(Draft) Actas de las VI Jornadas de Concurrencia. 1998, p. 117-128

Sistema de información distribuido orientado a objeto para la localización de móviles, una aproximación basada en CORBA[1]

P. R. Muro-Medrano (1), F. J. Zarazaga,
J.A. Bañares, J. Guillo,
D. Infante, R. López

Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior, Universidad de Zaragoza
María de Luna 3, 50015 Zaragoza

(1) prmuro@posta.unizar.es

Resumen

En este trabajo se presenta la motivación e ideas básicas de un sistema de información distribuido orientado a objeto para el seguimiento y control de móviles. El trabajo presenta la arquitectura de software adoptada y se centra en la infraestructura de objetos distribuidos a través de un bus CORBA. En este contexto se analizan las ventajas que se consiguen por medio de esta distribución y en las facilidades que esto aporta para la integración y expansión del sistema.

1. Introducción

La tendencia actual de las aplicaciones cliente/servidor parece indicar que todas las aproximaciones existentes convergerán en un futuro próximo hacia los sistemas de objetos distribuidos [OHE96]. La utilización de esta tecnología esta justificada por el deseo de construir software a partir de componentes reusables y la necesidad de integrar componentes que interoperen a través de la red.

El modelo de objetos favorece el trabajo cooperativo y la reutilización a través de la encapsulación. Sin embargo, se tiene la necesidad de hacer compatible nuestras nuevas aplicaciones con los sistemas y aplicaciones anteriores. Si a esto añadimos la rapidez con que evoluciona la tecnología, la variedad de sistemas operativos y lenguajes de programación, y la necesidad creciente de compartir información y recursos, el desarrollo y mantenimiento de estos sistemas es realmente costosa debido a su heterogeneidad.

Con objeto de extender estas ideas de abstracción y encapsulación del modelo de objetos a los sistemas distribuidos se constituye el Object Mangement Group (OMG) [OMG95], consorcio de fabricantes de software que tiene entre sus objetivos la obtención de una infraestructura común para el desarrollo de aplicaciones distribuidas orientadas a objeto. Como parte de esta infraestructura surge CORBA *(Common Object Request Broker Architecture)* [OMG96b], una arquitectura para la interacción entre distintas aplicaciones en un entorno distribuido. Las aplicaciones, independientemente de que estén o no programadas con un lenguaje orientado a objeto, son vistas como objetos y sus servicios como la interface del objeto [VINO97]. CORBA aporta un modelo estable para

[1] **Agradecimientos**

Este trabajo ha estado parcialmente financiado por el CONSYD de la Diputación General de Aragón a través del proyecto P-18/96. El trabajo relacionado con la arquitectura de objetos distribuidos está parcialmente basado en resultados originados por el proyecto TAP95-0574 de la CICYT.

sistemas orientados a objeto distribuidos que permite abordar la heterogeneidad y el inevitable cambio de las tecnologías.

Un ejemplo de aplicación que precisa de esta infraestructura son los sistemas de navegación y ayuda a la gestión de los transportes y su logística [KRAK93]. Si bien los precios de los componentes necesarios para su construcción van bajando paulatinamente, la dificultad técnica para su desarrollo radica en el hecho de que exigen al equipo de desarrollo conocimiento para la integración de diversas tecnologías: tarjetas con microprocesadores, integración de sensores, comunicaciones por radio terrestre o por satélite, comunicaciones entre redes de computadores, software de comunicaciones, sistemas de información geográfica, interfaces gráficos de usuario, acceso y gestión de bases de datos avanzadas, ingeniería del software, etc.

Entre los diferentes tipos de sistemas de navegación, son los sistemas de seguimiento y control de flotas [BBH94] donde se enmarca el sistema objeto del presente trabajo (Figura 1). El corazón de dichos sistemas es un centro de control que monitoriza los vehículos de la flota en un mapa digital de carreteras y calles. Esta arquitectura requiere un enlace de comunicaciones con los móviles y capacidades para el reconocimiento de direcciones. Suele resultar necesario también capacidades de persistencia en base de datos para poder realizar análisis de rutas y, eventualmente, dependiendo de la aplicación, la integración con información de las bases de datos corporativas. En relación con las tecnologías utilizadas para el posicionamiento, son varias las disponibles, pero es la tecnología basada en GPS [KAPL96] la que se ha impuesto en el mercado. Según datos de 1993 era ya utilizada por un 52% de los sistemas [KRAK93] y en la actualidad es utilizada ya por encima del 66% de los sistemas de control de flotas [KRAK96].

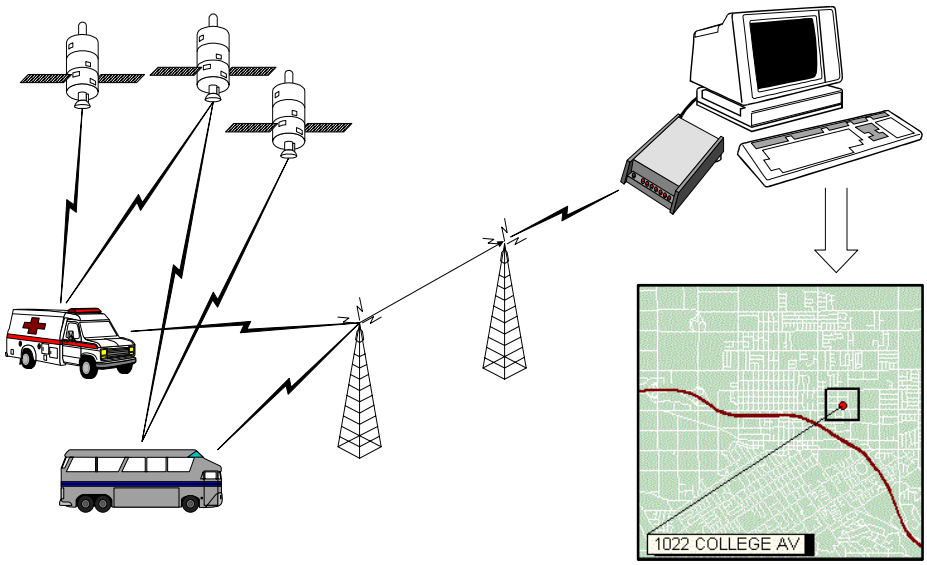

Figura 1: Sistema de navegación IVHS de tipo gestión de flotas.

