Simulazione a eventi discreti

- Definizioni
- struttura ed elementi di un simulatore DES
- esempi di modelli e implementazione DES
- Esecuzione di una simulazione DES

1

Introduzione alla Simulazione

- Definizione di simulazione
  - Imitazione delle operazioni eseguite nel tempo da un sistema o processo reale
- Scopo della simulazione
  - Generazione di una storia artificiale del sistema
  - Studio e valutazione delle caratteristiche del sistema (analisi delle prestazioni)
  - Risposta alla domanda "cosa accade se…"

2

Introduzione alla Simulazione

- Scopo della simulazione
  - Analisi di sistemi "ipotetici"
  - Confronto tra due o più sistemi
  - Determinare valore ottimale di parametri
  - Determinare i punti critici (bottlenecks)
  - Capacity planning
  - Predire le prestazioni del sistema nel futuro

3

Introduzione alla Simulazione

- Possibili alternative per l'analisi prestazionale di sistemi:
  - Simulazione
  - Monitor del sistema (benchmark)
  - Analisi

4

Introduzione alla Simulazione

- Perché la simulazione?
  - Se il sistema non esiste non posso usare benchmarks
  - Permette valutazioni **ripetibili** di sistemi (e nelle stesse condizioni)
  - Permette valutazioni di eventi e condizioni rare o rischiose
  - Permette stime non misurabili su sistemi reali
  - Non introduce overhead o alterazioni delle stime nel sistema (es. benchmark per valutare prestazioni S.O.)
  - Permette una buona accuratezza

5

Introduzione alla Simulazione

- Vantaggi della simulazione
  - Test di sistemi prima di investire sulle strutture
  - Possibilità di zoom sul tempo simulato
  - Possibilità di capire le cause di eventi
  - Esplorare nuove politiche di gestione
  - Diagnosi di problemi
  - Identificazione di requisiti
  - Comprensione del sistema
  - Ripetibilità

6

Introduzione alla Simulazione

- Svantaggi della simulazione
  - Costruzione dei modelli lunga e non banale
  - Interpretazione dei risultati lunga e complessa
  - Vengono in aiuto
    - **Simulatori** (primitive e strutture dati)
    - **Ambienti di simulazione** (+ analisi dati di output)

7

Introduzione alla Simulazione

- Concetti di Modeling e loro impiego in simulazione
  - Sistemi vs. modelli
  - Variabili di stato del sistema
  - eventi
  - Entità e attributi
  - Risorse
  - Gestione di liste
  - Attività e ritardi
  - Modelli di simulazione a eventi discreti

8

Introduzione alla Simulazione

- Sistema da analizzare:
  - una collezione di HW, SW e FW
- Metrica:
  - criterio usato per la valutazione delle prestazioni del sistema da analizzare
- Carico di lavoro (workload)
  - Le richieste fatte dagli utenti del sistema

9

Introduzione alla Simulazione

- Modello: astrazione o rappresentazione di un sistema reale
  - Analisi del sistema e isolamento delle caratteristiche da includere nel modello
  - Complessità: commisurata alle necessità dell'analisi
  - Semplicità: favorisce implementazione corretta e prestazioni del simulatore
  - Fedeltà: fornisce dati realmente descrittivi
  - Approssimazione: trade-off con semplicità
  - Tempo simulato vs. "wall-clock time"

10

Introduzione alla Simulazione

- Modelli a **tempo** continuo/discreto
  - tempo continuo: lo stato del sistema è definito sempre in ogni t
  - tempo discreto: lo stato del sistema è definito in istanti di tempo discreti: t, t+1…
    - Es. numero studenti del Mercoledi

11

Introduzione alla Simulazione

- Modelli a **stato** continuo/discreto
  - continuous-event model: le var. di stato sono continue
    - Es. livello del liquido in un bacino idrico
    - Possono essere approssimati da discrete-event models
  - discrete-event model: le var. di stato sono discrete e cambiano valori in corrispondenza di eventi discreti
    - Es. numero di persone in attesa alla posta

12

Introduzione alla Simulazione

– Modello deterministico/probabilistico
  • deterministico: output(input k)= x
  • probabilistico: output(input k)= x1, x2, x3…
– Modello statico/dinamico
  • statico: lo stato non dipende dal tempo
  • dinamico: lo stato dipende dal tempo

13

Introduzione alla Simulazione

– Modello lineare/non lineare
  • lineare: output(x)= k*x
  • non lineare: output(x) != k*x
– Modello aperto/chiuso
  • chiuso: nuovi job non entrano o escono dal sistema
  • aperto: “source” e “sink” per i job

14

Introduzione alla Simulazione

– Modello stabile/instabile
  • stabile: comportamento converge allo stato stazionario
  • instabile: comportamento che cambia continuamente (e se è oscillante?)

15

Introduzione alla Simulazione

– Variabili di stato del sistema
  • variabili che descrivono lo stato del sistema al tempo (simulato) t, e al livello di definizione sufficiente per l’analisi
  • permettono di interrompere e riprendere la simulazione
  • Sono strutture dati del modello

16

Introduzione alla Simulazione

– Evento
  • un cambiamento nello stato del sistema
  • avviene al tempo simulato t
  • eventi che accadono allo stesso tempo t devono essere “serializzati” opportunamente (e univocamente, per ripetibilità della simulazione).
– Evento interno (endogeno)
  • Riguarda variabili interne al modello
  • Es. inizio di un servizio di un job in coda
– Evento esterno (esogeno)
  • Riguarda variabili esterne al modello
  • Es. arrivo di un nuovo utente in coda

17

Introduzione alla Simulazione

– Attività
  • Periodo di tempo di durata pre-determinata caratterizzato da attività in corso
  • Scheduling di eventi di inizio/fine attività
  • Nella simulazione a eventi discreti (DES) le attività avanzano il tempo simulato t
– Durata dell’attività
  • Caratterizzata da distribuzione statistica
– Ritardo (delay)
  • Durata indefinita di un’attività, legata alle condizioni e all’evoluzione del sistema (attesa)

18

Introduzione alla Simulazione

- Entità (elementi che eseguono transazioni)
  - Oggetti esplicitamente definiti nel modello
  - sono gli elementi che "fluiscono" nel sistema
  - determinano l'accadere di eventi, oppure reagiscono ad eventi
  - Possono competere per ottenere le risorse ed essere accodati nelle corrispondenti code di attesa
- Attributi
  - Sono valori locali delle entità
  - Es. tempo di arrivo del cliente, dimensione della richiesta di servizio, ecc.

