(Draft) Actas de las VI Jornadas de Concurrencia. 1998, p. 129-136

Proporcionando capacidades de trabajo concurrente a aplicaciones GIS sobre un entorno distribuido basado en CORBA[1]

S. Comella, J. Ezpeleta, F.J. Zarazaga, J. Valiño, P. Muro-Medrano (1)

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 ponen de manifiesto algunas nuevas posibilidades que surgen del empleo de las tecnologías de objetos distribuidos, basados en el estándar CORBA, para solucionar problemas de interoperabilidad entre aplicaciones heterogéneas. Concretamente, se ilustra la solución de un problema específico de concurrencia para el acceso a un repositorio de datos común sobre un gestor de base de datos relacional, sobre el que trabajan aplicaciones de sistemas de información geográfica que precisan sus propios formatos y gestión de datos. Los problemas de integridad de la base de datos y el refresco automático en las aplicaciones GIS son analizados y se proponen soluciones de diseño. También se tratan algunos patrones de diseño aplicables en programación distribuida en general, que nos han resultado útiles para estos problemas.

1. Introducción

Las organizaciones que trabajan en el campo de la información geográfica precisan de unos sistemas de información que presentan unas peculiaridades y problemas diferentes a los de otras áreas. Particularmente, en el modelo de datos de estas organizaciones coexisten dos clases de entidades:

1. Por un lado están los datos georreferenciados ("features") definiendo georreferencia como "una relación unívoca entre un elemento de información y una localización geográfica". Estos datos son comúnmente almacenados y representados mediante herramientas SIG (Sistemas de Información Geográfica).
2. Por otra parte nos encontramos con un conjunto de entidades en las que no prima su carácter geográfico (que puede incluso no existir). La estructura de estos datos se ordena, normalmente, atendiendo a las relaciones entre ellos. Es común seguir un modelo relacional para su almacenamiento y representación.

Estos dos conjuntos de datos no son disjuntos; al contrario, hay entidades que mantienen una serie de relaciones con otras y por lo tanto pertenecen al ámbito relacional. Pero, además, también están georreferenciadas. Los SIG, normalmente, sólo son capaces de representar relaciones geométricas (inclusión, intersección...); así pues, si se quiere captar todo su significado semántico, estas entidades deben ser almacenadas en el modelo relacional. Pero también se deben poder visualizar y tratar con herramientas SIG. A estas entidades las denominaremos relacionales-georreferenciadas. Estas entidades poseen una representación dual: por un lado tienen persistencia en el modelo relacional; por otro lado también se pueden procesar en el subsistema geográfico, lo que conlleva dos formatos de representación para los mismos datos. Esta doble representación requiere programas de traducción que no solo consumen tiempo, sino que además causan diversos problemas, muy especialmente de integridad cuando trabajamos en entornos multiusuario.

[1] Este trabajo ha estado parcialmente financiado por el CONSYD de la Diputación General de Aragón a través del proyecto P-18/96.

Para salvar esta "impedancia" entre el subsistemas geográfico y relacional de un sistema de información es preciso hacer ambas partes interoperables. Interoperabilidad es un termino usado tan extensamente en la industria informática que ha perdido prácticamente su significado. Para Cliff Kottman, de Intergraph Corporation, interoperabilidad es la libertad de mezclar componentes en un sistema de información sin comprometer el éxito global.

Uno de las mayores dificultades para conseguir esta deseada interoperabilidad es la de los accesos concurrentes. El subsistema relacional, normalmente implementado sobre una Base de Datos Relacional (BDR), mantiene un eficiente control de concurrencia (definido este como un conjunto de técnicas que intentan evitar que los resultados de la ejecución de un programa sean incorrectos, incoherentes o se pierdan debido a la ejecución simultanea de otro programa que actúa sobre los mismos datos [Cam-84]); sin embargo, las herramientas SIG actuales, poseen un control de accesos simultáneos mucho más rudimentario, el problema se agrava al trabajar desde una herramienta SIG con datos que tienen persistencia en el modelo relacional. Se hace necesario, por lo tanto, implementar un control global de accesos concurrentes que coordine las políticas de concurrencia de ambos subsistemas.