La necesidad de optimizar los costes de los servicios y el incremento de la competencia en el sector del transporte ha inducido a muchas empresas a interesarse por estos sistemas de seguimiento y control de flotas. Así en los últimos años han surgido en España sistemas comerciales básicos (p.e. [BARR97]) y diversos proyectos de este tipo en los sectores de la paquetería, taxis, autobuses urbanos y de largo recorrido, así como en servicios públicos, fundamentalmente de protección civil (policía, bomberos, sanidad, …). Además este interés no ha estado limitado al transporte por carretera sino que ya hay sistemas en marcha para el sector naval y el transporte ferroviario [RP97].

Todo este extenso dominio de aplicaciones requiere una serie de servicios básicos comunes (adquisición, análisis, visualización, comunicaciones, …), sin embargo, los requisitos y necesidades específicas del cliente que han de ser satisfechas es muy variable en cuanto a magnitud de los recursos, máquinas, funcionalidad, etc., lo que exige generalmente soluciones “ad hoc”.

La estrategia elegida por nuestro grupo de I+D al introducirnos en este campo, ha consistido en desarrollar una infraestructura flexible que soporte las funcionalidades básicas y que posibilite la fácil integración de dichas funcionalidades en las aplicaciones. Por este motivo nos hemos planteado una arquitectura de diseño flexible donde aspectos como escalabilidad, interoperabilidad, sustituibilidad, extensividad y reusabilidad resultan de gran

importancia. La aproximación tecnológica adoptada se guía por los principios de sistemas abiertos en base a una arquitectura distribuida cliente/servidor utilizando las nuevas tecnologías de desarrollo de software orientado a objeto. En este sentido, nos hemos apoyado en el estándar CORBA, que reúne de manera muy adecuada los tres aspectos requeridos: orientación a objetos, distribución y arquitectura cliente/servidor.

Junto a CORBA, Java supone el complemento perfecto a CORBA. CORBA ofrece una infraestructura que permite la interoperabilidad entre aplicaciones a través de lenguajes, sistemas operativos y redes. Java ofrece una infraestructura para objetos portables a cualquier máquina y de esta forma podemos hacer que CORBA sea ubicuo en la red. Las aplicaciones cliente son *applets* Java que se pueden portar a cualquier máquina para posteriormente comunicarse con el servidor a través de CORBA.

El objetivo de este trabajo es presentar la motivación e ideas básicas de un sistema de información distribuido orientado a objeto para el seguimiento y control de móviles. El trabajo presenta la arquitectura de software adoptada y se centra en la infraestructura de objetos distribuidos a través de un bus CORBA. En este contexto se analizan las ventajas que se consiguen por medio de esta infraestructura de distribución y en las facilidades que esto aporta para la integración y expansión del sistema.

2. Panorámica de OODISMAL

OODISMAL (Object Oriented Distributed Information System for Automatic Movil Location) es un sistema de información que consiste en un conjunto de componentes distribuidos en una red de área local. Estos componentes interactúan entre sí utilizando el estándar CORBA y tienen como propósito la adquisición, almacenamiento y procesamiento de datos de localización provenientes de móviles que incorporen un dispositivo GPS.

Para ello, se definen distintos objetos CORBA, a través de los cuales se realizará cualquier comunicación que sea necesaria entre los componentes. De esta manera, el interfaz entre los componentes queda definido claramente, lo que permite reducir el acoplamiento entre ellos. Cada componente funcionará como servidor de los objetos CORBA que contenga y utilizará como cliente aquellos objetos CORBA que necesite de otros componentes. Por ejemplo, existen componentes que ofrecen información de localizaciones. Estos componentes funcionan principalmente como servidores de objetos que permiten el acceso a esta información. En cambio, los componentes que hagan uso de esta información para visualizarla o analizarla, utilizarán estos objetos como clientes. Un componente puede ser servidor y cliente de tantos tipos de objetos como se requiera, sin ningún tipo de restricción.

Según su funcionalidad, los componentes del sistema se pueden clasificar en:

- **Servicios**, que constituyen las utilidades básicas como:
    - adquisición de datos de los móviles a través de enlaces de comunicación (actualmente se dispone del software para la adquisición de datos de *GPS* vía radiotelefonía *trunking* en base al terminal T500 voz con opción GPS desarrollado por Teltronic),
    - utilidades para mejoras de precisión y de ajuste a rutas,
    - análisis de rutas (relacionados con distancias, tiempos, velocidades, posiciones) y correlación entre rutas reales y previstas,
    - gestión de persistencia,
    - simulación de móviles para depurar otras aplicaciones y hacer demostraciones,
    - interfaz gráfico de usuario para el acceso a los datos (estas visualizaciones pueden hacerse también por *internet*),
    - visualización de datos en mapas digitales (estas visualizaciones pueden hacerse también por *internet*).

Estos servicios proporcionan tanto las utilidades para el funcionamiento en *tiempo real* como el basado en datos almacenados.

