Simulador de Eventos Discretos

Algoritmos Distribuidos

MAESTRÍA EN CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

UNIVERSIDAD AUTÓNOMA METROPOLITANA
UNIDAD IZTAPALAPA

Ejecutando a mano un AD

- **Consideremos las actividades que realizamos cuando ejecutamos a *mano* un algoritmo distribuido:**
  - **Fijamos el grafo de comunicaciones**
  - **Para cada vértice en el grafo asumimos que ejecutará el algoritmo en cuestión**
  - **Hacemos un dibujo en el que vamos poniendo, para cada proceso, los valores de sus variables y los mensajes que llegan o se van conforme avanza la ejecución**

Ejecutando a mano un AD

- Decidimos empezar con algún proceso
- Mientras dure la simulación
  - Elegimos un evento por atender:
    - Ese evento usualmente:
      - Ocasiona que un proceso consuma un mensaje,
      - cambie los valores de sus variables y,
      - en ocasiones, envíe otro(s) mensaje(s).

- Observe que en cada paso puede haber más de un evento por ejecutar y nosotros decidimos arbitrariamente uno.
- Siempre ejecutamos completamente la reacción al evento elegido ... discretizamos la ejecución marcada por envíos y recepciones.

Ejecutando a mano un AD

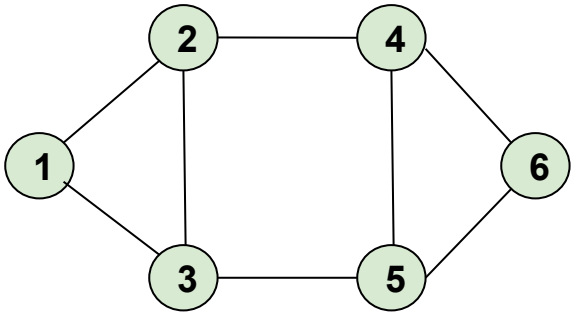
- **Lo que estamos haciendo es simulando el algoritmo sobre un modelo de SD y podemos observar aspectos de su comportamiento. Por ejemplo:**
  - **Cuántos mensajes envía**
  - **Cuánto tiempo se tarda**
- **¿Porqué no automatizar esta simulación?**

Simulando un AD

- **¿Que necesitamos?**

1. Un grafo para representar la infraestructura: por cada vértice un proceso y cada proceso con identidad única y conocimiento de sus vecinos.

El grafo:



Se representa como:

2 3

1 3 4

1 2 5

2 5 6

3 4 6

4 5

Simulando un AD

2. El algoritmo que cada proceso ejecutará

3. Una lista donde se coloquen todos los eventos por atender (normalmente una cola de mensajes)

Simulando un AD

COMIENZA

Crea la agenda

Lee la gráfica

Crea un proceso por nodo(id,vecinos) e indícale el algoritmo a simular

Genera un evento semilla

Inserta el evento en la agenda

Arranca el simulador

Mientras no esté la agenda vacía

- **Saca un evento**
- **Envíaselo al proceso que le corresponde**
- **El proceso ejecuta la reacción correspondiente**

TERMINA

Simulador de Eventos Distribuidos (SED)

Utiliza una agenda:

- **Ahí se almacenan los eventos ordenados de acuerdo con sus tiempos de atención.**
- **Se garantiza que un evento ocurre a tiempo.**

En cada ciclo de simulación, el evento fechado con el menor tiempo es removido de la agenda y se simula su tratamiento. Esto puede causar:

- **Nuevos eventos en el futuro (se agregan a la agenda) o**
- **Cancelación de eventos (se remueven de la agenda).**

El reloj se avanza entonces, hasta el tiempo del evento cuya simulación acaba de completarse.

