2. Simulazione discreta: approcci alla simulazione

Corso di Simulazione

Anno accademico 2008/09

Elementi di un programma di simulazione

- Controllore
- Tempo di simulazione
- Generatore dei dati di input
- Entità
- Eventi
- Attività
- Stati
- Processi

Simulazione per eventi: le classi

L'approccio basato sugli eventi permette di costruire programmi di simulazione molto compatti ed efficienti, ma anche proprio per questo più costosi dal punto di vista della manutenzione e degli aggiornamenti.

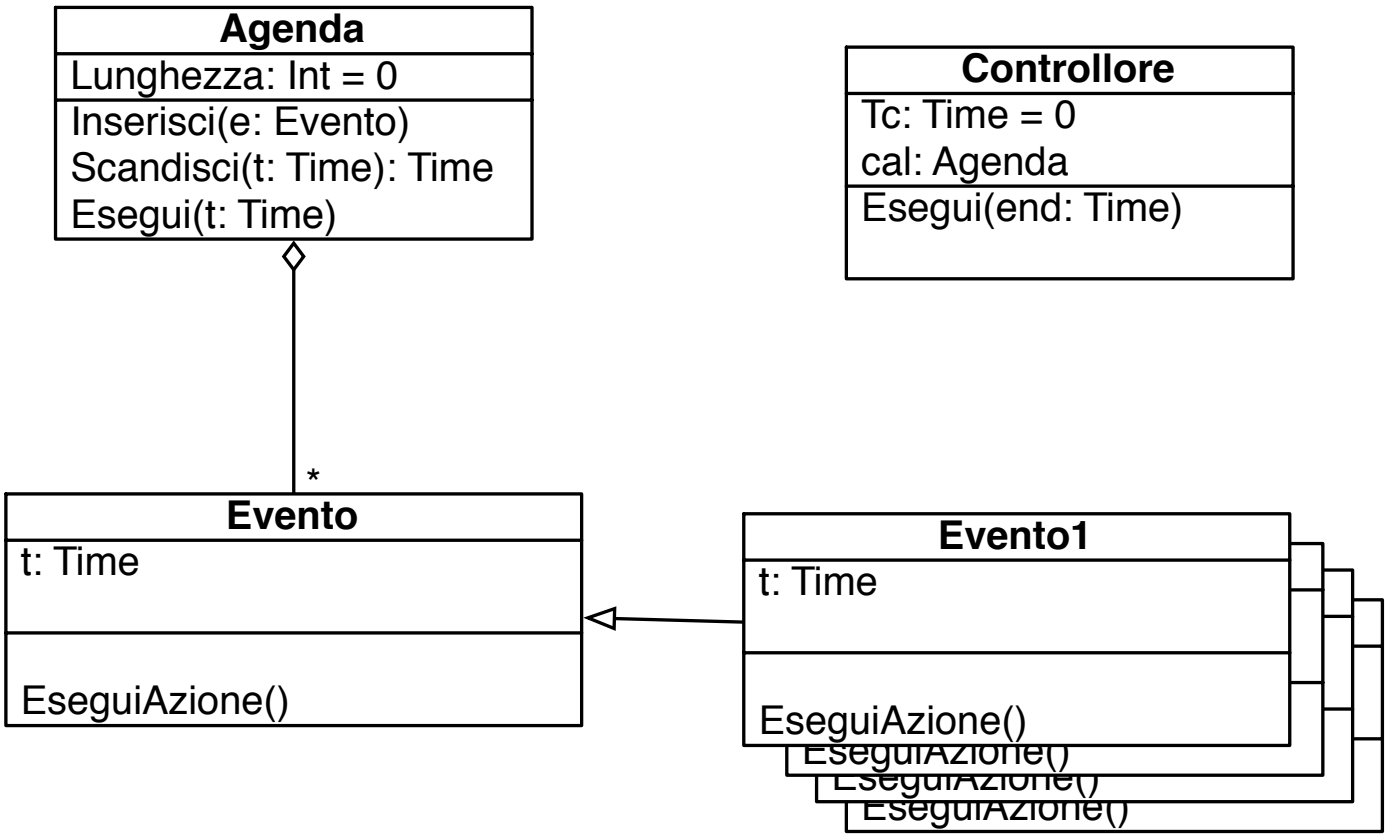
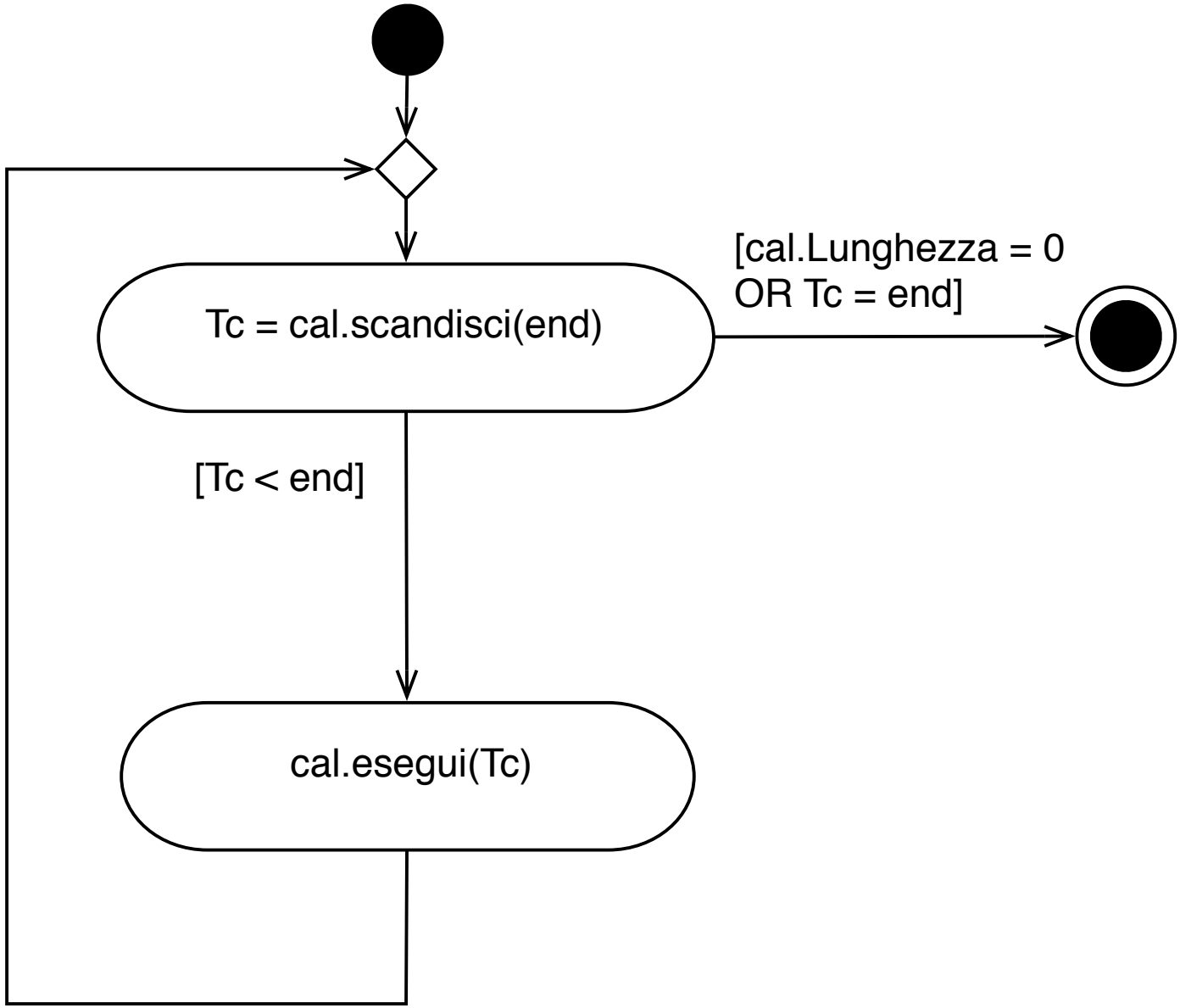


