



Un modello concreto

SCD

Anno accademico 2015/16
Sistemi Concorrenti e Distribuiti

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

Laurea Magistrale in Informatica, Università di Padova

1/19



Un modello concorrente concreto

Forma sintattica – 2

task type Controller;

task type Agent(Param : Integer);

task type Cashier_Attendant(Post : Post_Number := 1);

task type Yellow_Pages is
 entry Directory_Enquiry(
 Entity : in Name;
 Place : in Address;
 Number : out Telephone_Number);
end Yellow_Pages;

← Parti discriminante, pubblica e privata vuote

← Parte discriminante definita

← Parte discriminante definita, con valore iniziale predefinito modificabile (default)

← Parte pubblica definita con un punto di accesso per sincronizzazione

Laurea Magistrale in Informatica, Università di Padova

3/19



Un modello concorrente concreto

Forma sintattica – 1

❑ Tipo task come modello di processo nella tassonomia vista a lezione

○ Prima dichiarato poi istanziato

○ La dichiarazione include parte contrattuale (specification) e parte realizzativa (body)

○ La specifica contiene

- Nome del tipo
- Parte (opzionale) discriminante
simile al costruttore nel passare parametri (tipi a dimensione determinabile) all'oggetto
- Parte visibile
definisce ciò che di quel processo gli altri potranno sapere (inclusi i canali di accesso), incluso il discriminante
- Parte privata
definisce aspetti realizzativi interni, invisibili al resto del sistema

Laurea Magistrale in Informatica, Università di Padova

2/19



Un modello concorrente concreto

Una dichiarazione completa – 1

task type Customer(My_Id : Positive) is
 pragma Priority(...);
end Customer;
task body Customer is
 Outcome : Boolean;
begin
 Status.Signal(Outcome);
 if Outcome then
 Service.Wait;
 else
 ... -- alternate action
 end if;
end Customer;

protected Status is
 entry Wait;
 procedure Signal(Outcome: out Boolean);
private
 In_Line : Max_In_Line := 0;
 Present : Boolean := False;
end Status;

protected Service is
 entry Wait;
 procedure Signal;
private
 Calling_For_Service : Boolean := False;
end Service;

Una specie di Monitor che poi studieremo

type Customer_Ref is access Customer;
Customer_1 : Customer_Ref :=
 new Customer(1);
Customer_2 : Customer(2);

2

1

Laurea Magistrale in Informatica, Università di Padova

4/19

Unipd - SCD 2015/16 - Sistemi Concorrenti e Distribuiti

1



Un modello concorrente concreto

Una dichiarazione completa – 2

❑ ① e ② rappresentano distinte modalità dichiarative di oggetti (istanze di) *task*

❑ Il processo dichiarato tramite ① viene attivato solo alla fine dell’elaborazione della parte dichiarativa

All’ingresso della parte esecutiva del modulo che lo dichiara

❑ ② invece crea dinamicamente un processo che viene attivato all’ingresso della parte esecutiva del modulo creatore (subito, se ② è in parte esecutiva)

Può esserlo, essendo una assegnazione e non una dichiarazione

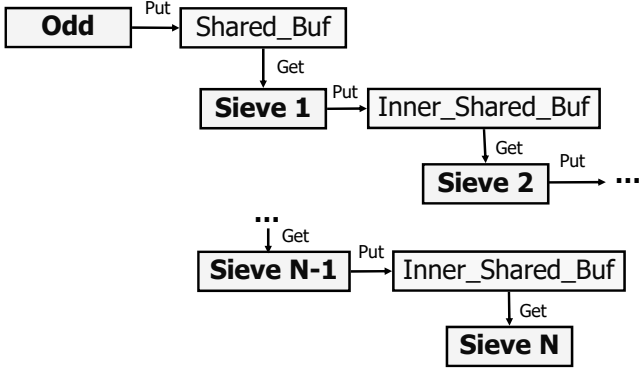
Laurea Magistrale in Informatica, Università di Padova

5/19



Un modello concorrente concreto

Un esempio completo – 2



Laurea Magistrale in Informatica, Università di Padova

7/19



Un modello concorrente concreto

Un esempio completo – 1

❑ Il crivello di Eratostene

Un processo (Odd) seleziona tutti i numeri dispari in un intervallo fissato

Un altro processo (S) rileva tra i numeri dispari ricevuti quelli non divisibili per un numero primo Prime noto

Il primo di questi numeri è certamente un nuovo numero primo (Num)

S passa Num e tutti i numeri dispari successivi non divisibili per Prime a un suo clone (Next_S)

Il clone ripete la sequenza sapendo che il primo numero che gli è pervenuto (Num) è per definizione primo

Ogni processo S opera come un filtro per un primo noto

Laurea Magistrale in Informatica, Università di Padova

6/19



Un modello concorrente concreto

Esercizio

❑ Replicare la struttura concorrente proposta nel riquadro precedente, utilizzando il linguaggio Java

❑ Raffrontare la soluzione proposta con la soluzione replicata

Laurea Magistrale in Informatica, Università di Padova

8/19

Un modello concorrente concreto

2



Un modello concorrente concreto

Stati di vita di processo – 1

❑ Un processo viene creato dall’elaborazione del modulo che ne contiene la dichiarazione

❑ L’esecuzione del processo attraversa 3 fasi distinte

- Attivazione : viene elaborata la parte dichiarativa del *task*
- Esecuzione : vengono eseguiti i comandi nella parte esecutiva (*body*) del *task*
- Finalizzazione : fine dell’esecuzione: viene eseguito il codice di finalizzazione di ogni oggetto creato dalla sua parte dichiarativa

