SLo 08/09 – Progetto di simulazione, I parte

Note a una possibile modellazione del caso di studio
Giovanni A. Cignoni

Entità

Le entità rilevanti del caso di studio sono definite dalle seguenti classi:

Ship, rappresenta una nave, incluse le sue caratteristiche in termini di tempi di servizio.

ScanArea, rappresenta un'area di scansione, incluse le caratteristiche del rilevatore installato;

BaseCC, rappresenta la base come informazioni della struttura della base e per la gestione delle politiche; di questa classe sono definite due specializzazioni, **BaseCCM** e **BaseCCS** rispettivamente per la politica a code multiple e a coda singola.

Nella soluzione proposta *Ship* e *ScanArea* sono modellate come entità attive. Il sistema è composto da un'istanza unica di *BaseCCM* o di *BaseCCS*, da *n* istanze di *ScanArea*, con *n* stabilito e fisso durante la simulazione, e da *k* istanze di *Ship*, con *k* variabile durante la simulazione.

Per la modellazione delle code di navi in attesa di scansione sono utilizzate istanze della classe *GS_Queue*, 1 nel caso di politica a coda singola, *n* nel caso di politica a code multiple.

Per la generazione dinamica delle istanze di *Ship* durante la simulazione è introdotta un'entità attiva dedicata definita dalla classe **ShipGen**.

Problematiche del caso di studio

L'aspetto più interessante del caso di studio riguarda la corretta valutazione delle politiche di gestione delle code e di anticipazione del ciclo di scarica quando alcuni eventi si verificano in contemporanea. Il problema principale è che le aree di scansione non possono reagire autonomamente all'arrivo delle navi. In particolare:

- politica a coda multipla, quando arriva una nave deve essere scelta la coda più favorevole, ma la scelta dipende dallo stato delle aree di scansione, una o più di queste potrebbero rendersi disponibili nello stesso momento;
- politica a cosa singola, la prima nave in coda deve essere servita dall'area più favorevole, anche in questo caso la scelta fra più aree che si rendono disponibili nello stesso momento deve essere deterministica e corretta rispetto ai criteri di valutazione dell'area più favorevole;
- politiche di anticipo nel caso di politica a coda singola, la decisione di anticipare il ciclo di scarica dipende da un'osservazione della coda; più aree che si rendono disponibili nello stesso momento devono coordinare le proprie osservazioni sulla coda, per fare le stime sulla nave corretta (la prima per l'area più favorevole, la seconda per la seconda, ...) o per identificare una pausa nel traffico (se in coda c'è una nave, per l'area più favorevole, la coda non è vuota, ma per la seconda sì).

Si tratta di casi particolari e probabilmente rari (dove probabilmente significa proprio in funzione della probabilità di eventi contemporanei). Il primo in particolare che implica la contemporaneità di almento tre eventi. È tuttavia un buon esercizio di modellazione trattarli correttamente o comunque in modo deterministico e definito da scelte consapevoli.

La soluzione proposta prevede in primo luogo l'introduzione di un segnale *StartScan* per gestire "a blocchi" prima tutti gli eventi che rendono disponibili le aree di scansione (fine delle scansioni e dei cicli di scarica), poi tutti quelli che ne richiedono l'utilizzo (arrivi e riposizionamenti delle navi) e infine tutti quelli che, appunto, ne determinano l'effettivo utilizzo (inizio delle scansioni). Inoltre, per ordinare gli eventi in modo che, quando contemporanei, la gestione sia deterministica, la priorità di alcuni eventi è assegnata dinamicamente.

Quest'ultimo accorgimento è sufficiente per gestire correttamente la scelta dell'area più favorevole nella politica a coda singola. Per garantire la scelta equiprobabile in caso di parità è sufficiente, quando si generano più *StartScan* contemporaneamente segnalarli con un ordine round-robin. Il caso più delicato riguarda l'arrivo di una nave con coda vuota che è ragionevolmente frequente, ma è

controllato dalla mediazione della base che applica il round-robin. Altri casi sono estremamente rari in quanto implicano situazioni di parità assoluta delle condizioni delle aree di scansione.

Nella soluzione proposta si è scelto di dare priorità agli eventi che rendono disponibili le aree. Questo è "conveniente" ai fini della scelta dell'area più favorevole (l'insieme di scelta include anche le aree resesi disponibili nello stesso istante dell'arrivo di una nave) è però "sconveniente" ai fini della predizione di una pausa per anticipare il ciclo di scarica (non si considerano le navi che stanno arrivando in quell'istante e si può anticipare per sbaglio). Tuttavia questo evento è più raro (oltre alla contemporaneità degli eventi l'area deve essere vicina al ciclo di scarica).

