García-Peñalvo, F. J., Alvarez Navia, I., Hernández Herrero, J. B., González Pérez, S., Costa Alba, Á., & Conde González, M. Á. (2001). Left CASE: Herramienta CASE basada en componentes Bonobo. In J. Hernández, J. Pavón, D. Sevilla, & A. Vallecillo (Eds.), *Actas del Segundo Taller de Trabajo en Ingeniería del Software basada en Componentes Distribuidos - IScDIS'01, desarrollado dentro de las VI Jornadas de Ingeniería del Software y Bases de Datos, JISBD'2001 (Almagro – Ciudad Real, 21-23 de noviembre de 2001)* (pp. 1-9). Murcia, España: Informe Técnico UM-DITEC-2001-112001, Departamento de Ingeniería y Tecnología de Computadores, Universidad de Murcia.

Left CASE: Herramienta CASE basada en componentes Bonobo

Francisco J. García Peñalvo, Iván Álvarez Navia, Juan B. Hernández Herrero, Santiago González Pérez, Ángel Costa Alba, and Miguel Ángel Conde González

Departamento de Informática y Automática
Facultad de Ciencias - Universidad de Salamanca (España)
fgarcia@usal.es

Resumen La utilización de la tecnología de componentes en el campo de las herramientas CASE abre las puertas a la interesante opción de contar con una arquitectura base, en la que se pueden ir incorporando componentes, cada uno de los cuales representa una técnica de modelado. Esta filosofía permite, contar, bajo una interfaz y política de uso comunes, con diferentes técnicas de modelado, integradas entre sí, para dar cobertura al ciclo de vida del software. Además, esta forma de trabajo permite que la herramienta evolucione de una manera más flexible, ya que al separar la arquitectura de los componentes, cada parte puede evolucionar por separado, siempre que se respeten las interfaces entre ambos. Esto es fundamental para el enfoque docente en el que ha surgido la herramienta Left CASE que aquí se presenta, porque una vez que se cuenta con la arquitectura adecuada, los componentes se van desarrollando paulatinamente, dado cobertura a las necesidades de modelado que presenta cualquier Ingeniería en Informática.
Incidiendo en el carácter abierto de una herramienta para fines docentes, el modelo de componentes que se ha elegido ha sido Bonobo, apostando por el auge que el entorno de escritorio GNOME está alcanzado en ciertos ámbitos.

1. Introducción

Contar con un conjunto de herramientas CASE que ofrecieran soporte a las principales técnicas de modelado que se estudian tanto en la Ingeniería Técnica como en la Ingeniería Informática de la Universidad de Salamanca, y que fueran accesibles para los alumnos sin las limitaciones propias de las herramientas propietarias, ha sido un objetivo perseguido desde hace tiempo en el Departamento de Informática de esta Universidad.

No obstante, las primeras versiones de las herramientas CASE construidas [1, 2, 3], aunque daban un soporte adecuado a los fines perseguidos, se caracterizaban por ser independientes las unas de las otras, tanto en lo referente a la interfaz como a la plataforma donde debían ejecutarse.

Así, surge la idea de crear un entorno CASE extensible, donde las técnicas soportadas puedan crecer, pero siempre bajo las restricciones de compatibilidad

con el entorno. Se piensa en una arquitectura de componentes, de forma que exista un conjunto de componentes, cada uno de los cuales debe proporcionar todas las características propias de una técnica de modelado concreta (por ejemplo, DFD, DER, DTE...), y un *framework* base o contenedor, que haga las veces de "anfitrión" para los componentes, ofreciéndoles los servicios comunes necesarios, así como el conjunto de interfaces necesarias para que dichos componentes puedan incorporarse al entorno construido.

