Trabajo Practico 1: ConcuCalesita

Arana Andrés, *P. 86.203*
and2arana@gmail.com

Arias Damián, *P. 89.952*
arias.damian@gmail.com

Sergio Matias Piano, *P. 85.191*
smpiano@gmail.com

2do. Cuatrimestre de 2014
75.59 Técnicas de Programación Concurrente 1
Facultad de Ingenieria, Universidad de Buenos Aires

Resumen

Este informe sumariza el desarrollo del trabajo practico 1 de la materia Técnicas de Programación Concurrente I (75.59) dictada en el segundo cuatrimestre de 2014 en la Facultad de Ingenieria de la Universidad de Buenos Aires. El mismo consiste en la construccion de una simulación del uso de una calesita utilizando técnicas de sincronización de procesos.

Índice

1. Análisis del problema

1.1. Descripción

El sistema deberá contar con la posibilidad de simular la llegada de los niños del barrio a la calesita del parque. A medida que van llegando, estos se irán encolando en la ventanilla única para comprar un boleto.

En ventanilla estará presente el cajero quien desacolará al primero de la fila, le cobrará el valor del boleto y pasará a atender al siguiente niño. El dinero recaudado es guardado en la caja, teniendo acceso exclusivo el cajero mientras la esté usando. Cada cierto tiempo vendrá el auditor a controlar la recaudación tomando el control momentaneamente de la caja.

Los niños que hayan comprado el boleto se dirigirán a la cola de entrada de la calesita donde aguardarán su turno para ingresar. La calesita tiene una capacidad fija y no empieza la vuelta hasta que estén todos sus lugares completos. Cuando termina una vuelta los niños que estaban en la calesita salen de a uno por la puerta de salida. Una vez que salieron todos, entra otra camada de niños que ingresan de a uno desacolandose de la cola de entrada y se van acomodando en sus lugares preferidos disponibles.

2. Hipótesis

- Los niños tienen su lugar preferido en la calesita e intentarán ocuparlo primero. Por lo tanto la calesita deberá tener alguna forma de identificar cuales son sus lugares disponibles.
- Antes de ingresar los niños a la calesita se deberá esperar que salgan los que estaban ocupandola hasta ese momento. No habrá ingresos y egresos en simultaneo.
- La vuelta en la calesita no comienza a menos que estén todos los lugares ocupados.

3. Desarrollo de la solución

3.1. Procesos, comunicación y sincronización

- **director**: inicializa los recursos compartidos, dispara el resto de los procesos, los detiene con señales de interrupción y libera recursos al terminar la simulación.
- **spawner**: genera un nuevo niño cada cierto tiempo random simulando la llegada de estos al parque. Cada niño es inyectado (escrito) en una fifo que luego será leído por el cashierq.
- **cashierq**: va desacolando la fifo escrita por el spawner y para cada niño simula la interacción con el cashier. Como el cashier es un proceso deberá sincronizar todas las transacciones que se realicen. Luego de esto inyecta los niños en la fifo que leerá carrouselq.
- **cashier**: realiza las ventas de boletos a cada niño. El dinero es guardado en la caja, la cual es lockeada para escritura cuando la accede.
- **audit**: cada cierto tiempo random lockea la caja para lectura inspeccionando la recaudación. En ese momento el cashier no puede usar la caja.
- **carrouselq**: espera que se abra la puerta de entrada de la calesita y va desacolando la fifo escrita por cashierq. Para cada niño crea un proceso childinc que simulará la elección del lugar preferido del niño en la calesita.
- **childinc**: genera un listado ordenado de los lugares favoritos del niño en orden decreciente. Siempre que el lugar esté desocupado, intentará ocupar el que más le gusta primero.
- **carrousel**: espera que todos los lugares estén ocupados y comienza la vuelta. Cuando la vuelta termina va inyectando los niños en la fifo se salida. Cuando salieron todos abilita la puerta de entrada.
- **exitq**: va desacolando la fifo de salida escrita por carrousel y despide a cada niño de la simulación.