Para que resulte más ilustrativa la explicación, el trabajo se centrará en un problema práctico específico donde se describe un escenario típico en el que coexisten los dos subsistemas antes presentados, así como los problemas que se presentan si no se establece una política de accesos concurrentes. Concretamente, el ejemplo consiste en un sistema de información sobre puntos de agua en el contexto de la gestión de cuencas hidrográficas.

2. Descripción detallada del problema.

Una organización, que gestiona una cuenca hidrográfica, cuenta con extensa información geográfica almacenada en coberturas[2], manteniendo además una base de datos relacional con información sore un basto conjunto de propiedades.

Estos dos conjuntos interseccionan en la entidad *Puntos de Agua* (PA), por una parte un PA es una característica geográfica y como tal debemos manipularla, por otra parte mantiene relaciones con otras entidades (subcuencas, municipios, análisis químicos, unidades hidrogeológicas, expedientes de tramitación, concesiones de explotación, …). En la solución "tradicional" estas relaciones obligan a dar persistencia a los PA en la BD relacional y establecer un mecanismo de traducción para poder procesarlos con las herramientas SIG, este mecanismo se divide en varias fases: en primer lugar se exportan la totalidad de los *puntos de agua* desde las tablas relacionales a una cobertura de *puntos de agua*, posteriormente se modifican/visualizan los *puntos de agua* en la herramienta SIG junto con el resto de los datos geográficos, por último, se exportan los cambios nuevamente a la base de datos relacional. Adicionalmente, es posible modificar directamente el inventario de puntos de agua desde la BD relacional.

Tal como ilustran los seis pasos de trabajo de la Figura 1, esta forma de operar presenta potenciales problemas de integridad semántica.

Con este método de trabajo los distintos subsistemas, aunque mantienen una forma básica de interacción, presentan problemas de interoperabilidad. Para ello resulta necesario desarrollar los mecanismos para coordinar, de un modo más estrecho, el trabajo de ambos subsistemas.

[2] En terminología SIG, "conjunto de datos de la misma clase (features) georreferenciados"