A la hora de elegir un entorno de desarrollo se pueden encontrar diferentes opciones en el mercado, incluyendo algunas de libre distribución. Dadas las características del proyecto, realizado por y para miembros de la comunidad académica, se optó, desde un principio, por trabajar con herramientas de libre distribución. Dentro de éstas nos podemos encontrar fundamentalmente con dos alternativas: GNOME[1] (y su modelo de componentes Bonobo) y KDE[2] (y su modelo de componentes Kparts). Se optó por trabajar con GNOME debido a varias razones:

- Empieza a contar con un importante apoyo, no sólo de los clásicos del desarrollo del software libre, sino también con el de algunos sectores importantes de la industria (SUN, HP).
- El mecanismo de interoperabilidad entre componentes se basa en CORBA [5], estándar en auge y con gran aceptación tanto en el mundo académico como en el empresarial.
- El modelo de componentes Bonobo [4] permite unos niveles de integración entre aplicaciones, dentro del entorno de escritorio GNOME, que, hasta ahora, tan sólo existía en algunas soluciones propietarias.

2. Arquitectura de Left CASE

Una vez fijado el contexto en el que se desarrolla el trabajo, en el presente apartado se va a presentar la arquitectura desarrollada.

En primer lugar, y como es obvio en una arquitectura de estas características, se ha diferenciado claramente la parte que va a hacer las veces de un framework de componentes y de los componentes en sí mismos. Esta estructura queda esquematizada en la Figura 1, donde se muestra, de forma genérica, la inclusión de componentes dentro de un contenedor utilizando Bonobo.

En la Figura 1 se describe un contenedor que tiene en su interior dos componentes. El contenedor ofrece a los componentes dos tipos de interfaces:

- `Bonobo::ClientSite`
- `Bonobo::ViewFrame`

Estas interfaces se definen en Bonobo utilizando IDL (*Interface Definition Language*), y son las encargadas de permitir que un componente pueda existir

[1] http://www.gnome.org/
[2] http://www.kde.org/

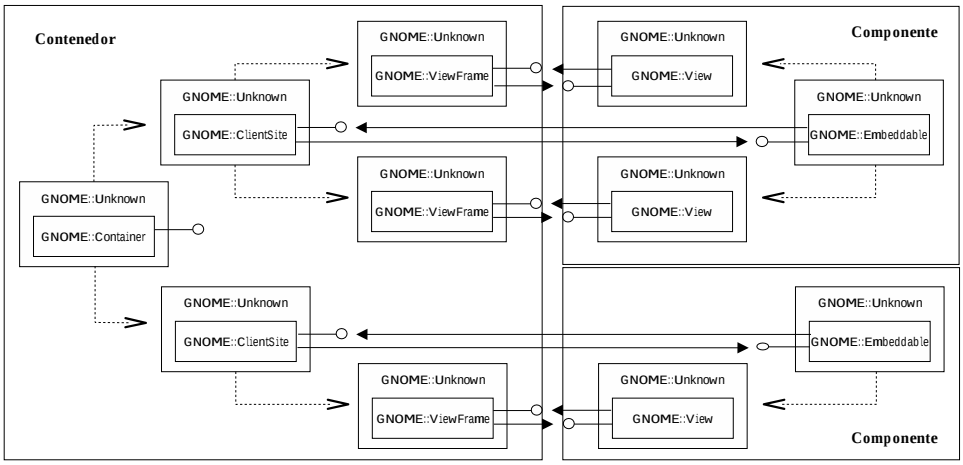

Figura1. Relación entre componentes y contenedores en Bonobo

dentro del contenedor. Par incrustar un componente dentro de un contenedor, se debe crear un *ClientSite* donde se almacena. Este componente puede tener varias vistas que comparten datos, pero que pueden mostrar diferentes aspectos de éstos. Cada instancia de un componente necesita que dentro del contenedor exista un *ViewFrame*, es decir, una vista para la instancia.

En cuanto a los componentes existe una interfaz fundamental: `Bonobo::View`. Mediante esta interfaz el componente se comunica con el contenedor, utilizando la interfaz del contenedor *ViewFrame*, siendo también el lugar por el que se pasan al componente los sucesos que puedan ocurrir en el contenedor.