3.2. Recursos compartidos

- **shared_data**: es un struct que contiene todos los valores compartidos entre los procesos. Estos valores son:
  - cashier_child_id: id del niño siendo atendido por el cajero.
  - cashier_child_money: dinero que recibe el cajero cuando le cobra al niño por el boleto.
  - cashier_child_change: vuelto entregado al niño durante al venta del boleto.
  - balance: dinero total recaudado en la caja.
  - config_price: precio del boleto.
  - config_capacity: cantidad de lugares en la calesita.
  - config_duration: duración de una vuelta en calesita.
  - sem_cashier: id del grupo de semáforos que sincronizan a los niños con el cashier.
  - sem_carrousel_entrance: id del semáforo que sincroniza la entrada a la calesita.
  - sem_carrousel_places: id del grupo de semáforos para los lugares disponibles en la calesita.

- sem_carrousel_full_places: id del grupo de semáforos para los lugares ocupados en la calesita.

- shmem_carrousel_places: id del vector de memoria compartida que guarda que niño se encuentra ocupando cada lugar de la calesita.

3.3. Semáforos

- **cashier_sem**: semáforo binario que bloquea al siguiente niño de la cola para comprar el boleto.
- **child_sem**: semáforo binario para bloquear al cajero mientras no haya niños por atender en la cola.
- **money_sem**: semáforo binario para bloquear al cajero mientras el niño le entrega el dinero para pagar el boleto.
- **change_sem**: semáforo binario para bloquear al niño mientras el cajero le entrega el vuelto y el boleto.
- **carrousel_entrance_sem**: semáforo binario para bloquear la entrada a la calesita cuando todavía tenga niños adentro.
- **carrousel_places_sem**: grupo de semáforos que se setean para cada lugar de la calesita indicando si el lugar está libre. Se utilizan para que no entre nadie a la calesita hasta que todos los lugares estén vacios.
- **carrousel_full_places_sem**: grupo de semáforos que se setean para cada lugar de la calesita indicando si el lugar está ocupado. Se utilizan para que la vuelta no comience hasta que todos los lugares de la calesita estén ocupados.

3.4. Colas de mensajes

Siempre se escriben y leen los ids de los niños.

- **cashier_FIFO**: escrita por el **spawner** y consumida por el **cashierq**.
- **carrousel_FIFO**: escrita por el **cashier** y consumida por el **carrouselq**.
- **exit_FIFO**: escrita por el **carrousel** y consumida por el **exitq**.

3.5. Locks

- **balance_file**: archivo ficticio utilizado por el cashier y el audit para lockear la caja mientras la acceden.
- **log_file**: el archivo de log del sistema es bloqueado para escritura por cada proceso cada vez que se agrega una entrada.

4. Uso de la aplicación

El proyecto incluye un makefile con varios targets para realizar todo tipo de tareas relacionadas. Toda salida de dichas tareas se realiza dentro del directorio build, generado automáticamente según sea necesario. Las tareas están correctamente configuradas para detectar dependencias; no es necesario correr una tarea como clean antes de hacer el build, o de correr all antes de run. Las tareas mas importantes son:

- `make all`

  Compila todos los ejecutables de cada proceso del proyecto generándolos en la carpeta build/exec.

- `make run`

  Ejecuta el director, proceso principal que dirige la simulación con parámetros por defecto.

- `make doc-preview`

  Visualiza el PDF del informe utilizando evince.

- `make clean`

  Elimina cualquier artefacto generado por el sistema de build.

Para ejecutar la aplicación manualmente se puede hacer por ejemplo:

