SEMANTICAS DE LAS RELACIONES DE COMPOSICION PARA APLICACIONES GEOGRAFICAS

María Fernanda Comelli
Adriana Laura Olivero

Director: Lic. Silvia Gordillo

INDICE

1. INTRODUCCIÓN

Un sistema de información geográfica (SIG) es un sistema usado para almacenar y manipular información georeferenciada (información que tienen definida una posición en el espacio). Estos sistemas trabajan con dos tipos de datos: datos conceptuales que describen entidades en términos de atributos descriptivos y datos que representan todos los aspectos relacionados con características espaciales. La tecnología SIG involucra aspectos complejos tales como la adquisición de datos, la corrección de los mismos, la representación de relaciones espaciales y de la topología y el diseño de la interfaz. La gran variedad de datos que caracterizan a estas aplicaciones, hacen el proceso de diseño dificultoso y complejo. En este contexto, modelar y diseñar actividades requiere de una gran práctica en el dominio de la aplicación. Hoy en día la construcción de este tipo de aplicación se vuelve una tarea completamente artesanal, es muy difícil encontrar documentación de estos sistemas, ya que la preocupación de los diseñadores pasa por obtener implementaciones eficientes, en vez de un buen diseño de la aplicación.

Un problema usual en este tipo de aplicaciones se produce cuando las entidades son diseñadas en término de su representación espacial en vez de su definición abstracta. Como consecuencia, el desarrollo de estos sistemas implica altísimos costos desde el punto de vista de reusabilidad, mantenimiento y extensibilidad. Una solución válida para este problema podría ser aplicar conceptos orientados a objetos para diseñar estructuras abstractas de aplicaciones geográficas. En particular, las relaciones de composición aparecen como una constante en este tipo de aplicaciones, pero para poder aplicarlas, es necesario estudiar cuál es el impacto de las mismas cuando se establecen entre objetos con geo-referenciamiento.

Desde el punto de vista del paradigma orientado a objetos, existen objetos que referencian a otros, a estas referencias las podemos definir con la semántica de la relación parte_de.

Desde el punto de vista de los sistemas geográficos, definir y entender la semántica exacta de la composición es muy importante, ya que permite establecer un diseño apropiado.

El objetivo de esta tesis es enriquecer los métodos de diseño de aplicaciones SIG, utilizando el Modelo Orientado a Objetos. En particular se pretende estudiar la semántica de las relaciones de composición de este modelo con objetos referenciados geográficamente. Para ello, es necesario estudiar cóomo se reflejan las relaciones de

composición conceptuales en un modelo geográfico. Además es necesario identificar composiciones conceptuales y composiciones que se refieren a las posiciones y a la geometría de los objetos. De esta forma se hace necesario ampliar la semántica de estas relaciones de manera de poder reflejar la composición en ambos niveles. Para realizar este trabajo se tomará como base el modelo de diseño propuesto en [Gordillo et al.97], que define dos niveles en el desarrollo de una aplicación geográfica: el nivel conceptual, en donde se definen las clases sin tener en cuenta sus características espaciales y el nivel geográfico en donde se refinan los objetos de manera de incorporar características espaciales. Este modelo utiliza el concepto de "Patrones de Diseño" para resolver situaciones recurrentes.

En el Capítulo 2 presentamos una introducción a los Sistemas de Información Geográficas (SIG). En el Capítulo 3, presentamos las nociones básicas de "Patrones de Diseño" y tratamos en particular algunos patrones que nos ayudan a resolver problemas que comúnmente encontramos en dominios de aplicaciones SIG. En el capítulo 4 hacemos una breve introducción al Modelo de Diseño propuesto en [Gordillo et al.97], en el cual nos basamos para ampliar las semánticas de las relaciones de composición. En el Capítulo 5 introducimos el concepto de Objetos Compuestos en el Paradigma Orientado a Objetos y describimos algunos trabajos relacionados que utilizan relaciones de composición para modelizar aplicaciones SIG. En el capítulo 6 extendemos las semánticas de las relaciones de composición en aplicaciones SIG. En el capítulo 7 se describen las conclusiones, así como los trabajos futuros que complementarán el proyecto.

2. SISTEMA DE INFORMACIÓN GEOGRÁFICA (SIG)

2.1 Introducción

Los Sistema de Información Geográfica (SIG), se caracterizan por trabajar con datos que describen fenómenos que ocurren en la superficie terrestre, también conocidos como Sistemas de Información Espacial. En este contexto, espacial es el término utilizado para referirse a datos con una ubicación en cualquier espacio, por lo que un SIG representa una perspectiva o manera de analizar y razonar acerca de este tipo de problemas.

Un mapa contiene una cantidad de información usada en diferentes formas, por diferentes individuos y organizaciones. Representa los significados de nuestra propia ubicación en relación al mundo que nos rodea. Los mapas se usan en diversas aplicaciones, desde la localización de líneas telefónicas hasta mostrar la deforestación de una zona boscosa.

Un SIG provee la facilidad para extraer los diferentes conjuntos de información de un mapa (calles, poblaciones, vegetación, etc.) y usarlos a medida que se requieran. Esto da una gran flexibilidad permitiendo producir rápidamente un mapa que satisfaga exactamente las necesidades del usuario. Sin embargo un SIG va más allá, debido a que los datos se almacenan en una computadora, haciendo posible el análisis y la modelización. Por ejemplo se podrían ubicar dos edificios, mostrar la información de cada uno de ellos desde una base de datos asociada (mucha más información de la que podría ser mostrada en un mapa en papel) y luego calcular la distancia más corta entre ambos.

Es decir, un SIG no es un sistema para crear simples mapas, aunque pueden hacerse en diferentes escalas, en diferentes proyecciones y con diferentes colores. Un SIG no guarda un mapa en un sentido convencional, no guarda una imagen particular de un vista de un área geográfica, sino que guarda los datos con los cuales se puede dibujar una vista deseada para alcanzar un propósito particular.

Esta tecnología se ha desarrollado tan rápidamente en las dos últimas décadas que actualmente es aceptada como una herramienta esencial para el uso efectivo de la información geográfica.

Si bien el SIG ha provisto un uso más sistemático de información espacial y en una diversidad mayor de disciplinas, la facilidad con la que puede manipular esta información ha creado una dificultad importante. Los usuarios no familiarizados con las técnicas de SIG o con la naturaleza de la información espacial pueden fácilmente derivar tanto en análisis inválidos como válidos. Válidos o no, los resultados tienen la precisión asociada con gráficos de computadoras sofisticados y grandes volúmenes de tabulaciones numéricas. Para establecer las políticas apropiadas para el uso de esta tecnología es crucial un entendimiento completo por parte de usuarios, gerentes y encargados de tomar decisiones en organizaciones públicas y privadas.

2.2 Generalidades

Un SIG recolecta, almacena y analiza objetos y fenómenos donde la localización geográfica es una característica importante o crítica para el análisis.

Por ejemplo la ubicación de una estación de bomberos en una aplicación de atención de emergencias o las localizaciones donde la erosión de los suelos es más severa en una aplicación ambiental, son consideraciones claves para el uso de la información. En cada caso se debe tener en cuenta qué es y dónde está.

Mientras que la manipulación y el análisis de los datos que son referenciados por una localización geográfica son capacidades claves de un SIG, la potencia del sistema se hace más evidente cuando la cantidad de datos involucrados es demasiado grande para ser manejada manualmente. Puede haber cientos o miles de características a ser consideradas, o puede haber cientos de factores asociados con cada característica o ubicación. Esos datos pueden existir como mapas, tablas de datos o aún como listas de nombres o direcciones. Estos grandes volúmenes de datos no son manejados eficientemente usando métodos manuales. Sin embargo cuando se introducen estos datos en un SIG, pueden manipularse y analizarse fácilmente en formas que serían prácticamente imposibles usando métodos manuales, porque serían muy costosos y consumirían mucho tiempo.

2.3 Uso de SIG para toma de decisiones

Existe una estrategia fundamental subyacente en todos los análisis de datos georeferenciados. Un entendimiento de esta estrategia no sólo conducirá a un mejor uso

de los métodos disponibles, sino que también a la necesidad de entender cómo se relacionan los diferentes niveles de abstracción de datos.

Tenemos que tomar decisiones que requieren conocimiento sobre nuestro mundo complejo. Ya que no tenemos un conocimiento completo, generalmente tomamos decisiones con información incompleta. Seleccionamos la información relevante para recordar y registrar. Usando este proceso de selección creamos un modelo conceptual de nuestro mundo. El término modelo se usa para significar un conjunto de relaciones o información sobre el mundo real. Nuestro modelo conceptual de algo es nuestro entendimiento de qué es ese algo y cómo se comporta. Cuando necesitamos tomar decisiones sobre el mundo real nos referimos a nuestro modelo, el cual es mucho más simple que el mundo real mismo. Es más simple ya que hemos seleccionado la información para incluir en el modelo las cosas que son relevantes para nosotros. Los detalles que no necesitamos tienden a ser olvidados selectivamente.

El proceso de usar un SIG comienza y finaliza en el mundo real. Recolectamos información sobre el mundo real, la que necesariamente es una abstracción. No podemos ni deseamos manejar hasta el último detalle. Usaremos esta información para tomar decisiones, y finalmente implementaremos esas decisiones, es decir aplicaremos nuestro razonamiento abstracto al mundo real.

Un "buen" sistema de información es aquel que nos provee de los datos necesarios, organizados de manera tal que podamos tomar decisiones correctas sobre el mundo real. Pero, Qué significa realmente la "decisión correcta"?. La decisión correcta es la que mejor logra los objetivos de aquel al que sirve el sistema. Para esto, es necesario saber cuáles son esos objetivos y poder predecir correctamente los resultados de elecciones alternativas. Tomar la "decisión correcta" requiere que los datos relevantes sean presentados en la estructura de un modelo apropiado, el cual sea evaluado usando criterios válidos.

El éxito de un sistema de información geográfica está determinado por varios factores que pueden agruparse en cuatro temas: el conjunto de datos, la organización de los datos, el modelo y los criterios.

2.4 Obtención de los datos relevantes

Los datos usados en un SIG representan algo sobre el mundo real en un punto en el tiempo. Siempre son una abstracción de la realidad, ya que no necesitamos ni deseamos todos los datos, sólo los que pensamos que serán útiles. Los datos que decidimos tomar constituyen la primera restricción en las capacidades del SIG: No se pueden usar los datos que no se tienen.

Entonces, por qué no tomar todos los datos?. Primero, porque nunca se podrían recolectar todos los datos, y segundo, porque aunque se pudieran conseguir todos, no se desean. Es muy costoso recolectar todos los datos. La recolección de datos más efectiva en costo es recabar sólo los datos que se necesitan.

Es costoso recolectar, almacenar y examinar grandes cantidades de datos innecesarios. El exceso puede hacer más difícil el uso de los datos que realmente se necesitan. Todo gasto de esfuerzo que no contribuye a la solución, representa tiempo, esfuerzo y recursos que podrían ser empleados en otra tarea para mejorar el análisis. El mismo argumento sirve para la calidad de los datos

Los aspectos más importantes de la calidad de los datos son: exactitud, precisión, tiempo, vigencia e integridad. La exactitud mide Cada cuánto, Por Cuánto y Cuán predeciblemente pueden ser corregidos los datos. La precisión mide la perfección de la escala usada para describir los datos. El tiempo indica en qué punto o en qué período de tiempo fueron obtenidos los datos. A veces el tiempo puede ser un factor crítico en la calidad de los datos. Existe información que puede quedar desactualizada rápidamente. La vigencia mide cuán recientemente fueron obtenidos los datos. En algunos casos la conveniencia de los datos dependerá de la estación o del año en que fueron obtenidos. En Canadá, por ejemplo se toman fotografías en verano para mapas de tipos de coberturas forestales. La integridad se refiere a la porción del área de interés para la cual los datos están disponibles. También se usa el término en referencia al sistema de clasificación usado para representar los datos. Siempre hay una correspondencia entre calidad y costo de los datos. La tendencia es que a mayor calidad, mayor es el costo de los datos.

2.5 La organización de los datos

La organización de los datos es el segundo factor en importancia a considerar en el uso exitoso de un SIG. Para proveer esta organización se usa una base de datos, la cual es crítica ya que:

Los datos no son de valor a menos que los datos correctos estén en el lugar correcto en el tiempo correcto.

Dependiendo de la cantidad de los datos y de la performance que necesitamos de nuestro sistema de base de datos, pueden ser suficientes formas simples de organización. Sin embargo, en un SIG, la cantidad de datos es lo suficientemente grande para que la forma y performance de los datos sea un factor crítico.

2.6 El modelo de decisión

Un modelo representa un objeto o fenómeno que existe en el mundo real. Un buen modelo es el modelo más simple que predice correcta y consistentemente el comportamiento del mundo real para los fenómenos de interés. Se crean los modelos para predecir cómo se comportarán ciertos aspectos del mundo real. Describen las relaciones entre elementos de datos para predecir cómo ocurrirán los eventos en el mundo real. La calidad del modelo está limitada por los datos que se han seleccionado y la forma en que éstos son organizados. También está limitada por el costo de usarlo. El modelo más complejo es el más costoso. El modelo más efectivo en costo es el modelo más simple que da los resultados que logran el nivel mínimo requerido de exactitud con el mínimo costo de datos.