- **Aplicaciones de usuario**, que constituyen soluciones integradoras con objetivos más especializados del cliente. Actualmente se está trabajando en las siguientes aplicaciones:
    - Un sistema de información de ayuda a los usuarios de autobuses urbanos de Zaragoza.
    - Un sistema de ayuda a la toma de decisiones para la asignación de taxis en base a peticiones telefónicas.
    - Un sistema de gestión y monitorización de alumbrado público.
    - Un sistema de monitorización de ascensores.

La distribución implica distintos procesos de ejecución que pueden estar residentes en la misma máquina o en máquinas distintas. De esta forma resulta fácil ajustar el sistema a distintas magnitudes de las necesidades de los

usuarios, así el sistema se puede ir escalando incrementando el número de máquinas para ajustarse a las prestaciones y puestos de trabajo deseados.

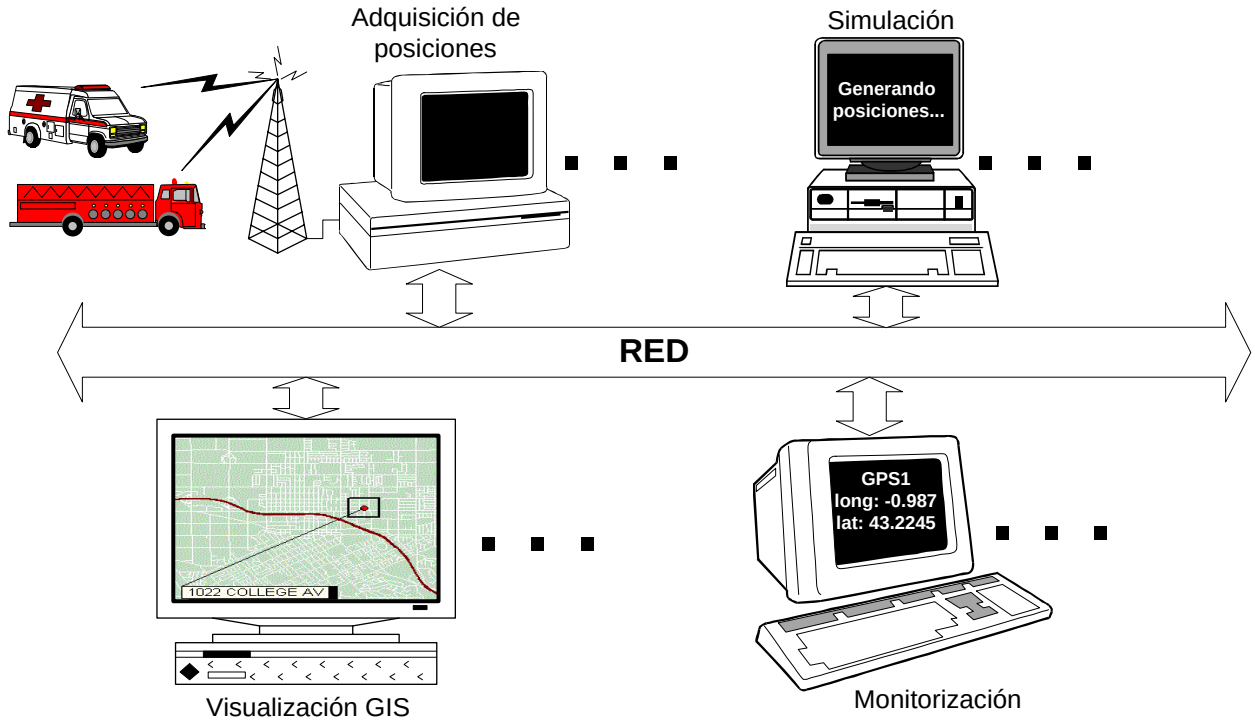

Figura 2: Arquitectura de aplicaciones

Esta tecnología distribuida posibilita el desarrollo de aplicaciones ligeras donde se resuelve la problemática específica de la aplicación mientras que están distribuidas el resto de funcionalidades y servicios.

Además, la arquitectura propuesta proporciona grandes posibilidades de escalamiento del sistema para ajustarlo a las peculiaridades del cliente.

Otro aspecto de interés lo constituye las facilidades para el trabajo conjunto de software heterogéneo. Dicha hetereogeneidad no sólo radica en la posibilidad de tener software corriendo en distintos sistemas operativos, lenguajes de programación y distintos tipos de máquinas físicamente distribuidas, sino que sería posible también integrar en el sistema el uso de eventuales aplicaciones ya disponibles y que no habían sido desarrolladas pensado en esta integración.

3. Arquitectura de OODISMAL como sistema distribuido orientado a objeto

3.1 Utilización del estándar CORBA

El paradigma de programación orientada a objeto proporciona técnicas de análisis, diseño e implementación para la construcción de software que es extensible, reusable y menos costoso de producir y mantener que el software orientado a la función. CORBA es una especificación para una arquitectura orientada a objeto distribuida estándar para aplicaciones. CORBA fue definido por el Object Management Group (OMG) [OMG96b] y actualmente existen varias implementaciones proporcionadas para los entornos y lenguajes más representativos.

