UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

Trabajo Final
La agencia de robots

Diseño y Aplicaciones de Sistemas Distribuidos
(SDI)

José Simó
Rev: septiembre 2022

Contenido

Trabajo final: La agencia de robots.

.1 Introducción.

La "agencia de robots" consiste en un grupo de robots que interactúan en un escenario. El comportamiento de los robots será totalmente simulado, es decir, no vamos a trabajar con robots reales y el escenario también será sintético.

Cada robot se implementará como un objeto distribuido ICE cuyo interfaz se especifica en el archivo "robot.ice" incluido en el anexo. El gestor del grupo de robots, al que llamaremos "cámara", también se implementará como un objeto ICE cuyo interfaz también se detalla en el archivo "robots.ice". Para visualizar el comportamiento de los robots, se suministra un componente "consola" que representará gráficamente la evolución del estado global de los robots. Resumiendo, cámara gestionará la pertenencia al grupo de tanto los robots como las consolas.

Los actores involucrados en la aplicación son:

- **Robot:** Este objeto es el miembro del grupo. (puede haber muchos)
  - El Robot es un objeto ICE que ofrece un interfaz adecuado para que otros objetos (p.e. el gestor del grupo) puedan obtener su información de estado.
  - Un Robot está interesado en pertenecer a un Grupo de Robots para poder recibir las difusiones destinadas al grupo y así coordinarse con el resto de Robots del grupo.
  - Para pertenecer al grupo el Robot debe suscribirse en el gestor del grupo (Cámara).
    - Para suscribirse, el Robot proporciona su IOR (proxy en formato cadena) al gestor del grupo.
    - Al suscribirse, el gestor del grupo le proporciona al Robot la identificación (IP y port) del canal de difusión del grupo. En el caso de usar JMS, el nombre del "topic" se compondrá de la siguiente forma: [IP]_[puerto] por ejemplo 228.7.7.7_4001

- **Cámara:** Este objeto es el gestor del grupo. (hay sólo uno)
  - Es un objeto ICE que ofrece un interfaz que permite que los miembros (objetos "Robot") se suscriban al grupo.
  - Para que los nuevos Robots puedan localizarlo, debe registrarse en el servicio de nombres (icegridregistry).
  - Mantiene una lista de los IOR de los miembros (Robots) que están suscritos al grupo.
  - Periódicamente realiza las siguientes operaciones:
    - Accede al estado de todos los miembros del grupo (a través del interfaz ICE que ofrecen los Robots).

  - Compone una lista con toda la información (Instantánea)
  - Difunde la "Instantánea" al grupo usando el canal de difusión.

- **Consola:** Este objeto es el miembro del grupo. (puede haber muchos)
  - Este componente se proporciona completamente implementado.
  - La consola es un es un objeto ICE.
  - Ofrece un interfaz gráfico que visualiza el estado de todos los robots que recibe por el canal de difusión.

El funcionamiento del sistema será el siguiente:

- La cámara iniciará su ejecución registrándose en el Servicio de Nombres ICE usando para el objeto el nombre "Camara" y para el adaptador el nombre "CamaraAdapter". Los robots localizarán la cámara (gestor del grupo) buscándola en el servicio de nombres.
- La cámara recibirá suscripciones de los robots (solicitud de adhesión al grupo) y mantendrá la lista de los robots que pertenecen al grupo (lista de IORs en forma de "String"). De la misma forma, también mantendrá la lista de las consolas adheridas al grupo. La cámara consultará periódicamente el estado de cada robot (método "ObtenerEstado") y compondrá una instantánea del grupo que "difundirá" al grupo. Los robots que no contesten a la llamada "ObtenerEstado" serán dados de baja del grupo. La cámara, para averiguar qué consolas debe dar de baja, hará lo mismo invocado al método "estoyViva" de las consolas.
- Un robot (o consola), al suscribirse en la cámara, recibirá un identificador del canal de difusión en la forma de un "IP" y un "puerto", además de una descripción del escenario en el que se encuentra. El canal de difusión podrá ser tanto un canal "Multicast" como un "topic" JMS. En el caso de usar "Multicast" la IP deberá encontrarse en el rango de direcciones de difusión. En el caso de usar JMS, el nombre del "topic" se compondrá de la siguiente forma:

  `[IP]_[puerto]` por ejemplo `228.7.7.7_4001`

  Así los robots (y consolas) podrán recibir la "instantánea" con la información del estado global.

Nota: sólo es necesario implementar un mecanismo de difusión que para este caso deberá ser **obligatoriamente JMS**. Observará que la consola que se suministra soporta ambos mecanismos de difusión.

- Cada robot implementará el comportamiento de "Ir a objetivo" y "Evitar obstáculos" utilizando los algoritmos de control suministrados en la biblioteca "khepera.jar". Usando la misma biblioteca también simulará su propio movimiento (avance) para actualizar su posición.
- Desde la consola se podrá cambiar el objetivo, posición, robot líder y escenario mediante la llamada a los métodos CORBA correspondientes.
- Cuando desde la consola se carga un nuevo escenario se invoca al método "ModificarEscenario" de la cámara. Cuando esto ocurra, la cámara deberá

informar a todos los robots y consolas suscritos del cambio de escenario invocando para cada uno su método “ModificarEscenario”.