Por qué este interés por el costo?. La razón es que siempre hay un costo. Puede no ser dinero, puede ser tiempo, pueden ser respuestas incorrectas. Solamente prediciendo y evaluando esos costos se puede tomar una decisión racional sobre la performance. Siempre hay una relación entre costo y performance. Un nivel de performance demasiado bajo puede ser muy costoso ya sea en errores causados por imprecisiones, obteniendo resultados demasiado tarde, o perdiendo las mejores soluciones. Un nivel de performance demasiado alto también puede ser muy costoso gastando un esfuerzo que no mejora el éxito.

Es costoso tolerar niveles de performance demasiado altos o demasiado bajos.

2.7 Criterios Válidos

El cuarto factor importante en las aplicaciones exitosas de SIG, es el grado en el cual los criterios usados para evaluar el modelo propiamente, reflejan que los valores de la gente son satisfechos. Al final del proceso de análisis de la información, se debe realizar una acción. Debe decidirse la acción a tomar pensando las alternativas y considerando las consecuencias de cada alternativa como predicha por nuestros modelos. La acción puede ser tan simple como enviar un aviso de deuda para una factura impaga, o tan compleja como decidir construir un embalse y anegar un valle.

Para el proceso son fundamentales los encargados de tomar las decisiones, quienes tienen el mandato y la responsabilidad de las consecuencias de las acciones tomadas. No importa cuán alta sea la calidad de los datos y cuán apropiados los modelos usados, si se usan criterios erróneos para evaluar la información producida por un SIG; probablemente los resultados no serán satisfactorios. Las personas toman decisiones en función de criterios. Los criterios usados por las personas para tomar la decisión deben ser los mismos que los usados por las personas a ser satisfechas.

2.8 Datos georeferenciados

Los datos geográficos tienen dos componentes fundamentales:

1. El fenómeno reportado, tal como una dimensión física o clase: Ejemplos de una dimensión física podrían ser: la altura de un bosque, la población de una ciudad, o el ancho de una ruta. La clase podría ser un tipo de roca, un tipo de vegetación o el nombre de una ciudad

2. La ubicación espacial del fenómeno: Generalmente se especifica la ubicación con referencia a un sistema de coordenadas común.

Una tercera componente fundamental de la información geográfica es el tiempo. La componente tiempo, a veces no se establece explícitamente pero puede ser crítica. La información geográfica describe un fenómeno en una localización, que existió en un punto específico en el tiempo. Un mapa de cobertura de tierras, describe la ubicación de diferentes clases de coberturas de tierras que existieron en el momento de la obtención de los datos. Si el área cambia rápidamente, esta información puede quedar desactualizada pronto. La información entonces puede quedar inservible para tomar decisiones que requieran el estado actual de la tierra. Sin embargo, los datos pueden ser

invalorables para analizar tendencias históricas, tal como la conversión de tierras agrícolas a otros usos.

Los datos geográficos son una forma inherente de datos espaciales (definen una posición en el espacio). Los datos geográficos pueden representarse en un mapa o en un sistema de información geográfica como punto, línea o polígonos. Los puntos se usan para representar la localización de fenómenos geográficos en un punto o para representar figuras de mapa demasiado pequeñas para ser mostradas como polígonos o línea. La ubicación de una ciudad, el pico de una montaña o una pista de aterrizaje podrían ser representados por un elemento punto. Una línea consiste de un conjunto ordenado de puntos conectados. Las líneas se usan para representar figuras de mapas demasiado delgadas para ser mostradas como polígonos o para representar figuras que teóricamente no tienen ancho, tal como un límite político. Un borde de playa, una línea de contorno o un límite administrativo son ejemplos de líneas. Un polígono es una región encerrada por líneas. La extensión geográfica de una ciudad, un bosque o un lago podrían ser representados por un polígono.

2.9 Componentes de un SIG

Tomado en un sentido más amplio, un sistema de información geográfica es un conjunto de procedimientos usados para almacenar y manipular datos referenciados geográficamente. Se podría tomar como definición la que sigue: Un SIG es un sistema computarizado que provee los siguientes conjuntos de capacidades para manejar datos georeferenciados: 1. Entrada; 2. Manejo de datos (almacenamiento y recuperación); 3. Manipulación y análisis y 4. Salida.

El ambiente en el que debe operar un SIG se ilustra en la figura 2.1.

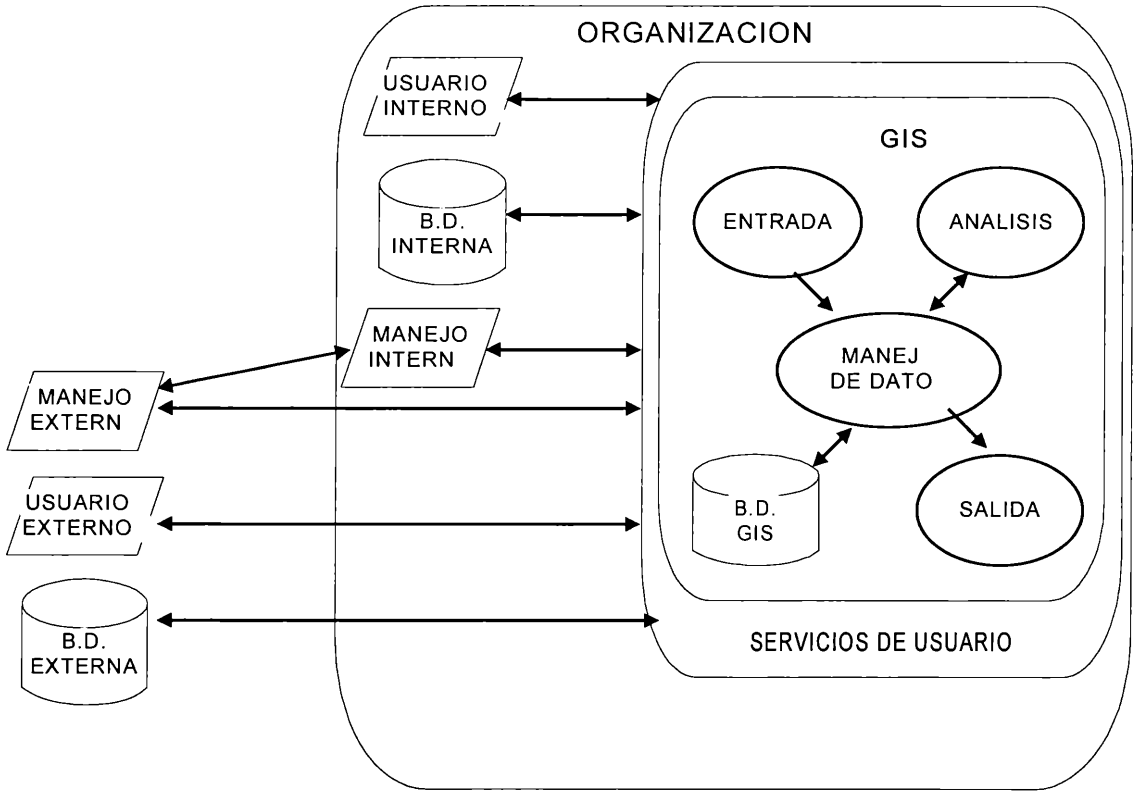


Figura 2.1 Ambiente para operar un SIG

A continuación se explican brevemente los componentes básicos de un SIG:

2.9.1 Entrada de datos

La componente entrada de datos convierte los datos desde su formato existente a alguno que puede ser usado por el SIG. Generalmente los datos georeferenciados son provistos como mapas, tablas de atributos, archivos, fotos aéreas, e imágenes satelitales. El procedimiento de entrada de datos puede ser directo como convertir un archivo de un formato electrónico a otro, o puede ser complejo. La entrada de datos es el principal cuello de botella en la implementación de un SIG.

El costo inicial para la construcción de la base de datos es comunmente de 5 a 10 veces el costo del hardware y software. Los datos que ingresan en un SIG son de dos tipos: datos espaciales, que representan la ubicación geográfica de los elementos (puntos, líneas y polígonos), usados para representar aspectos geográficos como calles, lagos, bosques, etc., y los datos no espaciales, representan la información descriptiva, como el nombre de la calle, la composisción del bosque,etc.

Generalmente los datos no espaciales son entrados por *teclado* o utilizando *archivos existentes*, mientras que los espaciales se pueden tomar a partir de *geometría de coordenadas, digitalización manual y scanners.*

Los *procedimientos de geometría* de coordenadas permiten ingresar datos por teclado y a partir de ellos se calculan las coordenadas espaciales, cada elemento espacial que se representa tiene un identificador que indica el tipo de elemento: punto, línea o polígono. Este método es uno de los más precisos pero es de 6 a 20 veces más caro que la digitalización manual [Aronoff91].

La *digitalización manual* es el método más usado para entrar datos espaciales de mapas. El mapa se pone sobre una mesa digitalizadora, y se van trazando los mapas con un cursor que calcula las coordenadas en forma digital. La mesa digitalizadora, electrónicamente codifica la posicisión del cursor con una precisión de fracciones de milímetros. La eficiencia de la digitalización depende de la calidad del software de digitalización y de la habilidad del operador.

Con el proceso de *scanning* se provee una medición más rápida de entrada de datos que la digitalización manual. Se produce una imagen digital del mapa, moviendo un detector electrónico que va recorriendo la superficie del mapa. Generalmente se utilizan dos tipos de scanners:

- "flat-bed", en este tipo de scanner, el mapa es ubicado sobre una superficie plana y el detector se mueve en dirección de X e Y.
- "Drum", en este scanner el mapa es montado sobre una superficie cilíndrica. El detector se mueve horizontalmente a medida que el cilindro gira.

El scanner genera una imagen digital y el nivel de detalle obtenido depende del tamaño del área del mapa que el detector puede ver.

Los métodos de entrada de datos y los standards de calidad de datos deben ser considerados cuidadosamente antes de comenzar la entrada de datos. Los diferentes métodos de entrada de datos deberían ser evaluados en términos del procesamiento a realizar, los standards de precisión a lograr, y el formato de salida a producir.

2.9.2 Manejo de Datos

La componente manejo de datos del SIG incluye las funciones necesarias para almacenar y recuperar datos desde la base de datos. Los métodos usados para

implementar esas funciones se refieren a cuán eficientemente el sistema realiza todas las operaciones con los datos. Hay una gran variedad de métodos para organizar los datos en archivos computarizados. La forma en que se estructuran los datos y la forma en que cada archivo se puede relacionar con otro, pone restricciones en el modo en que los datos pueden ser recuperados y la velocidad de la operación de recuperación.

En un SIG el almacenamiento y la presentación de los datos geográficos están separados. Los datos se pueden almacenar con un nivel alto de detalle y luego ser ploteados en un nivel más general y en una escala diferente. El mapa ploteado se convierte en una de las muchas formas de presentar los datos. Se convierte en efecto, en una vista de la base de datos geográfica. Se pueden ver los mismos datos como diferentes tipos de mapas. Cada uno puede ser customizado para un uso específico, además de los mapas los datos pueden ser presentados en forma de tabla o como descripciones en texto.

En un SIG los datos geográficos se representan como puntos, líneas o polígonos. La información de un elemento geográfico tiene cuatro componentes principales: su posición geográfica, sus atributos, sus relaciones espaciales y el tiempo. Simplemente, las cuatro componentes son: *donde está el objeto*, *qué es*, *cuál es su relación con otros elementos espaciales* y *cuándo existió* [Aronoff91].

Posición Geográfica

Los datos geográficos son principalmente una forma de datos espaciales. Cada objeto tiene una ubicación que debe ser especificada de una manera única. Las ubicaciones son registradas en términos de un sistema de coordenadas como por ejemplo Latitud/Longitud.

Un SIG requiere que se use un sistema de coordenadas común para todos los conjuntos de datos que van a ser usados juntos. Los datos geográficos se pueden almacenar en diferentes niveles de precisión posicional. Algunos datos pueden ser precisos en unos pocos centímetros, mientras otros solamente a diez metros por ejemplo.

Atributos

La segunda característica de los datos geográficos son sus atributos, es decir "qué es esto". Por ejemplo un bosque podría incluir la composición de las especies,

altura promedio, etc. Estos atributos suelen llamarse atributos no espaciales ya que no representan en sí mismos información de ubicación.

Relación espacial

La tercera característica de los datos geográficos son las relaciones espaciales entre los elementos del sistema. Por ejemplo, no solamente es importante conocer dónde hay un incendio, sino también dónde está la central de bomberos más cercana.

No es posible almacenar información sobre todas las relaciones espaciales, sino que algunas se definen explícitamente y el resto se calcula a medida que se necesita.

Tiempo

La información geográfica está referenciada a un punto en el tiempo o a un período de tiempo. Conocer el momento en que los datos geográficos fueron recolectados puede ser crítico para un uso apropiado de esos datos. Por ejemplo, un área puede estar cubierta de árboles un año y al siguiente éstos haber sido cortados.

La información histórica también puede ser una componente valiosa de la base de datos del SIG.

La representación del tiempo en un SIG es un nivel de complejidad agregado difícil de manejar.

Juntas, estas cuatro características, hacen que los datos geográficos sean difíciles de manejar. Al igual que con otros sistemas de bases de datos, se usa un modelo de datos para representar la información considerada de mayor relevancia.

Modelo de Datos Espacial

Existen básicamente dos modelos para representar los datos espaciales en un SIG:

- *El modelo vector*
- *El modelo raster*

En el *modelo vector* los objetos o condiciones del mundo real se representan por los puntos y líneas que definen sus límites, tal como si fueran dibujados en un mapa. La posición de cada objeto está definida por su colocación en un espacio de mapa por un sistema de referencia de coordenadas como se muestra en la figura 2.2c. Cada posición en el espacio del mapa tiene un valor de coordenada único. Se usan puntos, líneas y

polígonos para representar objetos geográficos distribuidos irregularmente o condiciones en el mundo real (un polígono es un área limitada por un lazo cerrado de segmentos de líneas rectas). Una línea puede representar una ruta, un polígono un bosque y así siguiendo.