Per la gestione deterministica dell'eventuale contemporaneità nell'arrivo o nel reincodamento di due (o più) navi, si è scelto di dare priorità ai reincodamenti (secondo la politica universale di favorire chi è già in coda) e alla nave con massa minore (secondo la politica universale di favorire chi richiede tempi di servizio più brevi).

Eventi

ShipArrival. Arrivo di una nave alla base. La nave è un parametro dell'evento.
Gestito dalla propria *Ship* (chiede la coda alla base, che o è unica o è viene identificata in base allo stato delle code e delle aree di scansione, si incoda, programma l'eventuale *ReCheck*). La richesta della coda a *BaseCCM* include la segnalazione di uno *StartScan* dedicato; la richiesta a *BaseCCS* include invece la segnalazione di *n StartScan* (segnalati secondo un ordine round-robin).

ReCheck. Rivalutazione della posizione in coda della nave, usato solo in caso di politica a code multiple. La nave è un parametro dell'evento. È in sostanza una ripetizione dell'arrivo, ma rimozione e re-incodamento sono condizionati dal confronto con l'attuale posizione in coda.
Gestito dalla propria *Ship* (chiede la coda alla base, che viene identificata in base allo stato delle code e delle aree di scansione, eventualmente si ri-incoda, programma il successivo *ReCheck*). La richiesta della coda a *BaseCCM* include la segnalazione di uno *StartScan* dedicato.

StartScan. Inizio dell'attività di scansione di una nave da parte di un'area di scansione. L'area di scansione è un parametro dell'evento. È un segnale a tempo 0 (rispetto a *ShipArrival*, *ReCheck*, *EndScan* o *EndDeCharge*) inviato alle aree di scansione tramite la base, come eco dell'arrivo o del riposizionamento di una nave, o dall'area di scansione a sé stessa, come eco della raggiunta disponibilità. Gestito dalla propria *ScanArea* (se la coda non è vuota preleva la nave, gli segnala lo *StartService* e programma il proprio *EndScan*).

StartService. Inizio del servizio di scansione di una nave. La nave è un parametro dell'evento. È un segnale a tempo 0 (rispetto a *StartScan*) inviato dall'area di scansione alla nave che sta servendo. Non è strettamente necessario: è introdotto per avere una gestione più esplicita dello stato delle navi.
Gestito dalla propria *Ship* (cambia stato aggiornando i dati di servizio).

EndScan. Fine dell'attività di scansione di una nave da parte di un'area di scansione. La nave e l'area di scansione sono parametri dell'evento.
Gestito dalla propria *Ship* (che termina), dalla propria *ScanArea* (che valuta, eventualmente anche in base a informazioni richieste alla base, se iniziare un ciclo di scarica e o si rende disponibile e segnala uno *StartScan* per sé, o inizia un ciclo di scarica e programma per sé un *EndDeCharge*).

EndDeCharge. Fine del ciclo di scarica. L'area di scansione è un parametro dell'evento.
Gestito dalla propria *ScanArea* (che si rende disponibile e segnala uno *StartScan* per sé).

Tutti gli eventi sono generati internamente al sistema, tranne *ShipArrival* che è esterno. Per farlo generare dinamicamente è introdotto l'evento *NewShip*, gestito dal generatore di navi *ShipGen* (genera un nuovo *NewShip* e il corrispondente *ShipArrival*).

Priorità fra gli eventi

Gli eventi a priorità più alta sono elencati per primi.

Prima tutti gli eventi *GenShip*; è una scelta arbitraria, coerente con l'osservazione che gli eventi di arrivo, concettualmente, esistono prima dell'inizio della simulazione, quindi l'evento di generazione viene sempre prima di ogni altro.
Priorità statica GenShip = 800 000

Poi tutti gli *EndDeCharge*; modificano la disponibilità delle aree di scansione, devono essere gestiti prima di arrivi e riposizionamenti, per valutare questi ultimi in uno stato definito e stabile. Poiché sicuramente l'area è disponibile a servire devono essere gestiti prima per aggiornare correttamente i puntatori necessari a gestire le politiche di anticipo.
Priorità statica EndDeCharge = 700 000

Poi tutti gli *EndScan*; modificano la disponibilità delle aree di scansione, devono essere gestiti prima di arrivi e riposizionamenti, per valutare questi ultimi in uno stato definito e stabile. Tuttavia gli *EndScan* causano anche la valutazione delle politiche di anticipo del ciclo di scarica, quindi devono essere ordinati per favorevolezza dell'area di scansione, in modo che, in caso di coda singola, l'area è certa di decidere rispetto alla nave che effettivamente servirebbe.
Priorità statica EndScan = 600 000 + d (normale)
500 000 + d (extra)

Poi tutti i *Recheck*; hanno la precedenza rispetto agli *ShipArrival*, devono essere valutati nello stato stabile delle aree e possono causare reincodamenti, condizionando così un'eventuale successiva valutazione della coda migliore da parte di *BaseCCM*.
Priorità dinamica: ReCheck = 500 000 – m/10.

