-CICLONES
Diseño de una aplicación GIS usando orientación a objetos. Un caso de uso.

G. Guaragna , S. Gordillo *
LIFIA-Departamento de Informática, Facultad de Ciencias Exactas, UNLP
CC 11 (1900) La Plata, Buenos Aires, Argentina
[gguarag, gordillo]@sol.info.unlp.edu.ar
Tel/Fax : (54) (21) 228252
* también CIC- Pcia. de Buenos Aires

Palabras Clave
GIS, Patrones de diseño, Ingeniería de Software, Orientación a Objetos.

Resumen

En este trabajo mostramos una aplicación GIS diseñada utilizando tecnología de orientación a objetos. Mostramos además el uso de algunos patterns en este ámbito y discutimos acerca del impacto que tiene el empleo de los mismos en el diseño de esta aplicación.

1- Introducción

Los sistemas de información geográfica (GIS) son sistemas de información diseñados para recolectar, procesar y mostrar datos referenciados por coordenadas espaciales o geográficas. En la actualidad los GIS también permiten almacenar y manipular información de otro tipo (no geográfica) y pueden estaár relacionados con otros sistemas de información (legacy systems, aplicaciones comerciales, etc.).

Durante el ciclo de desarrollo de este tipo de software, el mayor esfuerzo es puesto en realizar implementaciones eficientes y de alta performance. La complejidad inherente al dominio subyacente, la diversidad de la información , lo complejo de las relaciones existentes entre éstas y los cuantiosos volúmenes de datos que son precisos manipular, hace que en muchos casos no se tengan en cuenta las etapas previas del desarrollo. Debido a estos motivos, es común observar la obtención de productos con poca o ninguna documentación, que resultan difíciles de mantener y poco extensibles.

El uso de herramientas que nos provee la Ingeniería de Software en la actualidad, muchas veces es pasada por alto en la industria del desarrollo de aplicaciones GIS. La creación de este tipo de software se transforma en una tarea artesanal, donde la reusabilidad de soluciones de diseño existentes queda librado a la experiencia que tengan los diseñadores. Cada aplicación nueva se realiza sin aprovechar las experiencias previas sobre dominios semejantes, y en el mejor de los casos la reusabilidad solo se da a la hora de implementar y no en las etapas anteriores. En otros casos se emplean herramientas de diseño genéricas que resultan inadecuadas pues no fueron creadas para modelar problemas tan complejos como los que atacan los GIS.

Las técnicas de orientación a objetos proveen los mecanismos necesarios para diseñar las estructuras abstractas de las aplicaciones GIS. Es un hecho que en la actualidad hay un numero creciente de trabajos acerca del uso de estas técnicas en el campo de GIS ([Tryfona95] [OGIS96] [Shekhar97~~Shashi97~~]). En este trabajo nosotros proponemos no solamente el uso de técnicas de P.O.O. (Progamación Orientada a Objetos), sino también el uso de patrones de diseño como una manera de registrar experiencias previas en el desarrollo.

En la sección 2 damos una pequeña introducción a los patrones de diseño y explicamos como su uso influye positivamente en la construcción de este tipo de aplicaciones.

En la sección 3 hacemos una introducción al dominio del problema y definimos algunos de los conceptos necesarios para entender el diseño de las partes de la aplicación que aquí exponemos. Luego mostramos el uso de nuestro modelo orientado a objetos [DasNeves97] para diseñar aplicaciones GIS, ejemplificada en este caso por una aplicación que procesa datos que describen el efecto del paso de huracanes en una cierta región de la Tierra.
Explicaremos las bases de nuestras decisiones de diseño y el impacto que tuvo en las mismas el uso de patrones de diseño [Gordillo96], aplicada a este caso en particular.

2- Reusabilidad en Sistemas de Información Geográfica.

