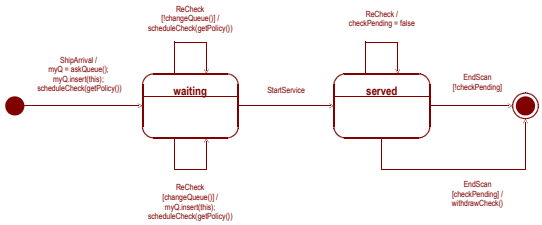
Meccanismi della soluzione

- Gestione a “blocchi” degli eventi
  - Tutti gli eventi che liberano le risorse
  - Tutti gli eventi che le richiedono (con politiche da valutare)
  - Tutti gli eventi che le occupano
  - Introduzione del segnale StartScan
- Priorità dinamica
  - È una caratteristica del motore, sfruttiamola
  - Evitare di gestire l’invio dei soli StartScan corretti
  - Mantenimento minimo di informazioni in BaseCC
  - Mettere gli StartScan in agenda ordinati
  - Fare in modo che quelli inutili siano ignorati

ScanArea, macchina a stati



Ship, macchina a stati



Metodi notevoli, 1

- void ScanArea::scan()
  - Toglie la nave dalla coda, aggiorna il tempo di uso effettivo, segnala alla nave lo *StartService*, programma l'*EndScan*.
- void ScanArea::deCharge()
  - Aggiorna i costi, programma l'*EndDeCharge*.
- GS_Queue BaseCCS::askQueue()

```
{
  sIdx = (sIdx + 1) % n;

  foreach scan area sa, starting from sIdx,
    scheduleStart(sa);

  return q;
}
```


Metodi notevoli, 2

- GS_Queue BaseCCM::askQueue()

```
{
  mark all scan areas as candidate;

  foreach candidate mark the best wrt queue;
  if unique sa { scheduleStart(sa); return sa.q; }

  foreach candidate mark the best wrt availability;
  if unique sa { scheduleStart(sa); return sa.q; }

  foreach candidate mark the best wrt type and actT;
  if unique sa { scheduleStart(sa); return sa.q; }

  sIdx = (sIdx + 1) % n;
  starting from sIdx, get the first candidate sa
  scheduleStart(sa); return sa.q;
}
```


Metodi notevoli, 3

- Bool ScanArea::needDeCharge()

```
{
  if (actT > DCTh*3600) {
    base.notServe(); return true;
  }
  if (deChargePolicy1)
    if (actT > DCTh*3600*dCP1P && base.askNextShip(saId) == -1) {
      base.notServe(); return true;
    }
  if (deChargePolicy2) {
    actDist = DCTh*3600 - actT;
    extOvfl = actT + base.askNextShip(saId)/NST - DCTh*3600;
    if ( extOvfl > actDist*dCP2P ) {
      base.notServe(); return true;
    }
  }
  base.serve(); return false;
}
```

Metodi notevoli, 4

- int BaseCCM::NextShip(saId)
  - Restituisce la massa della prima nave nella coda corrispondente alla *ScanArea*, o -1 se la coda è vuota
- int BaseCCS::NextShip(aId)
  - `{if (getCurT()!=cT) {cT=getCurT(); nS=0;} else nS++;`
    `if (q.lenght<=nS) return -1; else return q.getAt(nS).m;}`
- void BaseCC::serve()
  - `{if (getCurT()!=cT) {cT=getCurT(); nS=1;} else nS++;}`
- void BaseCC::notServe()
  - ```
    {if (getCurT()!=cT) {cT=getCurT(); nS=0;}
       else nS=max(0, nS-1);
    }
    ```
    ```