En el *modelo raster* el espacio está subdividido regularmente en celdas, generalmente cuadradas como se muestra en a Figura 2.2b. La ubicación de los objetos geográficos o condiciones está definida por la posición de la fila y columna de las celdas que ocupan. El área que cada celda representa define la resolución espacial disponible. Ya que las posiciones son definidas por los números de fila y columna de la celda, la posición de los elementos geográficos es registrada en la celda más cercana.

El valor almacenado en cada celda indica el tipo de objeto o condición que se encuentra en esa ubicación. Entonces, en la propuesta *raster*, el espacio es poblado por un gran número de celdas distribuidas regularmente, donde cada una puede tener un valor diferente. Las unidades espaciales son las celdas, cada una de las cuales corresponde a un área en una ubicación específica, tal como un área en la superficie de la tierra. Los valores de la celda reportan una condición en una ubicación y esa condición pertenece a la celda completa. A diferencia de las del modelo *vector*, las unidades del modelo *raster* no corresponden a las entidades espaciales que representan en el mundo real. Las entidades espaciales o unidades en el modelo de datos raster, no son los objetos que conceptualizamos, son las celdas individuales. Por ejemplo, una ruta no existe como una entidad raster; las celdas que representan las rutas son las entidades.

En ambos modelos, la información espacial se representa usando unidades homogéneas. En la propuesta *raster* las unidades homogéneas son las celdas. En la propuesta *vector* las unidades homogéneas son los puntos, líneas y polígonos.

Las diferentes propuestas tienen ventajas y desventajas, las que se resumen en la figura 2.3.

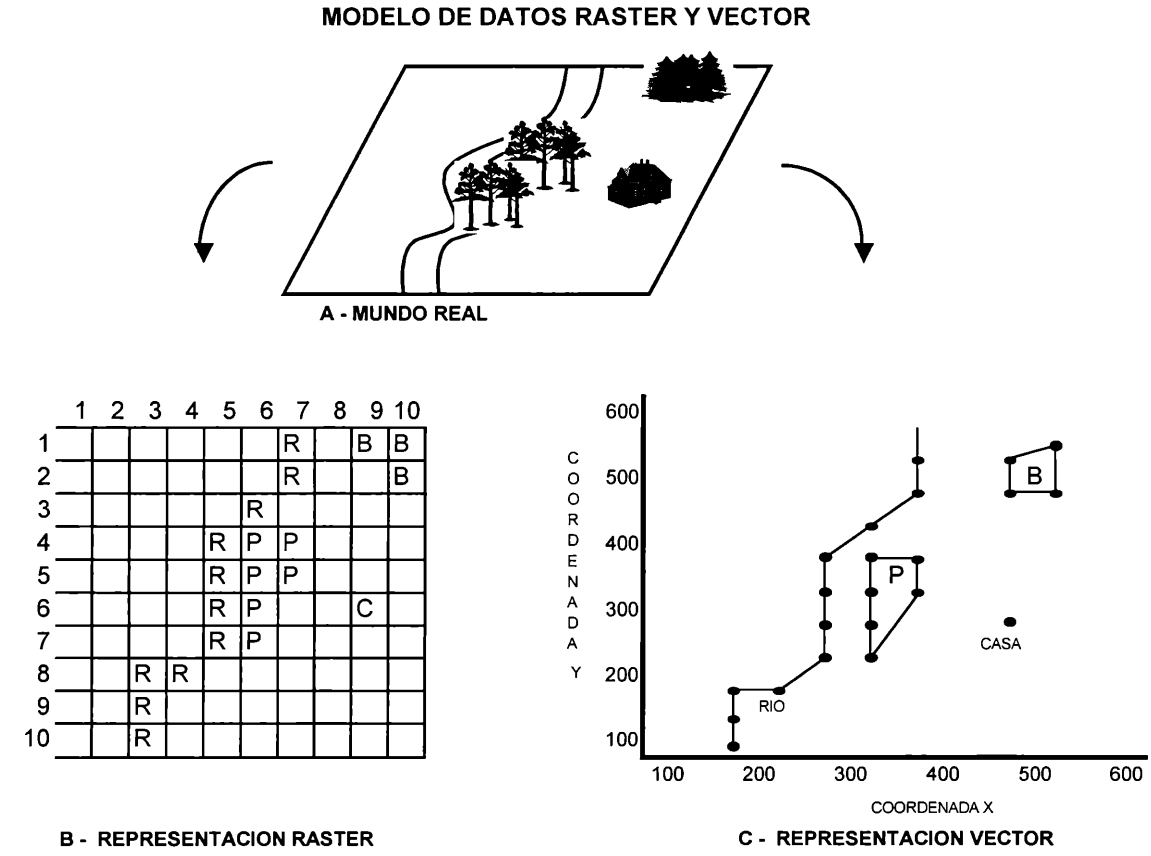


Figura 2.2 Modelos Raster y Vector

MODELO RASTER	MODELO VECTOR
Ventajas 1. Es una estructura de datos simple. 2. Las operaciones de cubrimiento son implementadas fácil y eficientemente. 3. Se representa eficientemente una gran variabilidad espacial. 4. Es requerido para una eficiente manipulación y mejora de imágenes digitales.	**Ventajas** 1. Provee una estructura de datos más compacta. 2. Provee un codificado eficiente de topología, y por lo tanto una implementación más eficiente de operaciones que requieren información topológica, tal como análisis de red. 3. Está mejor adaptado para soportar gráficos que se aproximan a los mapas dibujados a mano.

Desventajas	Desventajas
1. La estructura de datos es menos compacta. Las técnicas de compresión a veces solucionan este problema. 2. Son más difíciles de representar las relaciones topológicas. 3. La salida de gráficos es menos estética, ya que los límites tienden a tener una apariencia de bloque en lugar de las líneas de los mapas dibujados a mano. Esto puede solucionarse usando un gran número de celdas, pero puede resultar en archivos demasiado grandes.	1. Es una estructura de datos más compleja. 2. Las operaciones de cubrimiento son más difíciles de implementar. 3. La representación de mucha variabilidad espacial es ineficiente. 4. No se puede manipular y mejorar imágenes digitales efectivamente.

Figura 2.3: Comparación de los modelos Raster y Vector

2.9.3 Manipulación y análisis de datos

Respecto de la manipulación de la información hay tres situaciones básicas que se deben tener en cuenta:

1 - Acciones que requieren atributos no espaciales.

2 - Acciones que involucran sólo atributos espaciales.

3 - Acciones que combinan atributos espaciales y no espaciales.

La primera categoría incluye aquellas operaciones que manipulan atributos tales como los nombres de las entidades, la composición de las entidades o valores numéricos de los atributos. Las características de estas operaciones son que ninguna trata con atributos que tienen que ver con el georeferenciamiento de las entidades, es decir ni con su posición ni con las variaciones temporales que produzcan cambios en las posiciones. Estos atributos habitualmente son almacenados en una base de datos relacional y por lo tanto recuperados desde un lenguaje de consultas relacional.

Las acciones de la segunda categoría involucran operaciones tales como determinar cuáles son los países fronterizos a uno dado, medir el área de una región, determinar la longitud total de un sistema de cañerías. Es decir no está tratando con características descriptivas de las entidades, sino con características espaciales.

La tercera categoría incluye información tal como encontrar la cantidad de estudiantes que viven dentro de un área, o encontrar el área dañada de alguna región que ha sufrido tormentas.

En los dos últimos casos, donde se utilizan datos espaciales, la forma de resolver las consultas que se realizan depende directamente de cómo está organizada la información. Se podría pensar en realizar un pre-procesamiento de manera que la información espacial pueda ser almacenada en tablas de alguna manera y así compatibilizar los datos. Por ejemplo en el caso de los países fronterizos con uno dado, se podría pensar en reemplazar una representación de grafos en donde se manipulan caminos en una red, por una tabla en la que aparezcan cada uno de los países vecinos. La información de áreas y perímetros podría ser almacenada directamente en lugar de almacenar un conjunto de coordenadas y las reglas necesarias para calcularlos. Sin embargo no todas las operaciones de este tipo pueden resolverse sin procesamiento espacial, por ejemplo, encontrar qué propiedades podrían ser perjudicadas por la construcción de una autopista requiere este tipo de procesamiento.

En este tipo de situaciones es muy importante tener una buena organización de la información espacial y de los atributos descriptivos de manera que la conexión entre ambos tipos de datos sea fácil y rápida.

La eficiencia con que las operaciones se realizan no sólo depende de qué datos se requieren sino de cómo están organizados.

2.9.3.1 Organización de la información espacial

La información geográfica se organiza a menudo como un conjunto de temas, como por ejemplo, tipo de suelo, vegetación, caminos, etc.. Cada uno de estos temas es pensado como una "capa" de un mapa general que representa a una región, estas capas son llamadas *"layers"*. Cada layer puede ser manipulado como una unidad separada o superpuesto con el resto para formar el mapa completo.

Según [Aronoff91] un Data Layer consiste de un conjunto de características geográficas relacionadas. La elección del conjunto de características es arbitrario y depende de qué se está representando y el tipo de análisis que se quiere realizar.

Por otro lado, puede ocurrir que el área que se está tratando de representar sea muy grande y sea necesario dividirla. El área se divide en unidades más pequeñas llamadas *"Tiles"*. El tamaño de un tile depende de las restricciones del software. La grilla definida por la latitud y la longitud es ampliamente utilizada para definir tiles.

2.9.3.2 Operaciones sobre los datos espaciales

Una vez organizada la información, los datos están preparados para realizar diferentes tipos de análisis espacial. En forma general, el análisis que se realiza sobre los datos se clasifica en tres grandes líneas: el análisis local, el zonal y el focal. El análisis local involucra trabajar con posiciones comunes en distintos layers, por ejemplo saber qué conjuntos de características hay en alguna posición determinada. El análisis zonal se define en función de posiciones que tienen el mismo valor de un atributo, por ejemplo determinar dónde crece trigo. El análisis focal trata con relaciones de vecindad y proximidad entre posiciones del mismo layer, por ejemplo encontrar todas las posiciones donde se cultiva trigo que estén a menos de 5 Km. de una donde se cultiva maíz. Para poder realizar estas tareas, los sistemas cuentan con una serie de operaciones que son las que se utilizan más comúnmente cuando se hace análisis espacial. Una lista de estas operaciones es:

1) Reclasificación y agregación
2) Determinación del centroide
3) Conversión de estructuras de datos
4) Operaciones espaciales
   Conectividad y operaciones de vecindad
5) Medidas
   Distancia y dirección
6) Análisis estadístico
   Estadísticas descriptivas
   Regresión, correlación y tabulación cruzada

1) Reclasificación y agregación

Las operaciones que alcanzan esta categoría en general involucran dos situaciones:

- La categoría de la información necesita ser modificada de manera que pueda ser mejor comprendida en una aplicación determinada.

- La resolución de la información debe ser ampliada para captar características espaciales que no están disponibles en los datos.

En el primer caso, la codificación original de la información puede no ser la más eficaz o la correcta para realizar ciertos análisis. Por ejemplo, supongamos que tenemos codificados los distintos tipos de roca que componen el suelo desde el punto de vista

geológico; un ingeniero necesita estudiar la zona para ver si el suelo permite la construcción de cierto tipo de edificio. Proveer los códigos que se utilizan en la geología para que verifique la posible construcción resultaría bastante ineficiente desde su punto de vista. Seguramente en este caso, es mejor reconvertir los códigos en "construible" y "no construible" según determinadas características del suelo. Esta operación, no solamente sugiere el cambio en los códigos, sino también en el mapa, ya que los tipos de suelos tendrán límites entre sí, que seguramente se modificarán cuando se abstrae la información de manera de identificar solamente zonas en donde se puede construir y zonas en las que no; seguramente habrá límites en los datos originales que desaparecerán.

Este tipo de operaciones son más complejas de realizar sobre una estructura vector que raster. En la primera, remover una línea podría significar redefinir polígonos y sus atributos, que están definidos explícitamente. Superponiendo layers la operación es mucho más simple ya que podemos asignar un promedio para cada celda en la superposición.

La agregación es una operación que se utiliza cuando es necesario incrementar el tamaño de la unidad espacial elemental, para, por ejemplo ignorar ciertos detalles. Esta también es una operación mucho más común en una estructura raster que vector, aunque también existen operaciones (del tipo del join) entre entidades representadas como vectores para obtener la información como se la desea, en este caso se requieren reglas de decisión para establecer cómo se van a reorganizar los atributos.

2) Determinación del centroide

El centroide es una manera habitual de encontrar la ubicación promedio de un polígono o una línea. En un polígono bidimensional se puede calcular la ubicación promedio de sub-áreas muy pequeñas dentro del polígono y a partir de éstas determinar la ubicación del centroide.

3) Conversión de estructuras de datos

Estas operaciones se utilizan para transformar objetos almacenados en diferentes estructuras de datos, de manera de poder operar entre ellos. Incluyen tanto la conversión de estructuras internas de datos que están almacenados con la misma filosofía, por ejemplo dos objetos almacenados utilizando el modelo de vector se quieren almacenar bajo una estructura de tipo spaghetti, así como la posibilidad de

transformar objetos que fueron codificados bajo el modelo de vector a raster y viceversa.