Diseñar software es una tarea dura y diseñar software reusable es aun más dificil. Los diseños deben ser específicos al problema que están atacando, y a la vez lo suficientemente genéricos para adaptarse a futuros problemas y requerimientos. Como mencionábamos anterionmente, muchas veces los diseñadores de aplicaciones GIS saltean las etapas previas de analisis y diseño, y el reuso solo se da a la hora de implementar. En estos casos solo se estan reutilizando estrucuras de datos o algoritmos. Si se está trabajando dentro de un ambiente orientado a objetos, se reusan clases y jerarquias de herencia. En la medida que diseñábamos la aplicación, utilizando tecnologías de O.O., se hacía evidente que no podíamos tomar a las clases como la unidad mínima de diseño, porque se perdía información referente a las relaciones estáticas y dinámicas entre las mismas. Descubrimos que el nivel correcto para trabajar, durante el diseño de la aplicación, consistía en pensar en bloques de construcción constituidos por pequeños conjuntos de clases y la descripción de la relaciones existentes entre ellas. Estas componentes de alto nivel resultan ser instancias de soluciones a problemas más genéricos. Estas soluciones se las conoce como Patrones de Diseño y constituyen micro-arquitecturas de software que podemos reusar entre diferentes aplicaciones en las etapas de diseño (y no solo en la implementación). En la siguiente sección describiremos como empleamos estas soluciones genéricas en el contexto del diseño de una aplicación GIS.

2.1 Introducción a los patrones de diseño.

Una de las tendencias de la Ingeniería de Software para la solución de problemas de diseño es el uso de patrones de diseño [Alexander77] [Gama95]. Los patrones de diseño nombran, explican y evalúan soluciones a problemas de diseño genéricos e independientes de un dominio. Ellos capturan experiencia de diseño de manera que otras personas puedan usarlas correctamente. Usualmente son descriptos detallando un problema en el cual el patrón puede ser aplicado, los elementos que forman parte del diseño, sus relaciones, responsabilidades y colaboraciones. Los patrones trabajan sobre un conjunto pequeño de responsabilidades de un conjunto de clases del sistema y pueden ser usados en distintas situaciones; además una misma clase puede pertenecer a distintos patrones.

El uso de Patterns favorece el desarrollo de aplicaciones complejas (como los sistemas de información geográfica) tal como se discute en [Schmidt95]. Algunas de las principales ventajas son :

- *Los Patrones posibilitan un vasto reuso de arquitecturas de software* : En el caso de las aplicaciones GIS, esto permite reusar algo más que algoritmos o estructuras de datos.
- *Los Patrones mejoran la comunicación dentro y fuera de los grupos de desarrollo de software.* Dado que proveen un vocabulario común, poderoso y consiso. Cuando los desarrolladores conocen los patrones de diseño, el nivel de discurso entre ellos tiene un grado de abstraccion más alto.
- *Los Patrones capturan en forma explícita el conocimiento que los diseñadores expertos usan implícitamente.* Los diseñadores expertos usualmente toman buenas decisiones y encuentran buenas soluciones; los patrones son un medio ideal para documentar qué problema están resolviendo, la forma en que lo hacen y las motivos de esa solución.
- *La descripción de los Patrones almacenan explícitamente las decisiones de diseño y las alternativas.* Estas decisiones muchas veces no quedan documentadas y entonces es necesario un gran esfuerzo para justificarlas. Cuando el mismo problema aparece en otra aplicación, nosotros podemos comprender claramente las razones para adoptar una solución en particular.
- *Los Patrones ayudan a ir más allá de los puntos de vista centrados en el ambiente.* Este punto es crítico para el diseño de aplicaciones GIS, debido a que los diseñadores tienden a usar estructuras definidas en ambientes GIS durante el proceso de diseño.

3- Un caso de uso: Modelando el comportamiento de los Huracanes.

3.1 Objetivos

El modelo a construir describe cómo los huracanes afectan una determinada región de la tierra, en particular desde el punto de vista económico y estadístico. Todo dato acerca del huracán es conocido a través de las mediciones de estaciones meteorológicas, que proveen los valores de diferentes variables de la atmósfera en ciertos puntos de una región sensada. El cálculo de destrozos y pérdidas se hace a partir de información acerca de la densidad de población y construcción de cada región para poder prevenir las evacuaciones. La información relevante para el problema acerca de los huracanes incluye : como se comportan al moverse sobre una región del globo, conocer su intensidad, tamaño, los puntos por donde pasó y como afectó a esa región, asimismo como la probabilidad de que afecte otras zonas.

3.2 Estrategia

La metodología de diseño que usamos para atacar el problema es la propuesta en [DasNeves97], consistente en dividir la elaboración del modelo en dos etapas (un modelo conceptual y un modelo geográfico). La principal ventaja de esta metodología es que nos permite desacoplar la definición abstracta de la aplicación, de las representaciones espaciales de sus entidades.