Poi tutti gli *ShipArrival*; devono essere valutati nello stato stabile delle aree causano incodamenti, condizionando così un'eventuale successiva valutazione della coda migliore da parte di *BaseCCM*.
Priorità dinamica: ShipArrival = 400 000 – m/10.

Per tutti gli *StartScan*; la priorità deve assicurare che, in caso di coda singola, la prima nave della coda sia servita dal rivelatore più favorevole. La priorità è calcolata in base al tipo e alla distanza dal ciclo di scarica, se non ci sono altre navi per le altre aree di scansione disponibili lo *StartScan* sarà un falso allarme, mentre non sarà sentito dalle aree non disponibili a quel momento.
Priorità dinamica: StartScan = 200 000 + d (normale)
100 000 + d (extra)

Infine gli StartService, utilizzati solo per cambiare lo stato delle navi e non afflitti da problemi di contemporaneità.
Priorità statica: StartService = 000 000

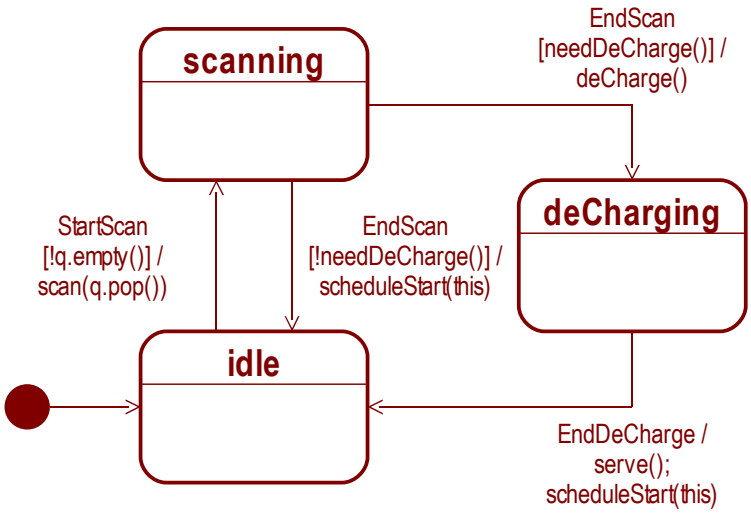

Fig. 1: diagramma della macchina a stati di *ScanArea*

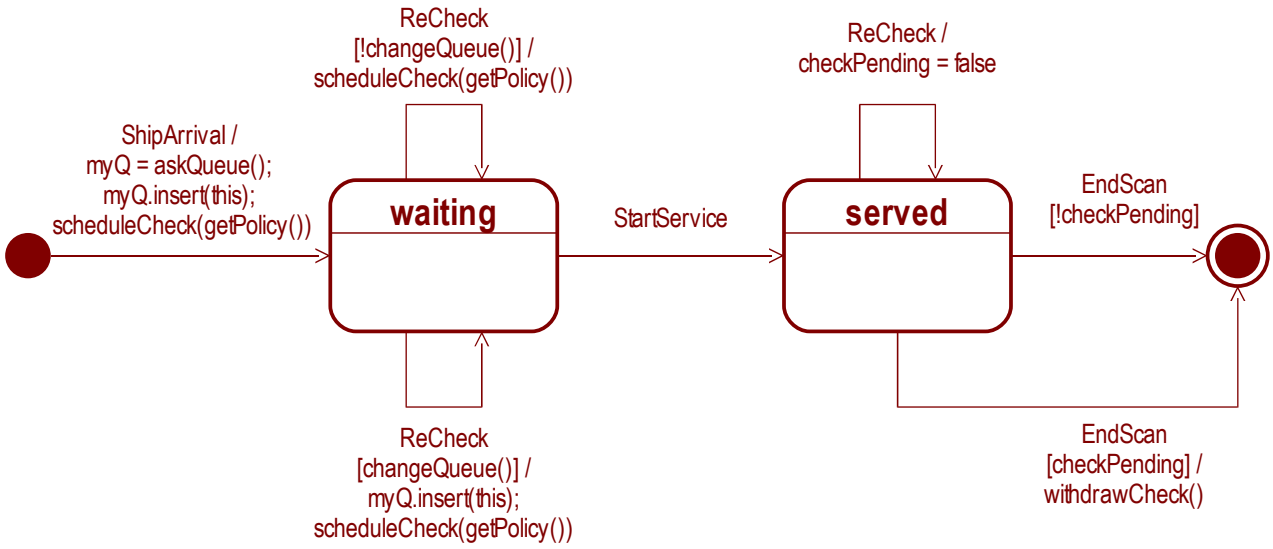

Fig.2: diagramma della macchina a stati di *Ship*

Metodi notevoli

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, segnala la non disponibilità, programma l'*EndDeCharge*.