4) Operaciones espaciales

El análisis espacial incluye operaciones que consideran las características espaciales de la información que existe en mas de un layer de datos. Se dividen básicamente en dos tipos: aquellas que involucran operaciones concernientes con la conectividad entre distintas posiciones y las relativas a características de vecindad.

i) Proximidad

Una de las operaciones principales en este tipo de sistemas es poder establecer la proximidad de los objetos. Esta operación se utiliza en un gran número de aplicaciones, incluso cuando se está en condiciones de establecer un criterio de distancia, se puede utilizar para ubicar entidades que queden a una distancia determinada de otra.

Obviamente, la operación de proximidad incluye objetos como polígonos, líneas, puntos y sus combinaciones.

ii)Vecindad

Este tipo de operaciones nos permiten determinar, por ejemplo, cuándo dos polígonos son adyacentes, o cuándo un polígono es adyacente a una línea o a un punto.

5) Medidas

Estas operaciones incluyen el cálculo de distancias, longitud de perímetros de áreas, volúmenes, etc. El cálculo de distancias entre dos áreas involucra generalmente definir un criterio para saber qué distancia se está buscando, por ejemplo puede ser distancia mínima, o puede ser la distancia entre los centroides. En los rasters, las distancias generalmente se toman como las distancias entre los centroides de los pixels.

6) Análisis estadísticos

Hay una variedad de técnicas para realizar este tipo de análisis, que incluyen estadísticas descriptivas que implican cálculo de media, varianza, co-varianza de valores de atributos de un layer o un área determinada de un layer. Histogramas y recuento de frecuencias, que es el estudio de la distribución de los atributos de una región. O el cálculo de correlaciones, a partir del cual se puede estudiar la distribución espacial de un atributo en distintos layers de datos.

2.9.4 Salida de Datos

Las funciones de salida de un SIG varían más por su calidad, precisión y facilidad de uso, que por las capacidades disponibles. El software necesario para mostrar la información que se está manipulando, es una de las herramientas más importantes con la que debe contar un SIG. Las salidas pueden ser en forma de tablas, mapas, grafos, etc.

La información en tablas puede ser mostrada, por ejemplo, como una lista de todas las áreas boscosas posibles de talar, dando la superficie del área, las especies, la edad y una estimación de la producción.

Los tipos de mapas más comunes que se utilizan son:

- Los mapas temáticos, que se concentran en las variaciones espaciales de un fenómeno simple (por ejemplo, la lluvia) o la relación que existe entre un fenómeno (por ejemplo, tipo de suelo).
- Los mapas "choropleth" se usan para comunicar magnitudes relativas de datos continuos que ocurren en ciertas áreas. Estos mapas se utilizan para representar información tal como, densidad de población, o el ingreso per cápita anual, etc. En estos mapas generalmente se indican con distintos colores los distintos valores que aparecen. En general, los contornos de las áreas de estudio están definidos y representan, en general áreas políticas o de otro tipo.
- Los mapas de contorno, son utilizados para representar cantidades por líneas o valores iguales. Las isobaras e isotermas son ejemplos de estos mapas o las curvas de nivel que definen regiones de igual altura.

Muchas veces también es necesario mostrar la información, no ya en forma de mapas, sino por ejemplo en forma de gráfico, por ejemplo gráfico de torta, de barras, etc., por lo que el SIG debe también proveer herramientas para este tipo de información.

3. PATRONES DE DISEÑO

3.1 Introducción

Basada en la tecnología de objetos, aparece en los últimos años una herramienta de diseño muy poderosa: los patrones de diseño [Fowler97], [Gamma et al.95].

Los patrones de diseño representan buenas soluciones para problemas que aparecen recurrentemente dentro de uno o más dominios. Christopher Alexander define a los patrones de diseño como: "Cada patrón describe un problema que ocurre una y otra vez en nuestro ámbito, luego exhibe el corazón de la solución al problema de manera que esta solución se pueda usar una y otra vez, sin hacer lo mismo dos veces".

Un patrón tiene cuatro elementos esenciales [Gamma et al.95]:

- **Nombre:** Se utiliza para describir el problema de diseño. Da al diseño un alto nivel de abstracción y brinda el vocabulario para la documentación. Encontrar un buen nombre es fundamental.
- **Problema:** Describe cuándo aplicar el patrón. Explica el problema y su contexto. Permite describir clases o estructuras de objetos que son características de un diseño inflexible. Algunas veces el problema incluirá una lista de condiciones que se deben cumplir para que tenga sentido aplicar el patrón.
- **Solución:** Describe los elementos que hacen al diseño, sus relaciones, responsabilidades y colaboraciones. La solución no describe una implementación o diseño particular, ya que un patrón es un "template" que puede ser aplicado en diferentes situaciones. Un patrón provee una descripción abstracta de un problema de diseño y cómo un conjunto de elementos lo resuelve (en nuestro caso, clases y objetos).
- **Consecuencias:** Son las ventajas y desventajas de aplicar el patrón. Son importantes para evaluar las alternativas de diseño, los costos y los beneficios de aplicar el patrón.

Un patrón de diseño identifica y abstrae aspectos claves de un diseño para crear otro diseño orientado a objetos reusable.

Los patrones de diseño identifican las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades. Cada patrón focaliza un

problema particular de diseño orientado a objetos. Describe cuándo aplicarlo, si puede ser aplicado en función de otras restricciones de diseño y las consecuencias de usarlo.

3.2 Descripción de los patrones de diseño

La notación gráfica es muy importante y útil para describir los patrones de diseño, pero no es suficiente. Esta capta simplemente el producto final del proceso de diseño, como relaciones entre clases y objetos. Con el objetivo que los patrones de diseño sean más fáciles de estudiar, comparar y usar, se define el siguiente modelo para su documentación [Gamma et al.95]:

Nombre y clasificación: El nombre del patrón transmite la esencia del mismo. Un buen nombre es muy importante, ya que este podría llegar a ser parte de nuestro vocabulario de diseño.

Intención: Descripción de qué hace el patrón, cuál es su sentido y qué problema o aspecto particular del diseño ataca.

Otros nombres: Otros nombres con que se conoce el patrón (si los hay).

Motivación: Descripción de un escenario que muestra un problema de diseño y cómo las estructuras de clases y objetos del patrón lo resuelven.

Aplicabilidad: Describe en qué situaciones puede ser aplicado el patrón y cómo reconocer esas situaciones.

Estructura: Representación gráfica de las clases del patrón, usando una notación basada en técnicas para modelar objetos (OMT). También se usan diagramas de interacción para ilustrar secuencias de requerimientos y colaboraciones entre objetos.

Participantes: Las clases y/o objetos participantes en el diseño y sus responsabilidades.

Colaboraciones: Cómo los participantes colaboran (entre sí) para efectuar sus responsabilidades.

Consecuencias: Ventajas y desventajas del uso del patrón.

Implementación: Ayudas o técnicas que se deben tener en cuenta al implementar el patrón.

3.3 Notación gráfica

Los patrones de diseño usan una notación formal para las relaciones e interacciones entre clases y objetos. Para ello se utilizan tres tipos de diagramas diferentes, basados en la notación OMT (Object Modeling Technique) [Rumbaugh et al.91]:

Diagrama de clases

Representa clases, sus estructuras y las relaciones estáticas entre ellas.

La figura 3.1.a muestra la notación OMT para clases abstractas y concretas. Una clase es representada por un recuadro con el nombre de la clase resaltado en la parte superior. Las operaciones de la clase aparecen debajo del nombre de la clase. Las variables de instancia se encuentran debajo de las operaciones. La información del "tipo" es opcional, se utiliza la convención de C++, colocando el nombre del tipo antes del nombre de la operación (significando el tipo retornado), variable de instancia o parámetro actual. Para indicar las clases y operaciones abstractas, se utiliza un tipo de letra inclinado.

En algunos patrones de diseño, es útil ver dónde las clases clientes referencian a las clases participantes. Cuando un patrón incluye una clase Cliente como una de sus participantes, es decir cuando el cliente tiene alguna responsabilidad en el patrón, entonces la clase Cliente se representa como una clase común. En cambio, cuando un patrón de diseño no incluye una clase Cliente como una de sus participantes, es decir el Cliente no tienen responsabilidades en el patrón, la clase cliente es mostrada en gris.

La figura 3.1.b muestra varias relaciones entre clases. La notación OMT para *herencia* de clase es representada como un triángulo conectando la subclase a su padre. La relación "*parte_de*" o de agregación, se representa con una flecha y un rombo en la base. La flecha apunta a la clase que es agregada o que es parte_de. Esta referencia puede tener un nombre. Una flecha sin rombo en la base representa *conocimiento*.

Cuando una clase instancia a otra, se representa con una flecha punteada. Se llama relación "*creación*". La flecha apunta a la clase que es instanciada

La notación OMT también define círculos coloreados que significa "*mas de uno*". Cuando un círculo aparece en la cabecera de una referencia, significa que múltiples objetos son referenciados o agregados.

Finalmente, la notación OMT es aumentada con anotaciones de pseudocódigo para representar la implementación de operaciones. La figura 3.1.c muestra la notación en pseudocódigo para la operación Dibujar de la clase Dibujo.

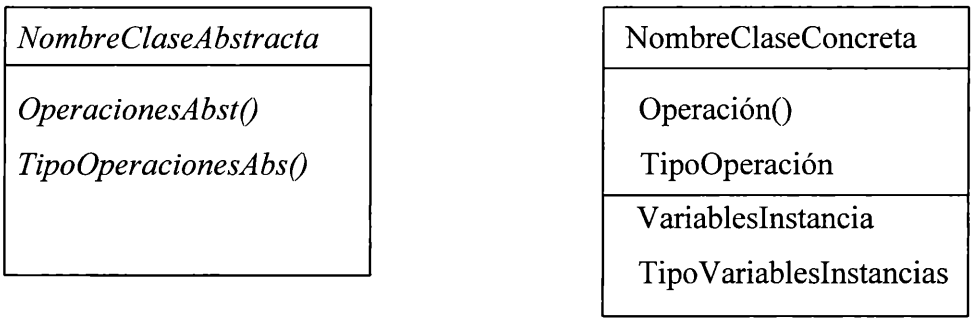


Figura 3.1.a Clases Abstractas y Concretas

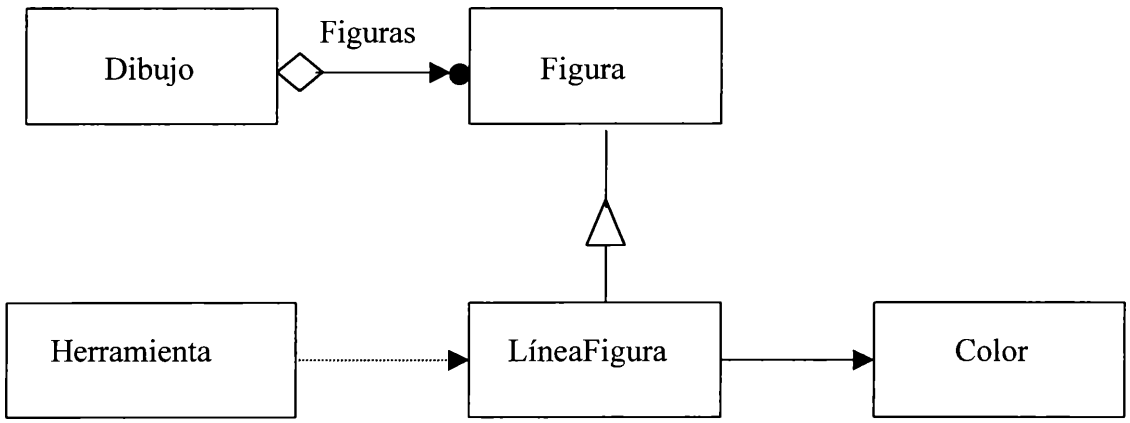


Figura 3.1.b Relaciones entre clases

Figura 3.1.c Pseudocódigo

Figura 3 Notación de Diagrama de Clase

Diagrama de objetos

Representa la estructura de un objeto particular en tiempo de ejecución.

Muestra exclusivamente instancias. Provee una "foto" de los objetos en un patrón de diseño. Los objetos son llamados "aNombre", donde Nombre es la clase del objeto. El símbolo para un objeto es un recuadro redondeado con una línea separando el nombre del objeto de las referencias de objetos. Las flechas indican el objeto refrenciado. La figura 3.2 muestra un ejemplo.

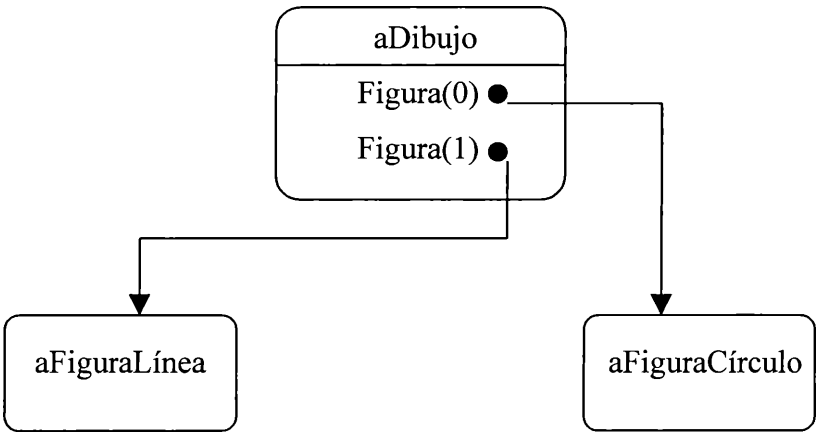


Figura 3.2 Notación Diagrama de Objeto

Diagrama de interacción

Muestra como se ejecutan los requerimientos entre objetos.