En el primer paso modelamos los objetos conceptuales que representan a los elementos del dominio, permitiéndonos comprender mejor el problema, sin tener en cuenta sus características geográficas. El resultado de este paso es un modelo orientado a objetos similar al que se obtiene cuando se modelan aplicaciones convencionales.

En el segundo paso (modelado geográfico) agregamos las características espaciales y temporales al resultado del paso anterior. No necesariamente todos los objetos del modelo conceptual tienen que tener características dependientes del

espacio/tiempo (objetos puramente conceptuales y sin representación geográfica). Además, en esta etapa, pueden aparecer objetos puramente geográficos, que no tengan asociado un modelo a nivel conceptual.

3.3 Un primer enfoque a la solución del problema

Un *Huracán (o Tifón* en el océano Pacífico*)* es un ciclón tropical tibio en el que los vientos son mayores o iguales que 119 Kph. Los huracanes aparecen en un punto en un momento determinado, y se mueven dentro de una región siguiendo una ruta. Una vez que son detectados reciben un nombre que los identifica. Durante su recorrido y dependiendo de la intensidad de los mismos, destruyen construcciones, plantaciones, etc. generando pérdidas humanas y materiales. El daño que ocasionan está directamente vinculado con las velocidades de los vientos y reciben una categorización (5-Catastrofico, 4-Extremo, 3-Extenso , 2-Moderado, 1-Mínimo). Más información acerca de ciclones y huracanes puede ser encontrada en [Hardy83], [Llauge86], [Donn78] y en el web del National Hurricane Center (*www.nhc.noaa.gov* o en *www.tiphoon.org).*

Las *Estaciones Meteorológicas* están distribuidas a lo largo de todo el mundo. Estas recolectan y almacenan información acerca de la atmósfera en períodos de tiempo regulares. Mas información acerca del tipo de datos que recolectan y registran las mismas puede ser encontrada en el "site" del National Climatic Data Center (*www.ncdc.noaa.gov)*.

Un problema de diseño recurrente en este contexto involucra aquellos objetos que poseen características conceptuales y geográficas al mismo tiempo. Por ejemplo, en el problema que aquí tratamos, las *Estaciones Meteorológicas* y los *Huracanes* tienen ambas características.

3.4 Usando decoradores para agregar características geoespaciales

Existe una manera de agregar a los objetos conceptuales sus características geoespaciales sin necesidad de redefinir los objetos, ni de subclasificarlos. Para ello aplicamos el pattern Decorator definido en [Gamma95]. La idea básica consiste en envolver al objeto conceptual (en nuestro ejemplo un *Huracán*) con otro objeto que le agrega las características geoespaciales (*GeoHuracan*). La estructura de la relación entre estas clases es la que se muestra en la Figura 1 [1].

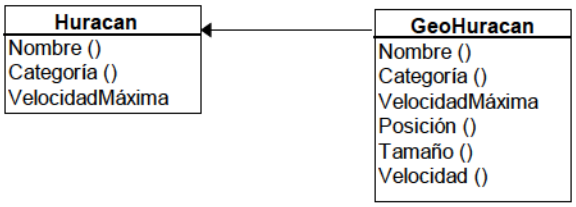

Figura 1 : Uso del decorador para agregar las características geoespaciales del Huracán

Algunas de las clases de objetos que aparecen en nuestro diseño y que no poseen modelo conceptual son las *regiones* afectadas por huracanes y las *rutas* que sigue un huracán. Las *rutas* están representadas por una colección de puntos

[1] La notacion que utilizamos para representar la estructura de los Patterns es la propuesta en [Rumbaugh91].

georeferenciados. Las *regiones* son polígonos cerrados que cubren áreas de la tierra. Estas clases de objetos son puramente geográficos.

Una posible implementación que se propone para este patrón consiste en tener una clase abstracta que agrupe el comportamiento común de aquellos objetos que envuelven al modelo conceptual más aquellos que pertenecen al modelo de la aplicación GIS. En la figura 2 vemos la jerarquía de Geo-Objetos que incluye decoradores y objetos que no se relacionan con objetos conceptuales. En la figura 3 mostramos una instanciación de esta jerarquía en el ámbito del problema que estamos estudiando.