Teniendo en cuenta esta estructura de aplicación que impone el uso de Bonobo, el diseño arquitectónico propuesto es el mostrado en la Figura 2, formado por tres paquetes bien diferenciados:

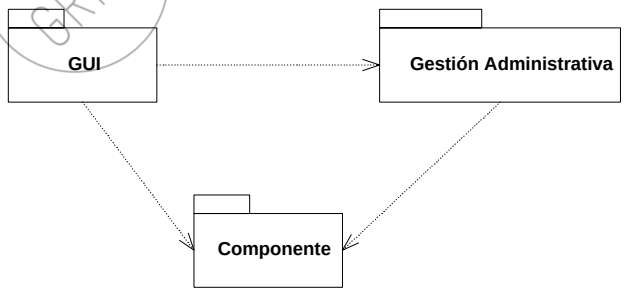

Figura2. Diagrama de paquetes del sistema

- **GUI:** representa la interacción del sistema con el usuario, intercambiando los datos con él y encargándose de responder a los eventos. También se incluye en esta parte los diálogos de la aplicación. Tiene dos clases:
  - *Contenedor:* proporciona el espacio para alojar el componente y define una serie de métodos para su manejo.
  - *Ventana:* representa la ventana que se muestra al iniciar la aplicación y va a englobar los menús, los aceleradores de teclado...
- **Gestión administrativa:** gestiona el almacenamiento de los diagramas creados por el sistema y cuyo componente haya sido descargado. Además, almacena todos los datos del proyecto en el que se está trabajando, así como la interfaz para poder interactuar con él. Las tres clases que lo componen son las siguientes:
  - *Proyecto:* en esta clase se guardará toda la información del proyecto en estudio y ofrecerá la interfaz para manejar todos los datos del proyecto.
  - *Diagrama:* aquí se especificará a que componente pertenece dicho diagrama y otros datos de interés a la hora de recuperar el componente.
  - *Instancia de modelado:* son los datos de cada uno de los elementos que pueda tener el diagrama, y que la aplicación guarda por si fuera necesario volver a cargar el componente.
- **Componente:** aunque se trata de un paquete independiente que se incrusta en el contenedor, se ha integrado en el sistema para su mejor comprensión y análisis. En cuanto a su diseño, depende del tipo de funcionalidad en concreto que ofrece, y por tanto, del tipo del componente. Sin embargo, debido a la característica de componente que se empotra en el contenedor, todos parten de una base arquitectónica común (Figura 3):
  - *GUI-Componente:* abarca toda la interfaz gráfica de usuario y gestiona la comunicación con el mismo.
  - *Gestión componente:* engloba los elementos que tiene el componente. En general, todos los componentes cuentan con tres clases fundamentales, a las que se pueden añadir otras, dependiendo del componente concreto. Las tres clases comunes son:
    - Navegador: gestiona el comportamiento del navegador y cómo éste debe ser actualizado según las modificaciones que se efectúen en el panel de dibujo.
    - Barra de herramientas: ofrece la interfaz necesaria para que se pueda cambiar el tipo de dibujo seleccionado en el sistema. Cuando sucede algo en la barra de herramientas ésta se encarga de actualizar el panel.
    - Panel: es el lugar donde se hacen los dibujos. Gestionará por tanto los sucesos que se producen en el área de dibujo para que se puedan crear y modificar los elementos.
  - *Elementos de modelado del componente:* se trata de los elementos que se pueden editar con el componente.

En cuanto a la activación de los componentes se ha utilizado OAF, la Infraestructura de Activación de Objetos (*Object Activation Framework*). OAF utiliza