La figura 3.3 es un diagrama de interacción que muestra como una figura es agregada a un dibujo.

El tiempo va desde arriba hacia abajo. Las líneas verticales indican el tiempo de vida de los objetos. Si el objeto se instancia después del comienzo del tiempo en el diagrama, la línea vertical aparece punteada.

Un rectángulo vertical indica que el objeto está activo, es decir manejando un requerimiento. Los requerimientos a otros objetos se indican con una flecha horizontal al otro objeto, con el nombre del requerimiento. Un requerimiento para crear otro objeto se muestra con una línea punteada.

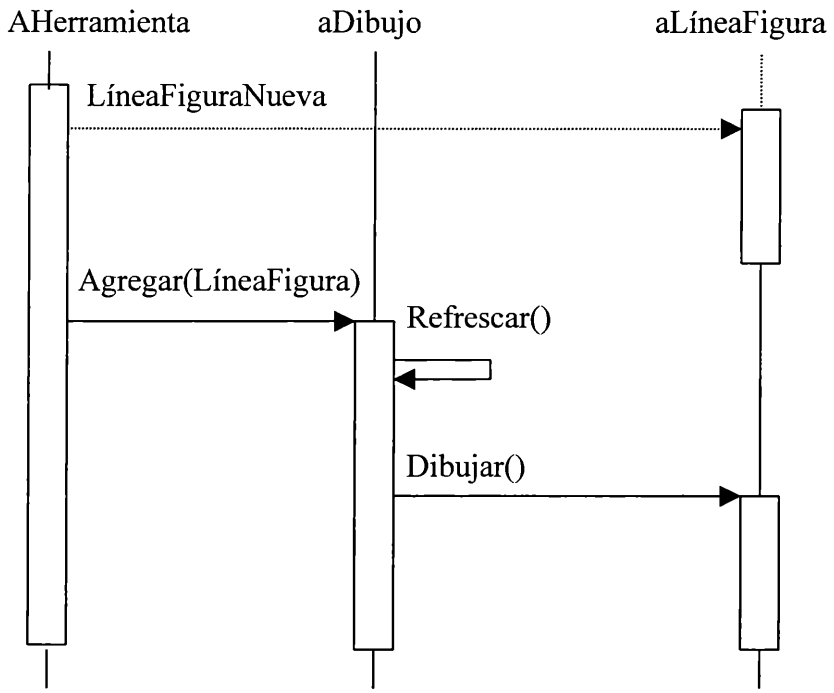


Figura 3.3 Notación Diagrama de Interacción

A continuación describiremos algunos patrones de diseño que se encuentran en [Gamma et al.95] y [Johnson et al.97], que serán utilizados en capítulos posteriores.

3.4 Patrón de Diseño: *DECORATOR* [Gamma et al.95]

INTENCIÓN

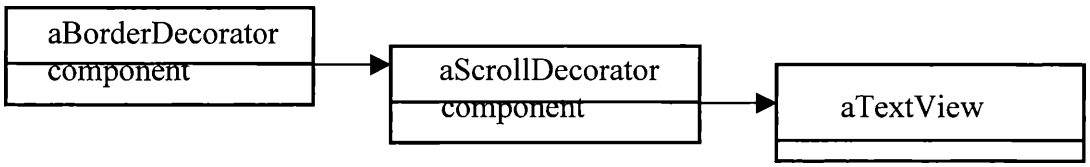
Adicionar responsabilidades a un objeto en forma dinámica. Provee una alternativa más flexible que la subclasificación para extender funcionalidad.

OTRO NOMBRE: Wrapper

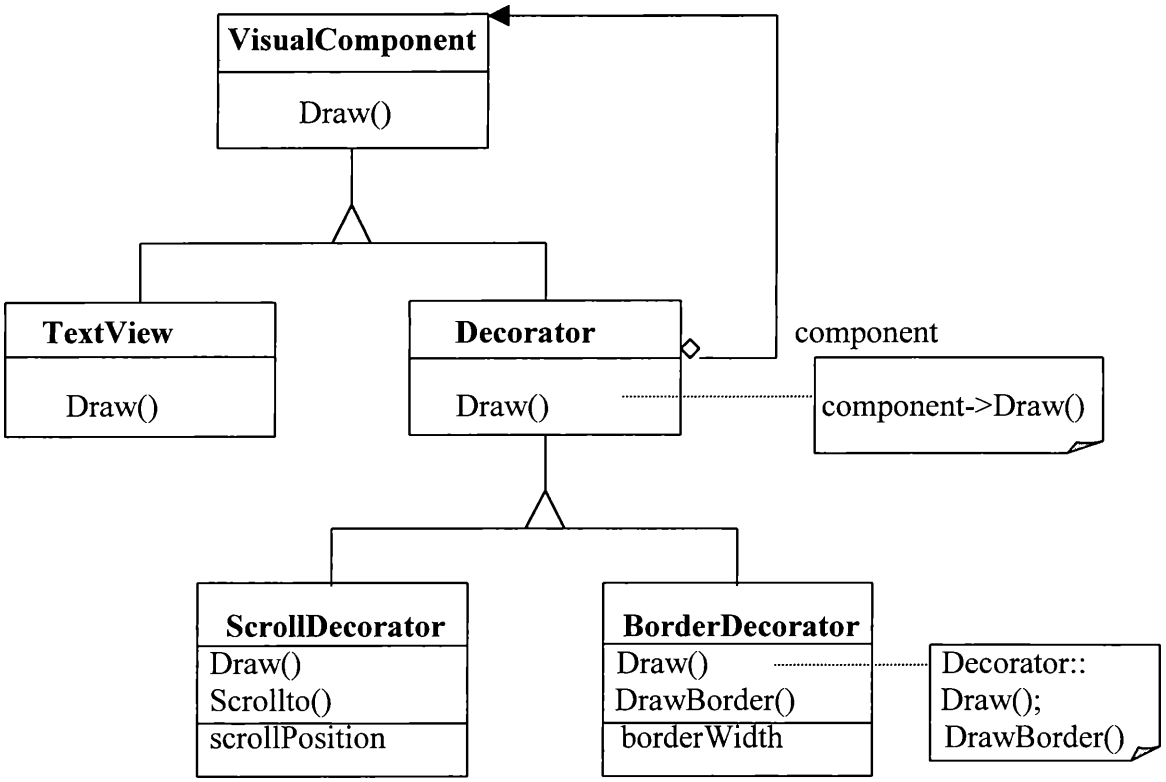
MOTIVACIÓN: A veces queremos agregar responsabilidades a objetos individuales, no a todas las instancias de una clase. Supongamos que tenemos una clase ventana que presenta texto en una ventana y se le quiere agregar propiedades como bordes o comportamiento como scroll. Una forma de hacer esto es crear una subclase de ventana que contenga una operación que realice el borde. Esto implica que todas las instancias de la subclase tengan un borde, esto es inflexible, ya que la elección del borde se hace estáticamente. No se puede controlar cómo y cuándo decorar la componente con el borde. Una solucuón más flexible es utilizar el patrón Decorator, éste replica la

interfaz del objeto al que decora y además define comportamiento adicional, de esta forma delega requerimientos que corresponden a la componente original y ejecuta las acciones adicionales. Todas las caracterísiticas adicionales pueden definirse y agregarse a los objetos en forma individual.

El siguiente diagrama de instancia muestra como componer un objeto TextView con objetos ScrollDecorator y BorderDecorator para producir un texto con borde y que se le pueda hacer scroll.



Las clases ScrollDecorator y BorderDecorator son subclases de Decorator, una clase abstracta para componentes visuales que decora otras componentes visuales.



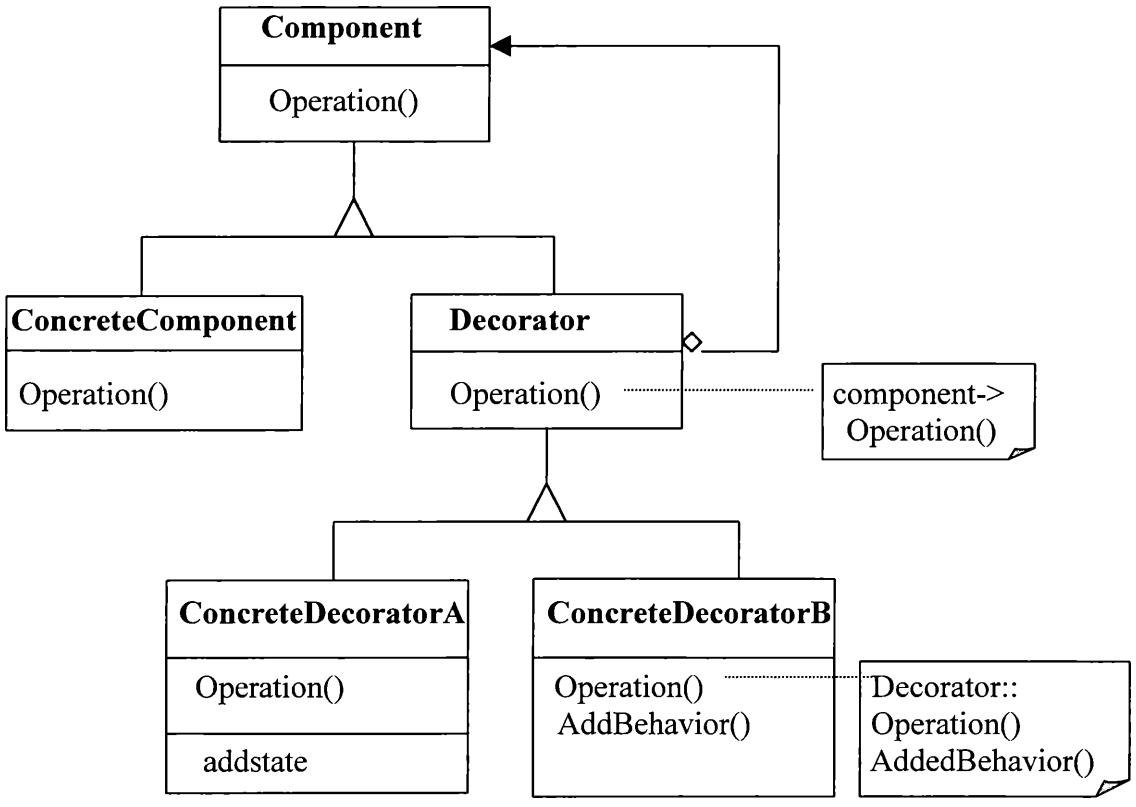
VisualComponent es una clase abstracta para objetos visuales. Esta define sus diseños y los eventos que manejan la interfaz. La clase Decorator simplemente solicita los dibujos requeridos a su componente y las subclases de Decorator pueden ejecutar la operación requerida.

Las subclases de Decorator pueden agregar operaciones para funcionalidad específica. Por ejemplo la operación Scrollto de ScrollDecorator permite a otros objetos hacer scroll en la interfaz, si ellos saben que hay un objeto ScrollDecorator en la interfaz.

APLICABILIDAD:

- Adicionar responsabilidades a objetos individuales en forma dinámica y transparentemente, sin afectar otros objetos.
- Para responsabilidades que deben ser redefinidas.
- Cuando extender funcionalidad por subclasificación no es práctico. Algunas veces es posible definir un gran número de extensiones independientes y podría producirse una explosión de clases para soportarlas.

ESTRUCTURA



PARTICIPANTES

- **Component (VisualComponent)**

Define la interfaz de los objetos a los cuales se les ha agregado responsabilidades.

- **ConcreteComponent (TextView)**

  Define un objeto al cual se le adicionan responsabilidades.

- **Decorator**

  Mantiene una referencia a un objeto *Component* y define la interfaz que conforma la interfaz del objeto decorado.

- **ConcreteDecorator (ScrollDecorator y BorderDecorator)**

  Adiciona responsabilidades a los objetos.

COLABORACIONES

El Decorator solicita los requerimientos a su objeto Component. Opcionalmente puede realizar operaciones antes y después de enviar el requerimiento.

CONSECUENCIAS

- *Más flexibilidad que la herencia estática.* El patrón Decorator provee una forma más flexible de agregar responsabilidades a objetos que la que se puede tener con herencia estática. Con los decoradores se pueden agregar y remover las responsabilidades en tiempo de ejecución simplemente ligándolos y desligándolos de los objetos. A diferencia de esto, la herencia requiere que se cree una nueva subclase para cada responsabilidad que se quiere agregar. Esto origina muchas clases e incrementa la complejidad de un sistema.

- *Evita clases sobrecargadas en la línea de herencia.* El uso de Decorator también permite agregar funcionalidad incrementalmente en lugar de definir clases que engloben todas las carácterísticas, de esta forma, si hay características que no se usan no hay costos adicionales.

- *El decorador y sus componentes no son idénticos.* El decorador actúa como una envoltura transparente para la componente. Pero desde el punto de vista de la identidad, una componente decorada no es idéntica a la componente en sí misma, por lo tanto no se debe asumir esto.

- *Abundancia de objetos pequeños.* Cuando se usan decoradores, a menudo se tiene sistemas con una proliferación de objetos pequeños, todos muy semejantes. Aunque estos sistemas son fáciles de "customizar" pueden ser dificultosos para realizar controles y seguimientos.