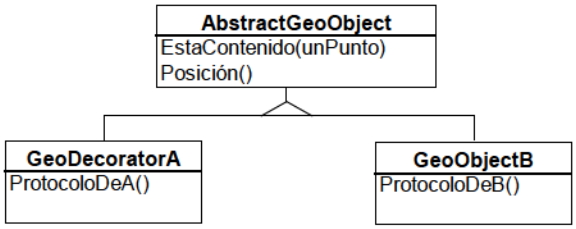

Figure 2 : Una jerarquía de Geo-Objetos que incluye objetos decoradores y objetos que no se relacionan con objetos conceptuales.

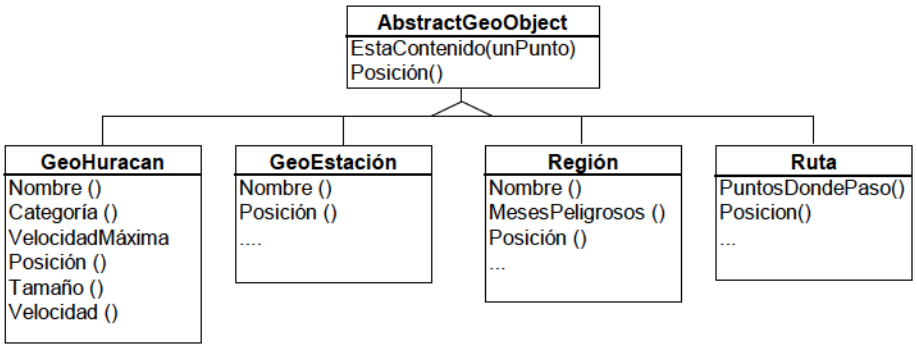

Figura 3 : Un ejemplo de la jerarquía de Geo-Objetos para algunas de las clases del sistema de Ciclones

El protocolo para manipular las características geo-referenciadas comunes a todos los objetos geográficos del sistema, esta definida en la clase abstracta GeoObject, que define el comportamiento necesario para manipular posiciones (Location) en el espacio y tiempo tal como esta definido en [OGIS96].

Gracias a la transparencia que nos brinda el uso de los decorators, podemos utilizarlos recursivamente, permitiéndonos agregar responsabilidades en forma incremental. Como ejemplo consideremos las *Regiones* dentro del contexto de nuestra aplicación. A través de datos históricos podemos conocer la probabilidad de que aparezca un huracán que azote una área determinada dentro de un período de tiempo dado (en general durante ciertas estaciones del año). Además una vez que ha aparecido un huracán sobre una región y gracias a modelos y heurísticas que se conocen acerca del comportamiento de los mismos, es posible predecir dentro de un período corto de tiempo (por. ej. 72 hs) la probabilidad de que ese huracán dado ataque ciertas partes de la región. De esta forma vemos como se hace necesario agregar nuevo comportamiento a un mismo objeto, en forma dinámica. Esto también

puede resolverse mediante el uso de este patrón. La figura 4 muestra un ejemplo del uso recursivo de los decoradores (decorar a otro decorador).

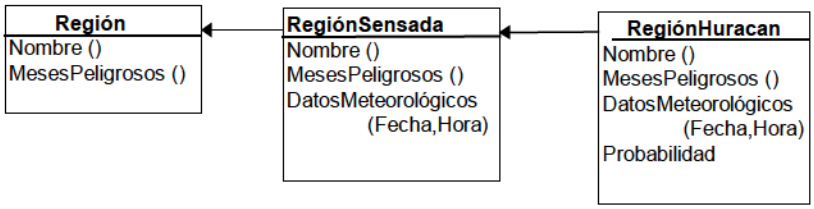

Figura 4 :Uso recursivo de decoradores

Como vemos el uso de los decoradores nos brinda más flexibilidad que la herencia estática, y nos permite agregar comportamiento a objetos en tiempo de ejecución. En lugar de hacer clases complejas y adaptables que intenten soportar todo el comportamiento, el uso decoradores nos permite agregar funcionalidad en forma incremental componiendo piezas simples. Sin embargo su uso trae algunos problemas como la gran cantidad de pequeños objetos que se agregan al sistema (lo cual agrega una sobrecarga en el dispatching de mensajes y problemas de eficiencia en el almacenamiento) y problemas de identidad de los objetos (una componente decorada no es idéntica a la componente en si).

3.5 Solucionando el problema del comportamiento variable

En algunas situaciones necesitamos emplear diferentes algoritmos para solucionar un mismo problema y la elección de cual será el algoritmo a usar, se toma en tiempo de ejecución. Durante el diseño del sistema de huracanes, este problema apareció en reiteradas ocasiones.