Figure: Diagramma delle classi in un simulatore per eventi

Simulazione per eventi: il controllore



Simulazione per eventi: gli eventi nel caso del botteghino teatrale

- *ArrivaCp*. Se L'impiegato è occupato il cliente viene inserito in Q_p, altrimenti inizia il servizio, e sia l'impiegato che il cliente passano allo stato di servizio.
- *ArrivaCt*. Se L'impiegato è occupato, oppure è arrivato contemporaneamente un cliente fisico, il cliente viene inserito in Q_t. Altrimenti, inizia il servizio, e sia l'impiegato che il cliente passano allo stato di servizio.
- *FineServizio*. Se la coda Q_p non è vuota, l'impiegato rimane nello stato di servizio ed estrae il primo cliente da Q_p, altrimenti estrae il primo cliente da Q_t, se tale coda non è vuota. In entrambi i casi, il cliente estratto passa allo stato di servizio,e viene programmato un nuovo evento *FineServizio*. Se entrambe le code sono vuote, l'impiegato si mette in attesa, cambiando così di stato. Il cliente appena servito esce dal sistema.

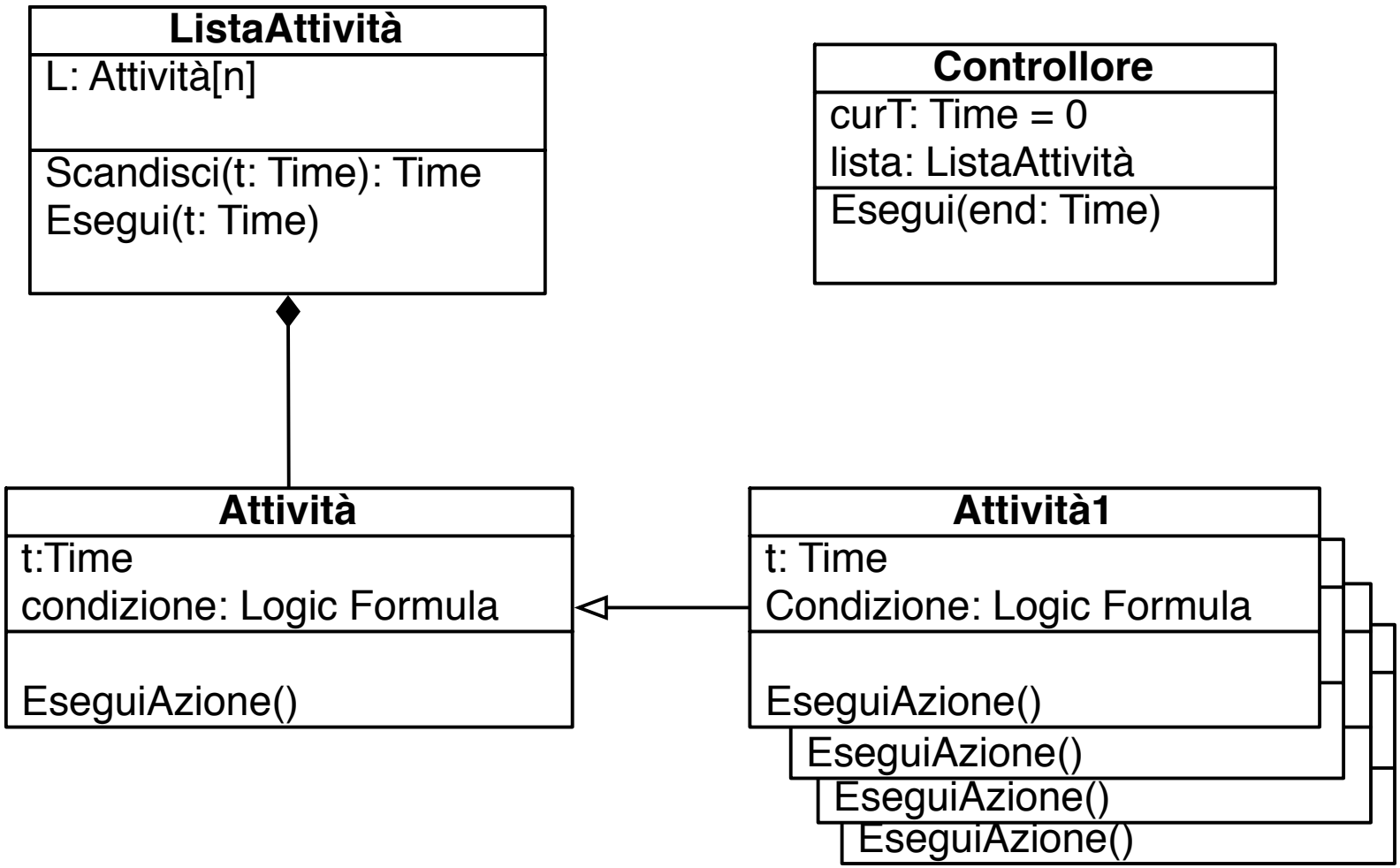
Simulazione per eventi: una variante

Il problema principale nella modellazione per eventi nasce dal fatto che le operazioni svolte da un singolo evento possono essere molto complesse e si propagano coinvolgendo diverse entità. Questo rende alta la possibilità di errori, rende difficile il mantenimento e l'aggiornamento del modello.
Una possibile variante è quella di distribuire le operazioni fra le diverse entità. Ognuna di esse mantiene l'informazione sullo stato in cui si trova (punto nel suo ciclo degli stati) e sul tipo di operazioni che deve svolgere in corrispondenza a ciascuno stato ed a ciascun evento.
Allora ogni evento contiene fra le sue informazioni la lista delle entità "attive", cioè che sono coinvolte quando esso viene attivato dal controllore. E l'operazione eseguita dall'evento è solo quella di attivare le entità coinvolte.
In questo modo le operazioni sono più parcellizate e distribuite e quindi il programma risulta più facile da controllare ed aggiornare.