GS_Queue BaseCCS::askQueue()
Restituisce la coda unica e segnala con ordine round-robin lo *StartScan* a tutte le aree di scansione. Il risultato è utilizzato dalla nave che sta gestendo lo *ShipArrival*.

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

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

  return q;
}
```

GS_Queue BaseCCM::askQueue()
Identifica la coda più favorevole la coda in cui incodarsi: la più corta, e a parità quella con dell'area di scansione più favorevole, in funzione del risultato segnala lo *StartScan* corretto. Il risultato è utilizzato dalla nave che sta gestendo lo *ShipArrival* o il *ReCheck* per incodarsi. Lo *StartScan* è necessario nel caso l'arrivo o il reincodamento avvenga in una coda vuota con area di scansione disponibile, negli altri casi sarà ignorato.

```
{
  mark all scan areas as candidate;

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

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

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

  sIdx = (sIdx + 1) % n;

  starting from sIdx, get the first candidate sa
  scheduleStart(sa); return sa.q;
}
```

bool Ship::changeQueue()
Chiede alla base la coda più favorevole la coda in cui incodarsi e confronta la lunghezza della coda restituita con la posizione attuale, se favorevole cambia il valore di myQ. L'invocazione del metodo askQueue() causa un segnale *StartScan* che è necessario solo se il reincodamento avviene e se avviene in una coda vuota. In tutti gli altri casi sarà ignorato.

```
{
  newQ = base.askQueue();

  if ( newQ.getSize() < myQ.getPos(this) ) {

    myQ.remove(this);
    myQ = newQ;
    return true;
  }

  return false;
}
```

Bool ScanArea::needDeCharge()
Serve a decidere se iniziare un ciclo di scarica, implementa le politiche (nessuna, anticipo per pausa, anticipo per eccessivo superamento, entrambi), invoca il metodo BaseCC::askNextShip(sId) per fare la previsione sul tempo di scarica e per sapere se c'è una pausa di traffico in corso, invoca i metodi BaseCC::notServe() e BaseCC::serve() per segnalare se il ciclo di scarica viene deciso o no.

```
{
  if (actT > DCTh*3600) {
    base.notServe();
    return true;
  }

  if (deChargePolicy1)
    if ( (actT > (DCTh*3600*dCP1Perc)) && (base.askNextShip(sId) == –1) ) {
      base.notServe();
      return true;
    }

  if (deChargePolicy2) {
    actDist = DCTh*3600 – actT;
    extOvfl = actT + base.askNextShip(sId)/NST – DCTh*3600;
    if ( extOvfl > actDist*dCP2Perc ) {
      base.notServe();
      return true;
    }
  }

  base.serve();
  return false;
}
```

int BaseCCM::NextShip(aId)
Invocato da una *ScanArea*, restituisce la massa della prima nave nella coda corrispondente alla *ScanArea*, o un valore di default (-1) se la coda è vuota (pausa nel traffico).

int BaseCCS::NextShip(aId)
Invocato da una *ScanArea*, restituisce la massa della prossima nave da servire, o un valore di default se non ci sono navi da servire (pausa nel traffico). L'identificatore della *ScanArea* è in effetti ignorato, ma il metodo deve tener conto di precedenti invocazioni e di eventuali navi non servite (perché la *ScanArea* anticipa il ciclo) restituendo quella che effettivamente è la prossima nave da servire.

```
{
  if (getCurT() != callT) { callT = getCurT(); next = 0; } else { next++; }

  if (q.lenght <= next) { return -1; } else { return q.getAt(next).mass; }
}
```

void BaseCC::serve()
Invocato da una *ScanArea* per segnalare la propria disponibilità quando non decide un ciclo di scarica o alla fine di ciclo di scarica, quando sicuramente non invoca needDeCharge(). Nel caso della politica a coda singola è necessario per indicare che una nave della coda sarà servita e permettere la corretta valutazione delle politiche di anticipo del ciclo di scarica.

```
{
  if (getCurT() != callT) { callT = getCurT(); next = 1; } else { next++; }
}
```

void BaseCC::notServe()
Invocato da una *ScanArea* per segnalare il ciclo di scarica. Nel caso della politica a coda singola è necessario per indicare che la nave non è servita e deve essere restituita anche in una successiva eventuale invocazione del metodo.

```
{
  if (getCurT() != callT) {
    callT = getCurT();
    next = 0;
  } else {
    next = max( 0, next-1);
  }
}
```