Por ejemplo, dada una región el sistema tiene que permitir obtener para cada punto de la misma la probabilidad de que un huracán pase dentro de un radio de 50 millas. La forma en que se calcula esta probabilidad es a través de modelos estadísticos y de ciertas heurísticas. Dado que pueden existir diferentes modelos para diferentes regiones, y que puede existir más de un modelo para una región queremos permitir que el sistema soporte distintos algoritmos. Además queremos que éstos puedan ser aplicados según sea necesario, sin necesidad de que los mismos queden atados a ningún objeto en particular, ni que haya mas de una representación para una misma región.

Otro caso en el que se presenta el mismo problema de diseño del ejemplo anterior es en la determinación del centro y la ruta que sigue un ciclón. El centro de un ciclón esta determinado usualmente por los patrones de nubes, viento y/o distribución de presiones. La ruta que sigue un ciclón esta dada por los puntos por los que pasó el centro del mismo. Hay varias formas de obtener el centro, algunas son por aviones de reconocimiento, satélite, radar, datos sinópticos, etc.. Dependiendo de como se originen los datos, hay diferentes algoritmos para obtener el centro y la ruta que realizó.

Un tercer caso donde aparece el ya citado problema de diseño es para el cálculo del "Movimiento Presente". El "Movimiento Presente" es la mejor estimación del movimiento del centro de un ciclón tropical en un momento dado y en una posición dada. Esta estimación no refleja las oscilaciones de pequeña escala, en periodos cortos, del centro del ciclón. Existen diferentes algoritmos para obtener esta estimación.

En general no es deseable codificar estos algoritmos dentro de las clases de los clientes que los requieren por varios motivos :

- los clientes de estos algoritmos se hacen mas complejos, grandes y difíciles de mantener
- los diferentes algoritmos serán útiles en diferentes momentos o circunstancias, no pudiendo asignarlos dinámicamente
- se hace difícil agregar nuevos algoritmos o variar los existentes, cuando son parte de los clientes

De esta forma una clase que tiene un comportamiento variable en diferentes contextos, tendría muchas instrucciones condicionales dentro de sus operaciones ; haciendo las mismas difíciles de comprender y extender.

Podemos evitar estos problemas definiendo clases que encapsulen diferentes algoritmos que atacan un problema común. Un algoritmo que esta encapsulado de esta forma es llamado un *strategy*

3.5.1 Aplicación del patrón *Strategy*

Este patrón define una familia de algoritmos, encapsulando cada uno y haciéndolos intercambiables. Strategy permite que los algoritmos varíen independientemente a partir de los clientes que los usan. Es decir que es posible configurar una clase con uno de muchos comportamientos (por ej. diferentes formas de obtener el centro del huracán). Un punto interesante es que un algoritmo podría tener que emplear datos que los clientes no tienen porqué conocer, gracias al strategy, evitamos la exposición de estructuras de datos complejas y específicas de algún algoritmo en particular. En la figura 5 mostramos el uso de strategy para resolver el problema de calcular la posición del centro del Huracán. En este caso el diseño solo contempla dos formas de calcular el centro del huracán (mediante el uso de información proveniente de radar y mediante datos satelitales). Nótese que si quisiéramos extender el diseño para que soporte otras estrategias (algoritmos) solo debemos agregar subclases a la clase abstracta HuracanStrategy. Un GeoHuracan puede ser configurado en tiempo de ejecución con el algoritmo necesario de acuerdo al tipo de información que se disponga en ese momento.

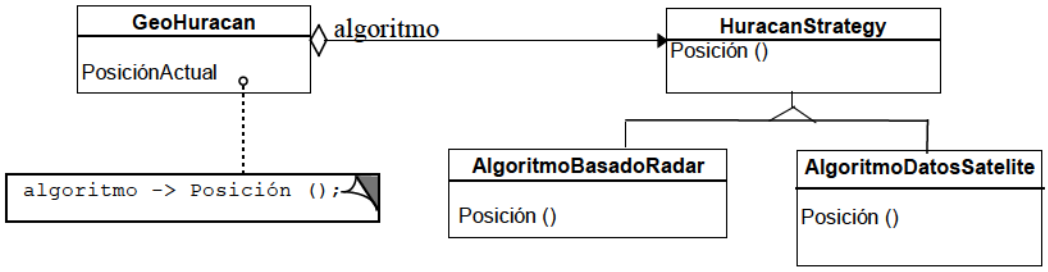

Figura 5: Estructura del patrón Strategy instanciado en el cálculo de la posición de un huracán.