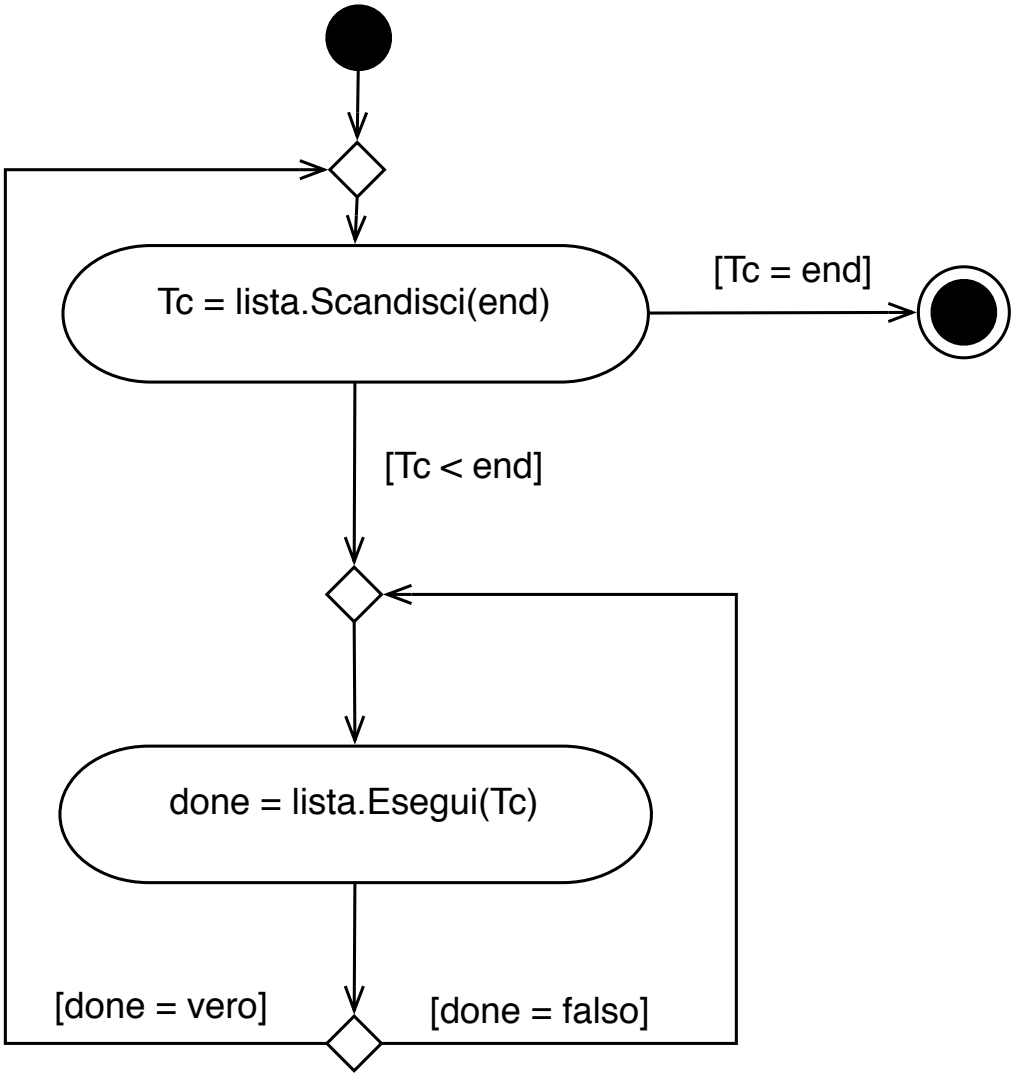
Simulazione per attività

In questo approccio, a partire dal diagramma degli stati, si decompongono le attività svolte nel sistema in attività elementari. Queste attività corrispondono agli eventi, cioè a quei fatti o azioni che portano ai cambiamenti di stato del sistema e che quindi danno origine alle transizioni. Il ruolo del controllore sarà quello di gestire la lista delle attività individuate in modo che vengano eseguite. Ogni attività è caratterizzata dalla condizione che la fa avvenire e dalle azioni che vengono corrispondentemente fatte, e viene rappresentato per mezzo di una tabella in cui sono riportate le informazioni rilevanti.

Simulazione per attività: le classi



Simulazione per attività: il controllore



Simulazione per attività: il caso del botteghino teatrale (1)

Le attività:

- $a[1]$ Inizio servizio allo sportello
- $a[2]$ Inizio servizio al telefono
- $a[3]$ Fine servizio allo sportello
- $a[4]$ Fine servizio al telefono
- $a[5]$ Arrivo cliente allo sportello
- $a[6]$ Arrivo cliente al telefono

Simulazione per attività: il caso del botteghino teatrale (2)

Attività	a[1]: Inizio servizio allo sportello