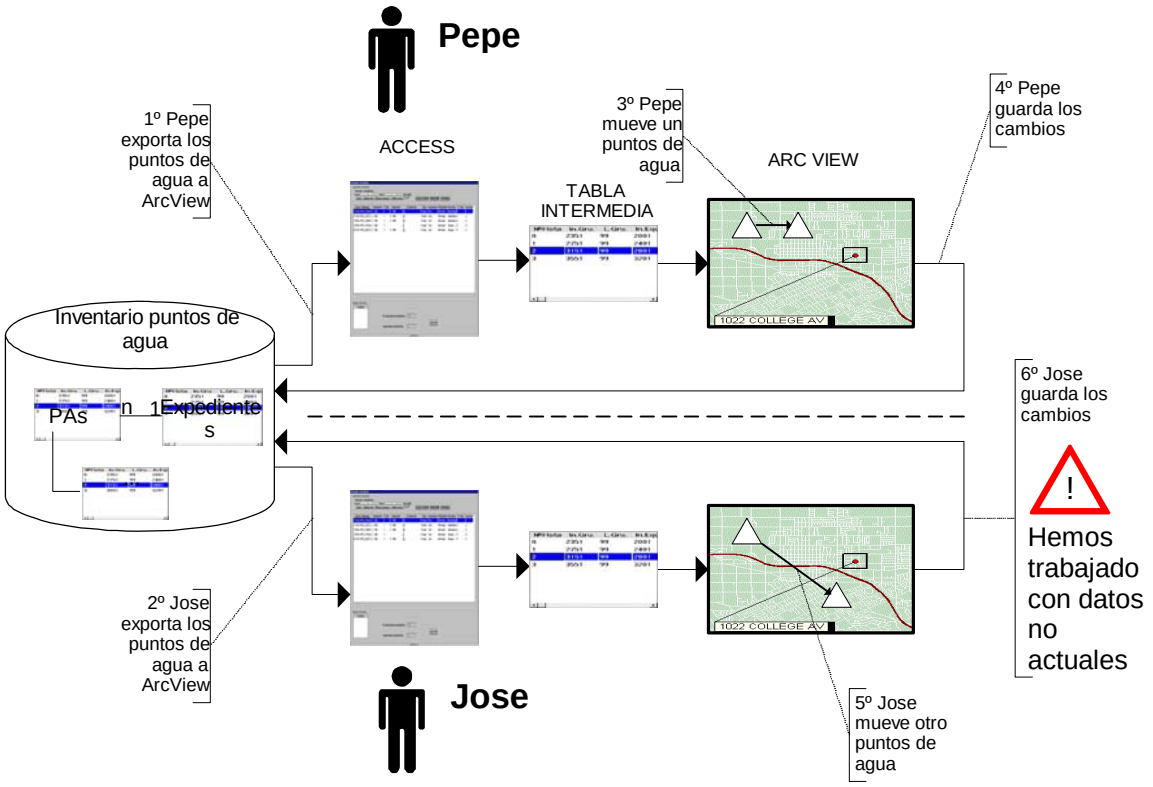

Figura 1: Problemática de acceso concurrente desde el SIG.

3. Estrategias para la interoperabilidad.

En la actualidad son numerosos los esfuerzos en la industria informática para conseguir la deseada interoperabilidad. De entre ellos hay dos que por su relevancia deben ser reseñados: OpenGIS® y CORBA®.

Open Geodata Interoperability Specification (OpenGIS u OGIS) [OCG-OSFSC, OCG-OAS] es una especificación de geoinformación para definiciones orientadas a objetos, que permitirá el desarrollo de geoprocesamiento auténticamente distribuido a través de grandes redes de datos, así como el desarrollo de aplicaciones geográficas realmente interoperables [SMB-96]. OpenGis proporciona:

1. Un modelo universal de datos y procesamiento espacio-temporal.
2. Una especificación para cada principal base de base de datos para implementar el modelo de datos OGIS.
3. Una especificación para cada gran modelo de procesamiento distribuido (incluyendo CORBA) para implementar el modelo de proceso OGIS.

El objetivo de interoperabilidad de OpenGIS es mucho más amplio que el que se pretende en este artículo. OpenGIS provee interoperabilidad a escala “galáctica” [OHE-97]. Esto es, todos los GIS en la red podrían interoperar, mientras que en el contexto presentado sólo se necesita interoperabilidad entre un GIS y una base de datos relacional, principalmente en términos de acceso concurrente.

The Common Object Request Broker Arquitecture (CORBA) [OHE-97, OMG-OMAG] es el producto de un consorcio llamado Object Management Group (OMG). CORBA facilita el desarrollo de sistemas distribuidos orientado a objetos de un modo independiente del sistema operativo, la plataforma y el lenguaje empleado.

Arquitecturalmente CORBA se distribuye en capas (Figura 2); en la más interna se encuentra el Object Request Broker (ORB) [OMG-CORBAS] que permite enviar y recibir, de un modo transparente, peticiones en un entorno distribuido; sobre el ORB los Object Services [OMG-COSS], una colección de servicios (objetos e interfaces) que soportan funciones básicas para usar e implementar objetos; por último las Common Facilities [OMG-CFS], una colección de servicios que comparten diversas aplicaciones, pero que no son tan básicos como los Object Services.

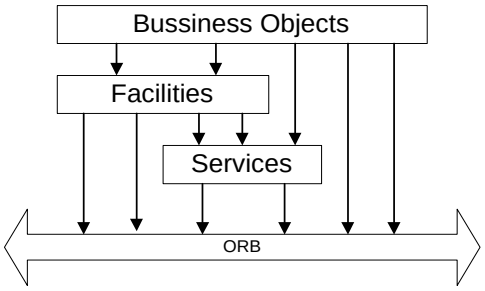

Figura 2: Arquitectura general de CORBA

Dos características de CORBA lo hacen especialmente interesante para resolver problemas de interoperabilidad.

1. Nos facilita un modelo OO distribuido, por lo que se puede emplear toda la potencia de este paradigma de programación extendiéndola a un ambiente distribuido.