3.64- Resolviendo la interacción con otros sistemas

Uno de los problemas que aparece en el diseño de esta aplicación es la necesidad de interactuar con otros sistemas o con conjuntos de subsistemas. En este caso particular hay que interactuar con sistemas de información catastral para poder

realizar proyecciones de daños económicos que podrían ser causados por el paso de un potencial huracán. Otro caso de uso sería para estudiar el volúumen de pérdidas que un huracán provocó al pasar por una región, teniendo un modelo probabilístico del impacto basado en la intensidad y otras variables. En ambos casos es necesario obtener información demográfica y de las densidades de construcción de una región. Esta información se obtiene a partir de otros sistemas (por ejemplo aplicaciones de catastro).

Estos sistemas catastrales pueden ser otras aplicaciones GIS, aplicaciones de Bases de Datos u otros. Nuestra aplicación actuará en este caso como un clientes de estos sistemas para recolectar cierto tipo de informacón específica.

Dejando de lado la complejidad de la conexión particular de cada caso (olvidándose de como un cliente recupera esos objetos para utilizarlos), lo cual no es un problema trivial, nuestra intención es desacoplar la forma en que nuestro sistema va a interactuar con otras aplicaciones. Dado que el sistema es externo al nuestro, es necesario tener una representación interna del mismo, de manera de poder interactuar con él. Esta representación cumple la función de abstraer los mecanismos de comunicación con el mismo (pensemos por ejemplo que algunos de los sistemas catastrales pueden ser remotos y ser accedidos mediante una W.A.N.), abstraer la forma en que se representan los datos y proveer un mecanismo uniforme para acceder a estos.

El problema que atacaremos aquí es cómo podemos acceder en forma uniforme a estas aplicaciones externas (ya existentes) mediante una interface común a todas ellas. No nos ocuparemos aquí de como manejar la comunicación con sistemas externos, ni como podríamos elegir (en el caso de que existan más de uno) que sistema de catastro nos resulta mas conveniente (por su granularidad, precisión, velocidad de respuesta, etc.).

Consideraremos que las operaciones que nuestra aplicación GIS va a solicitar de los otros sistemas tendrán una funcionalidad similar entre si, y en ciertos casos serán necesario realizar un preprocesamiento o postprocesamiento. Entonces el problema que tenemos es poder acceder a los otros sistemas mediante un protocolo unificado, aún cuando las interfaces de estos sean heterogéneas y diferentes de las que nuestro sistema espera.

Una forma de solucionar este problema es mediante el uso del pattern Adapter Facade[Gamma95]. AdapterFacade convierte la interface de una clase en otra interface que el cliente espera. Adapter permite que estas clases colaboren, cuando de otra forma no prodrían hacerlo debido a la incompatibilidad entre las interfaces. provee una protocolo unificado a un conjunto de interfaces de un subsistema complejo. Es decir que en este caso tendremos un único punto de entrada para estos sistemas externos. En la figura 6 se muestra la estructura de este pattern. Nuestra aplicación GIS (el sistema de huracanes) aparece como *Client*. *Target* es una clase abstracta que define la interface específica del dominio que nuestro cliente usa. *Adaptee* representa el sistema externo (Catastro) con el cual nos queremos comunicar. *Adaptee* define la interface existente que necesita ser adaptada. El *Adapter* es una implementación concreta que adapta la interface del *Adaptee* con la del *Target*.

Aquí planteamos utilizar el mismo concepto del pattern Adapter de [Gamma95], pero a diferencia del mismo la relación entre el Adapter y el Adaptee no es una simple referencia entre objetos sino que es un objeto que provee la comunicación con el sistema externo.

El *Adapter* puede realizar solo un simple cambio de protocolo (cambiando los nombres de los mensajes y el orden de los argumentos), o ser algo más complejo que realice tareas antes y/o después de enviar mensajes al *Adaptee*.