3.5 Patrón de Diseño: *COMPOSITE* [Gamma et al.94]

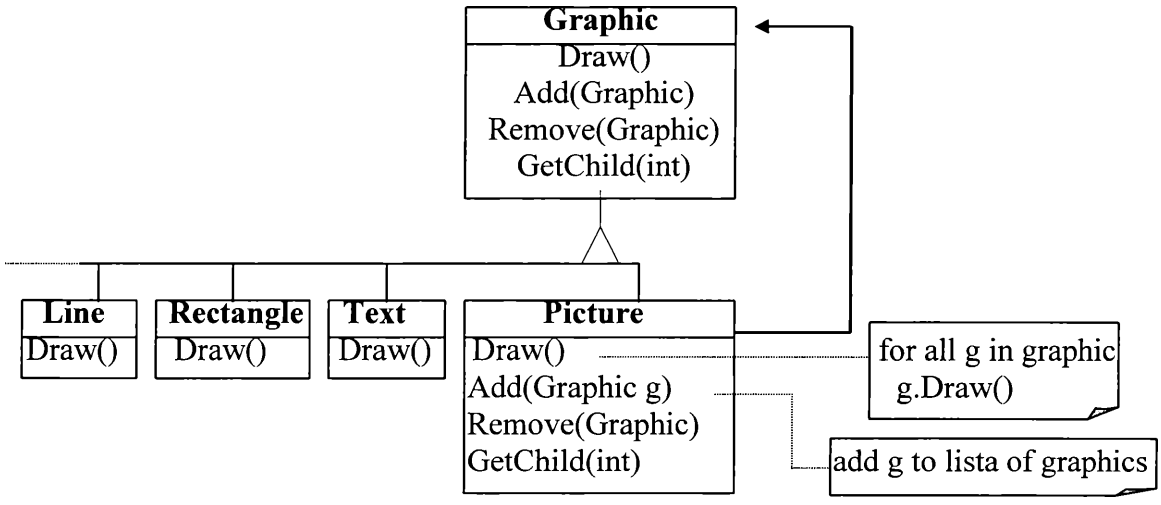
INTENCION

Componer objetos dentro de estructuras de árbol para representar jerarquías parte_de. Permite tratar con objetos individuales y composiciones de objetos uniformemente.

MOTIVACIÓN

Las aplicaciones gráficas, como editores de dibujo, permiten a los usuarios construir diagramas complejos a partir de componentes simples. El usuario puede agrupar componentes para formar componentes mas grandes, las cuales a su vez pueden ser agrupadas. Una implementación simple podría definir clases para primitivas gráficas tales como Text y Lines más otras clases que actúen como contenedores de estas primitivas.

Pero existe un problema con esta aproximación: El código que usan estas clases deben tratar con primitivas y objetos contenedores diferentemente, y la mayoría de las veces se los trata idénticamente. Tener que distinguir estos dos objetos hacen a la aplicación muy compleja. El patrón Composite describe cómo usar la composición recursiva para que los clientes no tengan que hacer esta distinción.



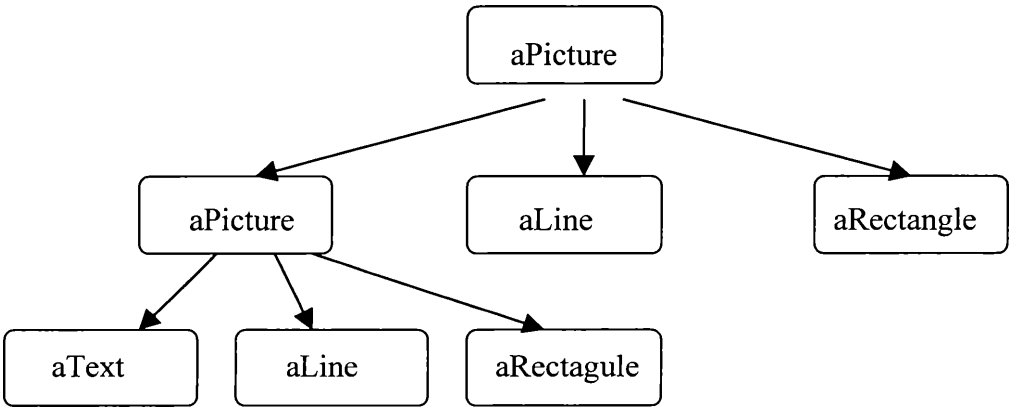
La clave para el patrón Composite es una clase abstracta que representa ambas, las primitivas y sus contenedores. Para los sistema gráficos esta clase es Graphic. Graphic declara operaciones como Draw que es específica para objetos gráficos y

también declara operaciones que todos los objetos compuestos comparten, tales como operaciones para acceder y manejar a sus hijos.

Las subclases Line, Rectangle y Text, definen objetos gráficos primitivos. Estas clases implementan Draw() para dibujar líneas, rectángulos y texto respectivamente. Ya que las gráficas primitivas no tienen gráficos hijos, ninguna de estas subclases implementan operaciones relacionadas con sus hijos.

La clase Picture define un agregado de objetos Graphic. Picture implementa Draw() para llamarla sobre sus hijos y ésta por consiguiente implementa la operación relacionada con sus hijos. Como la interfaz de Picture conforma a la interfaz Graphic, los objetos de Picture pueden componer otros Picture recursivamente.

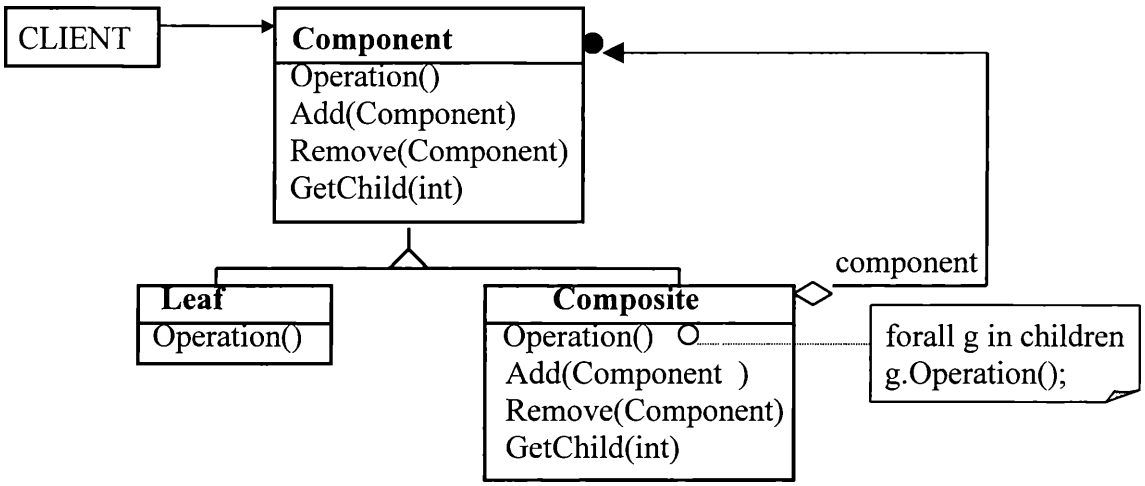
El siguiente diagrama muestra una estructura de objetos gráficos compuestos recursivamente:



APLICABILIDAD

- Representar jerarquías de objetos parte_de.
- Cuando se quiera clientes capaces de ignorar las diferencias entre composiciones de objetos y objetos individuales. Los clientes tratarán todos los objetos, en la estructura compuesta uniformemente.

ESTRUCTURA



PARTICIPANTES

- **Component (Graphic)**

  - Declara la interfaz para los objetos en la composición.

  - Implementa comportamiento por defecto para la interfaz común a todas las clases.

  - Declara una interfaz para acceder y manejar sus componentes hijos.

  - (opcional) Define una interfaz para acceder a un padre del componente en la estructura recursiva y la implementa si es apropiada.

- ***Leaf*** **(Rectangle, Line, Text, etc.)**

  - Representa a los objetos hojas en la composición. Una hoja no tiene hijos.

  - Define comportamiento para objetos primitivos en la composición.

- ***Composite*** **(Picture)**

  - Define el comportamiento para las componentes que tienen hijos.

  - Almacena los componentes hijos.

  - Implementa las operaciones relacionadas con los hijos en la interfaz de Component.

- ***Client***

  - Maneja objetos en la composición a través de la interfaz de Component.

COLABORACIONES

- Los clientes usan la interfaz de la clase Component para interactuar con objetos en la estructura compuesta. Si el receptor es un Leaf, entonces el requerimiento se maneja directamente. Si el receptor es un Composite, generalmente envía los pedidos a sus componentes hijos, posiblemente realizando operaciones adicionales antes y/o después del envío.

CONSECUENCIAS

- *Define jerarquías de clases que consisten de objetos primitivos y objetos compuestos*. Los objetos primitivos pueden ser compuestos dentro de objetos más complejos, los cuales pueden ser compuestos, y así siguiendo recursivamente. Siempre que el código del cliente espera un objeto primitivo, también puede esperar un objeto compuesto.
- *Simplifica al cliente*. Los clientes pueden tratar con estructuras compuestas y objetos individuales uniformemente. Normalmente ellos no conocen y no debieran preocuparse si tratan con una hoja o una componente compuesta. Esto simplifica el código del cliente.
- *Facilita el agregado de nuevos tipos de componentes*. Las nuevas clases de Composite o subclases de Leaf trabajan automáticamente con estructuras existentes y código de clientes. Los clientes no tienen que ser cambiados por nuevas clases de Component.
- *Hace que el diseño sea más general*. La desventaja de hacer fácil el agregado de componentes nuevos es que lo hace más difícil para restringir a los componentes de un Composite. A veces se quiere que un Composite tenga sólo ciertos componentes. Con Composite tendrá que usarse un chequeo en tiempo de ejecución, en vez de dejar que el sistema fuerce las restricciones.

3.6 Patrón de Diseño: *Type Object* [Johnson et al.97]

INTENCIÓN

Desacoplar instancias de sus clases para que éstas puedan ser implementadas como instancias de otra clase. Permite que las nuevas clases sean creadas en forma

dinámica en tiempo de ejecución, permite también que un sistema provea sus propias reglas de chequeo de tipo conduciendo a sistemas más pequeños y simples.

OTROS NOMBRES

Power Type, Item Descriptor, MetaObject, Data Normalization.

MOTIVACIÓN

Por lo general, no se conoce la cantidad de instancias que se tendrá de una clase, ni la cantidad de subclases de la misma. Aunque un sistema orientado a objetos puede crear nuevas instancias cada vez que sea necesario, generalmente no puede crear nuevas clases sin una recompilación. Un diseño en el cual una clase tiene un número desconocido de subclases puede ser convertido en otro, en el que la clase tenga un número desconocido de instancias.

Consideremos un sistema para el seguimiento de videos en el inventario de un videoclub. El sistema requerirá obviamente una clase llamada *Video*. Cada instancia de *Video* representará uno de los videos en el inventario del negocio. Sin embargo, ya que muchos de los *videos* son similares, las instancias de *Video* tendrán un conjunto de información redundante. Por ejemplo, todas las copias de "La guerra de las galaxias" tendrán el mismo título, precio de alquiler, clasificación, etc. Esta información es diferente para "Terminator", por ejemplo, pero todas las copias de "Terminator" tendrán los mismos datos. Repetir esta información para todas las copias de cada una de estas películas sería redundante.

Una forma de resolver este problema es crear una subclase de *Video* para cada película. Así dos de las subclases podrían ser *LaguerradelasgalaxiasVideo* y *TerminatorVideo,* de esta manera la información común a todas las copias de "La guerra de las galaxias" podría ser almacenada una sola vez. Ahora, *Video* podría ser una clase abstracta; el sistema no crearía instancias de ella. Cuando el negocio compre una nueva copia de "Terminator" y comience a alquilarla, el sistema debería crear una instancia para esa película, es decir, una instancia de *TerminatorVideo*.

Esta solución funciona, pero no es muy buena. Un problema es que si el negocio se abastece de lotes de diferentes películas, la clase *Video* podría requerir de un gran número de subclases. Otro problema que podría suceder cuando, con el sistema en funcionamiento, el negocio comience a almacenar una nueva película (por ejemplo "Día de la independencia"); no hay una clase *DíadelaindependenciaVideo* en el sistema.

Si el desarrollador no previó esta situación, tendría que modificar el código para agregar una nueva clase *DíadelaindependenciaVideo*, recompilar el sistema y ejecutarlo nuevamente. Si el desarrollador había previsto esta situación, él pudo prever una subclase especial de *Video* -como *DesconocidoVideo*- y el sistema podría crear una instancia de ella para todos los videos de la nueva película. El problema con *DesconocidoVideo* es que tiene la misma carencia de flexibilidad que tiene la clase *Video*. De la misma manera que *Video* requería de subclases, también lo requiere *DesconocidoVideo*, por eso, crear la subclase *DesconocidoVideo* no es una buena solución.

En vez de eso, ya que el número de tipos de videos es desconocido, cada tipo de video necesita ser instancia de una clase. Sin embargo, cada video necesita ser una instancia de un tipo de video. Los lenguajes de objetos basado en clases dan soporte para instancias de clases, pero no dan soporte para instancias de instancias de clases. Por esto, para implementar esta solución en un lenguaje típico basado en clases, se necesita implementar dos clases: una para representar un tipo de video (*Película*) y una para representar un video (*Video*). Cada instancia de *Video* tendría un puntero a su correspondiente instancia de *Película*.

El diagrama de clase de la figura 3.1 muestra como cada instancia de *Video* tiene una correspondiente instancia de *Película*. Se ve cómo las propiedades definidas para el tipo de video están separadas de aquellas que son diferentes para cada video particular.