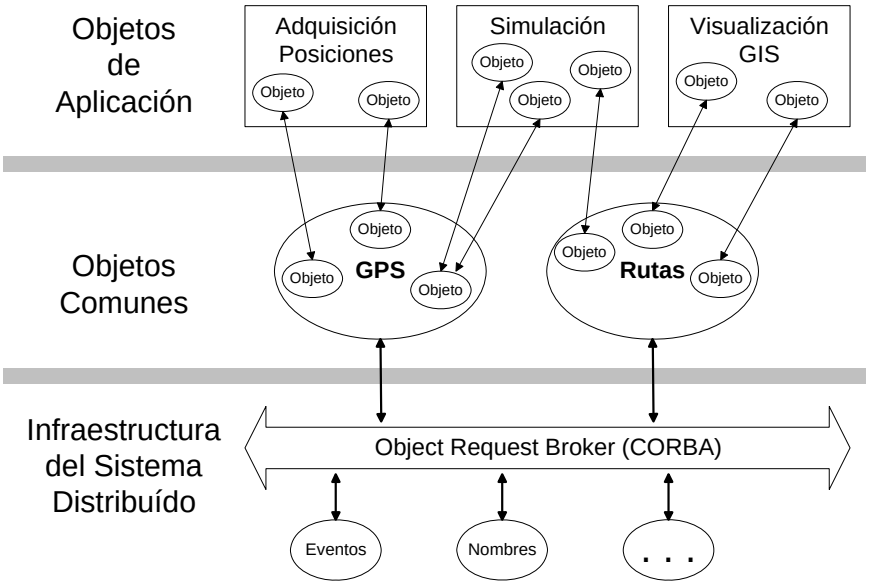

Figura 3: Arquitectura del entorno organizada por niveles

Por medio de la computación distribuida dos o más piezas de software pueden compartir información. Dichas piezas pueden estar ejecutándose en el mismo computador o en diferentes computadores conectados a la misma red. La mayor parte de la computación distribuida está basada en el modelo cliente/servidor (un software cliente, que requiere informaciones o servicios y, un software servidor, que proporciona la información o servicio).

La Figura 3 muestra las líneas básicas de la arquitectura de software de OODISMAL para soportar la distribución de objetos, en el que se ha adoptado una aproximación basada en el estándar CORBA. La arquitectura está formada en base a tres niveles de responsabilidad:

- La **infraestructura** proporciona el entorno de computación distribuido para la aplicación. Estos servicios incluyen comunicaciones, almacenamiento persistente de datos, interfaz de usuario, distribución de eventos, gestión de excepciones, etc. OODISMAL utiliza la infraestructura de servicios y facilidades definidas por la Arquitectura de Gestión de Objetos de OMG (OMA), cuyos principales elementos son:
    - Un modelo de objetos común que hace que los distintos componentes aparezcan como objetos distintos en el sentido clásico, con una identidad y un estado propio, y no sólo como un conjunto de servicios. CORBA utiliza un lenguaje declarativo denominado IDL *(Interface description language)* para describir el interfaz de las aplicaciones, permitiendo separar la interfaz de los objetos de su implementación. Los programadores tratan con los objetos CORBA utilizando su lenguaje nativo (C, C++, Ada, Java, COBOL), mientras que las interfaces de los servicios y componentes que residen en el bus CORBA vienen definidos por sus IDLs. Las figuras 5 y 6 muestran ejemplos de interfaces definidos como IDLs. Se puede apreciar el parecido a la definición de clases C++ o interfaces CORBA. La inclusión de tipos predefinidos en CORBA asegura la interoperabilidad entre diferentes plataformas. En la figura 6 también se puede apreciar como se especifica la herencia entre interfaces en CORBA.
    - El Object Request Broker es el bus CORBA que permite invocaciones remotas de objetos como si fueran locales, manejando los detalles de la red sobre la que se ha distribuido el sistema. Incluye la transmisión de parámetros y resultados de operaciones, así como la activación de las aplicaciones servidoras cuando sean requeridas.
- Los **objetos comunes** constituyen las entidades funcionales comunes a través de las aplicaciones. Los objetos comunes proporcionan un modelo común para estas entidades, posibilitando un desarrollo e integración más rápida. Estos objetos especifican datos y comportamiento requerido para la interoperabilidad entre las aplicaciones. En el caso de OODISMAL, estos objetos proporcionan la infraestructura para compartir datos de localización en tiempo real e informaciones sobre rutas almacenadas en soportes persistentes.
- Los **objetos de aplicación** proporcionan funcionalidades de aplicación específicas. Estos objetos definen datos y comportamiento construidos sobre los datos y comportamientos comunes, lo que permite a la aplicación

interoperar con otras aplicaciones. Los objetos de aplicación definen también las reglas del negocio (business rules) y el interfaz de usuario para las aplicaciones. En nuestro caso corresponden con componentes especializados en la adquisición, visualización, simulación, disponibilidad para internet y distintas aplicaciones para gestión de flotas y logística.

3.2 Patrón de diseño sujeto-observador

El funcionamiento del sistema de seguimiento se basa principalmente en la existencia de objetos que representan móviles con GPS, los cuales tienen datos acerca de la posición en la que se encuentra el móvil. Esta información puede utilizarse con diferentes propósitos como realizar visualizaciones sobre mapas geográficos, construcción de rutas, etc. Es preciso que la información utilizada por cualquiera de los clientes de esta información sea consistente y se actualice en cuanto cambie la posición de un móvil. Tenemos una relación de dependencia uno a muchos, de forma que cuando un objeto cambia su estado, todos los objetos que dependen de éste deben ser informados.

