Arquitectura abierta para un Sistema de Información Geográfica corporativo

Andrés Pazos, José Poveda , Michael Gould

Departamento de Lenguajes y Sistemas Informáticos, Universidad Jaume I,
12071 Castellón, España
Andres.Pazos@anubis.uji.es
{albalade, gould}@uji.es

Abstract. En el presente artículo se introduce una arquitectura flexible para el diseño de Sistemas de Información Geográfica en entornos corporativos. Esta arquitectura cliente-servidor de tres capas define un conjunto de paquetes que se sitúan en el lado del servidor y que responden en tiempo de ejecución a las demandas de los clientes. Un núcleo y *n* paquetes externos son descargados y configurados automáticamente de forma que se compone la aplicación SIG final. Se incluye una capa intermedia *middleware* encargada de la autenticación de usuarios y del control de versiones. Para demostrar la arquitectura propuesta, se ha desarrollado una aplicación práctica que permite la composición de un cliente SIG con paquetes de software libre. Este cliente SIG definido, por sus características, podría constituir de forma natural, un elemento importante en la explotación de una Infraestructura de Datos Espaciales.

1 Introducción

Los SIG corporativos permiten compartir datos espaciales, dar servicio coordinado y optimizado al nivel de mediana o gran organización, y facilitan la comunicación entre los distintos departamentos. El problema inherente a la naturaleza de dichos SIG es que el número de incidencias técnicas también se ve aumentado notoriamente en comparación con los SIG "*stand-alone*" debido al elevado número de usuarios capaces de explotar simultáneamente el sistema. Cada usuario dentro de un mismo departamento tiene requisitos específicos que implican configuraciones de hardware y software distintas, además estos requisitos son susceptibles de cambiar en el tiempo. Estudios realizados en diferentes organizaciones públicas y privadas muestran que la mayoría de los usuarios sólo utiliza un porcentaje mínimo de las funcionalidades que por defecto son instaladas en el Sistema de Información Geográfica, teniendo el administrador que instalar y mantener obligatoriamente un sistema completo, que en general será infrautilizado y que además contribuirá a un importante desembolso económico de la organización en concepto de licencias.

En organizaciones con un elevado número de usuarios de SIG y especialmente en el sector público, resulta de especial interés el desarrollo de un sistema SIG flexible a dos niveles: por una parte flexibilidad del sistema al nivel de la organización al pre-

sentar el sistema escalabilidad que permita responder con reutilización total del sistema a nuevas necesidades de la organización y por otra parte a nivel del usuario al permitir la personalización, en el sentido de que el usuario a solicitud propia es capaz de instalar sólo los componentes necesarios para el desarrollo de su trabajo. Esta flexibilidad guarda estrecha relación con los niveles de *tailoring* (configuración a medida) presentados por Morch en [1]. A grandes rasgos el sistema estaría formado por una colección de paquetes entre los cuales es de destacar un núcleo central ligero, el cual contiene la funcionalidad básica del sistema, y un conjunto de paquetes externos que responderían a funcionalidades adicionales. Cada uno de estos paquetes implementaría una parte específica del SIG y su descarga y conexión al cliente SIG se realizaría automáticamente a solicitud del cliente.

Otro aspecto relevante de la arquitectura es la flexibilidad en la distribución del software. En la arquitectura cliente-servidor que se propone es posible descargar e instalar el software deseado en cada ordenador, sin la necesidad de efectuar instalaciones específicas y completas desde CDs u otro tipo de dispositivo de almacenamiento. El núcleo del sistema y los paquetes de funcionalidad adicional son almacenados en un servidor central, de este modo cada usuario, bajo autentificación, puede conectar y configurar su propia cliente SIG en base a sus necesidades y sus permisos. Esta arquitectura centralizada también presenta un claro beneficio en la administración del sistema, ya que permite actualizaciones con coste mínimo, así como el seguimiento de las descargas y del uso “real” del software en la organización.