19

Introduzione alla Simulazione

- Risorse
  - Sono entità che forniscono servizi passivi (risorse) o attivi (server) ad entità dinamiche (che possono richiedere una o più unità della risorsa passiva, e mai due server contemporaneamente)
  - Le risorse richieste, se occupate, possono determinare un'attesa in coda
    - gestione della coda, prima e dopo ogni servizio
  - Possono fornire servizi in parallelo (server parallelo) può servire più entità contemporaneamente

20

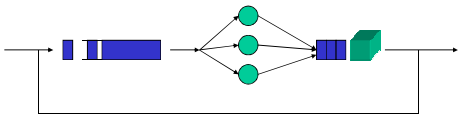
Introduzione alla Simulazione

- Liste
  - Sono strutture usate per implementare code
  - Possono essere mantenute ordinate, secondo i valori di attributi per ragioni di efficienza
  - Discipline di attesa/servizio
    - Es. FIFO, LIFO
    - Shortest Request First (basata su attributo locale)
    - ecc...

21

Introduzione alla Simulazione

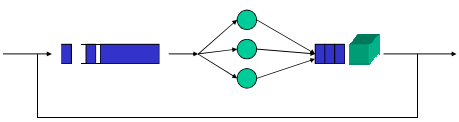
- DES: una prima visione transaction-flow
  - sistema costituito da unità di traffico (transactions) discrete che "fluiscono" attraverso le componenti e le risorse del sistema



22

Introduzione alla Simulazione

- DES: lo stato del sistema cambia solo a determinati istanti di tempo (discreto), anche se "casuali"...



23

Introduzione alla Simulazione

- Esperimento di simulazione
  - si eseguono più repliche (trials)
    - si usano sequenze pseudo-random diverse
    - producono set di risultati indipendenti e identicamente distribuiti
    - ogni replica viene detta RUN di simulaz.

24

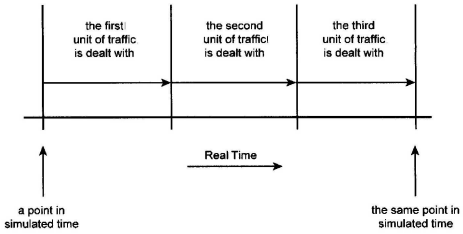
Introduzione alla Simulazione

– un RUN di simulazione
  • consiste nell'esecuzione del modello
  • avanza il CLOCK di tempo simulato
    – diverso dal Wall Clock Time
  • in DES ci sono due elementi del simulatore ciclicamente eseguiti
    – esegui tutti gli eventi/processi del tempo T
    – avanza il T simulato al prossimo evento/processo

25

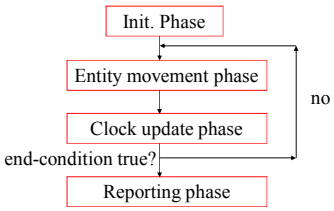
Introduzione alla Simulazione

– esecuzione eventi del tempo simulato T



Introduzione alla Simulazione

– un RUN di simulazione



27

Introduzione alla Simulazione

- Componenti del simulatore
  – routine di inizializzazione:
    • chiamate per prime, definiscono lo stato iniziale del sistema, gli stream pseudo-casuali, ecc.
  – Varie routine di gestione dei singoli eventi, o scheduling dei processi (simulatore + implementazione del modello)
  – Report generator:
    • procedure che al termine della simulazione generano i dati di stima
  – routine di trace:
    • procedure per notificare eventi o stime a run-time
  – gestione dinamica memoria e garbage collection

28

Introduzione alla Simulazione

Traduzione del modello
- Il modello concettuale ottenuto viene codificato in una forma interpretabile dal calcolatore, ottenendo un modello operazionale
- Dipende dal simulatore e dalla tecnica di implementazione della simulazione usata
- Occorre definire le strutture dati, la loro gestione, le funzioni le procedure deputate alla gestione, le funzioni e le procedure di supporto
- Verifica e Validazione

29

Introduzione alla Simulazione

- Tipo di simulazione:
  – **Discrete-event simulation (DES)**
    • simulazione che usa un modello del sistema a **tempo discreto**
    • opposto di continuous-event simulation
    • può essere basata su stato continuo o discreto
    • Le variabili di stato cambiano solo in corrispondenza ad eventi discreti, determinati a loro volta da attività e ritardi.

30

Introduzione alla Simulazione

- Strutture di modeling e DES
  - Esistono essenzialmente quattro metodologie
  - noi vediamo rapidamente 1 e soprattutto 2

1. Interazione tra Processi
2. Scheduling di Eventi
3. Scansione di attività
4. "Tre fasi"

31

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - Il flusso di esecuzione di un processo in esecuzione emula il flusso di un oggetto attraverso il sistema
  - L'esecuzione procede finchè il flusso non viene bloccato o entra in una nuova attività
    - Attesa in coda, servizio (ritardo), sink
  - Quando il flusso di un'entità viene bloccato, il tempo simulato avanza al tempo di inizio previsto dalla prima successiva entità in esecuzione
  - occorre un processo che rileva deadlock?