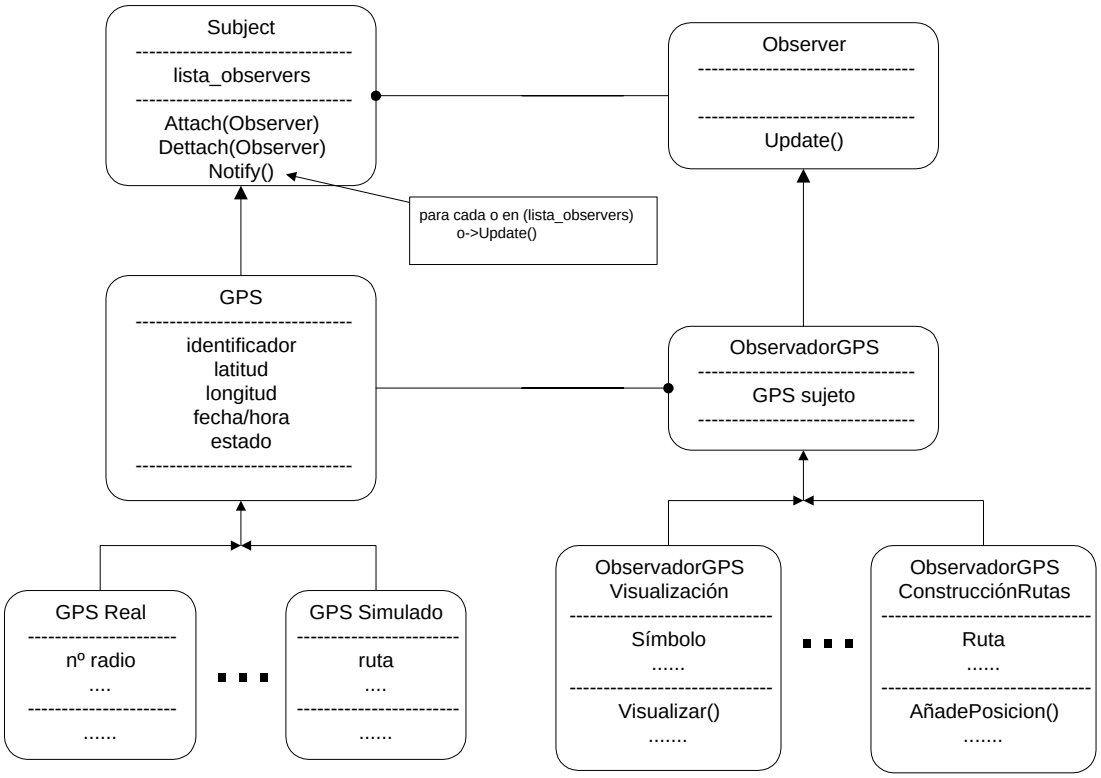

Figura 4: Modelo de objetos sujeto-observador

Una gran ayuda en el diseño del sistema ha sido la revisión de patrones de diseño que nos permitieran reusar la experiencia de diseño. Los patrones de diseño presentados en [GHJV94] son diseños orientados a objetos útiles, reusables y flexibles. En el sistema de seguimiento la mayor parte de las interacciones entre clientes y servidores dentro del sistema se ajustan al patrón sujeto-observador que describe la situación presentada.

Según este patrón, se identifican a los objetos de los que interesa conocer sus datos como **sujetos**, y objetos que desean consultar dichos datos y ser avisados cuando se produzca algún cambio en ellos como **observadores**. Los objetos que muestran el estado de los móviles en componentes de visualización o monitorización siguen este patrón siendo avisados automáticamente cuando se produce un cambio en la posición del móvil al que observan. El modelo de objetos se puede ver en la figura 4 (se sigue la notación de J. Rumbaugh [RUMB91]), y los IDLs que especifican la interfaz de estos objetos se muestran en las figuras 5 y 6. Este patrón consta de dos niveles:

Al nivel más alto están los objetos sujeto y observador (subject y observer). En este nivel se establece el funcionamiento general del patrón sujeto-observador:

- El sujeto *(Fig. 5)* tiene conocimiento de cuales son sus observadores y tiene servicios para que los observadores se den de alta o de baja como tales. Además tiene un servicio Notify(), que es el encargado de notificar a cada uno de sus observadores que se ha producido un cambio.
- El observador *(Fig. 6)*, en cambio, no sabe quién es su sujeto a este nivel del modelo. Tiene un servicio Update() por medio del cual se le notificarán los cambios producidos en el sujeto. Este servicio es sobrescrito en los niveles inferiores según el comportamiento que se quiera tener en cada caso

En un segundo nivel, este patrón genérico se concreta para unos datos determinados, en este caso datos de localización de GPS.

- El objeto GPS *(Fig. 7)* representa el dispositivo GPS conectado a un determinado móvil y contiene información acerca de la posición captada por el dispositivo, el momento en que se tomó dicha posición y otros datos de estado del GPS.
- El observador de GPS sí conoce cuál es su sujeto y sabe consultar su información.

En el sistema se ha añadido un tercer nivel para tener objetos GPS que consiguen su información a través de distintos métodos (por comunicación a través de radio, por simulación de una ruta almacenada, etc…) y observadores que tratan la información de distintas maneras (para visualizarla, para construir rutas, etc…).

A través de CORBA se distribuyen los objetos GPS del segundo nivel. Esto quiere decir que un objeto GPS que se comunica por radio y un objeto GPS simulado a partir de una ruta se distribuyen en el sistema como objetos GPS genéricos. Por eso, para los observadores ambos objetos GPS son del mismo tipo, quedando ocultas las distintas implementaciones. Esto permite que el sistema sea más heterogéneo, y en consecuencia, también más flexible y escalable. De manera parecida, los observadores sólo son objetos CORBA al nivel más alto, de manera que un sujeto GPS puede tener observadores de cualquier tipo.

```
// Sujeto generico. Debe almacenar la lista de observadores
interface Subject {

  exception Failure { string reason; };
  // Excepcion general de fallo de funcionamiento.
  // Utilización: throw Subject_Failure("mensaje").

  unsigned long Attach(in Observer obs);
  // Conecta un observador al sujeto.
  // Devuelve un numero que debe ser almacenado en el atributo
  // number del observador

  void Dettach(in unsigned long obsnum) raises (Failure);
  // Desconecta el observador del sujeto.
  // Compara los observadores a traves de sus atributos number.

  void Notify();
  // Notifica a los observadores que los datos del sujeto han cambiado
  // Debe ser ejecutado cada vez que se realicen cambios en los datos
  // del sujeto.
  // Ejecuta el servicio Update de cada uno de sus observadores
};
```