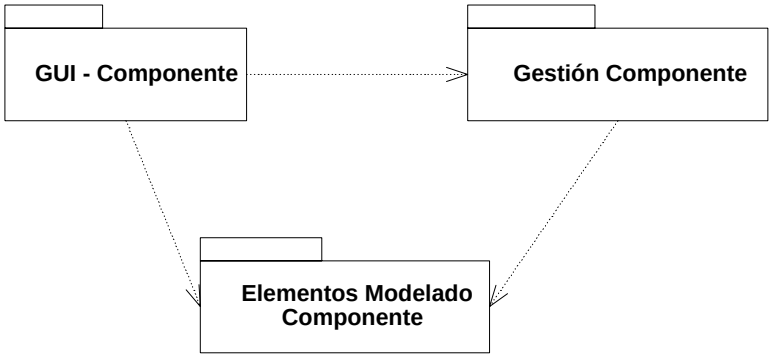

Figura3. Diagrama de paquetes del componente

un sistema muy sencillo de configuración consistente en una serie de archivos XML situados en un directorio conocido por todas las aplicaciones (generalmente `/usr/share/oaf`), en los cuales se incluye una entrada por cada servidor CORBA instalado en el sistema.

El uso de esta arquitectura separa el desarrollo del contenedor del desarrollo de los componentes, de manera tal, que estos últimos son instancias de componentes que pueden ser activados en cualquier momento, sin que el usuario tenga conocer nada sobre su localización.

3. Presentación de la aplicación

La aplicación que se presenta es una herramienta CASE que permite al usuario la creación, modificación, verificación y almacenamiento de diferentes tipos de diagramas, todo ello en un producto software que presenta una interfaz común sencilla de manejar y amigable. Se integran, en principio, cuatro tipos de diagramas distintos, que pueden ser utilizados por los alumnos de la Ingeniería Técnica en Informática de Sistemas e Ingeniería en Informática de la Universidad de Salamanca bajo las plataformas UNIX/GNOME. Los diagramas soportados son: diagramas E/R, DFDs, diagramas de transición de estados y diagramas de clase UML.

Como se puede apreciar en la Figura 4, al arrancar la aplicación no aparece ninguna característica asociada a cualquiera de las técnicas de modelado soportadas, correspondiéndose este esqueleto con la noción de contenedor. Es el momento en el que se crea un nuevo proyecto (o se selecciona uno existente), ver Figura 5, cuando se determinan que componentes van a poder utilizarse (dependiendo de si se elige un proyecto estructurado o UML), cargándose los controles propios del componente elegido.

En la Figura 6 se puede apreciar que tras haber seleccionado un proyecto estructurado, se ha cargado el componente que permite la creación de modelos