32

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - è un approccio naturale alla simulazione per chi sa programmare (transaction flow = flow chart)
  - il comportamento di ogni entità viene codificato come "codice da eseguire" o procedura
  - esistono regole chiare e univoche su chi viene eseguito e fino a quando. (non lo scheduler di processi gestito dal S.O, a meno che non si creino primitive di gestione e sincronizzazione ad hoc...)
  - primitive di temporizzazione e di sincronizzazione tra processi

33

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - es. linguaggi SIMULA, C-SIM, ecc.
  - implementazione di processi pseudo-paralleli
  - reentrant-code (un programma eseguibile da più istanze di processi "diversi", con stato locale), threads...
  - processi eseguiti in MODEL-time (non real time) che può essere "accelerato o rallentato".

34

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - Vita di un processo nel modello
    - fase "attiva" di esecuzione (richiede 0 model-time)
      - aggiornamento e gestione stato locale e globale
    - fase "passiva" di esecuzione (in >0 model-time discreto)
      - es. attesa per risorsa o servizio ricevuto
  - ordine di esecuzione dei processi governato dal "calendar" (lista di puntatori a processi ordinati per model-time di "risveglio") e gestito attraverso un processo-scheduler (kernel)

35

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - Possibili stati dei processi
    - Passive: il processo è definito, ma non ha nessuna entry relativa nel calendar, cioè non si sa se e quando si "risveglia"
    - Planned: il processo è definito, e ha una entry relativa nel calendar, cioè si sa quando si "risveglia"
    - Active: il processo è eseguito (la sua entry è in testa al calendar)
    - Terminated: il processo è terminato (non può essere riavviato) e i suoi dati sono ancora mantenuti in memoria
  - le transizioni di stato avvengono attraverso l'esecuzione del processo stesso o da parte di terzi

36

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - Primitive di transizione di stato dei processi
    - passivate(),hold(): l'attuale processo ATTIVO diventa PASSIVO e viene rimosso dal calendar. Il primo processo in testa al calendar diventa il processo attivo.
    - wait(DMT): il processo x ATTIVO diventa PLANNED al Model-Time attuale + DeltaModelTime (DMT). Il primo processo in testa al calendar diventa il processo attivo.
    - activate(Y,MT): il processo x ATTIVO cambia lo stato del processo Y da PASSIVO a PLANNED(tempo MT). Il processo x rimane ATTIVO.
    - cancel(Y): come activate, ma x ATTIVO cambia Y da PLANNED a PASSIVO.

37

Introduzione alla Simulazione

- Simulazione per **scheduling di eventi**
  - non ci sono processi, ma esecuzione sequenziale di "gestori di eventi"
  - Si avanza il tempo simulato al tempo T dell'evento successivo (di solito il termine o l'inizio di un'attività) e si eseguono tutti gli eventi di T
  - Simulatore = Lista ordinata di eventi e scheduler di eventi
  - Il termine di un'attività coincide con la nuova allocazione di risorse rilasciate tra le entità in attesa e con lo scheduling di nuove attività causalmente determinate

38

Introduzione alla Simulazione

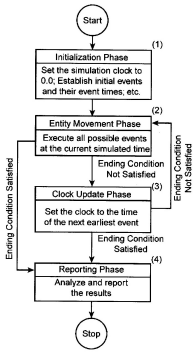
– **scheduler** di eventi
  - mantiene struttura di lista ordinata per tempo simulato (multi-linked) di eventi futuri (schedulati, cancellati, rinviati, bloccati).
  - Gestisce l'avanzamento del tempo simulato
    - event-driven: clock=(tempo del prossimo evento il lista)
    - unit-time: clock= clock+Delta... Sono accaduti eventi? Struttura dati Evento=(time, puntatore al codice della routine di evento)
  - La routine di evento aggiorna le var. di stato e la lista di eventi (inserisce, cancella, o rinvia eventi)

39

Introduzione alla Simulazione

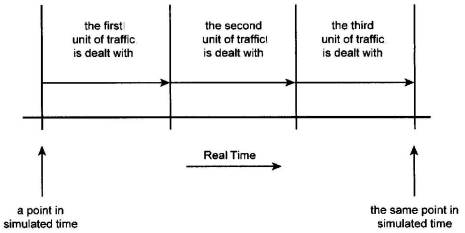
Flusso di esecuzione di un RUN di simulazione DES:

Ciclo tra EMP e CUP



Introduzione alla Simulazione

(Simulated) Model-time e (Real) WallClock-Time



41

Introduzione alla Simulazione

– Entità: possono essere in 5 stati possibili
  - Attivo
    - solo un'entità alla volta (quella in esecuzione al wall clock time attuale)
  - Ready
    - durante una EMP ci possono essere più entità pronte a "muoversi" al tempo T, che diventeranno ATTIVE una alla volta
  - Time-delayed
    - entità che diventeranno di nuovo READY a un tempo futuro già determinato (per le quali la prossima attività è un evento nel futuro)

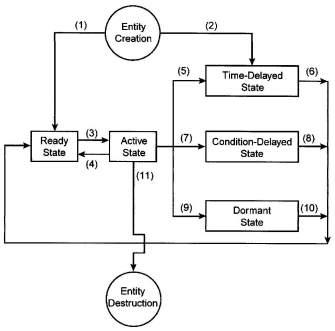
42

Introduzione alla Simulazione

– Entità: possono essere in 5 stati possibili
  • Condition-delayed
    – entità che diventeranno di nuovo READY a un tempo futuro non determinato, che dipende dal verificarsi di una certa condizione (es. il rilascio di una risorsa)
  • Dormienti
    – sono entità per le quali non è prevista una condizione di risveglio automatico, ma che possono essere rese attive dalla logica prevista dal creatore del modello, a seconda dei casi.

