UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

Trabajo Final
La agencia de robots

Diseño y Aplicaciones de Sistemas Distribuidos
(SDI)

José Simó

Rev: Septiembre 2013.

Contenido

Trabajo Final 1. La agencia de robots.

1.1 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 CORBA cuyo interfaz se especifica en el archivo "robot.idl" incluido en el anexo. El gestor del grupo de robots, al que llamaremos "cámara", también se implementará como un objeto CORBA cuyo interfaz también se detalla en el archivo "robots.idl". 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.

El funcionamiento del sistema será el siguiente:

- La cámara iniciará su ejecución registrándose en el Servicio de Nombres CORBA usando el nombre "Camara". En nuestro caso usaremos el ORB del JDK, por lo tanto el servicio de nombres será el "orbd". 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.
- 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", demás de una descripción del escenario en el que se encuentra. El canal de difusión podrá ser tanto un canal "Multicast" como un "topic" JMS (sólo es necesario implementar un mecanismo pero se recomienda el uso de 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.
- 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”.

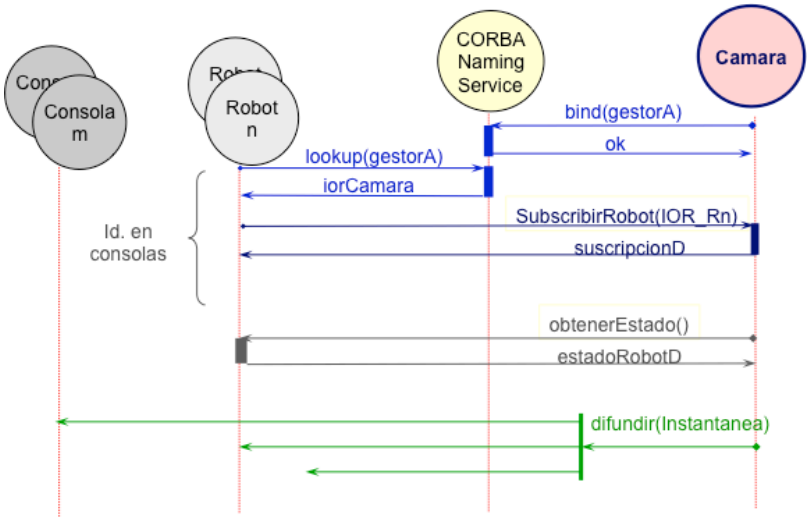


Figura 1: Esquema de colaboración entre objetos

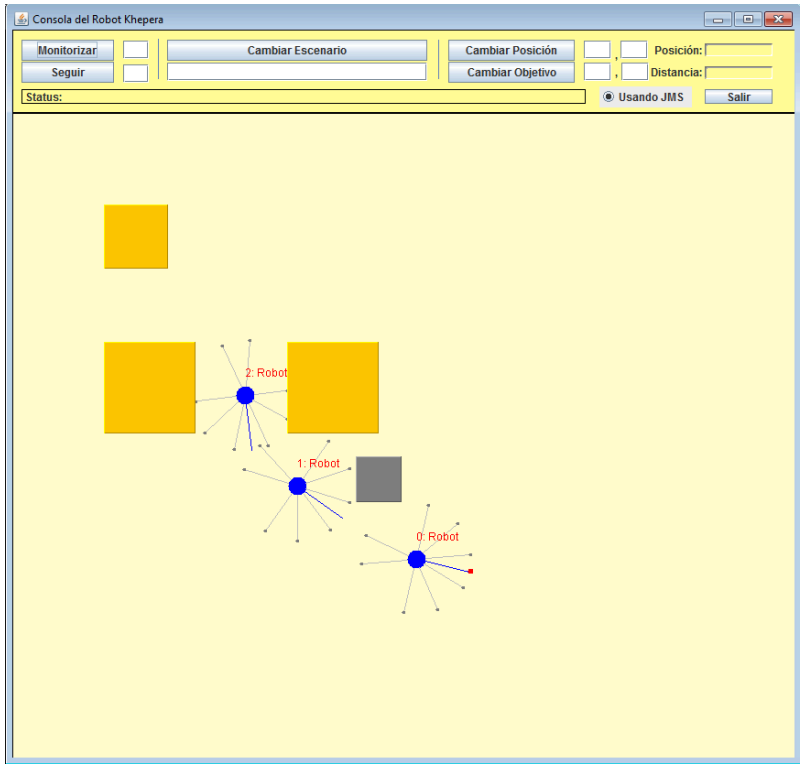


Figura 2: Aspecto de la representación gráfica de la “consola”.

1.2 Ejercicio

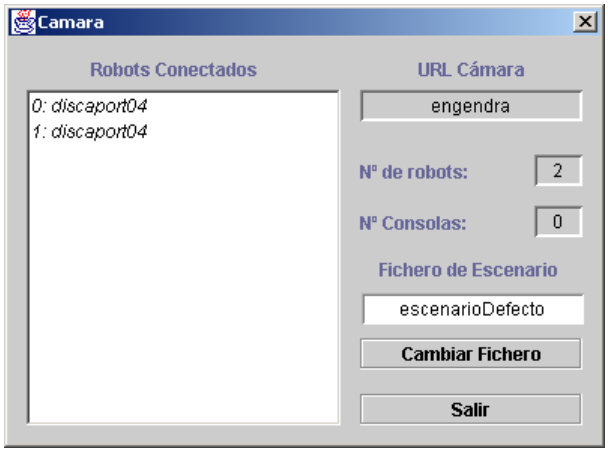
Se pide implementar el código del "robot" y el de la "cámara" de forma que se respete el interfaz "idl" suministrado y que el resultado funcione perfectamente con la "consola" que se suministra. El sistema deberá soportar varios robots en ejecución y gestionar la caída de robots y consolas. Se debe implementar soluciones para todos los métodos CORBA especificados en el interfaz.