Figura 5: IDL del sujeto genérico

```
// Observador generico.
```

```
// No tiene asociado el sujeto (esto es propio del observador concreto)
interface Observer {

  unsigned long IdNumber();
  // Almacena el numero identificativo devuelto por el sujeto al hacer Attach

  oneway void Update();
  // Este servicio es ejecutado por el sujeto cuando realiza un Notify.
  // Informa al observador que los datos del sujeto han cambiado.
  // Debe ser virtual, el observador concreto realizara la operaciones
  // que considere oportunas.
};
```

Figura 6: IDL del observador genérico

```
#include "subject.idl"

enum TpEstadoGPS {
                POSICION_VALIDA,        // La posición es válida
                NO_SIGNAL_GPS,          // No hay señal de GPS
                ERROR_ELEVADO,          // El error (PDOP) es elevado
                SIN_SATELITES,          // No hay satélites en cobertura
                UN_SATELITE,        // Sólo hay un satélite en cobertura
                DOS_SATELITES,          // Sólo hay dos satélites en cobertura
                TRES_SATELITES,         // Sólo hay tres satélites en cobetura
                SATELITE_NO_UTIL,       // El satélite no es útil
                GPS_SIMULADO };         // El GPS es simulado

struct TpDate {
  unsigned short Year;
  unsigned short Month;
  unsigned short Day;
  unsigned short Hour;
  unsigned short Minute;
  unsigned short Second;
  unsigned short DayOfWeek;
};

struct TpInfo {
  double latitud;
  double longitud;
  TpDate hora;
  unsigned short velocidad;
  short rumbo;
  TpEstadoGPS estado;
};

interface GPS:Subject {
    readonly attribute string identificador;
    readonly attribute string parametro;
    readonly attribute boolean activo;
    readonly attribute unsigned short periodo;
    readonly attribute unsigned short umbral;
    // Tienen servicios especiales de modificación
    attribute TpInfo info;
    // Información propia del GPS.
    // Para configurar el GPS en su periodo y umbral se deben utilizar
    // los servicios específicos setPeriodo y setUmbral.
```

```
    // Cuando se modifica la información se ejecuta el servicio
    // Notify de forma automática.

    // Consulta individual de la información
    double latitud();
    double longitud();
    TpDate hora();
    unsigned short velocidad();
    short rumbo();
    TpEstadoGPS estado();

    oneway void setPeriodo(in unsigned short p);
    oneway void setUmbral(in unsigned short u);
    // Servicios de configuración.
    // NO se ejecuta automaticamente el servicio Notify

            void activar(in boolean a);
            // intenta activar o desactivar el GPS

            oneway void Actualizar();
    //      Solicita que se actualicen los atributos del GPS con la
    //      información actual del mismo. Cuando ésta se actualiza
    // realmente, se ejecuta automáticamente el sevicio Notify
    // del Subject.
};
```

Figura 7: IDL del Sujeto GPS

3.3 Utilización de los servicios CORBA

Además de la infraestructura aportada por el bus CORBA para la comunicación de objetos a través de la red, es importante contar con mecanismos que nos permitan localizar los servicios [OMG96a]. Así, las aplicaciones adquieren una mayor independencia, ya que no tienen que conectarse a ordenadores concretos, sino que pueden interactuar entre ellas sin tener en cuenta donde se hayan ubicadas. CORBA ofrece además de la infraestructura básica otros servicios como el que permite buscar objetos a lo largo del sistema sin que sea necesario conocer en qué máquina se encuentran físicamente.

En el sistema de seguimiento, una aplicación cliente puede conectarse simultáneamente a uno o varios servidores y observar objetos GPS de distintas fuentes de forma totalmente transparente al usuario. De la misma manera, un mismo objeto GPS puede tener varios observadores que se encuentren en distintas máquinas y que realicen tareas distintas con la información adquirida, sin que el propio objeto GPS deba ser consciente de ello. Aparte de los objetos sujeto GPS, se exportan también objetos que gestionan dichos sujetos y servidores de otros tipos, como aquellos que ofrecen servicios de persistencia, etc.

4. Estructura de los distintos componentes

A continuación se muestran algunos de los componentes presentes en OODISMAL:

4.1 Servicio de Adquisición de Datos

Este componente es el encargado de adquirir datos de posición de los móviles a través de una comunicación por radio trunking. Este componente interactúa con una radio base conectada al puerto serie del ordenador en el que se ejecuta. A través de dicha radio, se comunica con los móviles y sus dispositivos GPS y recibe periódicamente datos de localización provenientes de los mismos. Para hacer públicos dichos datos, este componente ofrece al bus CORBA los objetos sujeto GPS que representan los dispositivos conectados a los móviles. Los clientes de este componente se conectan primero con un objeto gestor de GPS que les ofrece una lista con los objetos GPS disponibles, para que los clientes puedan conectarse a ellos como observadores.

4.2 Servicio de Simulación de móviles

En algunos casos resulta interesante hacer funcionar el sistema utilizando información que no proviene de móviles reales, sino de posiciones grabadas con anterioridad. Esto puede ser útil para comprobar el buen funcionamiento del resto de los componentes o para hacer demostraciones. Para esto se utiliza el componente de simulación.

