Modellazione discreta per eventi, esempi

Simulazione & Logistica, I modulo
Esercitazione n. 4

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

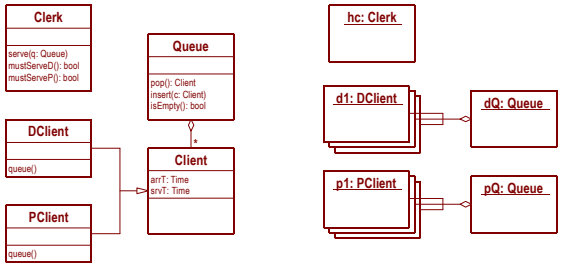
Contenuti

- L'impiegato sempre tormentato (a eventi)
- Modellazione e simulazione per eventi
  - Simulatori e modelli
  - Un modello tutto in uno
    - modellazione per eventi classica
    - compatto, doAction() diretta, rischi di manutenibilità
  - Modellazione con GS DSlibs
    - relazioni fra doAction() e handle()
    - OO, ereditarietà e polimorfismo
- Esecuzione di una simulazione

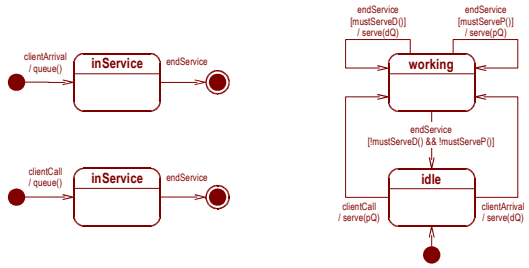
L'impiegato tormentato (a eventi)

- Un servizio al pubblico
  - I clienti arrivano di persona
  - Oppure telefonano
  - L'impiegato serve i clienti allo sportello o al telefono
  - I clienti che arrivano allo sportello aspettano (in ordine)
  - Il centralino ordina le telefonate in attesa
  - I clienti allo sportello hanno la priorità
  - I tempi di servizio dipendono dai clienti
- Modellazione a eventi
  - Costruendo un unico diagramma a stati (classica)
  - Mantenendo un diagramma per ogni entità attiva (GS DSlibs)

Classi e oggetti

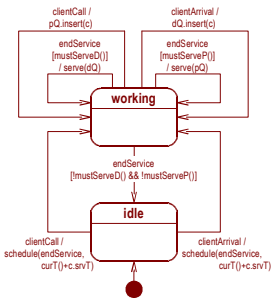


Macchine a stati



Modellazione classica: tutto in uno

- Diagramma del sistema
- Semplificazioni (?)
  - Clienti derivati inutili
  - Metodo queue() inutile
  - Clienti come entità passive
  - Se non impegnato, l'impiegato serve direttamente
- Metodo doAction()
  - Ricavabile dal diagramma
  - Come il metodo handle(), ma partizionato per eventi



Problemi

- Codice organizzato per eventi
  - Il metodo doAction() è specifico di un evento
  - Il comportamento delle entità è prima riunito ...
  - ... ma poi diviso per eventi
  - Difficoltà di intervento e possibilità di errori
- Il codice interessa? Possiamo generarlo
- Un solo diagramma, più complicato
  - Si perde la visione di sistema
  - Azioni diversificate, il codice risulta ottimizzato, ma la comprensione del diagramma è resa più difficile
  - Specialmente quando i sistemi sono complessi

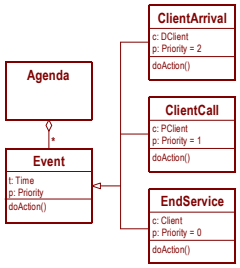
Modellazione con GS DSlibs

- Mantenere il modello originale
  - Diagrammi delle macchine a stati per ogni entità attiva
  - Sistema, non blocco unico
- Metodi doAction() e handle()
  - Propagazione dell'evento alle entità interessate, gestione da parte delle entità stesse
  - Disaccoppiamento della gestione, rispetto alle entità
  - Coesione di tutto il comportamento
- OO: ereditarietà, polimorfismo (m. virtuali)
- Costo: ripetizione delle condizioni sugli eventi

Classi e metodi

- Specializzare la classe evento
  - Riferimento al cliente
  - Definire doAction()
- Metodi

```
ClientArrival::doAction() {
    c.handle(this);
    hc.handle(this);
}
ClientCall::doAction() {
    c.handle(this);
    hc.handle(this);
}
EndService::doAction() {
    hc.handle(this);
    c.handle(this);
}
```



Esecuzione di una simulazione

- Eventi
  - Caricati nell'agenda prima di lanciare la simulazione
  - *Type* *arrT* *srvT*

ClientArrival	5	10
ClientCall	10	10
ClientArrival	15	10
ClientCall	15	10
ClientCall	20	10
ClientArrival	20	10
ClientCall	60	10
ClientArrival	85	10

- Tempo di simulazione: 60
- Cosa succede?

Giovanni A. Cignoni - SLo1: Simulazione - www.di.unipi.it/~giovanni/