Attributi	- $I.Libero = vero \ \& \ Qp.lunghezza > 0$
Azioni	a[3].t := curT + Qp.Estrai().t; $I.Libero := falso$

Simulazione per attività: il caso del botteghino teatrale (3)

Attività	a[2]: Inizio servizio al telefono
Attributi	- $I.Libero = vero$ & $Qp.lunghezza = 0$ & $Qt.lunghezza > 0$
Azioni	a[4].t := curT + Qt.Estrai().t; $I.Libero := falso$

Simulazione per attività: il caso del botteghino teatrale (4)

Attività	a[3]/a[4]: Fine servizio
Attributi	t vero
Azioni	*I.Libero* := *vero*

Simulazione per attività: il caso del botteghino teatrale (5)

Attività	a[5]: Arrivo cliente allo sportello
Attributi	t vero
Azioni	Qp.Inserisci(nuovoCliente(rndp1()) a[5].t := curT + rndp2()

Simulazione per attività: il caso del botteghino teatrale (6)

Attività	a[6]: Arrivo cliente al telefono
Attributi	t vero
Azioni	Qp.Inserisci(nuovoCliente(rndp1()) a[6].t := curT + rndp2()

Simulazione per processi

In questo approccio tutti gli eventi del ciclo degli stati di una entità, con le relative operazioni, vengono vengono accorpati in una sequenza detta processo. Un processo è sostanzialmente la sequenza delle operazioni descritte dal diagramma degli stati.