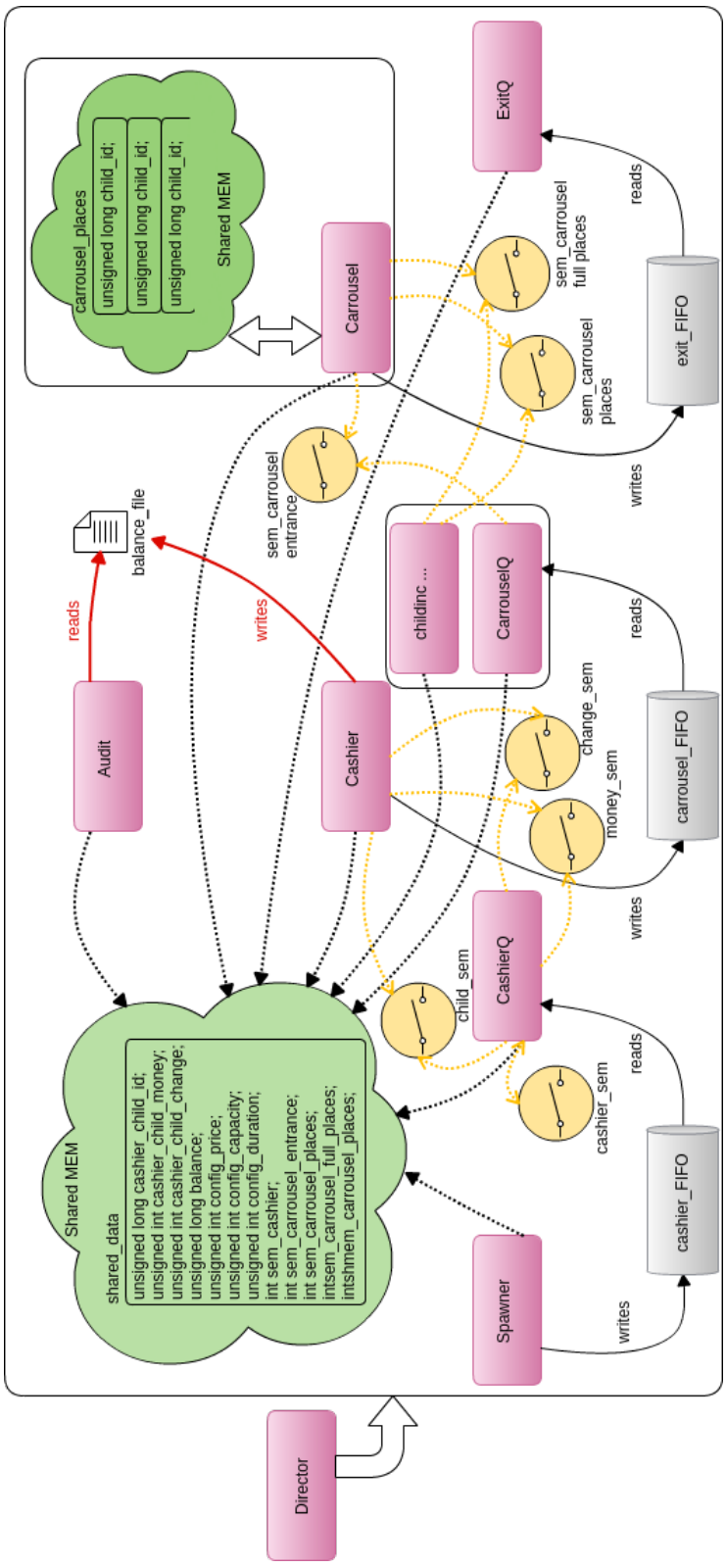
```
build/exec/director -l 0 -p 30 -d 5 -c 5
```

Los parámetros significan:

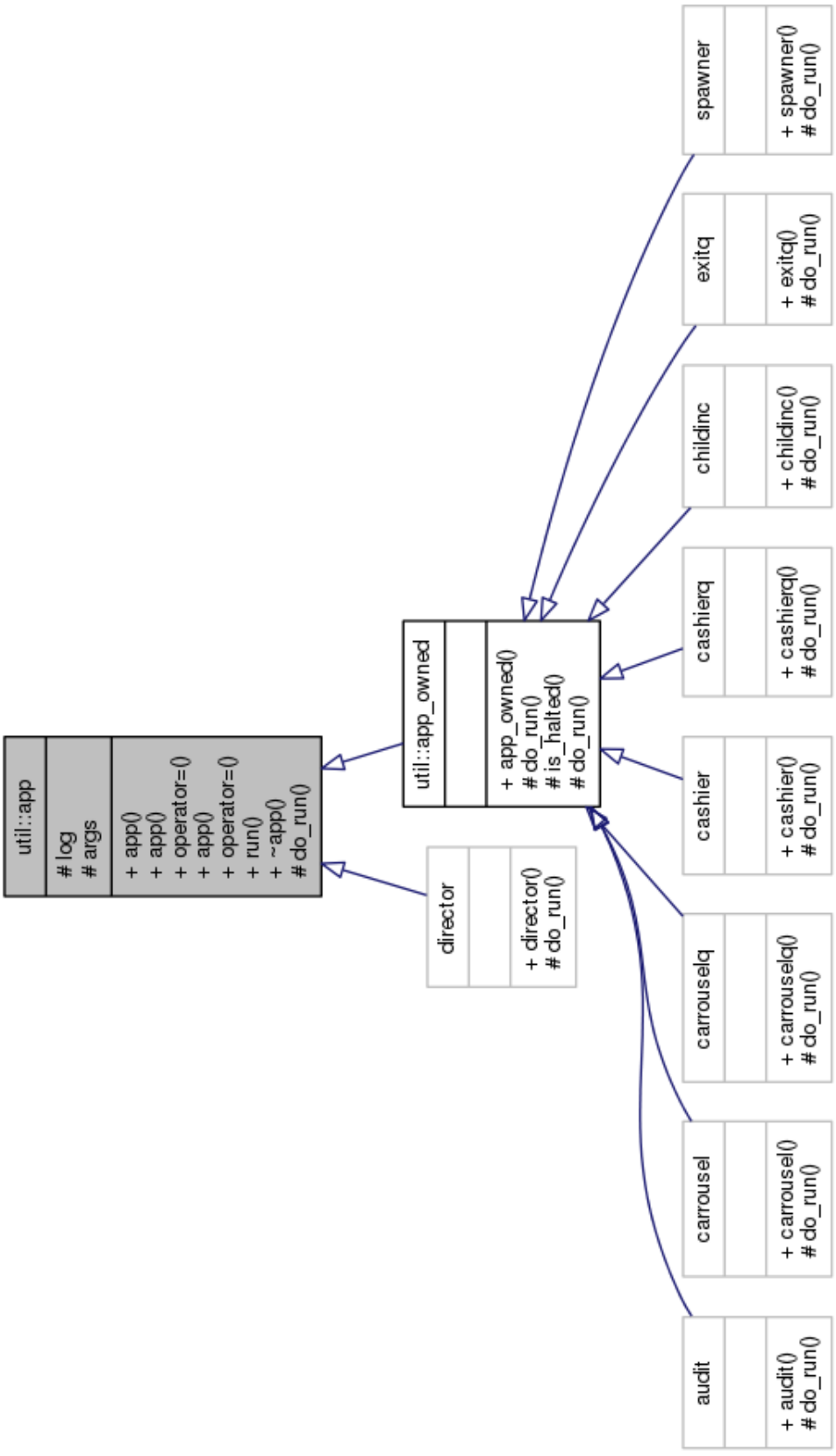
- -l: setea el nivel de loggeo (0:debug y 1:info).
- -p: es el precio del boleto.
- -d: es la duración de la vuelta en calesita en segundos.
- -c: es la cantidad de lugares disponibles en la calesita.