Como valor añadido a la arquitectura que se propone y pieza clave para el crecimiento del sistema en un entorno abierto, se han reutilizado librerías de programación *Open Source* bajo licencia GNU. Gracias a la disponibilidad del código, se pueden modificar y extender este tipo de librerías para generar componentes software que añadan nuevas funcionalidades al sistema [2].

Es por tanto objetivo del presente artículo, la introducción de una arquitectura para el desarrollo de clientes SIG en entornos corporativos. El resultado final que se persigue es la definición un cliente SIG flexible, capaz de extender su funcionalidad por medio de paquetes externos y en el que cada usuario sea capaz de instalar y configurar un cliente SIG acorde a sus necesidades específicas. Se presenta finalmente en este artículo un prototipo que cumple los objetivos de la arquitectura propuesta. Dicho prototipo ha sido implementado utilizado la plataforma de programación *Java Web Start* y en su desarrollo han sido integrados distintos componentes de software libre.

2 Descripción de la Arquitectura

En esta sección se presenta una solución que cubre las necesidades que se han planteado previamente. Es nuestra pretensión, por tanto, modificar la concepción tradicional de cliente SIG monolítico para orientarlo a un entorno cambiante, corporativo e integrado en red. El ancho de banda disponible actualmente en la mayoría de las redes hace posible esta integración, algo que era una limitación en estudios anteriores [3]. En esta línea, la arquitectura que se propone se fundamenta en dos aspectos básicos, por una parte el de facilitar la distribución y actualización del software así como por

otra parte el de definir el sistema conceptualmente de modo que tenga la capacidad de ser extendido con nuevas funcionalidades sin necesidad de reingeniería del sistema y siguiendo la línea de desarrollos *end-user development* [4].

2.1 Estructura General

Se propone añadir una capa intermedia en la arquitectura SIG cliente-servidor. Esta nueva capa actuaría a modo de *middleware*, siendo la encargada de dar soporte a algunas nuevas funcionalidades del cliente como son la identificación y validación del usuario así como la personalización de los paquetes enviados desde el servidor a un cliente SIG específico.

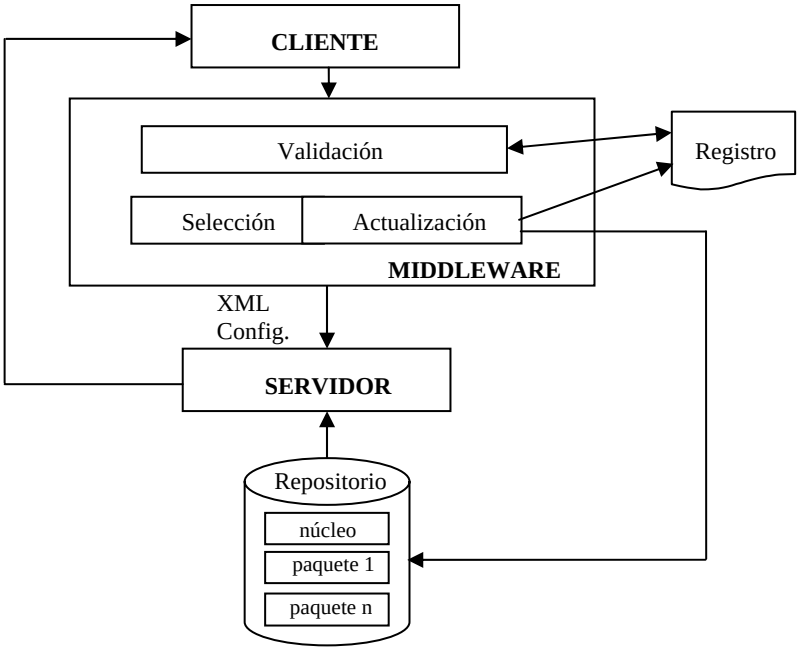

Fig. 1. Arquitectura SIG propuesta, a tres capas

El diagrama de la arquitectura se muestra en la figura 1. Los pasos a seguir para que el usuario defina la aplicación SIG que atiende a sus necesidades son los que seguidamente se especifican:

a) Conexión a la capa *middleware*. Este paso se realiza únicamente la primera vez que el usuario desea definir y descargar la aplicación SIG que quiere utilizar.