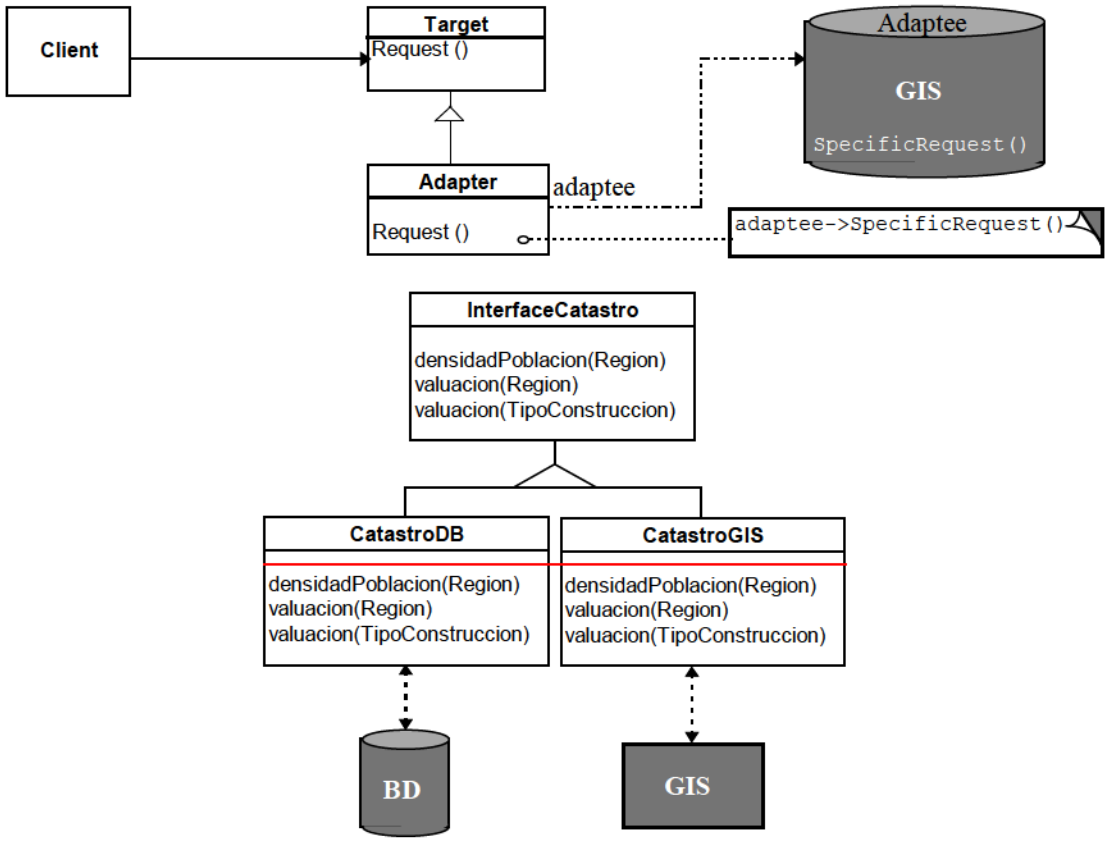

Figura 6 :Ejemplo del uso de AdapterFacade para interactuar con dos sistemas catastrales.

Es posible implementar una jerarquía de Facade para acceder a subsistemas que cumplan el mismo objetivo, pero que tengan interfaces diferentes. Esto se muestra en la figura 6. Para ello utilizaremos una clase abstracta que especifique la interface común y subclases concretas que implementen la forma de acceder a los diferentes subsistemas. De esta manera una clase concreta conoce a que objetos del subsistema (en nuestro caso un sistema catastral) delegar los pedidos del cliente (el sistema de huracanes). Los subsistemas implementan la forma de resolver el trabajo asignado por el facade y no tienen conocimiento del mismo (no tienen referencias al facade).

Este pattern define una interface de alto nivel que hace más fácil de usar a estosun sistemas externos. Los clientes envían mensajes a instancias de Adapter. El adapter llama a operaciones del Adaptee para poder entregar las respuestas.

Desde el punto de vista de los clientes –(en este caso nuestro sistema de Huracanes) esto implica que en vez de trabajar con diferentes protocolos dependiendo del sistema al que podemos acceder, todos los sistemas externos se accederán con un protocolo uniforme.

un conjunto de objetos para acceder a los aspectos externos de los mismos se tiene un único objeto que lo representa dentro del espacio del cliente.. Esto nos da independencia de los otros subsistemas y portabilidad.

45- Conclusiones

En este artículo presentamos la aplicación de técnicas de la programación orientada a objetos en el contexto del diseño de sistemas de información geográfica.