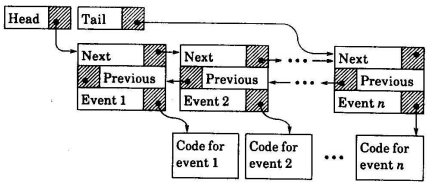
Introduzione alla Simulazione

Ciclo dei possibili stati di un entità in un RUN di simulazione DES



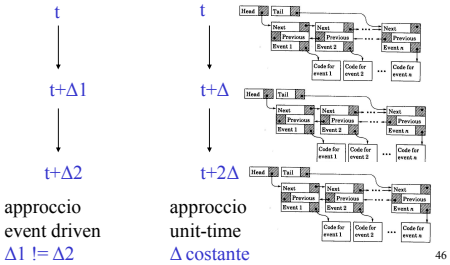
Introduzione alla Simulazione

Es. di entry di una lista di gestione eventi



Introduzione alla Simulazione

Es. di lista di gestione eventi futuri (FEL)



approccio event driven
Δ1 != Δ2

approccio unit-time
Δ costante

Introduzione alla Simulazione

– Strutture dati per il controllo delle entità
– tipicamente per ogni stato delle entità esiste una lista relativa
  • lista (singolo elemento) Active Entity (AEL)
    – tale entità viene eseguita fino a che non cambia stato o si elimina dal sistema
  • Current Event List (CEL)
    – contiene tutte le entità ATTIVE (pronte all'esecuzione)
    – contiene le possibili entità "clonate" dalla entità attiva
    – possibili criteri di priorità per diventare ATTIVE

Introduzione alla Simulazione

  • Future Event List (FEL) (vedi figura Jain)
    – lista di eventi del tempo T1, T2...
    – contiene le entità in stato TIME-DELAYED
    – ordinata per tempo simulato di "movimento" dell'entità
    – quando termina la EMP, il CUP avanza al tempo simulato della prima entry della FEL, poi tutta la lista attuale di FEL viene spostata nella CEL (diventano entità READY), e la EMP riparte a "eseguire" le entità...
  • Delay List (condition-delayed entities)
    – related waiting (se conosco le cause o eventi che possono liberare) o polled waiting (controllo ciclico condizioni)
    – entità "liberate" vanno aggiunte alla CEL (ready entity)

Introduzione alla Simulazione

- User Managed List
  - lista di entità in stato "dormiente"
  - la gestione e la semantica è a carico del creatore del modello

Introduzione alla Simulazione

- Elementi di controllo
  - sono elementi che agiscono da trigger di eventi e rilevano combinazioni di condizioni di stato
  - es. se (coda piena) e Tsimulazione>300 allora...

Introduzione alla Simulazione

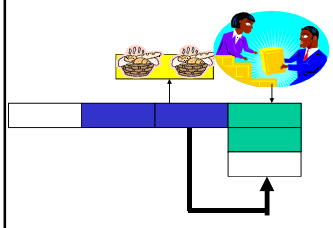
- es. Problemi di implementazione e gestione
  - risorsa rilascia e tenta di acquisire subito la risorsa
    - 1) allocare la risorsa prima di rendere l'entità un contendente?
    - 2) rendere l'entità un contendente prima di allocare la risorsa?
    - 3) risorsa ri-catturata immediatamente?
  - Il primo della coda viene sempre ritardato
    - A è in coda prima di B per la risorsa R. A chiede 2 R mentre B chiede 1 R... una sola R diventa disponibile: cosa fare?

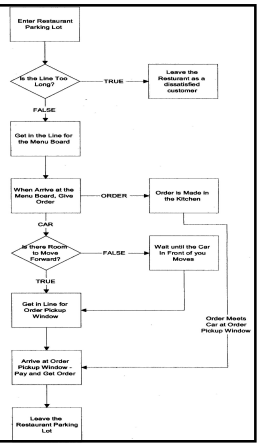
Introduzione alla Simulazione

- es. Problemi di implementazione e gestione
  - Yielding control
    - la Active entity vuole dare controllo ad altre Ready entities, poi vuole ri-acquisire il controllo, prima che il CUP avanzi il tempo simulato: come si fa?
    - es. apro porta per 10 entità e poi riblocco la porta...
  - Condizioni di attesa che coinvolgono il clock
    - es. wait until buffer empty OR clock = 1000 (non >=)
    - tipicamente si clonano le entità (dummy entity) e poi si gestiscono in coppia (la prima che si libera elimina l'altra)
  - Condizioni di attesa che coinvolgono lo stato delle risorse
    - es. A wait finchè R viene rilasciata da chi la detiene ora (B)...
    - occorre garantire che la entità A sia sbloccata PRIMA dell'eventuale cattura della risorsa R da parte di altre entità che erano in attesa

Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

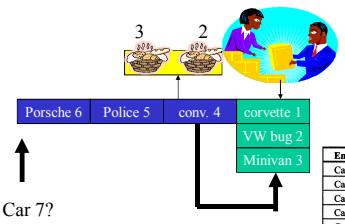




Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Time = 12:00:00



- Car 1 (the Corvette) is at the Order Pickup Window receiving its order.
- Cars 2 (the convertible VW Bug) and 3 (the minivan) are in line waiting for the Order Pickup Window.
- Car 4 (the 2-seat convertible) is ordering at the Menu Board.
- Cars 5 (the Police Car) and 6 (the Porsche) are waiting in line for the Menu Board.
- The Order for Car 2 is being cooked in the Kitchen.
- The Order for Car 3 is waiting in line for the Kitchen.
- Car 7 (of undetermined type) is scheduled to arrive at the restaurant in the future.