b) Validación del usuario por medio de su *login* y *password*. El usuario debe haber sido dado de alta por el administrador del SIG para la utilización del sistema. Dependiendo de los privilegios del usuario el *middleware* mostrará una lista de paquetes accesibles para incorporar a la aplicación SIG.
c) Selección de los paquetes que el usuario desea instalar. En función de las necesidades del usuario, éste seleccionará los paquetes que cubran sus necesidades específicas.
d) Configuración XML. Después de recibir el resultado de la selección, el *middleware* genera un archivo XML que contiene la información de los paquetes seleccionados y del usuario. Seguidamente, este archivo XML de configuración es enviado al servidor.
e) Interpretación del servidor. El servidor procesa los archivos de configuración recibidos y envía el software de instalación específico para cada cliente.

2.2 Composición de la aplicación

Tras la descarga por parte del cliente de los paquetes seleccionados, el núcleo generará de forma automática la aplicación SIG. En la figura 2 se muestra la estructura de la composición de paquetes.

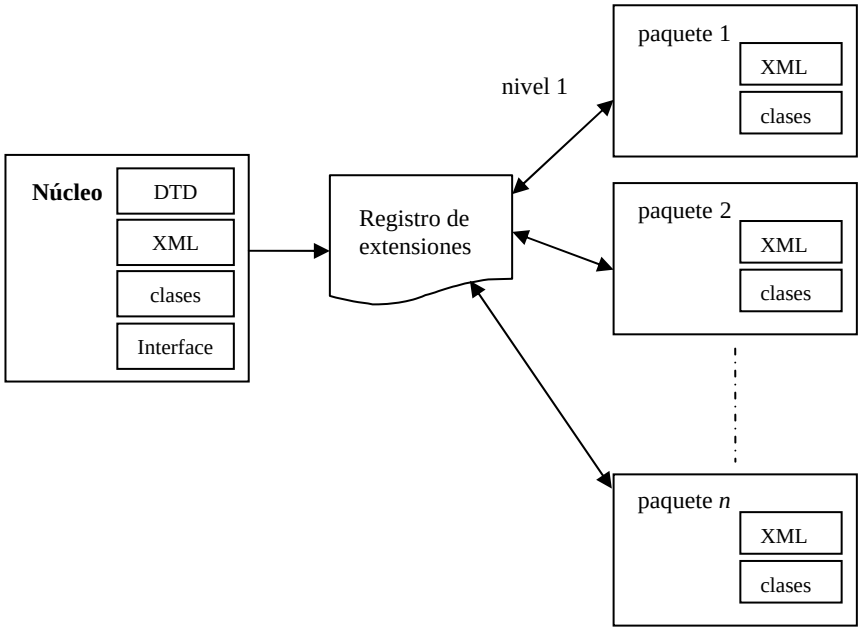

Fig. 2. Composición de paquetes

La aplicación SIG se diseña del modo que los paquetes encajan en la estructura general del núcleo. Con anterioridad a la implementación del núcleo, cada parte acce-

sible debe definirse de acuerdo con una especificación de implementación. Estas partes son denominadas puntos de extensión [5]. Al recibir el cliente cada uno de los paquetes seleccionados, el núcleo los analiza y extrae la información contenida en el archivo XML de configuración. Dicho archivo debe estar definido de acuerdo con el DTD del núcleo, en el cual se definen los distintos puntos de extensión existentes. Por ejemplo, es posible definir un punto de extensión del menú de la aplicación. El archivo XML debe contener una etiqueta que indique que el nuevo paquete debe de ser incluido en el menú y que tiene una acción determinada asociada. Los paquetes también contienen clases que implementan una funcionalidad determinada. El núcleo leerá del archivo XML el nombre las clases registradas y las cargará de forma que se genere la aplicación SIG que finalmente utilizará el cliente (usuario).