Un processo può essere attivo oppure in attesa. In quest'ultimo caso, si può trattare di una attesa condizionata, quando la ripresa dell'attività del processo dipende dal realizzarsi di condizioni esterne, oppure di un'attesa programmata, quando il tempo in cui il processo verrà riattivato è predefinito.

Simulazione per processi: il botteghino teatrale(1)

1. Arrivo del cliente C quando il tempo di simulazione è uguale al suo tempo di arrivo, cioè $curT = P(C).t$.
2. Calcolo del tempo di arrivo , $P(C').t$ del cliente successivo, C';
3. Generazione dell'entità C' e creazione del processo $P(C')$;
4. Il processo $P(C')$ viene posto in stato di attesa fino a che non risulti $curT = P(C').t$;
5. Il cliente C viene inserito nella prima coda, e viene posto in stato di attesa fino a quando non si trovi in testa alla coda e risulti Impiegato.Libero = *vero*;
6. Si pone Impiegato.Libero := *falso*, si pone $P(C)$.Tempo_Prossima_Attività $= curT + C.t$, e si pone il processo in stato di attesa fino a quando $curT = P(C)$.Tempo_Prossima_Attività;
7. Si pone Impiegato.Libero := *Vero* e si interrompe il processo.

Simulazione per processi: il botteghino teatrale (2)

Nell'esempio considerato, l'impiegato può essere considerato come una risorsa che viene utilizzata dai processi (clienti). In casi più complessi si possono avere diverse classi di entità e quindi tipi di processi che competono per l'uso di risorse comuni. In questo caso si hanno processi che interagiscono, ciascuno condizionato dallo stato degli altri.

Simulazione per agenti

- Agenti
- Ambiente
- Controllore

Simulazione per agenti: gli agenti

- Un agente è una entità discreta dotata di proprietà e di obiettivi che ne caratterizzano e guidano il comportamento, e dotata della capacità di prendere decisioni.
- Un agente è situato in un ambiente con il quale interagisce. Parimenti un agente interagisce con gli altri agenti. Devono quindi essere chiaramente definiti i protocolli che consentono tali interazioni.
- Un agente può avere obiettivi e la capacità di valutare gli esiti delle sue azioni rispetto a tali obiettivi.
- Un agente è autonomo e, almeno rispetto ad un insieme di possibili azioni, può funzionare indipendentemente nel suo ambiente e nei rapporti con gli altri agenti.
- Un agente è flessibile ed ha la capacità di imparare ed adattare i suoi comportamenti sulla base dell'esperienza. Deve avere quindi una qualche forma di memoria ed, eventualmente, delle metaregole che gli permettano di modificare le sue regole di comportamento.

Simulazione per agenti: l'ambiente

L'ambiente rappresenta tutte quelle realtà ed informazioni che hanno un effetto sul sistema da modellare e quindi in particolare sugli agenti. Ad esempio fanno parte dell'ambiente le risorse disponibili, la domanda (quando non sia anch'essa rappresentata per mezzo di agenti), parametri quali tasso di sconto, costi, probabiltà di guasto di una macchina, ...

Simulazione per agenti: il controllore

Il programma di controllo ad ogni passo fa avanzare il tempo (in genere per intervalli regolari) e fa eseguire le operazioni previste per ciascun agente.
Alcune operazioni che vengono effettuate ad ogni avanzamento del tempo, ed operazioni condizionate che vengono realizzate solo quando si verificano opportune condizioni.
Il risultato della simulazione può non essere indipendente dall'ordine con cui il controllore fa eseguire le diverse operazioni.