2. CORBA está especialmente diseñado para asimilar y adaptar "legacy software" (software ya existente y con un funcionamiento satisfactorio) no necesariamente OO a través de una construcción denominada "wrapper" de la que hablaremos más adelante.

La solución que se presenta a continuación se basa en CORBA para implementar la política de accesos concurrentes y garantizar la interoperabilidad entre los dos subsistemas ya comentados.

4. Diseño de la solución.

Como primer paso para alcanzar la interoperabilidad, se debe dotar a la solución de un mecanismo flexible para salvar la diferencia de formato entre los PA (*Puntos de Agua*) como registro en una tabla relacional y los PA como elemento geográfico. La estrategia "tradicional" por fases, en la que la primera consiste en un la exportación total de los datos de un formato a otro, no es suficientemente potente para nuestras exigencias, ya que introduce un excesivo aislamiento entre un subsistema y otro: durante la manipulación de los PA nos imposibilita la comunicación entre la herramienta GIS y la BD relacional. Para salvar esta dificultad se introduce el componente **DataServer** (Figura 3) que no es sino una sesión del gestor de la BD relacional que sirve las peticiones de datos de las sesiones GIS. Se consiguen dos importantes objetivos; por un lado la transferencia de información es mucho más dinámica (no se interrumpe en ningún momento durante el proceso); por otro, cada sesión del gestor de la BD bloquea, cuando es necesario, los registros que emplea, esto incluye al **DataServer** que bloqueará los registros servidos a las sesiones GIS.

Es importante resaltar que los registros bloqueados por el **DataServer** son un recurso compartido para todas las sesiones GIS, son pues fuente de potenciales problemas de acceso concurrente. Se establece un mecanismo de bloqueos (que denominamos de segundo nivel) asociado a este recurso mediante el componente **LockManager** (Figura 3). Este componente es un repositorio/gestor de Bloqueos (**Lock**), que a su vez son objetos unívocamente asociados con un registro mediante un PID (Persistence Identifier), y que mantienen información sobre su propietario. Esta información se emplea para establecer comunicaciones (Figura 3, Figura 4, Figura 5).

Aunque CORBA posee ya un servicio de concurrencia [OMG-COSS, OHE-97], éste resulta demasiado "pesado" para los presentes requisitos, por lo que una versión simplificada como el **LockManager** se ajusta mejor a nuestras necesidades.

Atendiendo a la clasificación de las reservas en tipo X y tipo S [Cam-84], el sistema presentado sólo contempla bloqueos de tipo X o de escritura (Figura 3, Figura 4, Figura 5), lo que conlleva un nivel 1 de aislamiento entre procesos [Date-95]: Un proceso concurrente no actualiza elementos afectados por modificaciones no confirmadas por otros procesos. Este nivel de aislamiento, por si sólo, nos garantiza la integridad semántica. Sin embargo, exige un mecanismo que avise al resto de procesos de la confirmación de una actualización.

En CORBA esta necesidad está cubierta por el Servicio de Eventos [OMG-COSS, OHE-97], que permite a los objetos registrar su interés en eventos específicos. Un evento es la ocurrencia en un objeto de un hecho de interés para uno a más objetos. El servicio de eventos se basa en tres entidades: 1) el **Suplier** que produce eventos 2) el **Consume**r que procesa los eventos vía un "handler" y 3) el **EventChannel** (Figura 3), que es un objeto que es, a la vez, productor y consumidor de eventos. Permite a múltiples productores comunicarse con varios consumidores

asíncronamente y sin saber nada los unos de los otros. Un **EventChannel** es un objeto standard CORBA que se sitúa en el bus y desacopla la comunicación entre consumidores y productores.