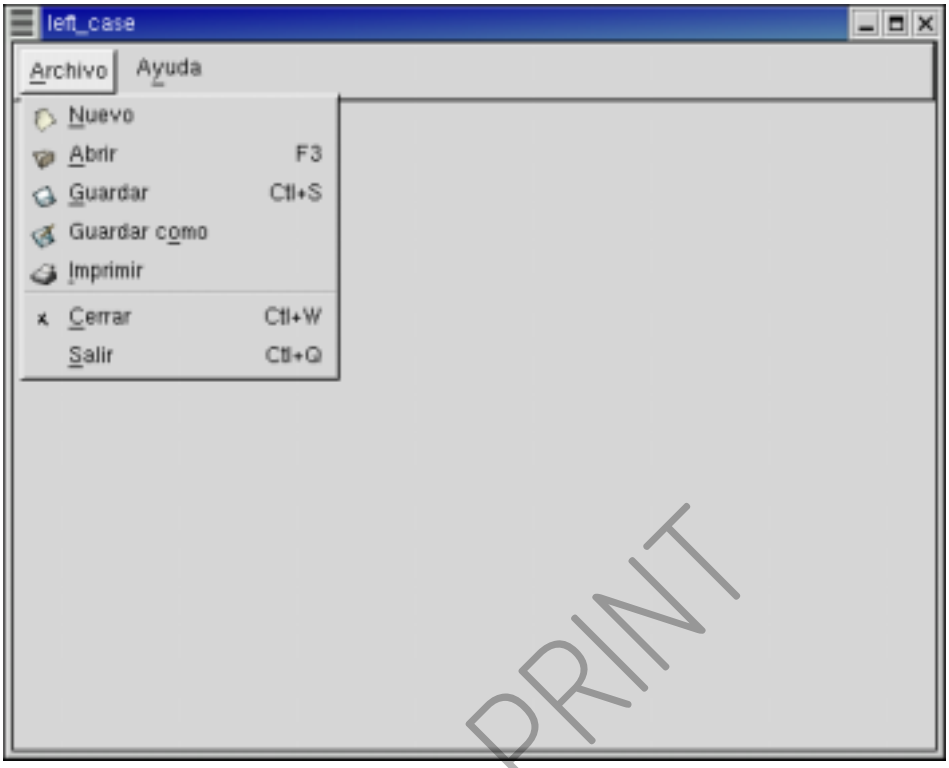

Figura4. Contenedor de componentes Left CASE

entidad/relación, estando a disposición del usuario todos los controles propios de dicho componente.

Además de las tareas propias ya indicadas de los componentes de edición de diagramas, se ha recurrido a los servicios Bonobo para tareas concretas, como puede ser la impresión de diagramas. En cada uno de los componentes de edición es posible acceder a las propiedades individuales de cada elemento (clase, relación...), tanto semánticas como de aspecto, sin más que seleccionar el elemento con el botón derecho del ratón. También se puede realizar un acceso rápido a los diferentes elementos de un diagrama a través del navegador (parte izquierda de la Figura 6).

4. Conclusiones

Se ha creado una herramienta CASE funcional con capacidad para crear diferentes tipos de diagramas, utilizando una arquitectura basada en componentes, lo que le confiere unas características de extensibilidad y flexibilidad no disponibles en otras herramientas previamente desarrolladas en este campo. Se proporciona

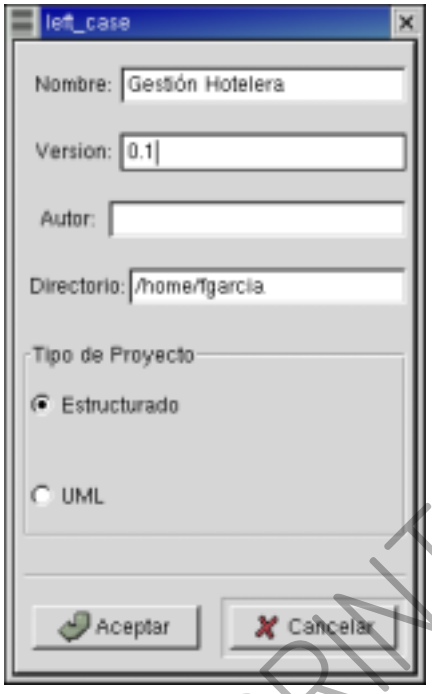

Figura5. Selección de tipo de proyecto Left CASE

un marco en el que resulta sencillo la adición de nuevas características, sin más que crear el correspondiente componente.

Las características básicas que se ofrecen se pueden resumir en la creación, edición, verificación y almacenamiento de diagramas E/R, DFDs, DTE y diagramas de clase UML. Así mismo, se ofrece la posibilidad de imprimir los diagramas creados y exportarlos a ficheros en formato gráfico (PNG).

En cuanto a las herramientas utilizadas, cabe destacar el uso del modelo de componentes Bonobo (proyecto GNOME) que, por un lado, ha facilitado el desarrollo de la aplicación por la arquitectura utilizada y los servicios que ofrece, pero por otro lado, la falta de documentación y los fallos debidos a que se trata de sus primeras versiones estables, ha complicado el desarrollo de la aplicación.

En cuanto a las líneas de trabajo futuro, parece claro que una de ellas sea el desarrollo de nuevos componentes que ofrezcan soporte para nuevas técnicas de modelado, satisfaciendo así nuevas necesidades del ingeniero de software que use este entorno. Así, se crearán nuevos componentes para la edición de casos de uso, diagramas de secuencia, diagramas de colaboración, tarjetas CRC...

Otro aspecto a tener en cuenta es, desde el punto de vista de la arquitectura, la adaptación de la misma y de los componentes a la futura especificación GNOME 2.0.