3 Prototipo Desarrollado

Como aplicación práctica y prototipo de pruebas de la arquitectura diseñada, se ha implementado un Sistema de Información Geográfica que cumple las características definidas en este estudio. El sistema ha sido desarrollado haciendo uso de la plataforma de programación Java, gracias a las ventajas que reportaba para este tipo de proyecto [6]. Se ha dividido en tres partes correspondientes al cliente, a la capa *middleware* y al servidor.

3.1 Cliente SIG

La aplicación está compuesta por un núcleo principal y por paquetes adicionales que proporcionan funcionalidades específicas al cliente. En el núcleo se ha implementado la interfaz gráfica de usuario de la aplicación, la cual contiene: la vista principal, la vista de datos, la barra de menú y la barra de herramientas.

Cada uno de estos elementos definidos está asociado a una clase independiente que muestra públicamente una interfaz que permite su extensibilidad. A través de estas interfaces los paquetes externos pueden acceder al núcleo del sistema. La conexión entre el núcleo y los paquetes de funcionalidad adicional que previamente a su descarga selecciona el cliente, es posible gracias a los archivos XML de conexión, los cuales cumplen el esquema definido para los puntos de extensión. Dentro del núcleo existe una clase encargada de parsear dichos archivos XML y extraer la información necesaria para conectar y añadir cada paquete al registro general de paquetes. Desde un punto de vista de mantenimiento y ampliación del sistema, las nuevas funcionalidades con las que se quiera proveer al SIG deben ser implementadas en forma de paquetes de acuerdo al DTD que define todos los posibles archivos XML de conexión. Finalmente sería el administrador del sistema el que ubicaría dicho paquete en el servidor para su uso por todo aquel cliente que tuviera los privilegios necesarios. Al objeto de optimizar el tiempo de descarga de la aplicación SIG que requiera instalar el cliente los paquetes son almacenados en el servidor como ficheros comprimidos.

En el prototipo de pruebas desarrollado se han implementado cuatro paquetes con las siguientes funcionalidades:

- Paquete1: Incluye las herramientas de *zoom*, *pan* y selección. Dichas herramientas se muestran en el menú de la aplicación así como en iconos gráficos en la barra de herramientas. Se incluye como ejemplo una parte del archivo XML necesario para que los iconos aparezcan en la aplicación principal así como para que la acción asociada sea ejecutada (código 1). La etiqueta *class* indica en qué clase es implementa dicha funcionalidad.

- Paquete2: Carga y visualiza en la vista principal del cliente un mapa de datos vectoriales. Este paquete ha sido desarrollado haciendo uso del API Open Source *GeoTools*. Es por tanto, que éste paquete supone una capa intermedia que permite el uso de librerías de libre distribución o desarrolladas *ad hoc* en la arquitectura SIG que se define.

- Paquete 3: Carga una nueva vista en el entorno gráfico de la aplicación. Esta vista realiza búsquedas sobre los datos, selecciona los resultados en el mapa y los muestra en la vista de datos. Análogamente al paquete 2 para su desarrollo ha sido reutilizado código del API *GeoTools*.

