



Un modello di *rendez-vous*



Anno accademico 2012/13
Sistemi Concorrenti e Distribuiti

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

Laurea Magistrale in Informatica, Università di Padova1/20



Un modello di *rendez-vous*
Modello base – 1

- **Interazione di tipo cliente-servente**
 - Il servente dichiara i servizi che è disposto a fornire ai clienti
 - La specifica del servente dichiara i canali d'accesso (*entry*) che corrispondono a ciascuno dei servizi esposti
 - Ogni canale specifica il suo proprio protocollo di scambio parametri
 - Il cliente effettua una richiesta (*entry call*) nominando servente e canale
 - Il servente fornisce uno dei servizi richiesti esprimendone esplicitamente la propria accettazione
 - La comunicazione tra servente e cliente è sincrona e non richiede necessariamente il passaggio di dati

Laurea Magistrale in Informatica, Università di Padova2/20



Un modello di *rendez-vous*
Modello base – 2

```
task type Operator is
  entry Query (A_Person : in Name;
              An_Address : in Address;
              A_Number : out Number);
end Operator;
Ann : Operator;
```

Specifica di punto di accesso e protocollo

```
task type User;
task body User is
  My_Number : Number;
begin
  ...
  Ann.Query(
    "...", "...",
    My_Number);
end User;
```

```
task body Operator is
begin
  loop
    accept Query(A_Person : in Name;
                An_Address : in Address;
                A_Number : out Number) do
    ...
  end loop;
end Operator;
```

Invocazione Realizzazione di accettazione

Laurea Magistrale in Informatica, Università di Padova3/20



Un modello di *rendez-vous*
Modello base – 3

- **Storicamente chiamato *rendez-vous***
 - Per rappresentare il fatto che cliente e servente si incontrano su uno specifico canale nello stesso istante temporale
- **Al momento dell'incontro i parametri di modo *in* passano dal cliente al servente**
- **Il servente esegue il servizio richiesto come una normale procedura e poi restituisce i parametri di modo *out* al cliente sul canale**
- **A quel punto la sincronizzazione si interrompe e i processi riprendono la loro esecuzione concorrente**

Laurea Magistrale in Informatica, Università di Padova4/20

Un modello di rendezvous

Un modello di *rendezvous*

Modello base – 4

□ Nella forma base

- Il server si sospende in attesa di una richiesta
 - Come previsto per l'entità *Server* nel modello di concorrenza di lezione C01
- Il cliente si sospende fino alla disponibilità del servizio
- La chiamata del cliente viene posta in una coda associata al canale (*entry queue*)
- L'ordine di accodamento è normalmente FIFO ma può essere configurato diversamente
 - Per esempio su base prioritaria
 - Ma con le conseguenze che questo può comportare in termini di *starvation*

Laurea Magistrale in Informatica, Università di Padova

5/20

Un modello di *rendezvous*

Modello base – 5

Laurea Magistrale in Informatica, Università di Padova

6/20

Un modello di *rendezvous*

Esempio – 1

□ Il crivello di Eratostene rivisitato

- Una diversa realizzazione dell' algoritmo visto nella lezione precedente
- Ogni coppia di processi nella sequenza comunica tramite *rendez-vous*
- L'effetto di sincronizzazione rende superflua la mutua esclusione
 - L'accodamento FIFO sulla coda d'accesso preserva l'ordine dei valori da esaminare

Laurea Magistrale in Informatica, Università di Padova

7/20

Un modello di *rendezvous*

Esempio – 2

Laurea Magistrale in Informatica, Università di Padova

8/20

Un modello di *rendezvous*

Esercizio

- ❑ Studiare le modalità di terminazione del programma «Erathostenes-II» associato alla lezione corrente
- ❑ Spiegare come e perché esse differiscono da quelle della versione precedente dello stesso programma («Erathostenes-I»)

Laurea Magistrale in Informatica, Università di Padova 9/20

Un modello di *rendezvous*

Sincronizzazione tripartita – 1

- ❑ Il modello *rendez-vous* è
 - Sincrono rispetto alla comunicazione
 - Asimmetrico rispetto all'interfaccia
 - Bidirezionale rispetto al flusso dei dati
- ❑ Le azioni svolte dal servente durante la sincronizzazione possono coinvolgere processi terzi
 - Ciò permette di realizzare forme complesse di sincronizzazione sempre preservando separazione funzionale tra i processi coinvolti

Laurea Magistrale in Informatica, Università di Padova 10/20

Un modello di *rendezvous*

Sincronizzazione tripartita – 2

- ❑ Il coinvolgimento di processi terzi nella sincronizzazione di lato *server* ammette due forme distinte e duali
 - Annidando accettazioni
 - Modellando una **macchina a stati** in cui alcuni stati sono raggiungibili solo a partire da un dato stato iniziale
 - Invocando accettazioni nella realizzazione di una accettazione
 - Realizzando la fornitura di un **servizio composito** che racchiude il possibile contributo di più serventi che si siano ripartiti tra loro il lavoro

Laurea Magistrale in Informatica, Università di Padova 11/20

Un modello di *rendezvous*

Sincronizzazione tripartita – 3

```
graph LR; User((User)) -- "1 Service(...)" --> Controller((Controller)); Controller -- "2 Start" --> Device((Device)); Device -- "3 Read(...)" --> D[D]; Device -- "4 Finish(...)" --> Controller;
```