Este componente es muy parecido al anterior, en la forma en que exporta los datos de localización. La diferencia estriba en que este componente no ofrece datos adquiridos de móviles reales, sino que realiza una simulación basada en rutas grabadas previamente. Sin embargo, este hecho resulta transparente a los clientes, ya que el mecanismo de exportación de datos y el tipo de objetos que distribuye es idéntico al anterior.

4.3 Componente de Visualización de Móviles sobre un sistema GIS

Una de las funcionalidades más obvias y necesarias que se pueden requerir al sistema es la visualización, en tiempo real, de las posiciones en las que se encuentran los distintos móviles. Este componente proporciona esta y otras funcionalidades interesantes, como el almacenamiento de dichas posiciones en una base de datos y su posterior visualización.

Este componente funciona como cliente de los servicios anteriores y realiza una visualización de la localización de los móviles a los que observa sobre mapas base de las zonas en las que se encuentren.

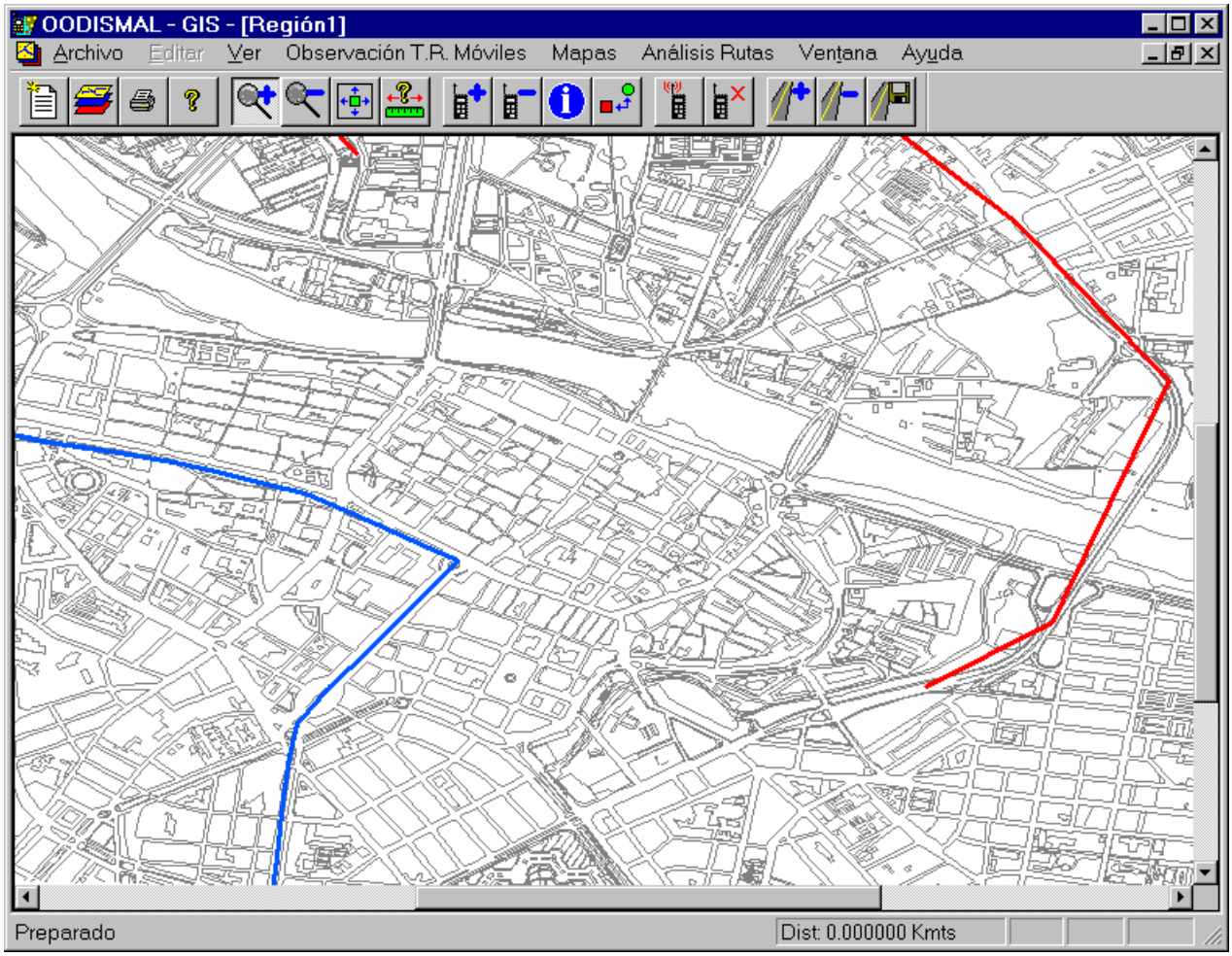

Figura 8: Apariencia del componente de visualización GIS

El componente se conecta a uno o varios servidores de objetos GPS y muestra al usuario una lista de los móviles que se pueden visualizar. El usuario puede entonces visualizar los móviles que desee, almacenar las rutas que siguen en soporte persistente e incluso realizar llamadas de voz a las radios que se encuentran en dichos móviles, si éstos no son simulados. Para realizar el seguimiento de estos móviles, el componente crea observadores a cada uno de ellos. Estos observadores son de distintos tipos según se desee visualizar el móvil o almacenar la ruta que ha seguido. Como es natural, el componente puede asignar un número indeterminado de observadores de cualquiera de

los dos tipos a un mismo sujeto GPS, ya que puede desear visualizar al móvil en distintas ventanas, almacenar distintos fragmentos de la ruta que sigue, etc. Gracias a la arquitectura distribuida del sistema, esto puede realizarse sin ninguna complicación.