- Paquete 4: Carga una nueva vista en el entorno gráfico de la aplicación con datos ráster. Gran parte de las funcionalidades sobre procesamiento y análisis de ortoimágenes de la librería GIA (*Geo Image Analyser*) han sido transferidas al cliente.

```
<?xml version="1.0"?>
<id>Paquete1
  <menu>
    <label>Herramientas</label>
    <memo>3</memo>
    <submenu>Zoom</submenu>
  </menu>
  <action>
       <kind>menu</kind>
       <label>Zoom</label>
       <class>paquete1/Zoom</class>
  </action>
 <action>
       <kind>toolbar</kind>
       <label>Zoom</label>
       <class>paquete1/Zoom</class>
  </action>
  <toolbar>
       <tooltip>Zoom</tooltip>
       <icon>paquete1/zoom.gif</icon>
  </toolbar>
</id>
```

Código 1: Extracto del fichero de conexión correspondiente al paquete 1

En las figuras 3 y 4 se muestran dos clientes con diferentes funcionalidades. En la primera de ellas se han cargado todos los paquetes de manipulación de datos vectoriales que han sido implementados, es decir, paquetes 1, 2 y 3. En la figura 3 se visualiza un mapa de los estados mejicanos así como su tabla de datos asociados. La operación que se muestra es la búsqueda de una ciudad, concretamente la capital Méjico D.F. El resultado de la búsqueda es marcado tanto en la tabla de datos como en el mapa activo. En la segunda figura se carga el paquete 4 que nos permite la lectura de datos ráster que posteriormente podrán ser procesados y analizados.