1.3 Ampliaciones propuestas

Se proponen dos posibles ampliaciones al trabajo:

1.3.1 Ampliación 1: Cámara gráfica

La cámara, como gestor del grupo, mantiene información que puede ser relevante para el usuario. En esta ampliación se plantea la programación de un interfaz gráfico para la consola que, al menos, muestre los elementos que aparecen en la la figura.



1.3.2 Ampliación 2: Cámara tolerante a fallos

La cámara, como gestor del grupo, es un componente crítico del sistema ya que centraliza la pertenencia al grupo y la coherencia del estado global. En esta aliación se propone la programación de la cámara de forma que este componente soporte redundancia. El comportamiento de la cámara sería el siguiente:

- Cuando arranca la cámara, intenta suscribirse al servicio de nombres. Si lo consigue, se comportará como "cámara principal" de forma normal. Si no consigue registrar el nombre en el servicio de nombres es que ya hay otra cámara en funcionamiento, en este caso se comportará como "cámara segundaria".

- La cámara segundaria mantendrá una copia del estado global (robots y consolas suscritas) mediante consulta periódica o recibiendo notificaciones (a elección del programador). Cuando detecte que la cámara principal no responde, la cámara segundaria pasará a ser la cámara principal.
- Adicionalmente, la cámara puede encargarse de lanzar otras cámaras para mantener la redundancia en caso de fallo.

1.4 Ficheros de apoyo:

1.4.1 Fichero khepera.jar

En este fichero 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.

1.4.1 Fichero robots_consola.jar

Este fichero 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.

1.4.1 Fichero corba.jar

Este fichero es el resultado de la compilación del archivo "robot.idl". Se suministra por comodidad. Si se incluye como depencencia en los proyectos, no será necesario compilar el idl para generar las clases Java corrrespondientes.

1.4.2 Fichero robot.idl

```
module corba{

  module khepera {

    module robot{
      struct PosicionD{
        float x;
        float y;
      };
    };

    module escenario{
      struct RectanguloD{
        float x;
        float y;
        float ancho;
        float alto;
        unsigned long color;
      };

      struct EscenarioD{
          sequence<RectanguloD> recs;
```

```
          unsigned long  nrecs;
          unsigned long color;
      };
     };
  };

  module robot{
    interface RobotSeguidorInt;
  };

  module instantanea{

   struct PuntosRobotD{
     corba::khepera::robot::PosicionD centro;
     corba::khepera::robot::PosicionD sens[9];
     corba::khepera::robot::PosicionD finsens[9];
     corba::khepera::robot::PosicionD inter[8];
    };

    struct EstadoRobotD {
      string nombre;
      unsigned long id;
      //Referencia en formato String IOR
      string IORrob;
      //Referencia en formato binario
      corba::robot::RobotSeguidorInt refrob;
      PuntosRobotD puntrob;
      corba::khepera::robot::PosicionD posObj;
      unsigned long idLider;
    };

     struct InstantaneaD{
       sequence<EstadoRobotD> estadorobs;
    };
  };

  module robot{
    interface RobotSeguidorInt{
       void ObtenerEstado(out corba::instantanea::EstadoRobotD est);
       void ModificarEscenario(in corba::khepera::escenario::EscenarioD esc);// raises
(CORBA::COMM_FAILURE);
       void ModificarObjetivo(in corba::khepera::robot::PosicionD NuevoObj);
       void ModificarPosicion(in corba::khepera::robot::PosicionD npos);
       void ModificarLider(in unsigned long idLider);
    };

  };

  module consola{
    interface ConsolaInt{
       void ModificarEscenario(in corba::khepera::escenario::EscenarioD esc);
       boolean estoyviva();
    };
  };

  module camara{

     struct IPYPortD{
        string ip;
        unsigned long port;
     };

     struct suscripcionD{
         unsigned long id;
         IPYPortD iport;
```

```
         corba::khepera::escenario::EscenarioD esc;
    };

    struct ListaSuscripcionD{
      //IORs en formato string
      sequence<string> IORrobots;
      sequence<string> IORconsolas;
    };

    interface CamaraInt{
       suscripcionD SuscribirRobot(in string IORrob);
       suscripcionD SuscribirConsola(in string IORcons);
       void BajaRobot(in string IORrob);
       void BajaConsola(in string IORcons);
       ListaSuscripcionD ObtenerLista();
       IPYPortD ObtenerIPYPortDifusion();
       corba::instantanea::InstantaneaD ObtenerInstantanea();
       void ModificarEscenario(in corba::khepera::escenario::EscenarioD esc);
       corba::khepera::escenario::EscenarioD ObtenerEscenario();
    };
  };
};
```