4.4 Componente de Supervisión y Acceso

Este componente, realizado en lenguaje Java, permite observar distintos objetos GPS de distintos servidores, de forma análoga a como se realiza en el componente anterior. En este caso, la visualización se produce en una ventana que muestra los datos del móvil. Este componente no sólo permite visualizar dichos datos, sino que además da la posibilidad de configurar distintos aspectos del objeto GPS, como el período de actualización de los datos.

Java ofrece una infraestructura para objetos portables a cualquier máquina. El beneficio de esta portabilidad es doble. Por un lado el *Abstract Windowing Toolkit* de Java nos permite desarrollar interfaces que son independientes de cualquier máquina. Por otro lado, la portabilidad del código Java nos permite descargar aplicaciones clientes y la infraestructura CORBA en cualquier máquina y de esta forma podemos hacer que CORBA sea ubicuo en la red. Las aplicaciones cliente son *applets* Java que se pueden portar a cualquier máquina mostrando la interface la apariencia de típica de la máquina en la que se ejecuta, para posteriormente comunicarse con el servidor a través de CORBA. Además, las últimas versiones de Netscape incluyen las clases Java que soportan la infraestructura CORBA, por lo que basta con portar el código de la clase cliente.

Al estar implementado en lenguaje Java, este componente puede ser utilizado a través de Internet, aumentando enormemente la capacidad de interacción y escalabilidad del sistema.

4.5 Otros componentes

Además de los componentes descritos, existen otros que añaden funcionalidades adicionales al sistema. En general, estos componentes realizan tareas que no están relacionadas con la adquisición de información, sino con el postproceso de la misma, por lo que no se ajustan al patrón sujeto-observador. Entre estos componentes se encuentran, por ejemplo, los servicios de persistencia, componentes de análisis de rutas, etc.

Por encima de todos estos componentes, se construyen aplicaciones que hacen uso de ellos en un dominio de aplicación específico.

5. Conclusiones y expectativas

El trabajo ha presentado una arquitectura distribuida orientada a objeto para el desarrollo de un sistema de localización de móviles. Esta aproximación, además de proporcionar las ventajas de ingeniería del software provenientes de las aproximaciones orientadas a objeto, proporciona un alto grado de flexibilidad y reusabilidad al permitir fácilmente un gran rango de desarrollo de sistemas, desde pequeños basados en un único computador personal, hasta la integración de varios computadores personales y/o estaciones de trabajo basadas en UNIX. En este sentido, el estándar CORBA nos ha proporcionado varias ventajas substanciales:

- Como puede observarse, esta arquitectura permite un desarrollo flexible, fácilmente escalable y muy versátil, pudiendo utilizarse para una amplia gama de aplicaciones.
- Es posible añadir nuevas funcionalidades a la aplicación sin necesidad de modificar lo desarrollado.
- Esta infraestructura distribuida puede ser, además, ampliada para su utilización en otros campos, y no solamente en el seguimiento de móviles. Así, por ejemplo, las componentes de comunicación por radio trunking nos permite abordar aplicaciones de monitorización y control de alarmas. De esta forma los servicios desarrollados en unas aplicaciones pueden ser fácilmente integrados en otras, e incluso ser ofertados como valores añadidos.
- El avance de estas tecnologías está permitiendo el desarrollo de sistemas a precios muy asumibles por una creciente gama de clientes institucionales, industriales y privados, con unas utilidades que hace muy pocos años sólo podían ser desarrollados por empresas líder en tecnología y comprados por clientes con grandes presupuestos.

En la actualidad se están desarrollando entre otras aplicaciones un sistema de información para líneas de autobuses interurbanos que, aprovechando esta infraestructura, la utilicen para dar servicios de seguimiento de la flota de autobuses, además de aplicaciones de gestión, como billetaje o facturación.

6. Bibliografía

[BBH94] A. Bethmann, K. Brocke, S. Harrer. “Sistema automático de localización de vehículos”. *Comunicaciones Eléctricas*, pp. 129-135. 2º trimestre, 1994.

[BARR97] E. Barrios. “Seguimiento Competitivo de Vehículos”. *RedesLan*, pp. 110-111. Febrero, 1997.

[GHJV94] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Desing Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Publishing Company. 1994.

[KAPL96] E.D. Kaplan (ed.). *Understanding GPS Principles and Applications*. Artech House Publishers. 1996.

[KRAK93] E.J. Krakiwsky. “Tracking the Worldwide Development of IVHS Navigation Systems”. *GPS World*, V 4, N 10, pp. 40-47. October, 1993.

[KRAK96] E.J. Krakiwsky. “Driving ITS Development: Technology and Market Forces”. *GPS World*, V 7, N 10, pp. 50-55. October, 1996.

[OHE96] R. Orfali, D. Harkey. *Client/Server Programming with Java and CORBA*. Wiley Computer Publishing. 1997.

[OMG96a] Object Management Group. *CORBA Services: Common Object Services Specification*.. Framingham, MA: Object Management Group, July, 1996.

[OMG96b] Object Management Group. *The Common Object Request Broker: Architecture and Specification, Revision 2.0*. Framingham, MA: Object Management Group, July, 1996.

[OMG95] Object Management Group. *Object Management Architecture Guide, Version 3.0*. Framingham, MA: Object Management Group, 1995.

[RP97] J. Roca, J.L. Pérez. “All Aboard!. On track with Catalonia´s Trains”. *GPS World*, V 8, N 6, pp. 20-28. June, 1997.

[RUMB91] J. Rumbaugh et al. *Object Oriented Modeling and Design*. Englewood Cliffs, N.J.: Prentice Hall, 1991.

[VINO97] S. Vinoski. *CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments*. *IEEE Communications Magazine*, pp 46-55. Feb. 1997.