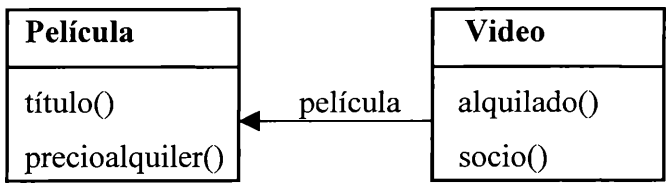


Figura 3.1 Clases del sistema de videoclub

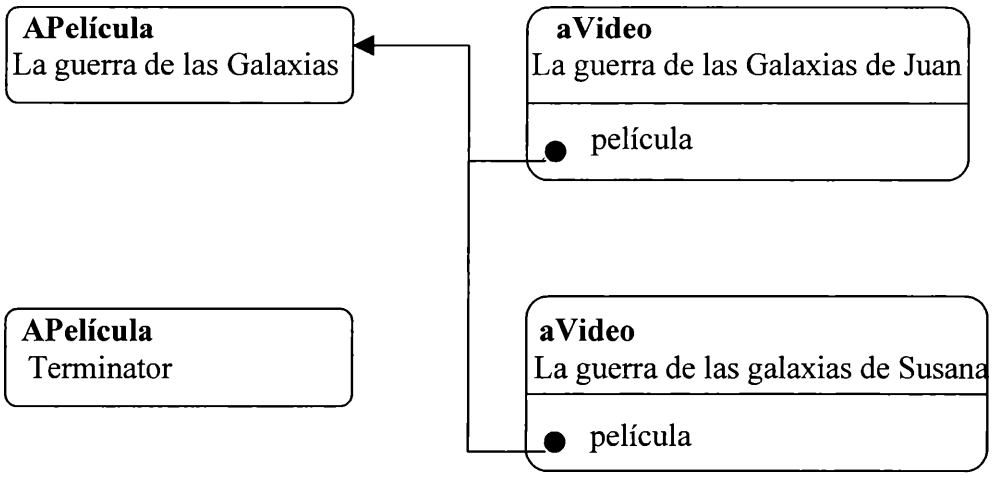


Figura 3.2 Instancias del sistema de videoclub

En este caso, los datos correspondientes al título de la película y cuánto cuesta el alquiler van separados de los datos de si el video está alquilado y quien lo alquiló.

El diagrama de instancias de la figura 3.2 muestra que existe una instancia de *Película* para representar cada tipo de video y una instancia de *Video* para representar cada video que hay en stock. "La guerra de las galaxias" y "Terminator" son películas; los videos son, la copia de "La guerra de las galaxias" que Juan alquiló y otra que Susana alquiló. Esto también muestra cómo cada *Video* conoce de qué tipo es, debido a su relación con una instancia particular de la clase *Pelicula.*

Si una nueva película como "Día de la independencia" fuera alquilada por José, el sistema debería crear una nueva *Película* y un nuevo *Video* que apunte a esa *Película.* La película es "Día de la independencia" y el video es la copia de "Día de la independencia" que José alquiló.

Video, Película y la relación *es_instancia _de* entre ellas (un Video es una instancia de una Película) es un ejemplo del patrón TypeObject. Este patrón se utiliza para crear instancias de un conjunto de clases cuando el número de clases es desconocido. Permite que una aplicación cree nuevas "clases" en tiempo de ejecución, ya que las clases son realmente instancias de una clase. La aplicación debe entonces mantener la relación entre las instancias reales y sus instancias como clases.

La clave de este patrón es: dos clases concretas, una cuyas instancias representan las instancias de la aplicación y la otra cuyas instancias representan tipos de

las instancias de la aplicación. Cada instancia de la aplicación tiene un puntero a su correspondiente tipo.

El patrón TypeObject incorpora un framework que tiene tres características:

- Dos clases, una clase tipo y una clase instancia.
- La clase instancia tiene una variable de instancia cuyo tipo es la clase tipo.
- La clase instancia delega su comportamiento de tipo a la clase tipo a través de la variable de instancia.

El framework también puede incluir las siguientes variaciones en el patrón:

- El sistema puede mantener una lista de las instancias de su clase tipo.
- Las instancias de la clase tipo pueden mantener una lista de sus instancias.

APLICABILIDAD

- Cuando las instancias de una clase necesitan ser agrupadas juntas de acuerdo a sus atributos y/o comportamientos comunes.
- Cuando la clase necesita una subclase para cada grupo, para implementar los atributos y comportamiento comunes del grupo.
- Cuando la clase requiere un gran número de subclases y/o la variedad total de subclases que puedan ser requeridas es desconocida.
- Cuando se quiere poder crear nuevos agrupamientos en tiempo de ejecución que no fueron previstos en el diseño.
- Cuando se quiere poder cambiar una subclase de un objeto luego que ha sido instanciada, sin tener que cambiarlo a una nueva clase.
- Cuando se quiere poder anidar agrupamientos recursivamente, de manera que un ítem sea el mismo ítem en otro grupo.

ESTRUCTURA

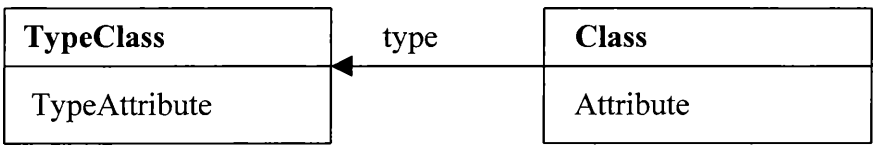


Figura 3.3 Estructura del patrón Type Object

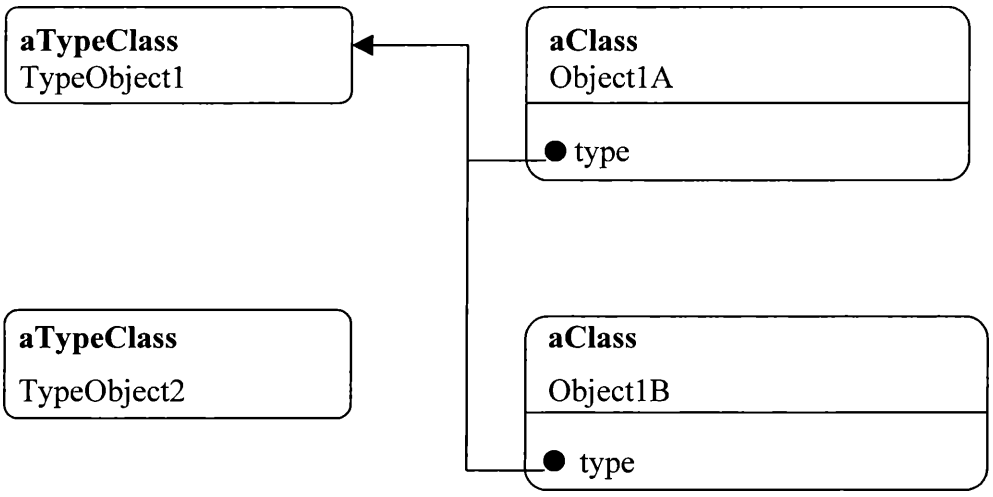


Figura 3.4 Instancias del patrón Type Object

Como muestra la figura 3.3, el patrón Type Object tiene dos clases concretas, una que representa objetos y otra que representa sus tipos. Cada objeto tiene un puntero a su tipo correspondiente.

La figura 3.4 muestra cómo el sistema usa un Typeobject para representar cada tipo en el sistema y un Object para representar cada una de las instancias de estos TypeObjects. Cada Object tiene un puntero a su TypeObject

PARTICIPANTES

- **TypeClass (Película):**

  -Es la clase TypeObject.

  -Tiene una instancia separada para cada tipo de objeto.

- **TypeObject (La guerra de las galaxias, Terminator, Día de la independencia):**

  -Es una instancia de la clase TypeClass.

  -Representa un tipo de Object. Establece todas las propiedades de un objeto que son las mismas para todos los objetos del mismo tipo.

- **Class (Video):**

  -Es la clase del Object

  -Representa instancias de TypeClass.

- **Object(La guerra de las galaxias de Juan, La guerra de las galaxias de Susana):**

-Es una instancia de Class

-Representa un item único que tiene un contexto único.

-Tiene un TypeObject asociado que describe su tipo. Delega propiedades definidas por su tipo a su TypeObject.

TypeClass y Class son clases. TypeObject y Object son instancias de sus clases respectivas. Como con cualquier instancia, TypeObject y Object conocen cuál es su clase. Un Object tiene un puntero a su TypeObject, así éste conoce de que TypeObject es. El Object usa su TypeObject para definir su comportamiento tipo. Cuando el objeto recibe requerimientos que son de un tipo específico pero no de una instancia específica, delega esos requerimientos a su TypeObject. Un TypeObject puede tener punteros a todos sus objetos.

Por lo tanto, Película es un TypeClass y Video es una Class. Las instancias de Película, como "La guerra de las galaxias", "Terminator" y "Día de la independencia" son TypeObject. Las instancias de Video como La guerra de las galaxias de Juan, La guerra de las galaxias de Susana son Object. Ya que un Object tiene un puntero a su TypeObject, el video de Juan y el video de Susana tienen punteros a su correspondiente Película, lo que en este caso es "La guerra de las galaxias" para ambos Videos. Esto es ya que los Videos saben que ellos contienen "La guerra de las galaxias" y no cualquier otra película.

COLABORACIONES

- Un Object tiene dos categorías de requerimientos: los definidos por su instancia y los definidos por su tipo. Maneja los pedidos de instancia el mismo y delega el requerimiento tipo a su TypeObject.
- Algunos clientes pueden querer interactuar con Typeobjects directamente. Por ejemplo: más que iterar a través de todos los Videos que el negocio tiene en stock, un socio podría querer ver todas las películas que ofrece el negocio.
- Si es necesario, TypeObject podría tener un conjunto de punteros a sus Objects. De esta forma, el sistema puede recuperar fácilmente un Object que se adapte a la descripción de TypeObject. Por ejemplo, una vez que un socio encuentra una Película que le gusta, podría querer saber cuáles son los videos del negocio que se corresponden con la descripción.

CONSECUENCIAS

- *Creación de clases en tiempo de ejecución.* El patrón permite que se creen nuevas "clases" en tiempo de ejecución. Estas, no son clases actualmente, son instancias llamadas TypeObject, creadas por el TypeClass, como cualquier instancia es creada por su clase.
- *Evita la explosión de subclases.* El sistema no necesita numerosas subclases para representar diferentes tipos de objetos. En lugar de numerosas subclases, el sistema puede usar una TypeClass y numerosos TypeObject.
- *Oculta la separación de instancia y tipo.* Un cliente de objetos no necesita conocer la separación entre Object y TypeObject. El cliente realiza requerimientos del Object, y el Object a su vez, decide qué requerimiento envía al TypeObject. Los clientes que conocen los TypeObject pueden colaborar con ellos directamente sin ir a través de los Objects.
- *Cambio de tipo dinámico.* El patrón permite que el Object cambie dinámicamente su TypeObject, lo cual tiene el efecto de cambiar su clase. Esto es más simple que cambiar un objeto a una nueva clase.
- *Subclases independientes.* Class y TypeClass pueden ser subclasificadas independientemente.
- *Objetos de tipo múltiple.* El patrón permite que un Objeto tenga múltiples TypeObject, donde cada uno define alguna parte del tipo de Object. El Object debe entonces decidir cuál comportamiento tipo delegar a cuál TypeObject.

4 - UN MODELO ORIENTADO A OBJETOS PARA SIG

4.1 Introducción

En este capítulo se presenta un modelo para el diseño de aplicaciones geográficas (SIG) basado en el Paradigma de Orientación a Objetos. En particular, el modelo se basa en el uso de patrones de diseño que permiten resolver problemas recurrentes en esta área.

Esta aproximación permite al diseñador desacoplar la definición conceptual de aplicaciones de objetos de su representación espacial, y es también útil para extender aplicaciones convencionales construidas con tecnología orientada a objetos para soportar características espaciales.

La tecnología de SIG involucra tratar con aspectos complejos tales como, adquisición de datos, la corrección de los mismos, la representación de las relaciones espaciales y de la topología y el diseño de la interfaz. Uno de los problemas más graves que existen en este dominio, es la carencia de un método de diseño que permita reflejar todas las características mencionadas y permitir obtener un producto con las propiedades deseables desde el punto de vista de la ingeniería de software, como por ejemplo, reusabilidad, modularidad, facilidades para la evolución y la modificación, etc..

En efecto, hoy en día la construcción de este tipo de aplicaciones se vuelve una tarea completamente artesanal, en donde aquellos diseñadores experimentados utilizan su conocimiento anterior sin registrarlo, mientras que aquellos que no poseen experiencia realizan la tarea basándose en pruebas y errores. Por supuesto, es sumamente difícil encontrar documentación sobre estos sistemas, pues la preocupación pasa más por la precisión y la corrección de los datos que se están almacenando (es decir en cómo la información es representada) y en resolver las dificultades que presentan las relaciones espaciales, que por obtener un buen diseño (es decir en qué información o comportamiento es necesario).

Esta metodología lleva a lo que ya se ha conocido y vivido en otras áreas, los costos de mantenimiento, reusabilidad y evolución de los sistemas resultan altísimos. De hecho, ocurre más de una vez que grupos que realizan desarrollos basados en el mismo tipo de aplicación (por ejemplo catastral) deben comenzar cada trabajo como si fuera completamente nuevo, cuando en realidad la mayor parte de los módulos de una

aplicación catastral pueden ser utilizados para otra.

La tecnología orientada a objetos ha demostrado ser una buena herramienta a la hora de resolver problemas de gran envergadura, en los que la complejidad de la información que se manipula es alta y es imprescindible contar con un alto grado de integración de esa información. Como resultado de su uso, se obtienen no solamente soluciones en donde las características anteriores se mantienen (reusabilidad, evolución, modularidad), sino también sistemas interoperables, ya que los objetos pueden encapsular conocimiento que puede agruparse según las necesidades.

