Università degli Studi di Padova

Interazione con OOP

SCD

Anno accademico 2006/7
Corso di Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it

Università degli Studi di Padova

Inheritance anomaly – 1

- **L'interazione tra concorrenza e orientazione a oggetti (OO) produce effetti indesiderabili**
  - **Metodi che contengono codice di sincronizzazione (per mutua esclusione, sospensione, risveglio) possono essere non direttamente ereditabili senza preventiva analisi dettagliata e/o modifiche della classe sorgente**
    - S. Matsuoka and A. Yonezawa
      Analysis of inheritance anomaly in object-oriented concurrent programming languages
      In: G. Agha, P. Wegner, and A. Yonezawa (editors), **Research directions in concurrent object-oriented programming**, pp. 107-150. MIT Press, 1993
  - **Tale eventualità viola il principio di incapsulazione**

Università degli Studi di Padova

Inheritance anomaly – 2

- **Forme canoniche di sincronizzazione**
  - **Esecuzione condizionale**
    - Le operazioni effettuabili possono dipendere da
      - **Stato interno** della risorsa
      - **Storia** di esecuzione
      - **Parametri** della richiesta
    - La natura della risorsa può necessitare accesso esclusivo
  - **Controllo di ordinamento**
    - Le richieste non immediatamente soddisfacibili possono venire accodate

Università degli Studi di Padova

Inheritance anomaly – 3

- **L'anomalia ha luogo tipicamente all'introduzione di**
  - **Diversi criteri di ammissibilità**
    - La sottoclasse può aver bisogno di introdurre una corrispondenza diversa tra stati interni e operazioni ammissibili
  - **Modifica degli stati interni**
    - La sottoclasse può aver bisogno di introdurre nuovi stati interni, ignoti e non trattati dalla classe sorgente
  - **Modifica delle dipendenze dalla storia d'esecuzione**
    - La sottoclasse può aver bisogno di imporre nuovi e diversi vincoli sulla ammissibilità di particolari richieste
- **Ciò richiede conoscenza precisa della logica originaria della classe sorgente e la possibilità di modificarla**

Università degli Studi di Padova

Esempio

- ***Bounded buffer*** **con `get()` e `put()`**
  - Non vuoto → `get()`
  - Non pieno → `put()`
  - **Diversi criteri di ammissibilità**
    - Prelievo simultaneo di 2 articoli: aggiunta di metodo `get2()` e oscuramento dei metodi ereditati
      - Stato non originario → modifica della classe sorgente
  - **Modifica degli stati**
    - Risorsa inaccessibile → ereditarietà multipla da `lock()`
    - Risorsa accessibile → ereditarietà multipla da `unlock()`
      - Stato non originario → modifica della classe sorgente
  - **Modifica delle dipendenze della storia**
    - Preleva 1 articolo ogni N immissioni: aggiunta di metodo `Nget()` e oscuramento dei metodi ereditati
      - Stato non originario → modifica della classe sorgente