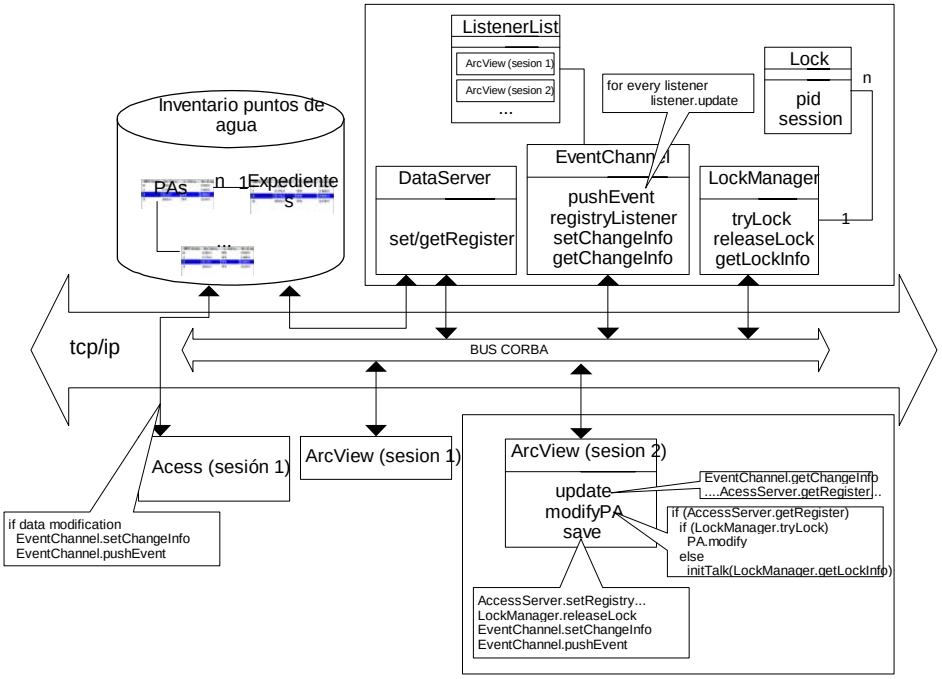

Figura 3: Diseño del sistema de control de bloqueos y refresco

5. Un escenario típico de funcionamiento.

En la siguiente secuencia de figuras esquematiza el funcionamiento del diseño presentado dinámicamente.