Simulador de eventos discretos (SED)

1. **Event**: Representa un paquete de información que se intercambia entre entidades activas
2. **Model**: Representa la rutina para la atención de eventos, está basada en un modelo de máquina de estados finitos. También efectúa funciones de reporte y generación de trazas.
3. **Process**: Representa una entidad activa relacionada con un modelo.
4. **Simulator**: Representa al calendarizador de eventos así como tareas de gestión de reloj.
5. **Simulation**: Representa al administrador que establece los canales entre procesos y coordina su comunicación. También efectúa funciones de inicialización.

Diagrama de Clases

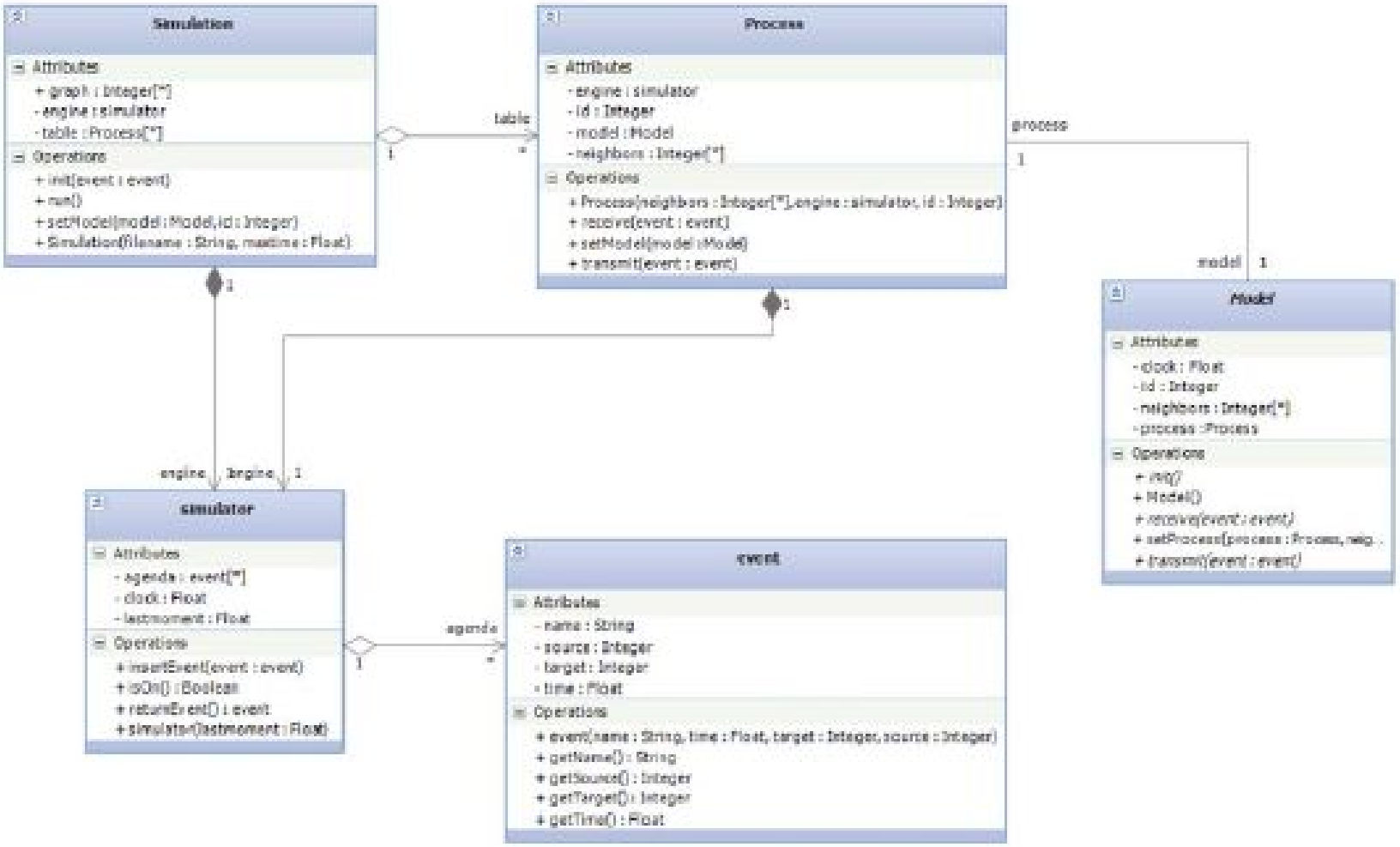


Diagrama de secuencias

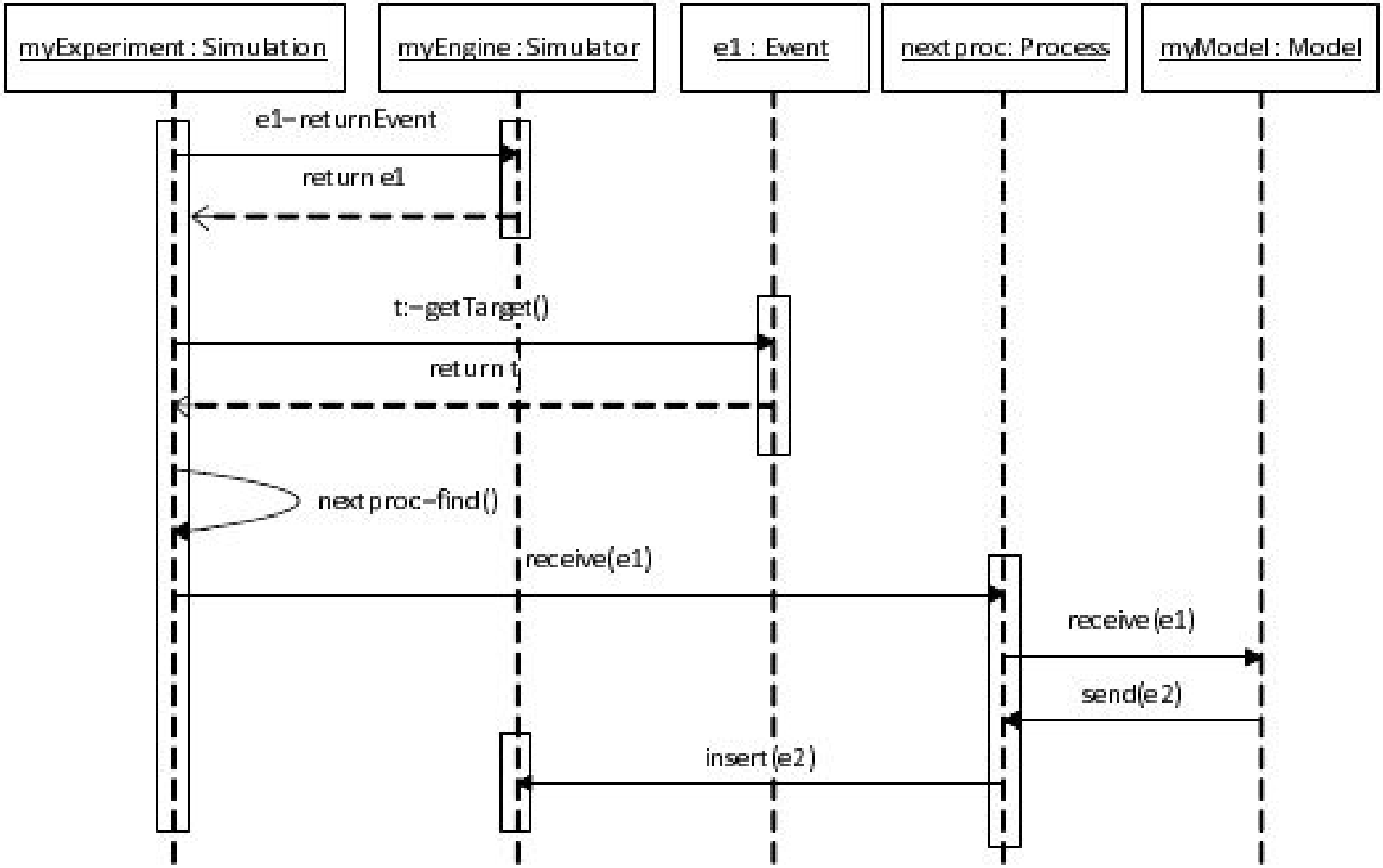


Figura A.2 Diagrama de secuencias

Ejemplo DFS Cheung

```
# Este archivo sirve de modelo para la creación de
  aplicaciones, i.e. algoritmos concretos
# Implementa la simulación del algoritmo de recorrido en
  profundidad de Cheung

import sys
from event import Event
from model import Model
from process import Process
from simulator import Simulator
from simulation import Simulation
```

Ejemplo DFS Cheung

```
class AlgorithmDFS(Model):
  # Esta clase desciende de la clase Model e implementa los
    métodos "init()" y "receive()", que en la clase madre se