5. Diagramas

5.1. Diagrama de comunicación entre procesos

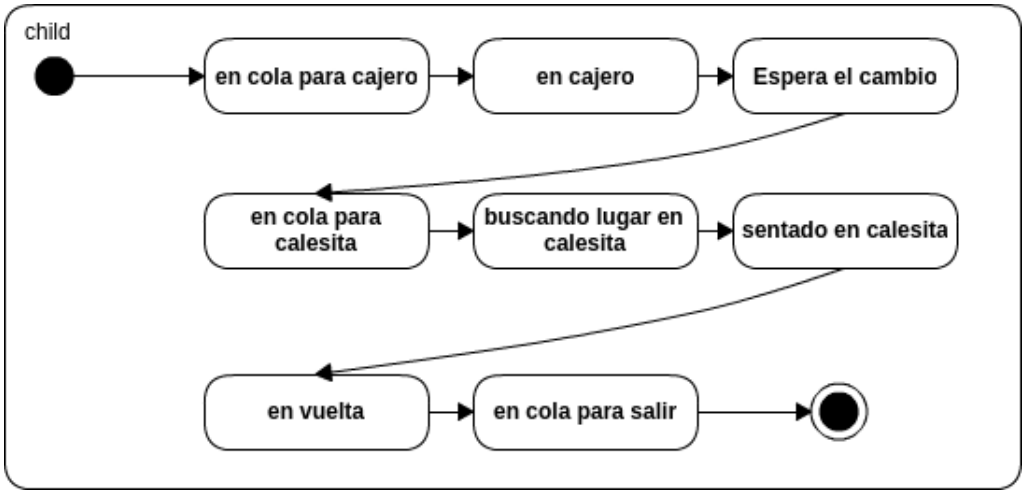


5.2. Diagrama de clases



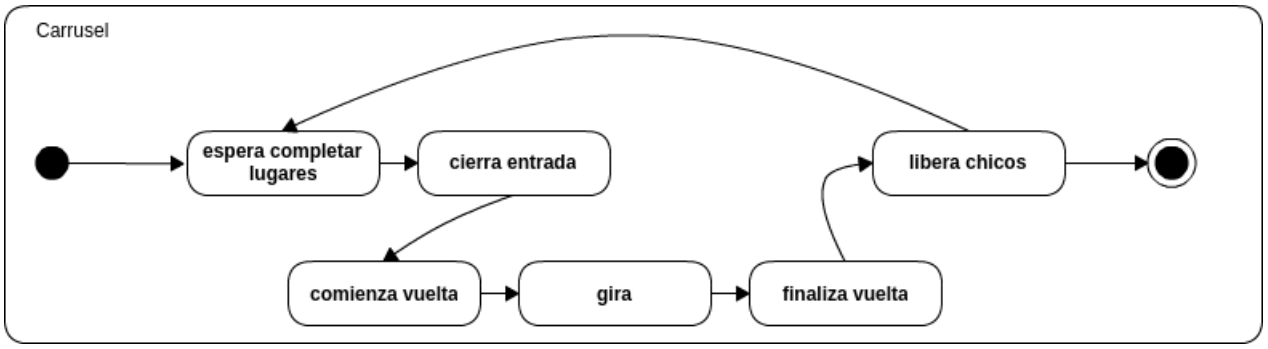
5.3. Diagramas de transición de estados

5.3.1. Niño

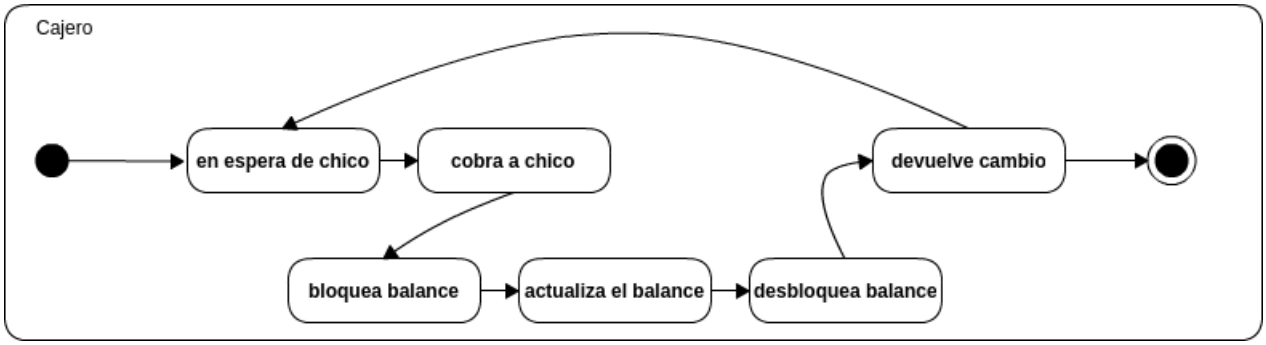


5.3.2. Director

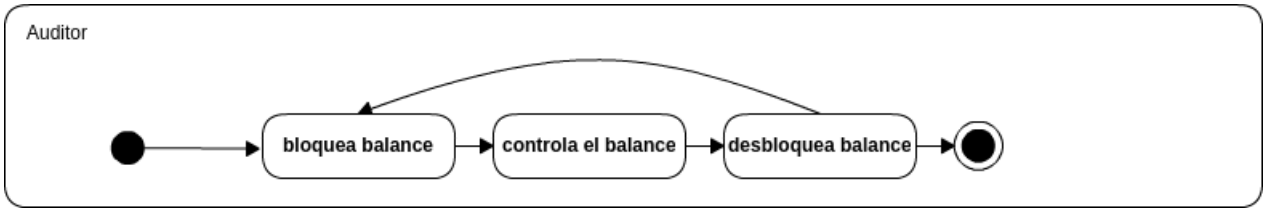
5.3.3. Calesita



5.3.4. Cajero



5.3.5. Auditor



Primer Proyecto: ConcuCalesita

75.59 - Técnicas de Programación Concurrente I

Objetivo

A los niños del barrio les encanta ir a la calesita del parque. El objetivo de este proyecto consiste en implementar la simulación parcial del funcionamiento de dicha calesita.