Table 2: Entity Attributes at Noon

Entity	StartTime	OrderValue
Car(1)	11:54:20 AM	$ 10
Car(2)	11:55:50 AM	$ 6
Car(3)	11:57:10 AM	$ 4
Car(4)	11:58:20 AM	$ 14
Car(5)	11:59:30 AM	$ 14
Car(6)	12:00:00 PM	$ 10
Car(7)	---	---

Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Time = 12:00:00

Table 1: The Calendar at Noon

Entity	Event	Event Time
Car(7)	Arrive at Restaurant	12:00:20 PM
Car(1)	Order Pickup Window Complete	12:00:40 PM
Ordr(2)	Kitchen Complete	12:00:56 PM
Car(4)	Menu Board Complete	12:01:10 PM

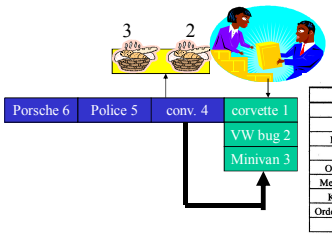


Table 3: Statistics at Noon

Statistic	Value	Time/Obs
Revenue	$ 504	
Lost Revenue	$ 74	
MenuBoard Utilization	0.9984	1:00:00
Kitchen Utilization	0.7006	1:00:00
OrderWindow Utilization	0.9678	1:00:00
MenuBoard Waiting in Line	1.2306	1:00:00
Kitchen Waiting in Line	0.0822	1:00:00
OrderWindow Waiting in Line	1.0311	1:00:00
Time In System	5.5969	44

Introduzione alla Simulazione

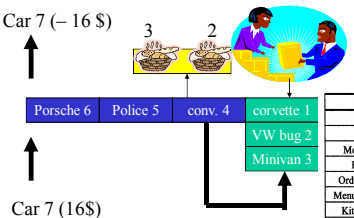
Un esempio di sistema:
Drive-Through Fast Food

Time = 12:00:20

Table 1: The Calendar at Noon

Entity	Event	Event Time
Car(7)	Arrive at Restaurant	12:00:20 PM
Car(1)	Order Pickup Window Complete	12:00:40 PM
Ordr(2)	Kitchen Complete	12:00:56 PM
Car(4)	Menu Board Complete	12:01:10 PM

Car 7 (– 16 $)



Car 7 (16$)

Table 5: Statistics at 12:00:20 PM

Statistic	Value	Time/Obs
Revenue	$ 504	
Lost Revenue	$ 90	
MenuBoard Utilization	0.998409	1:00:20
Kitchen Utilization	0.702254	1:00:20
OrderWindow Utilization	0.967978	1:00:20
MenuBoard Waiting in Line	1.234851	1:00:20
Kitchen Waiting in Line	0.087271	1:00:20
OrderWindow Waiting in Line	1.036453	1:00:20
Time In System	5.5969	44

Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Time = 12:00:40

Table 4: The Calendar at 12:00:20 PM

Entity	Event	Event Time
Car(1)	Order Pickup Window Complete	12:00:40 PM
Order(2)	Kitchen Complete	12:00:56 PM
Car(4)	Menu Board Complete	12:01:10 PM
Car(8)	Arrive at Restaurant	12:01:30 PM

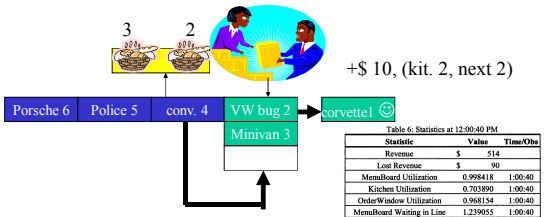


Table 6: Statistics at 12:00:40 PM

Statistic	Value	Time/Obs
Revenue	$ 514	
Lost Revenue	$ 90	
MenuBoard Utilization	0.998418	1:00:40
Kitchen Utilization	0.703890	1:00:40
OrderWindow Utilization	0.968154	1:00:40
MenuBoard Waiting in Line	1.239055	1:00:40
Kitchen Waiting in Line	0.092286	1:00:40
OrderWindow Waiting in Line	1.041747	1:00:40
Time in System	5.613265	45

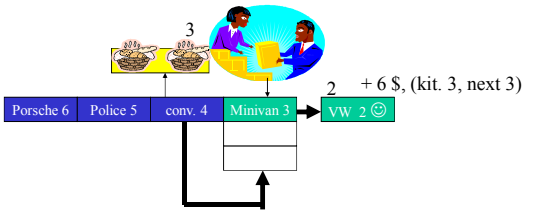
Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Time = 12:00:56

Table 4: The Calendar at 12:00:20 PM

Entity	Event	Event Time
Car(1)	Order Pickup Window Complete	12:00:40 PM
Order(2)	Kitchen Complete	12:00:56 PM
Car(4)	Menu Board Complete	12:01:10 PM
Car(8)	Arrive at Restaurant	12:01:30 PM



58

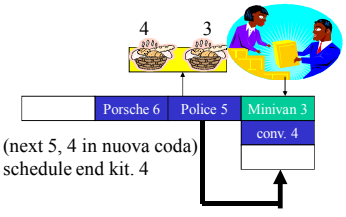
Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Time = 12:01:10

Table 4: The Calendar at 12:00:20 PM

Entity	Event	Event Time
Car(1)	Order Pickup Window Complete	12:00:40 PM
Order(2)	Kitchen Complete	12:00:56 PM
Car(4)	Menu Board Complete	12:01:10 PM
Car(8)	Arrive at Restaurant	12:01:30 PM



(next 5, 4 in nuova coda)
schedule end kit. 4

59

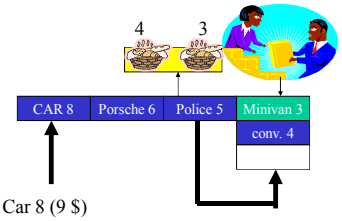
Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Time = 12:01:30

Table 4: The Calendar at 12:00:20 PM

Entity	Event	Event Time
Car(1)	Order Pickup Window Complete	12:00:40 PM
Order(2)	Kitchen Complete	12:00:56 PM
Car(4)	Menu Board Complete	12:01:10 PM
Car(8)	Arrive at Restaurant	12:01:30 PM



Car 8 (9 $)

60

Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Time = 13:00:00

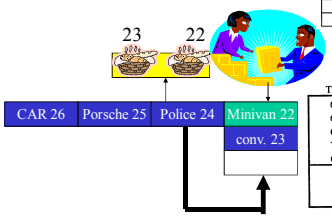


Table 7: Statistics at End of Simulation

Statistic	Value	Time/Obs
Revenue	$ 1,638	
Lost Revenue	$ 170	
MenuBoard Utilization	0.9724	3:00:00
Kitchen Utilization	0.7250	3:00:00
OrderWindow Utilization	0.9846	3:00:00
MenuBoard Waiting in Line	0.8785	3:00:00
Kitchen Waiting in Line	0.133	3:00:00
OrderWindow Waiting in Line	1.1567	3:00:00
Time In System	5.0487	140