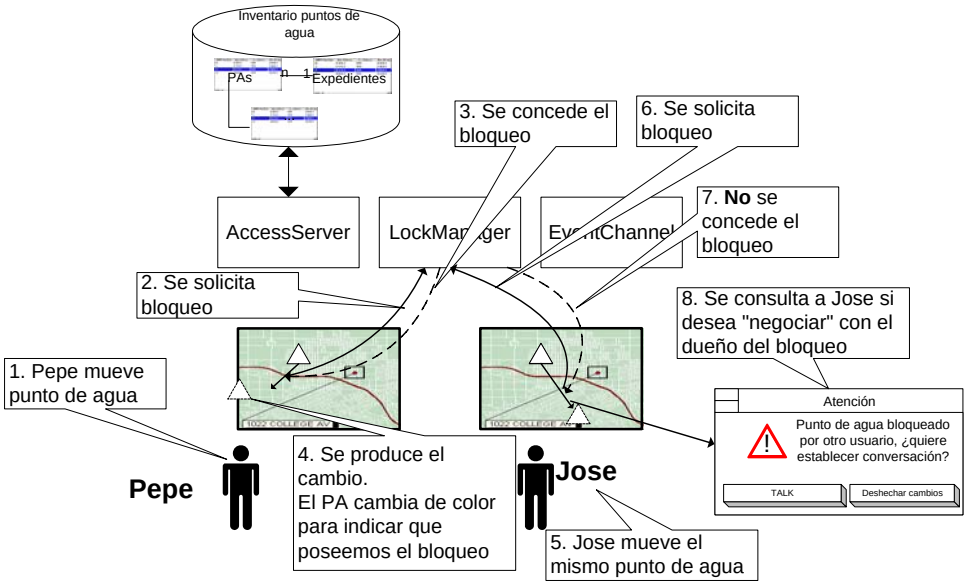

Figura 4: Modificación de un PA y bloqueo del mismo

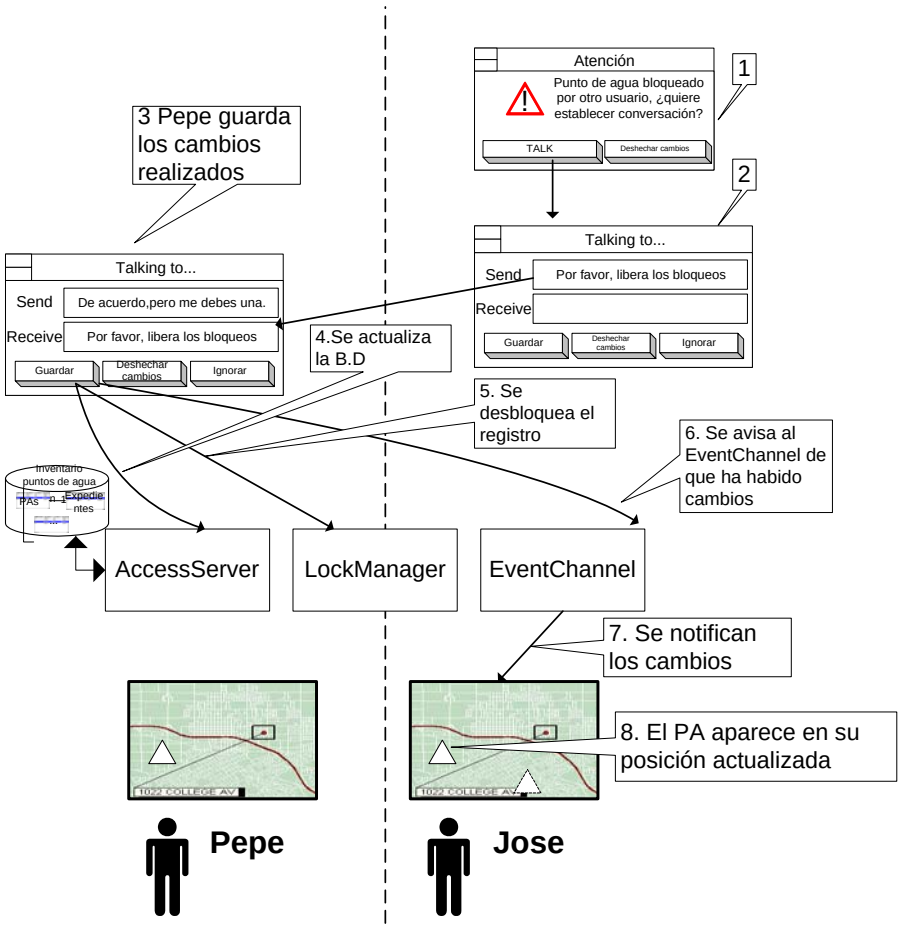

Figura 5: Modificación en BD del PA y refresco

6. Detalles de diseño: Patrones.

Uno de los aspectos que más notablemente ha influido en el diseño de aplicaciones en la actualidad son los patrones. Un patrón de diseño sistemáticamente nombra, explica y evalúa un importante y recurrente diseño en un sistema orientado a objetos [FOWL96].

En la arquitectura presentada coexisten diversos patrones de diseño, algunos de los cuáles están explícitamente representados, mientras que otros no se muestran por motivos de claridad.

6.1 Event channel

El event channel no es sino una variante del patrón sujeto - observador [GHJV-95, BMHP-96]. En el patrón sujeto-observador uno o más observadores son notificados de cambios en el sujeto. Aquí, para disminuir el acoplamiento entre los sujetos y los observadores insertamos un gestor intermedio denominado **EventChannel**.

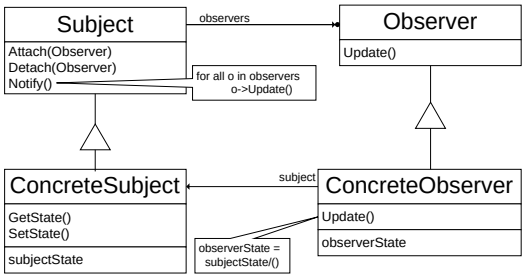

Figura 6: Observer

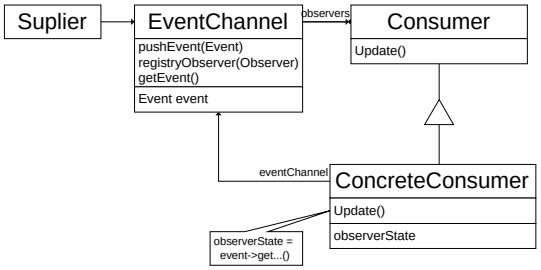

Figura 7: EventChannel

6.2 Singleton

Tanto el **LockManager**, el **EventChannel** y el **AccesServer** son objetos únicos en el sistema. Son, pues, candidatos para aplicarles el patrón Singleton. El patrón singleton asegura que una clase tiene una sola instancia y provee de un acceso global a la misma [GHJV-95].