Requerimientos Funcionales

Los requerimientos funcionales son los siguientes:

1. Los niños que quieren dar una vuelta a la calesita deben primero comprar el boleto. Para ello, deben formar fila por orden de llegada en la única ventanilla disponible de venta de boletos.
2. Una vez que compraron el boleto, los niños deberán esperar en fila su turno para poder subir a la calesita. La fila se acomoda también por orden de llegada de los niños que poseen boleto.
3. Al finalizar una vuelta de la calesita, los niños que salen lo hacen por la puerta de salida de a uno por vez, mientras que los que entran lo hacen por la puerta de entrada, también de a uno por vez.
4. Los niños ingresan a la calesita y corren para ocupar los lugares disponibles. Cuando no hay más lugares disponibles, se bloquea la entrada y se comienza con la vuelta.
5. El señor que vende los boletos por la ventanilla va guardando la recaudación en la caja.
6. El administrador de la calesita puede entrar en cualquier momento a consultar la recaudación.
7. Solamente uno a la vez puede utilizar la caja, ya sea para consultar el saldo o para actualizarlo.
8. Para que todos los niños puedan divertirse, la simulación finalizará cuando todos los niños del barrio tengan oportunidad de dar una vuelta en la calesita.
9. Se deberá poder configurar sin necesidad de recompilar el código la duración de la vuelta, el precio del boleto y la cantidad de lugares disponibles en la calesita.

Requerimientos no Funcionales

Los siguientes son los requerimientos no funcionales de la aplicación:

1. El proyecto deberá ser desarrollado en lenguaje C o C++, siendo este último el lenguaje de preferencia.
2. La simulación puede no tener interfaz gráfica y ejecutarse en una o varias consolas de línea de comandos.
3. El proyecto deberá funcionar en ambiente Unix / Linux.
4. La aplicación deberá funcionar en una única computadora.

5. El programa deberá poder ejecutarse en "modo debug", lo cual dejará registro de la actividad que realiza en un único archivo de texto para su revisión posterior. Se deberá poder seguir el recorrido de cada niño a medida que va pasando por las distintas etapas, desde que forma fila para comprar el boleto hasta que sale de la calesita.

6. Las facilidades de IPC que se podrán utilizar para la realización de este proyecto son las que abarcan la primera parte de la materia, es decir, hasta el primer parcial. Dichas facilidades son:

   a) Memoria compartida

   b) Señales

   c) Pipes y fifos

   d) Locks

   e) Semáforos

   Cualquier otra facilidad queda expresamente excluida para este proyecto.

Tareas a Realizar

A continuación se listan las tareas a realizar para completar el desarrollo del proyecto:

1. Dividir el proyecto en procesos. El objetivo es lograr que la simulación esté conformada por un conjunto de procesos que sean lo más sencillos posible.

2. Una vez obtenida la división en procesos, establecer un esquema de comunicación entre ellos teniendo en cuenta los requerimientos de la aplicación. ¿Qué procesos se comunican entre sí? ¿Qué datos necesitan compartir para poder trabajar?

3. Tratar de mapear la comunicación entre los procesos a los problemas conocidos de concurrencia.

4. Determinar los mecanismos de concurrencia a utilizar para cada una de las comunicaciones entre procesos que fueron detectadas en el ítem 2. No se requiere la utilización de algún mecanismo específico, la elección en cada caso queda a cargo del grupo y debe estar debidamente justificada.

5. Realizar la codificación de la aplicación. El código fuente debe estar documentado.

Entrega

La entrega del proyecto comprende lo siguiente:

1. Informe, se deberá presentar impreso en una carpeta o folio y en forma digital (PDF) a través del campus

2. El código fuente de la aplicación, que se entregará únicamente mediante el campus

La entrega en el campus estará habilitada hasta las 19 hs de la fecha indicada oportunamente.

El informe a entregar debe contener los siguientes ítems:

1. Breve análisis del problema, incluyendo una especificación de los casos de uso de la aplicación.

2. Detalle de resolución de la lista de tareas anterior.

3. Diagrama que refleje los procesos, el flujo de comunicación entre ellos y los datos que intercambian.

4. Diagramas de clases realizados.

5. Diagrama de transición de estados de un niño.