Table 9: 30 Iterations of Throwing Dice 100 Times

6.72	6.95	6.78	7.14	6.62	6.81
6.75	7.17	6.62	7.3	6.92	7.04
6.79	7.13	7.17	7.12	6.82	7.29
7.13	7.26	7.19	6.52	6.8	7.3
6.95	6.96	6.78	7.17	7.13	6.68

Average: 6.967
Standard Deviation: 0.22992
95% Confidence Interval: [6.8847,7.0493]

61

Introduzione alla Simulazione

Un esempio di sistema:
Drive-Through Fast Food

Dopo 30 RUN....

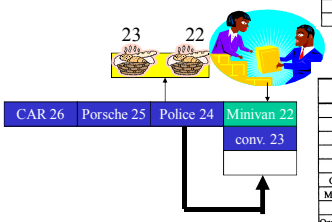


Table 7: Statistics at End of Simulation

Statistic	Value	Time/Obs
Revenue	$ 1,638	
Lost Revenue	$ 170	
MenuBoard Utilization	0.9724	3:00:00
Kitchen Utilization	0.7250	3:00:00
OrderWindow Utilization	0.9846	3:00:00
MenuBoard Waiting in Line	0.8785	3:00:00
Kitchen Waiting in Line	0.133	3:00:00
OrderWindow Waiting in Line	1.1567	3:00:00
Time In System	5.0487	140

Table 10: Confidence Intervals after 30 Iterations

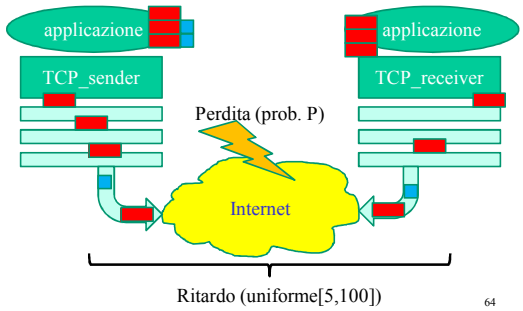
Statistic	Value	CI Lower	CI Upper
Revenue	$1,649	$1,627	$1,670
Lost Revenue	$155	$135	$176
Lost Customers	12.83	10.99	14.67
MenuBoard Utilization	0.97	0.96	0.98
Kitchen Utilization	0.73	0.72	0.74
OrderWindow Utilization	0.99	0.99	0.99
MenuBoard Waiting in Line	0.95	0.89	1.01
Kitchen Waiting in Line	0.12	0.11	0.13
OrderWindow Waiting in Line	1.23	1.18	1.28
Time In System	5.31	5.18	5.44

Esempio: modello trasmissione TCP

- Simuliamo un sistema con un sender e un receiver al livello trasporto usando il protocollo TCP, e immaginiamo tutta la rete sottostante come un black-box model caratterizzato da un tempo medio di trasmissione end-to-end e una probabilità di perdita dei dati *p*
- Vediamo i 2 approcci per la simulazione:
  - orientata ai processi
  - ad eventi

63

Esempio: modello trasmissione TCP



64

Interazione tra processi

- Simuliamo un sistema con un sender e un receiver al livello trasporto usando il protocollo TCP, e immaginiamo tutta la rete sottostante come un black-box model caratterizzato da un tempo medio di trasmissione end-to-end e una probabilità di perdita dei dati *p*
- Vediamo i 2 approcci per la simulazione:
  - orientata ai processi
  - ad eventi

65

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - es. linguaggi SIMULA, C-SIM, ecc.
  - implementazione di processi pseudo-paralleli
  - reentrant-code (un programma eseguibile da più istanze di processi "diversi", con stato locale), threads...
  - processi eseguiti in MODEL-time (non real time) che può essere "accelerato o rallentato".

66

Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - Vita di un processo nel modello
    - fase "attiva" di esecuzione (richiede 0 model-time)
      - aggiornamento e gestione stato locale e globale
    - fase "passiva" di esecuzione (in >0 model-time discreto)
      - es. attesa per risorsa o servizio ricevuto
  - ordine di esecuzione dei processi governato dal "calendar" (lista di puntatori a processi ordinati per model-time di "risveglio") e gestito attraverso un processo-scheduler (kernel)

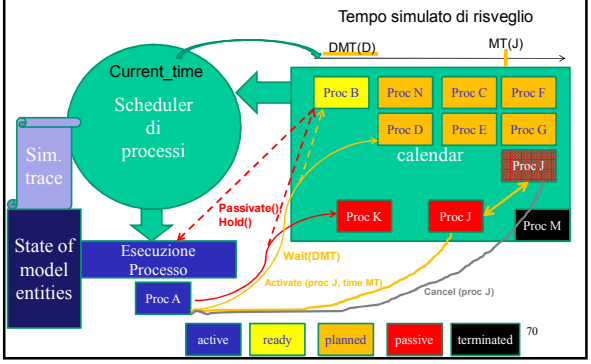
Introduzione alla Simulazione

- Simulazione per **Interazione tra Processi**
  - Possibili stati dei processi
    - Passive: il processo è definito, ma non ha nessuna entry relativa nel calendar, cioè non si sa se e quando si "risveglia"
    - Planned: il processo è definito, e ha una entry relativa nel calendar, cioè si sa quando si "risveglia"
    - Active: il processo è eseguito (la sua entry è in testa al calendar)
    - Terminated: il processo è terminato (non può essere riavviato) e i suoi dati sono ancora mantenuti in memoria
  - le transizioni di stato avvengono attraverso l'esecuzione del processo stesso o da parte di terzi