    definen como abstractos

  def init(self):
    # Aquí se definen e inicializan los atributos particulares
    del algoritmo
    self.father = self.id
    self.visited = False
    self.unvisited = self.neighbors
```

Ejemplo DFS Cheung

```
def go_explore_more(self):
  if len(self.unvisited) > 0:
    to_visit = self.unvisited[0]
    self.unvisited = self.unvisited[1:len(self.unvisited)]
    newevent = Event("DESCUBRE",self.clock+1.0, to_visit,
    self.id)
    self.transmit(newevent)
  elif self.father != self.id:
    newevent=Event("REGRESA",self.clock+1.0, self.father,
    self.id)
    self.transmit(newevent)
```

Ejemplo DFS Cheung

```
def receive(self, event):
 if  event.getName() == "DESCUBRE":
   if  event.getSource() != self.id:
                self.unvisited.remove(event.getSource())
   if self.visited == True:
          print "soy ", self.id, " ya estoy visitado, envio rechazo a ", event.getSource()
       newevent = Event("RECHAZO", self.clock + 1.0,
                      event.getSource(), self.id)
                self.transmit(newevent)
   else:
                self.visited = True
                self.father = event.getSource()
                print "soy ", self.id, " y mi padre es ", self.father
                self.go_explore_more()
```

Ejemplo DFS Cheung

```
#continua la definición de: def receive(self, event):
  ...
  elif  event.getName() == "REGRESA" or
        event.getName() == "RECHAZO":
       self.go_explore_more()
```

Ejemplo DFS Cheung

```
# "main()"
# construye una instancia de la clase Simulation recibiendo
    como parámetros el nombre del archivo que codifica la lista
    de adyacencias de la gráfica y el tiempo max. de simulación


if len(sys.argv) != 2:
    print "Please supply a file name"
    raise SystemExit(1)
experiment = Simulation(sys.argv[1], 100)
```

Ejemplo DFS Cheung

```
# asocia un pareja proceso/modelo con cada nodo de la gráfica
for i in range(1,len(experiment.graph)+1):
    m = AlgorithmDFS()
    experiment.setModel(m, i)

# inserta un evento semilla en la agenda y arranca

seed = Event("DESCUBRE", 0.0, 1, 1)
experiment.init(seed)
experiment.run()
```

Ejemplo DFS Cheung

> python DFSCheung.py g1.txt

soy 1 y mi padre es 1

soy 2 y mi padre es 1

soy 3 y mi padre es 2

soy 1 ya estoy visitado, envio rechazo a 3

soy 4 y mi padre es 3

soy 1 ya estoy visitado, envio rechazo a 4

soy 2 ya estoy visitado, envio rechazo a 4

soy 5 y mi padre es 4

soy 3 ya estoy visitado, envio rechazo a 5

soy 6 y mi padre es 5

soy 4 ya estoy visitado, envio rechazo a 6