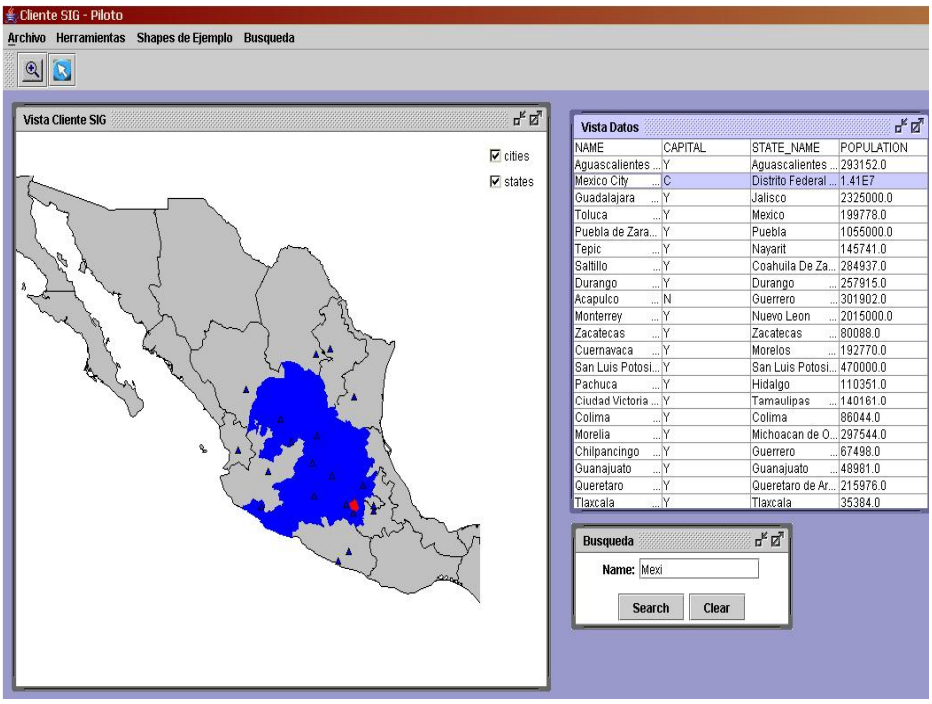

Fig. 3. Cliente SIG definido para el análisis de datos vectoriales

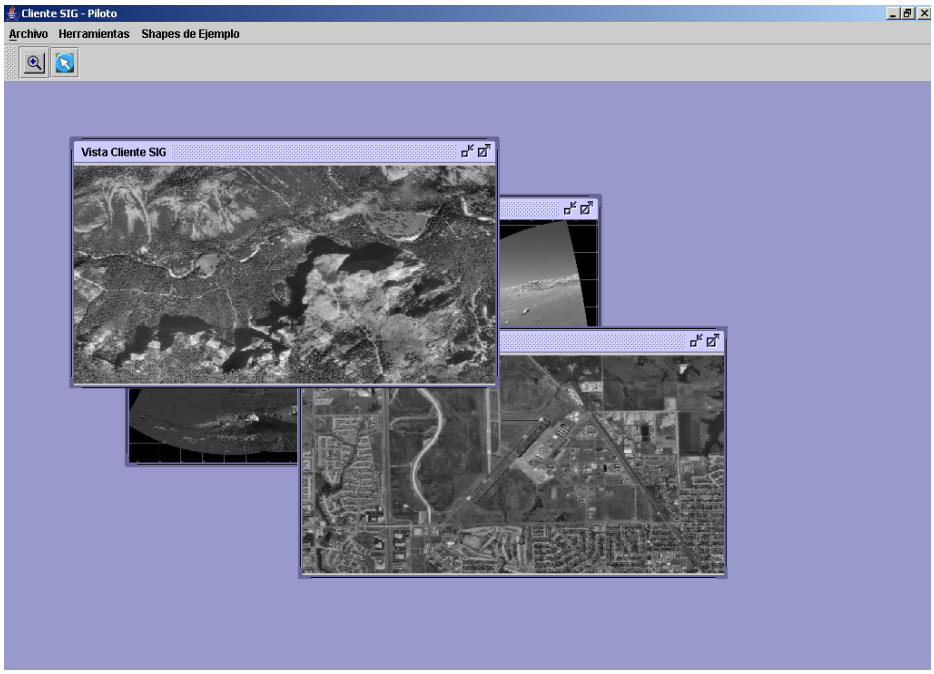

Fig. 4. Cliente SIG definido para el análisis de datos ráster

3.2 Middleware

La capa intermedia entre cliente y servidor ha sido implementada haciendo uso de la tecnología *servlets* de Java. Cada vez que un cliente se conecta al sistema pasa a través del *middleware* que valida internamente contra una base de datos al usuario. La implementación de la base de datos ha sido realizada sobre *PostgreSQL*. En base a los privilegios de paquetes y usuarios se muestra una lista con los paquetes que un determinado usuario puede ver y por tanto descargar. Para el caso de que el usuario posea privilegios de administrador, también se puede subir, borrar o actualizar paquetes en el servidor. Los usuarios cuyo software actual utiliza paquetes obsoletos son avisados por el sistema, pero es decisión del usuario su actualización o el de instalar un cliente con diferente configuración.

Una vez que el usuario ha seleccionado los paquetes que desea incorporar a su aplicación SIG, se genera dinámicamente un archivo XML de configuración del cliente, que debe ser accesible desde el servidor. En este archivo se incluyen los nombres de los paquetes *jar* que tienen que ser descargados.

3.3 Servidor

El servidor recibe el archivo XML de configuración del cliente enviado por la capa *middleware* y tras su procesamiento envía al cliente los paquetes de funcionalidad seleccionados. Para la implementación de la parte servidora se ha utilizado la plataforma de programación Java y el protocolo de comunicación *JNLP* [7]. Por medio de dicho protocolo se puede establecer una canal de comunicación a través de Internet con el sistema del usuario con objeto de enviar la aplicación desde el servidor. La aplicación queda instalada de forma transparente en el PC del usuario, lo que permite ejecutarla localmente. Al nivel de seguridad el protocolo JNLP está construido sobre la plataforma Java 2, que proporciona una amplia arquitectura de seguridad. Las aplicaciones se ejecutarán de forma predeterminada en un entorno restringido ("zona protegida") con acceso limitado a los archivos y a la red. Por tanto, para que la aplicación tenga acceso ilimitado los paquetes a descargar deben estar firmados y será el usuario el que aceptará o no la descarga. Este aspecto resulta de especial interés para el caso de redes abiertas y múltiples servidores.