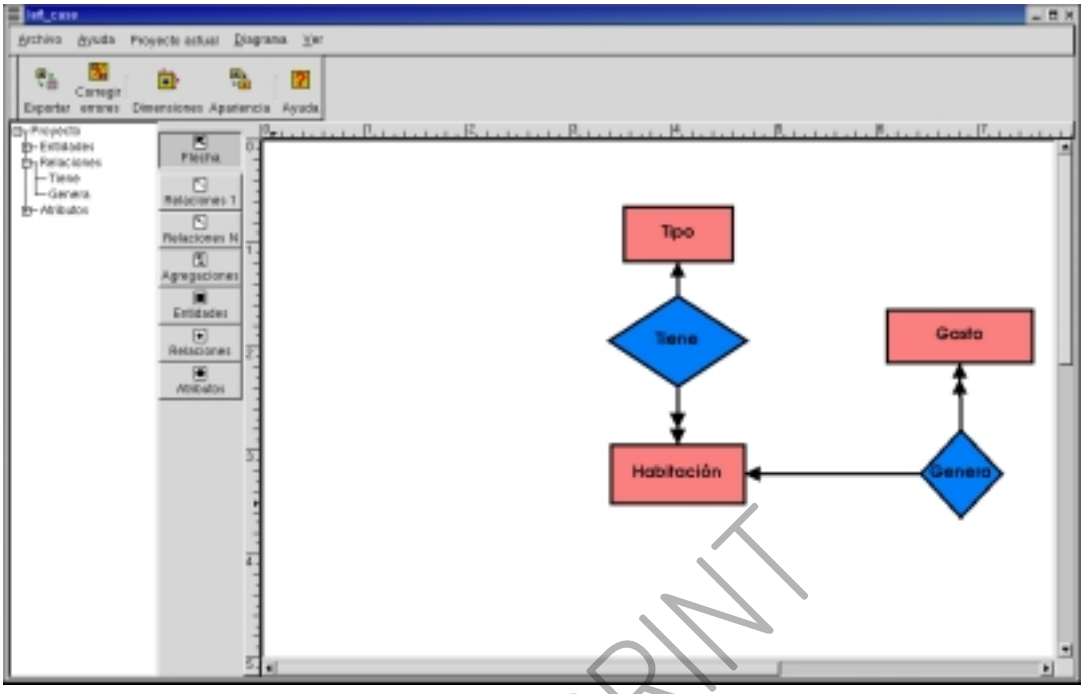

Figura6. Componente cargado en Left CASE

Desde un punto de vista crítico hacia el trabajo realizado, sería conveniente que para almacenar los diagramas se empleara la tecnología que Bonobo dispone para ello (tecnología que no se ha podido emplear por el momento al no funcionar correctamente). Además, se podría ampliar y mejorar los informes realizados por la aplicación, así como la exportación a otros tipos de documentos como RTF o PDF, siendo el trabajo cooperativo en red otra mejora que no se descarta.

5. Agradecimientos

Este trabajo ha sido parcialmente subvencionado por la Consejería de Educación y Cultura de la Junta de Castilla y León mediante el proyecto JAPQ.

Referencias

1. Blasco Martín, V.: ER CASE. Proyecto Fin de Carrera. Ingeniería Técnica en Informática de Sistemas. Facultad de Ciencias - Universidad de Salamanca.(2000)
2. Cuesta Carranza, A.: CRC CASE. Proyecto Fin de Carrera. Ingeniería Técnica en Informática de Sistemas. Facultad de Ciencias - Universidad de Salamanca.(2001)
3. García, F. J., Moreno, Mª. N., Moreno, A. Mª, González, G., Curto, B. y Blanco, F. J.: ADAM CASE. Using upper CASE tools in Software Engineering Laboratory. In

Computers and Education: Towards an Interconnected Society. Manuel Ortega and José Bravo editors. Kluwer Academic Publishers. (2001)
4. Icaza, M. de.: Bonobo. Ximian White Paper. http://www.ximian.com/devzone/tech/bonobo.html. (1999)
5. Object Management Group: Complete CORBA 2.4.2 Specification. Document formal/01-02-33. http://www.omg.org. (2001)