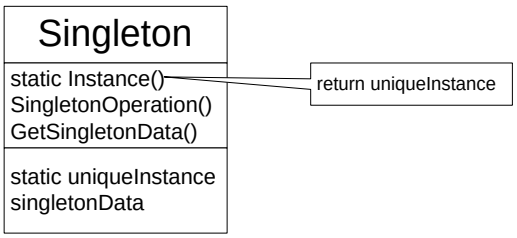

Figura 8: Singleton.

6.3 Adapter.

Una sesión de una herramienta GIS y una sesión con un gestor de BDs relacionales no son, ni mucho menos, objetos CORBA (ni siquiera son objetos). Por lo tanto tenemos que implementar una capa software que encapsule estos paquetes y simule el comportamiento de un objeto CORBA. Esto es precisamente lo que proporciona el patrón Adapter, también denominado Wrapper. Este patrón se emplea para convertir el interface de una clase en otro interface que un cliente espera. Adapter permite a dos clases trabajar juntas aunque tengan interfaces incompatibles [GHJV-95].

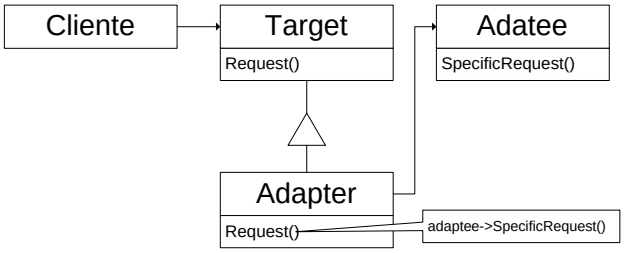

Figura 9: Adapter

7. Conclusiones

De una manera muy superficial hemos presentado un esquema de arquitectura útil para la integración de aplicaciones heterogéneas (opcionamente, también distribuídas) en un entorno en que la información de carácter geográfico es fundamental. Utilizando el estándar de objetos distribuídos CORBA pretendemos solventar, de una manera sencilla y elegante, los posibles problemas de inconsistencia de información en un sistema en el que es posible acceder desde distintas aplicaciones al repositorio de datos. En la propuesta, esto se resuelve mediante la implantación de un gestor externo que se encargue de coordinar las peticiones de acceso a datos por parte de las sesiones SIG.

8. Bibliografía.

[BMHP-96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerland, Michael Stal “A system of Patterns” WILEY 1996

[Cam-84] Benet Campderrich “Técnicas de bases de datos” Editores Técnicos Asociados S.A. 1984

[Date-95] C.J Date “Locking and Recovery in a shared database system”

[FOWL96] Martin Fowler “Analisis Patterns” Addison Wesley 1996

[GHJV-95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides “Design Patterns” Addison-Wesley” 1996.

[OCG-OAS] OCG “The OpenGIS Abstract Specification” 1996

[OCG-OSFSC] OCG “OpenGis Simple Feature Specification for CORBA” 1996

[OHE-97] Rober Orfali, Dan Harkey, Jeri Edwards “The essential Distributed Objects Survival guide”

[OMG-CFS] OMG “Common Facilities Arquitecture”

[OMG-CORBAS] OMG “Common Object Request Broker Arquitecture and Specification”

[OMG-COSS] OMG “Common Object Services Specification”

[OMG-OMAG] OMG “Object Management Arquitecture Guide”

[Red-97] Frank E. Redmond III “DCOM” IDG Books 1997

[SMB-96] David Schell, Lance McKee, and Kurt Buehler “Geodata interoperability – A Key NII Requirement” http:\\www.opengis.org\articles\nii2000.html. 1996