Introduzione alla Simulazione

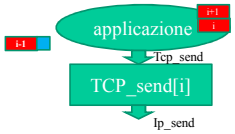
- Simulazione per **Interazione tra Processi**
  - Primitive di transizione di stato dei processi
    - passivate(),hold(): l'attuale processo ATTIVO diventa PASSIVO e viene rimosso dal calendar. Il primo processo in testa al calendar diventa il processo attivo.
    - wait(DMT): il processo x ATTIVO diventa PLANNED al Model-Time attuale + DeltaModelTime (DMT). Il primo processo in testa al calendar diventa il processo attivo.
    - activate(Y,MT): il processo x ATTIVO cambia lo stato del processo Y da PASSIVO a PLANNED(tempo MT). Il processo x rimane ATTIVO.
    - cancel(Y): come activate, ma x ATTIVO cambia Y da PLANNED a PASSIVO.

Scheduler e calendar



TCP (processi)

```
Tcp_send[i]
{
 sent[i] := false
 while(sent=false)
 {
  IP_Send(Packet, i)
  new Timer[i]
  activate(Timer[i](t), now)
  new Tcp_receive()
  activate(Tcp_receive(), now)
  passivate()
  \\ svegliato o da timeout o da ricezione ack da livello IP
  if (timeout == false) \\ se svegliato da ack
   {
   activate(Tx_buffer_send(i+1), now)
   sent[i] :=true
   }
 }
}
```



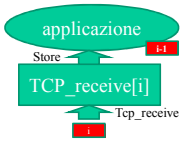

```
IP_Send[i]
{
 received[i] := false
 if RND()>p   \\ pacchetto arrivato
 {
  received[i] := true
  delay[i] := uniform(5,100)
  Wait(delay[i])
  activate(TCP_receive[i])
 }
}
```


TCP (processi)

```
Tcp_receive[i]
{
 passivate()
 // svegliato da passaggio pacchetto da livello IP a Tcp o Timeout//

 if timeout[i] == false \\ se svegliato da IP sottostante
 {
    if (OK(Packet, i)) \\ se pacchetto ricevuto ok
      {
      activate(IP_Send(Ack, i))   \\ invia ack
      activate(Store(Packet, i))   \\ bufferizza pacchetto per applicazione
      }
 terminate()
}
```

```
Timer[i](t)
{
 timeout[i] := false
 wait(t)          \\ t := TIMEOUT
 if (ack_received[i] == false)
  {
  timeout[i] := true
  activate(Tcp_receive[i], now)
  activate(TCP_send[i], now)
  }
 terminate()
}
```



Introduzione alla Simulazione

- Simulazione per **scheduling di eventi**
  - non ci sono processi, ma esecuzione sequenziale di "gestori di eventi"
  - Si avanza il tempo simulato al tempo T dell'evento successivo (di solito il termine o l'inizio di un'attività) e si eseguono tutti gli eventi di T
  - Simulatore = Lista ordinata di eventi e scheduler di eventi
  - Il termine di un'attività coincide con la nuova allocazione di risorse rilasciate tra le entità in attesa e con lo scheduling di nuove attività causalmente determinate

73

Introduzione alla Simulazione

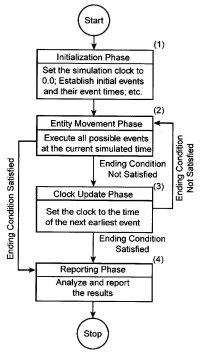
- **scheduler** di eventi
  - mantiene struttura di lista ordinata per tempo simulato (multi-linked) di eventi futuri (schedulati, cancellati, rinviati, bloccati).
  - Gestisce l'avanzamento del tempo simulato
    - event-driven: clock=(tempo del prossimo evento il lista)
    - unit-time: clock= clock+Delta... Sono accaduti eventi? Struttura dati Evento=(time, puntatore al codice della routine di evento)
  - La routine di evento aggiorna le var. di stato e la lista di eventi (inserisce, cancella, o rinvia eventi)

74

Introduzione alla Simulazione

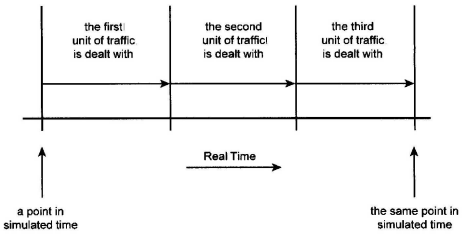
Flusso di esecuzione di un RUN di simulazione DES:

Ciclo tra EMP e CUP



Introduzione alla Simulazione

(Simulated) Model-time e (Real) WallClock-Time



76

Introduzione alla Simulazione

– Entità: possono essere in 5 stati possibili
  - Attivo
    - solo un'entità alla volta (quella in esecuzione al wall clock time attuale)
  - Ready
    - durante una EMP ci possono essere più entità pronte a "muoversi" al tempo T, che diventeranno ATTIVE una alla volta
  - Time-delayed
    - entità che diventeranno di nuovo READY a un tempo futuro già determinato (per le quali la prossima attività è un evento nel futuro)

77

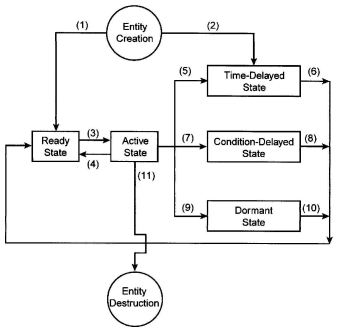
Introduzione alla Simulazione

– Entità: possono essere in 5 stati possibili
  - Condition-delayed
    - entità che diventeranno di nuovo READY a un tempo futuro non determinato, che dipende dal verificarsi di una certa condizione (es. il rilascio di una risorsa)
  - Dormienti
    - sono entità per le quali non è prevista una condizione di risveglio automatico, ma che possono essere rese attive dalla logica prevista dal creatore del modello, a seconda dei casi.