Laurea Magistrale in Informatica, Università di Padova

9/19



Un modello concorrente concreto

Stati di vita di processo – 3



Laurea Magistrale in Informatica, Università di Padova

11/19



Un modello concorrente concreto

Stati di vita di processo – 2

```
declare
  task type T_Type;
  task A;
  B, C : T_Type;
  task body A is ... end A;
  task body T_Type is ... end T_Type;
begin
  ...
end;
```

1A è un processo anonimo e viene **creato** a questo punto

2B e C sono istanze di un processo **tipato** e vengono **create** a questo punto

3Tutti i processi creati nel corrispondente blocco dichiarativo (A,B,C, ecc.) vengono **attivati** a questo punto per poi procedere indipendentemente

4Questo blocco di comandi comincia a **eseguire** solo dopo l’attivazione dei processi

Laurea Magistrale in Informatica, Università di Padova

10/19



Un modello concorrente concreto

Stati di vita di processo – 4

❑ In un linguaggio a blocchi

- I processi possono essere dichiarati in ciascun blocco
- I blocchi possono essere annidati gerarchicamente
- Un processo è esso stesso un blocco

❑ Ciò consente di realizzare gerarchie di processi

- Il processo padre è soggetto alle stesse regole del blocco contenitore per quanto concerne attesa per l’attivazione dei processi figli

Laurea Magistrale in Informatica, Università di Padova

12/19



Un modello concorrente concreto

Stati di vita di processo – 5

❑ L’attesa per la terminazione di un processo è responsabilità della regione dichiarativa nella quale è avvenuta la sua attivazione (*master*) e della sua gerarchia superiore

- Il processo padre non è sempre *master* dei suoi processi figli, ma può esserlo un blocco (*scope*) interno a esso
- Nel caso di creazione dinamica di processo (come al punto ② di pagina 4) il *master* è determinato dalla regione dichiarativa che ne definisce il tipo
- Un *master* non ha figli ma dipendenti!

Laurea Magistrale in Informatica, Università di Padova

13/19



Un modello concorrente concreto

Esempio – 2

```
task type Dependent;
task body Dependent is ... end Dependent;

...
declare
  type Dependent_Ref is access Dependent;
  A : Dependent_Ref;
begin -- scope 1
  ...
  declare
    B : Dependent;
    C : Dependent_Ref := new Dependent;
    D : Dependent_Ref := new Dependent;
  begin -- scope 2
    ...
    A := C;
  end;
end;
...
```

Scope 1

Activation of task C
Activation of task D

Scope 2

Activation of task B
Termination of task B

Termination of task D
Termination of task C

Laurea Magistrale in Informatica, Università di Padova

15/19



Un modello concorrente concreto

Esempio – 1

```
task type Dependent;
task body Dependent is ... end Dependent;

...
declare
  type Dependent_Ref is access Dependent;
  A : Dependent_Ref;
begin -- scope 1
  ...
  declare
    B : Dependent;
    C : Dependent_Ref := new Dependent;
    D : Dependent_Ref := new Dependent;
  begin -- scope 2
    ...
    A := C;
  end;
end;
```

① è *master* di tutti i processi creati a partire dal tipo *Dependent_Ref*

Quali sono?

② è *master* solo del processo B

Perché?

Quando può terminare ②?

Quando può terminare ①?

Laurea Magistrale in Informatica, Università di Padova

14/19



Un modello concorrente concreto

Esercizio

❑ Studiare l’ordine di attivazione e di terminazione nel programma «Master» associato alla lezione corrente

❑ Studiare come l’ordinamento di esecuzione dei processi influenza l’ordine di uscita dai blocchi e la loro terminazione

Laurea Magistrale in Informatica, Università di Padova

16/19



Un modello concorrente concreto

Stati di vita di processo – 6

❑ Nel programma che realizza il crivello di Eratostene (pagg. 6-7) il modulo *package SoE* definisce vari processi (quali?) senza esserne il *master*

❑ In questo caso è il programma principale che diventa *master* mediante inclusione (with) del modulo creatore

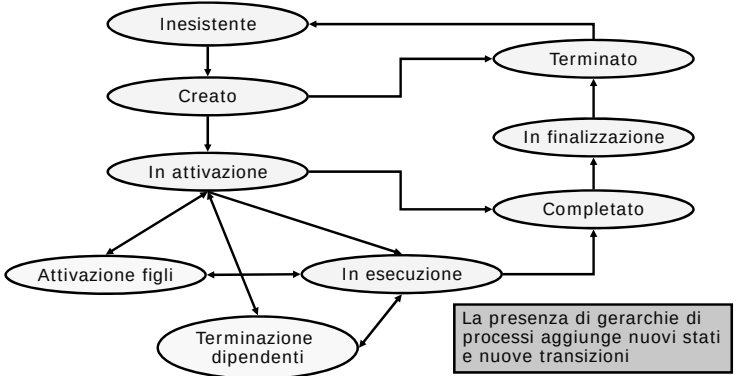
Laurea Magistrale in Informatica, Università di Padova

17/19



Un modello concorrente concreto

Stati di vita di processo – 7



La presenza di gerarchie di processi aggiunge nuovi stati e nuove transizioni

Laurea Magistrale in Informatica, Università di Padova

19/19



Un modello concorrente concreto

Esercizio

❑ Studiare la gerarchia di *master* nel programma «Erathostenes-I» associato alla lezione corrente

❑ Studiare come tale gerarchia influenza o risponda alle modalità di terminazione osservate

Laurea Magistrale in Informatica, Università di Padova

18/19

UnPD - SCD 2015/16 - Sistemi Concorrenti e Distribuiti

5