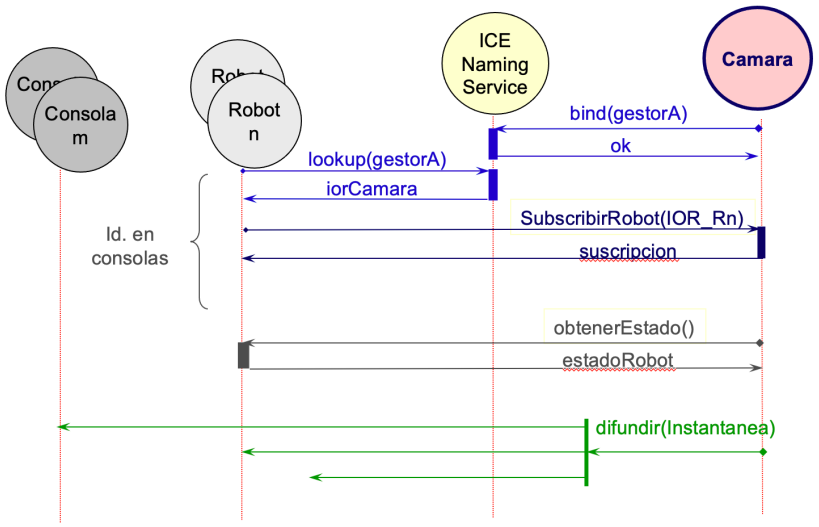

Figura 1: Esquema de colaboración entre objetos

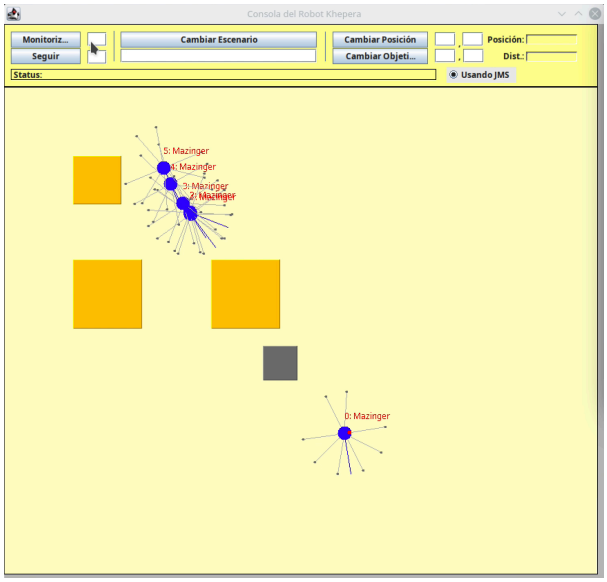

Figura 2: Aspecto de la representación gráfica de la “consola”.

.2 Ejercicio

Se pide implementar el código del “robot” y el de la “cámara” de forma que se respete el interfaz “idl“ suministrado y que el resultado funcione perfectamente con la “consola” que se suministra. El sistema deberá soportar varios robots en

ejecución y gestionar la caída de robots y consolas. Se debe implementar soluciones para todos los métodos ICE especificados en el interfaz.

.3 Archivos de apoyo:

.3.1 Archivo khepera.jar

En este archivo residen las bibliotecas necesarias para realizar el control del robot y simular su movimiento. Se suministra ya compilado y no se proporciona el código fuente.

.3.1 Archivo consolaRobots.jar

Este archivo es un "jar" ejecutable que implementa la consola gráfica. Se suministra ya compilado y no se proporciona el código fuente. El sistema que se desarrolle debe funcionar perfectamente en colaboracion con esta consola gráfica.

.3.2 Archivo robot.ice

```
//////////////////////////////////////////////////////////////////////
//// Module agencia

module agencia {

//////////////////////////////////////////////////////////////////////
//// Module agencia.khepera

  module datos {

      struct Posicion {
        float x;
        float y;
      }

      sequence<Posicion> ListaPosiciones;

      struct Rectangulo {
        float x;
        float y;
        float ancho;
        float alto;
        int color;
      }

      sequence<Rectangulo> ListaRectangulos;

      struct Escenario {
        ListaRectangulos rects;
        int  nrecs;
        int color;
      }


      struct PuntosRobot{
        Posicion centro;
        ListaPosiciones sens; // son 9
        ListaPosiciones finsens; // son 9
        ListaPosiciones inter; // son 8
      }

      struct EstadoRobot {
        string nombre;
        int id;
        string IORrob;  //Referencia en formato String IOR
        PuntosRobot puntrob;
        Posicion posObj;
        int idLider;
      }

      sequence<EstadoRobot> ListaEstadosRobot;

      struct Instantanea {
```

```
          ListaEstadosRobot estadorobs;
       }

      struct IPYPort{
         string ip;
         int port;
      }

      struct suscripcion {
          int id;
          IPYPort iport;
          Escenario esc;
      }

                                       sequence<string> ListaStrings;

      struct ListaSuscripcion{
        //IORs en formato string
        ListaStrings IORrobots;
        ListaStrings IORconsolas;
      }
}

////////////////////////////////////////////////////////////////////
//// Module agencia.objetos

  module objetos {

    interface RobotSeguidor{

       agencia::datos::EstadoRobot ObtenerEstado( );
       void ModificarEscenario( agencia::datos::Escenario esc);
       void ModificarObjetivo( agencia::datos::Posicion NuevoObj);
       void ModificarPosicion( agencia::datos::Posicion npos);
       void ModificarLider( int idLider);
    };



    interface Consola{
       void ModificarEscenario( agencia::datos::Escenario esc);
       bool estoyviva();
    };



     interface Camara{

        agencia::datos::suscripcion SuscribirRobot(string IORrob);
        agencia::datos::suscripcion SuscribirConsola(string IORcons);
        void BajaRobot(string IORrob);
        void BajaConsola(string IORcons);
        agencia::datos::ListaSuscripcion ObtenerLista();
        agencia::datos::IPYPort ObtenerIPYPortDifusion();
        agencia::datos::Instantanea ObtenerInstantanea();
        void ModificarEscenario(agencia::datos::Escenario esc);
        agencia::datos::Escenario ObtenerEscenario();
      };

  }
}
```