78

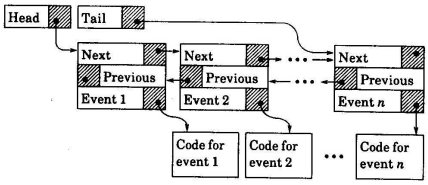
Introduzione alla Simulazione

Ciclo dei possibili stati di un entità in un RUN di simulazione DES



Introduzione alla Simulazione

Es. di entry di una lista di gestione eventi



Introduzione alla Simulazione

Es. di lista di gestione eventi futuri (FEL)

t

t+Δ1

t+Δ2

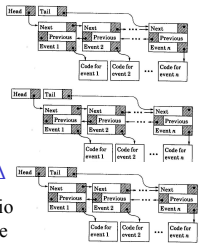
approccio event driven
Δ1 != Δ2

t

t+Δ

t+2Δ

approccio unit-time
Δ costante



Introduzione alla Simulazione

- Strutture dati per il controllo delle entità
- tipicamente per ogni stato delle entità esiste una lista relativa
  - lista (singolo elemento) Active Entity (AEL)
    - tale entità viene eseguita fino a che non cambia stato o si elimina dal sistema
  - Current Event List (CEL)
    - contiene tutte le entità ATTIVE (pronte all'esecuzione)
    - contiene le possibili entità "clonate" dalla entità attiva
    - possibili criteri di priorità per diventare ATTIVE

Introduzione alla Simulazione

- Future Event List (FEL) (vedi figura Jain)
  - lista di eventi del tempo T1, T2...
  - contiene le entità in stato TIME-DELAYED
  - ordinata per tempo simulato di "movimento" dell'entità
  - quando termina la EMP, il CUP avanza al tempo simulato della prima entry della FEL, poi tutta la lista attuale di FEL viene spostata nella CEL (diventano entità READY), e la EMP riparte a "eseguire" le entità...
- Delay List (condition-delayed entities)
  - related waiting (se conosco le cause o eventi che possono liberare) o polled waiting (controllo ciclico condizioni)
  - entità "liberate" vanno aggiunte alla CEL (ready entity)

Introduzione alla Simulazione

- User Managed List
  - lista di entità in stato "dormiente"
  - la gestione e la semantica è a carico del creatore del modello

Introduzione alla Simulazione

– Elementi di controllo
  - sono elementi che agiscono da trigger di eventi e rilevano combinazioni di condizioni di stato
  - es. se (coda piena) e Tsimulazione>300 allora...

85

Introduzione alla Simulazione

– es. Problemi di implementazione e gestione
  - risorsa rilascia e tenta di acquisire subito la risorsa
    – 1) allocare la risorsa prima di rendere l'entità un contendente?
    – 2) rendere l'entità un contendente prima di allocare la risorsa?
    – 3) risorsa ri-catturata immediatamente?
  - Il primo della coda viene sempre ritardato
    – A è in coda prima di B per la risorsa R. A chiede 2 R mentre B chiede 1 R... una sola R diventa disponibile: cosa fare?

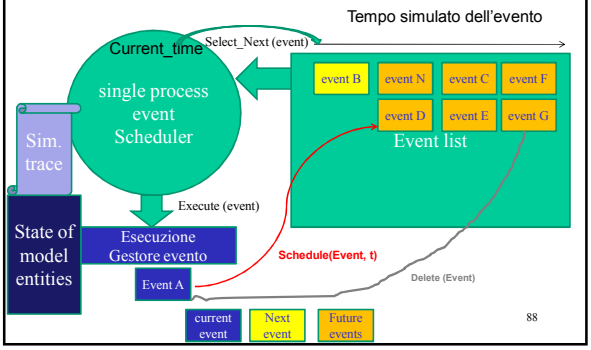
86

Introduzione alla Simulazione

– es. Problemi di implementazione e gestione
  - Yielding control
    – la Active entity vuole dare controllo ad altre Ready entities, poi vuole ri-acquisire il controllo, prima che il CUP avanzi il tempo simulato: come si fa?
    – es. apro porta per 10 entità e poi riblocco la porta...
  - Condizioni di attesa che coinvolgono il clock
    – es. wait until buffer empty OR clock = 1000 (non >=)
    – tipicamente si clonano le entità (dummy entity) e poi si gestiscono in coppia (la prima che si libera elimina l'altra)
  - Condizioni di attesa che coinvolgono lo stato delle risorse
    – es. A wait finchè R viene rilasciata da chi la detiene ora (B)...
    – occorre garantire che la entità A sia sbloccata PRIMA dell'eventuale cattura della risorsa R da parte di altre entità che erano in attesa

87

Scheduler and event list



88

TCP (eventi)

```
Tcp_send[i]
{
  schedule(IP_Send_packet(Packet, i), NOW)
  schedule(Timer_expired(i), NOW+TIMEOUT))
}

Tcp_timeout
{
  schedule(Tcp_send[i]...)
}

Timer_expired(i)
{
Tcp_timeout(i)
}

Tcp_receive \\pacchetto arrivato al lato ricevente TCP
{
  schedule(IP_Send_ack(i), NOW)   \\ invia ack
  schedule(Store(Packet, i), NOW)   \\ bufferizza pacchetto per applicazione
}

Tcp_receive _ack \\ ack arrivato al lato trasmittente TCP
{
  delete(Timer_expired[i])
  schedule(TCP_send(i+1), NOW)
}
```

```
IP_Send_packet[i]
{
 received[i] := false
 if RND()>p   \\ pacchetto arrivato
  {
   received[i] := true
   delay[i] := uniform(5,100)
   schedule((TCP_receive[i]), NOW+delay[i])
  }
}

IP_Send_ack[i]
{
 received_ack[i] := false
 if RND()>p   \\ ack arrivato
  {
   received_ack[i] := true
   delay_ack[i] := uniform(5,100)
   schedule((TCP_receive_ack[i]),
NOW+delay[i])
  }
}
```

89