- Utilizzando sincronizzazione tripartita, **Controller** realizza – tramite **Device** – l'astrazione di un dispositivo **D** che produce valori solo quando richiesto
- L'entità **Device** è una macchina a stati che usa l'accettazione delle sue *entry call* come evento di transizione

Laurea Magistrale in Informatica, Università di Padova 12/20

Un modello di *rendezvous*

Sincronizzazione tripartita – 4

```

task User;
task Device;
task Controller is
  entry Service (I : out Integer);
  entry Start;
  entry Finish (K : out Integer);
end Controller;
        
```

```

task body User is
  ...
  Controller.Service (Val);
  ...
end User;
        
```

```

task body Controller is
  begin
  loop
  accept Service (I : out Integer) do
  accept Start;
  accept Finish (K : out Integer) do
    I := K;
  end Service;
  end loop;
end Controller;
        
```

```

task body Device is
  Val : Integer;
  procedure Read
    (I : out Integer);
  begin
  loop
    Controller.Start;
    Read(Val);
    Controller.Finish(Val);
  end loop;
end Device;
        
```

Laurea Magistrale in Informatica, Università di Padova
13/20

Un modello di *rendezvous*

Sincronizzazione tripartita – 5

```

graph LR
    User((User)) -- Request_Part(...) --> CS((Customer Service))
    subgraph Server
        CS -- Enquiry(...) --> Warehouse((Warehouse))
    end
    
```

- Strutturazione gerarchica con *encapsulation*
- Il servizio Request_Part(...) nasconde al cliente la necessità di eventuale approvvigionamento presso componenti incapsulati all'interno del server

Laurea Magistrale in Informatica, Università di Padova
14/20

Un modello di *rendezvous*

Sincronizzazione tripartita – 6

```

task Warehouse is
  entry Enquiry
    (Item : Part_Number;
     In_Stock : out Boolean);
end Warehouse;

task Customer_Service is
  entry Request_Part
    (Order : Part_Number;
     Part : Spare_Part;
     Order : Order_Number);
end Customer_Service;
        
```

```

task body Customer_Service is
  In_Stock : Boolean;
  ...
  begin
  loop
  ...
  accept Request_Part
    (Order : Part_Number;
     Part : Spare_Part;
     Order : Order_Number) do
  ...
  if In_Stock then
    Part := The_Part; Order := None;
  else
    Warehouse.Enquiry(Order, In_Stock);
    if In_Stock then
      -- go get part from Warehouse
      Part := The_Part; Order := Next_Order_Nr;
    end if;
  end if;
  end Request_Part;
  end loop;
end Customer_Service;
        
```

Difetto: i servizi di Warehouse sono in questo modo visibili a tutti e non solo a Customer_Service

Laurea Magistrale in Informatica, Università di Padova
15/20

Un modello di *rendezvous*

Punti d'accesso privati – 1

- Un server non deve necessariamente esporre al pubblico tutti i suoi canali
- Alcuni possono essere ristretti per motivi di incapsulazione e/o di astrazione
- La dichiarazione dei canali deve in tal caso distinguere tra pubblici e privati

Laurea Magistrale in Informatica, Università di Padova
16/20

Un modello di *rendezvous*

Punti d'accesso privati – 2

```

task User;
task Controller is
  entry Service (I : out Integer);
private
  entry Start;
  entry Finish (K : out Integer);
end Controller;
                    
```

In questo modo la visibilità ai canali privati è ristretta al solo ambito (*scope*) del processo **Controller**

```

task body User is
  ...
  Controller.Service(Val);
  ...
end User;
                    
```

```

task body Controller is
  task Device;
  task body Device is
    Val : Integer;
    procedure Read (I : out Integer) is ... ;
  begin
    loop
      Controller.Start;
      Read(Val);
      Controller.Finish(Val);
    end loop;
  end Device;
  ...
  continues in sidebox
end Controller;
                    
```

```

begin -- Controller
loop
  accept Service (I : out Integer) do
  accept Start;
  accept Finish (K : out Integer) do
    I := K;
  end Completed;
  end Service;
end loop;
end Controller;
                    
```

Laurea Magistrale in Informatica, Università di Padova
17/20

Un modello di *rendezvous*

Casi d'errore

- ❑ Una eccezione sollevata durante la sincronizzazione ne causa l'abbandono e si propaga a entrambi i partecipanti
- ❑ Emettere una richiesta d'accesso verso un processo terminato è un errore a tempo di esecuzione e solleva una eccezione nel chiamante

Laurea Magistrale in Informatica, Università di Padova
18/20

Un modello di *rendezvous*

Esercizio

- ❑ Studiare le soluzioni al problema dei filosofi a cena proposte nei programmi associati alla lezione 3 (C02)
- ❑ Verificare quali di tali soluzioni usino il *rendez-vous* per assicurare mutua esclusione nelle rispettive sezioni critiche

Laurea Magistrale in Informatica, Università di Padova
19/20

Un modello di *rendezvous*

Stati d'esecuzione di processo

Laurea Magistrale in Informatica, Università di Padova
20/20