4 Conclusiones

Se ha presentado una arquitectura que plantea un conjunto de ideas de interés para el diseño y desarrollo de un SIG en un entorno corporativo: lo que según la arquitectura conceptual de la IDE definida en la iniciativa Europea INSPIRE correspondería a la aplicación cliente. Dichas ideas son el resultado de una investigación previa en el campo de las aplicaciones SIG para la Administración Pública y de las posibilidades del software libre en este campo. Se ha tratado de buscar un diseño adecuado para un entorno corporativo, orientado no sólo en aumentar la funcionalidad del sistema sino también en la minimización de gastos en concepto de administración y licencias así como en potenciar la interoperabilidad y la flexibilidad de los usuarios de aplicaciones SIG.

Se ha implementado un prototipo de pruebas al objeto de contrastar las ideas presentadas al respecto de la definición de un SIG en un entorno heterogéneo. Este desarrollo ha sido clave para entender mejor la arquitectura y poder demostrar su validez. Mencionar que la arquitectura propuesta pretende ser universal, en el sentido de que no esta ligada a ningún software o protocolo determinado. No obstante, para la implementación realizada se han utilizado desarrollos ya existentes como el caso del protocolo de comunicaciones *JNLP* a través de la plataforma de programación *Java Web Start*.

Uno de los aspectos clave de la arquitectura que se propone es la propia definición en red. Este hecho permite la distribución y posterior instalación de la aplicación SIG de forma optimizada por medio de la personalización interactiva que los propios usuarios pueden llevar a cabo.

Finalmente mencionar que la arquitectura descrita define un marco abierto para el desarrollo de SIGs pero que solamente con la inclusión de un conjunto de paquetes que cubran un amplio abanico de funcionalidades se podrá satisfacer las necesidades de un colectivo numeroso. Es por tanto necesario para un uso real la implementación

de nuevos paquetes que extiendan la aplicación SIG existente dotándole de nuevas funcionalidades. El uso de librerías Open Source como por ejemplo *Terralib* [8] y *Geotools 2* permite el desarrollo de paquetes con un mínimo coste de desarrollo.

Por otra parte el sistema puede ser extendido en la capa de *middleware* incluyendo funcionalidades de monitorización, de modo que generen estadísticas acerca del uso real del sistema.

Reconocimientos

Esta investigación ha sido subvencionada en parte por el Ministerio de Educación, Cultura y Deporte (Beca-colaboración), Consellería de Infraestructuras y Transporte (Generalitat Valenciana), y el MCyT (TIC-2003-09365-C02-02).

Referencias

1. Morch, A.: Three Levels of End-User Tailoring: Customization, Integration, and Extension. In: M. Kyng & L. Mathiassen (eds.): Computers and Design in Context. The MIT Press, Cambridge, MA (1997) 51-76
2. Anderson, G., Moreno-Sanchez, R.: Building Web-Based Spatial Information Solutions around Open Specifications and Open Source Software. Transactions in GIS, Vol. 7, (2003) 447
3. Anitto, R. N., Patterson, B. L.: A new Paradigm for GIS data Communications, URISA Journal, Milwaukee (1994) 64-67
4. Morch A., Stevens G., Won M., et al.: Component-based technologies for end-user development. Communications of the ACM, Vol 47, Issue 9, (2004) 59-62
5. Bolour, A.: Notes of the Eclipse Plug-in Architecture. http://www.eclipse.org/articles, (2003)
6. Gosling, J., McGilton, H.: White Paper: The Java Language Environment. Java Articles: http://java.sun.com (1996)
7. Schmidt R.: Java Networking Launching Protocol & API Specification. Java Articles http://java.sun.com (2001)
8. Camara, G., et al.: Terralib: Technology in Support of GIS Innovation. II Workshop Brasileiro de Geoinformática, Geoinfo2000. Sao Paulo (2000)