En particular comentamos el desarrollo de una aplicación empleando las guías propuestas en [DasNeves97]. Estas consisten en realizar una modelización de la aplicación en dos etapas (modelo abstracto y modelo geográfico). Una primera etapa en la que se reconocen los objetos conceptuales de una aplicación y se ponen de manifiesto las responsabilidades y colaboraciones entre estos. En la segunda etapa se tienen en cuenta las características geoespaciales de los objetos conceptuales y se agregan al modelo aquellos objetos que son puramente geográficos.

A partir del reconocimiento de que en algunos casos la clase no es la unidad de diseño adecuada para proyectos de mediano porte, investigamos el uso de patrones de diseño como organización de microarquitecturas.

En particular comentamos el empleo de :

- Decorator como mecanismo para agregar información geoespacial a objetos del dominio conceptual de una aplicación.
- El pattern Strategy que nos permite disponer diferentes algoritmos para resolver un mismo problema, desacoplados de los objetos que van a ser clientes de los mismos. Estos algoritmos pueden ser intercambiados en tiempo de ejecución.
- El pattern Adapter~~Facade~~ que permite a nuestra aplicación interactuar con otros sistema de información geográfica u otras aplicaciones, como pueden ser sistemas de bases de datos, legacy-systems, etc. ; disminuyendo el grado de acoplamiento con los mismos, teniendo una representación interna de estos en nuestro modelo y una interface estandar para acceder ~~único punto de entrada~~ a los mismos.

La experiencia acumulada en el desarrollo de este proyecto nos lleva a dos conclusiones. La primera es que la utilización de herramientas de la ingeniería de software moderna permite obtener buenos diseños de aplicaciones geográficas. La segunda es que la aplicación de los patrones de diseño sobrepasa el ámbito del desarrollo de software convencional, pudiendo ser aplicados en áreas que incluyen el manejo de grandes volúmenes de información y características particulares. A través de este trabajo intentamos ejemplificar esto y motivar la búsqueda de nuevos patrones de diseño específicos para aplicaciones GIS.

5- Referencias

[Alexander77] C. Alexander, S.Ishikawa, M.Silverstein, M.Jacobson, I.Fiksdahl-King and S.Angel : "A Pattern Language". Oxford University Press, New York, 1977.

[DasNeves97] Das Neves F., Gordillo S., Mostaccio C., Levato A.*"Toward a foundation for Object Oriented GIS Design"*. Join European Conference. Viena, Austria, April 1997.

[Donn78] William L. Donn. *" Meteorología "* . Editorial Pereité S.A.

[Fouler97] M.Fouler : "Patterns : Reusable Object Model". Addison Wesley, 1997

[Gamma95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *"Design Patterns. Elements of reusable Object-Oriented Software"*. Addison Wesley, 1995.

[Gordillo96] Gordillo S., Das Neves F., Mostaccio C., and Levato A. *"Design Patterns for Geographic Information Systems"*. Proceedings of First International Conference on Geographic Information Systems in Urban Regional and Environmental Planning. Island of Samos, Greece 1996.

[Hardy83] R. Hardy, P. Wright, J. Gribbin, J. Kington. *"El Libro del Clima"*. Herman Blume Ediciones.

[Llauge86] Félix Llauge. *" Iniciación a la Meteorología "*. Marcombo-Boixareu Editores.

[OGIS96] Open GIS Consortium (OGC) (1996b), The Open GIS Guide -A Guide to Interoperable Geoprocessing, Available at http://ogis.org/guide/guide1.html

[Rumbaugh91] J. Rumbaugh, M. Blaha, M. Premerlani and W. Lorensen: *"Object-Oriented Modelling and Design"*. Prentice Hall, Englewoods Cliff, New Jersey, 1991

[Schmidt95] D. Schmidt: *"Using Design Patterns to Develop Reusable Object-Oriented Communication Software"*. Comm. Of the ACM, October 1995, pp 65-74.

[Shekhar~~Shashi~~ 97] Shashi Shekhar, Mark Coyle, Brajesh Goyal, Duen-Ren Liu and Shyamsundar Sarkar : *"Data Models in Geographic Information Systems"*. Comm. Of the ACM, April 1997, pp 103-111.

[Tryfona95] N. Tryfona and T. Hadzilacos: *"Geographic Applications Development: Models and Tools for the Conceptual Level"*. Proceedings of ACM-GIS'95. Baltimore, Maryland, USA, pp 19-28.