4.2 El uso del Modelo Orientado a Objetos en Aplicaciones Geográficas

La preocupación de los diseñadores de aplicaciones geográficas por obtener información correcta desde el punto de vista de la representación (precisión, escala, etc.) genera un error bastante común en el diseño que parte de considerar las características de la representación espacial de las entidades, en lugar de pensarlas en función de su definición abstracta. El énfasis entonces, está puesto en **cómo** la información se representa y no en **qué** información se debe representar. Como una consecuencia de esto, las entidades que tienen distintas representaciones (por ejemplo distintas escalas) se implementan usualmente como objetos diferentes cuando en realidad no lo son.

Existen muchos trabajos que se han abocado al estudio del uso del modelo de objetos como una solución a estos problemas [Medeiros et al.94], [Tryfona et al.95], [Kosters et al.95]. Sin embargo, la mayoría de estos trabajos se basan en el modelo para representar entidades complejas pero no han estudiado cuestiones tales como la separación de las representaciones físicas de los objetos (puntos, líneas, polígonos) de su comportamiento conceptual, lo que fuerza al diseñador a asignar prematuramente características espaciales a las entidades y hace dificultoso trabajar con diferentes representaciones del mismo objeto.

Por otra parte, también se ha vuelto una práctica común extender sistemas convencionales para que manipulen características espaciales. Ejemplos de estas extensiones pueden encontrarse en [Postmesil97].

4.3 Un Modelo para Aplicaciones Geográficas

A continuación se describe un modelo orientado a objetos para diseñar aplicaciones geográficas. Este modelo está presentado en varios artículos [Gordillo et

al.97], [Balaguer et al.97]. El modelo está basado en el uso de patrones para resolver desde el diseño de las características espaciales de las entidades, hasta situaciones particulares como la representación del sistema de referencia utilizado para las posiciones de las mismas. Este modelo es utilizado como base para el diseño, descripto en el capítulo 6.

4.3.1 El Modelo Conceptual

Existen en las aplicaciones geográficas dos tipos de datos bien diferenciados. Los datos conceptuales describen a las entidades en términos de atributos descriptivos, y son los que se utilizan habitualmente en las aplicaciones convencionales. Por ejemplo, si estamos representando un país, su nombre, su lengua, la moneda que se utiliza, son atributos que describen al país. Por otro lado está la información geográfica, como la ubicación de ese país, la definición de sus fronteras, etc..

Estos dos tipos de datos, definen en la mayoría de los SIGs, dos bases de datos bien diferenciadas: una, generalmente relacional, conteniendo los atributos descriptivos, y otra, espacial, que almacena las características geográficas. En el nivel funcional pasa algo parecido, la operaciones que se deben realizar se pueden considerar de dos tipos: las que operan con los atributos descriptivos y obtienen información basándose en ellos, y aquellas operaciones espaciales que no solamente van a manipular información espacial, sino que además pueden estar determinadas por diferentes fenómenos.

La discusión anterior sugiere que las características (atributos y operaciones) conceptuales y espaciales pueden ser tratadas en niveles diferentes de abstracción, sin embargo, los dos aspectos deben estar cubiertos correctamente, de manera que interactúen en forma consistente.

Como la complejidad del diseño y del desarrollo reside básicamente en la definición de las características espaciales, la propuesta es realizar el proceso de diseño en dos pasos, separando el diseño de las características conceptuales, de las geográficas, de manera de desacoplar los problemas haciendo posible que la tarea resulte más fácil y clara.

Durante el primer paso, llamado Modelo Conceptual, se describe el modelo en términos del mundo real. La intención es comprender el problema proveyendo una descripción abstracta en la cual se ignoran tanto las características geográficas como los detalles de implementación. Se utilizan los conceptos de la orientación a objetos y, sólo

aquellas responsabilidades que son independientes del espacio, son tenidas en cuenta en esta etapa; de esta forma, el resultado obtenido no es diferente de una aplicación convencional diseñada bajo este paradigma.

En el segundo, se especifican las características geográficas de los objetos definidos en la etapa anterior y, posiblemente, aparezcan nuevos objetos que sólo contienen características espaciales. El método para adicionar estas características está descripto en el punto siguiente.

4.3.2 El modelo Geográfico

Una vez que el modelo conceptual está definido, puede enriquecerse agregándole las características geográficas. La propuesta es identificar aquellas clases del modelo conceptual que deben ser enriquecidas y definir una nueva clase que adiciona las nuevas características. Para hacer esto proponemos el uso del patrón de diseño "Decorator" detallado en el capítulo 3, y luego se explica cómo puede ser usado para diseñar el nivel geográfico en el modelo propuesto.

En la figura 4.1 se muestra una clase *Río,* que es parte de una aplicación para representar diferentes espectos de un país.

En la primera etapa, la clase fue pensada para que soporte características conceptuales, dejando de lado la información geográfica.

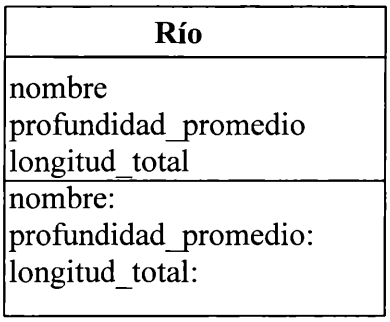


Figura 4.1 Clase Río

Luego, se desea agregar la información geográfica acerca del río. Por ejemplo, necesitamos conocer la profundidad en diferentes lugares, su ancho en alguna latitud, etc.. Una alternativa para agregar funcionalidad es la herencia (como se muestra en la figura 4.2).

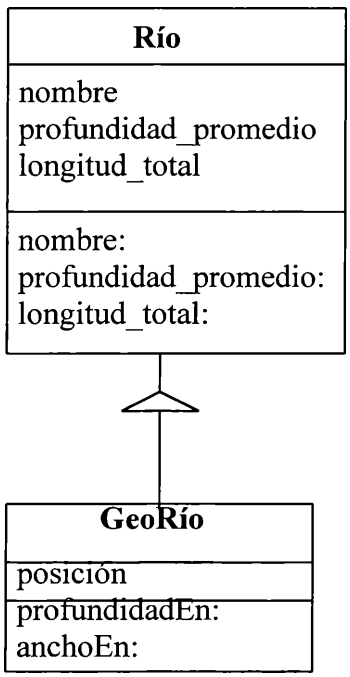


Figura 4.2 Subclase que agrega características espaciales a la clase

Pero en realidad, el uso de decoradores ayuda a actualizar la funcionalidad sin modificar el esquema de clases existente. La solución es definir una clase GeoRío, cuyas instancias trabajen como decoradores de los objetos río. GeoRío define el comportamiento geográfico, por ejemplo incluye un método para calcular la profundidad del río en diferentes coordenadas. Ya que GeoRío conoce la clase que decora (Río) puede delegar a esta clase la ejecución del comportamiento definido existente.

El esquema resultante es el que se nuestra en la figura 4.3.

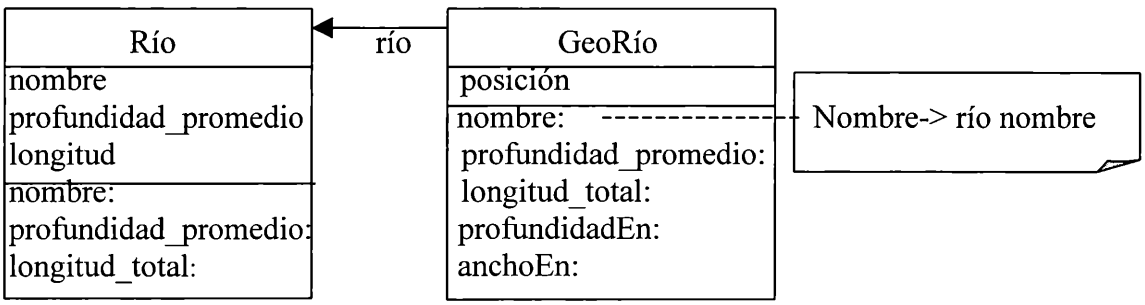


Figura 4.3 Relación entre Río y su clase Decorada

Utilizando la idea de este patrón de diseño es que se construye el modelo

geográfico, agregando características espaciales a los objetos en forma dinámica, tanto en aplicaciones que se están comenzando a desarrollar como en aplicaciones convencionales a las que se les quiere agregar comportamiento geográfico.

Como resultado de usar este patrón, las aplicaciones resultantes manipulan dos tipos de objetos: los conceptuales y los geográficos, que definen dos niveles diferentes. Así el modelo resulta más flexible, ya que podemos definir varios decoradores de un objeto o decorar un decorador, es decir anidar especificaciones.

Si lo que estamos haciendo es ampliar aplicaciones ya existentes, el modelo conceptual que se toma como base no sufre modificaciones, es decir, este modelo puede seguir siendo utilizado tal como fue concebido.

Cuando a un objeto se le agregan características geográficas, siempre se le debe asociar una posición. Esta posición no solamente tiene información acerca de las coordenadas terrestres del objeto, sino que también posiblemente, tiene información acerca del tiempo. En particular, aquellos objetos que están en movimiento poseen una posición que depende del tiempo, por ejemplo, para ubicar un río alcanza con la posición ya que ésta es estática y será la misma en todo momento, pero la posición de un huracán depende esencialmente del momento en que se lo está analizando, entonces la misma debe estar asociada a un período de tiempo determinado. Para evitar confusiones de aquí en adelante, llamaremos Location a la información referente a la posición y al tiempo (cuando sea necesario), mas detalles sobre esta definición pueden ser encontrados en [OpenGIS96].

La definición de la Location de los objetos geográficos es uno de los puntos cruciales en este tipo de aplicaciones y uno de los elementos mas difíciles de manejar. Existen varias formas de realizar proyecciones de datos que han sido tomados de la esfera terrestre para llevarlos a un sistema de coordenadas planares. El uso de cada una de ellas depende sustancialmente del tipo de datos que se está adquiriendo y de la forma en que éste será manipulado.

Una de las formas más comunes de representar posiciones es con el sistema de latitud/longitud, pero no siempre es el más conveniente, ya que por ejemplo, este sistema hace dificultoso el cálculo de distancias y áreas.

Otro sistema de referencias usual es el "Universal Transverse Mercator". Este sistema produce una proyección secante de la superficie terrestre y la divide en zonas de

6 grados de longitud por 8 de latitud, proveyendo un mecanismo que permite el cálculo de áreas en una grilla.

Cuando tratamos con las posiciones de los objetos, es imprescindible manejar información acerca de cuál es el sistema de referencia que se está utilizando. Mas aún, es probable que en una misma aplicación existan objetos referenciados con distintos sistemas, y si en algún momento estos objetos necesitan ser combinados, se deben tener disponibles las operaciones de conversión de un sistema de referencia a otro.

Otro elemento que se debe definir asociado a la Location es la geometría, la cual establece la "forma" que tienen los puntos definidos en la Location.

Dada la complejidad de las posiciones de los objetos y todos los elementos que están relacionados, en el modelo se define la Location como un objeto. Este objeto tiene asociado un sistema de referencias, que es el encargado de interpretar los valores de una posición en un contexto determinado, así como transformar posiciones de un sistema de referencias en otro; y una geometría que es la que brinda la información acerca de la topología de los objetos.

La arquitectura resultante en el modelo es la que se muestra en la figura 4.4.

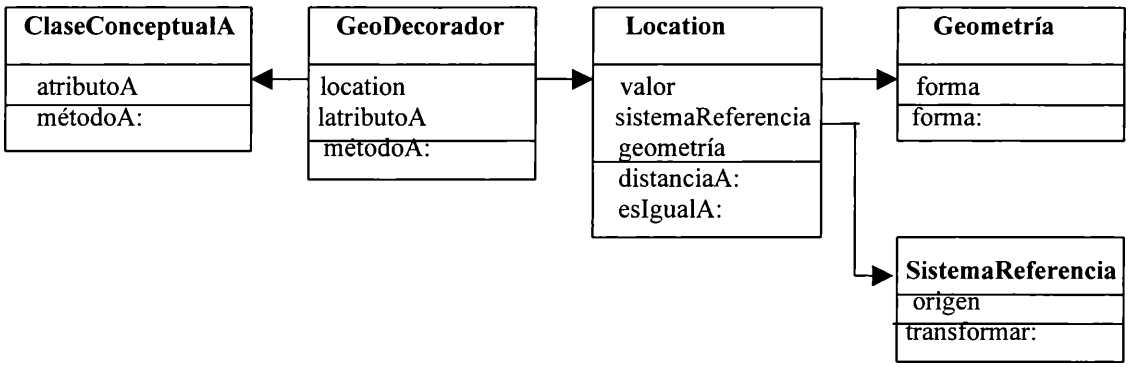


Figura 4.4 El Decorator agrega características espaciales como la Location

En el modelo geográfico pueden aparecer objetos espaciales puros, es decir que no están decorando un objeto conceptual, sino que sólo poseen características geográficas. Por ejemplo, supongamos que queremos en el contexto de una aplicación climatológica, representar el viento como un objeto. La información de interés acerca del viento está dada por su dirección y su velocidad en una posición y en un momento determinado. Está claro que este objeto no va a representar ninguna vista particular en el modelo conceptual.

Para representar este tipo de situaciones se crea una clase abstracta que agrupa el comportamiento, tanto de los objetos decorados como de los espaciales puros. El esquema resultante se muestra en la figura 4.5

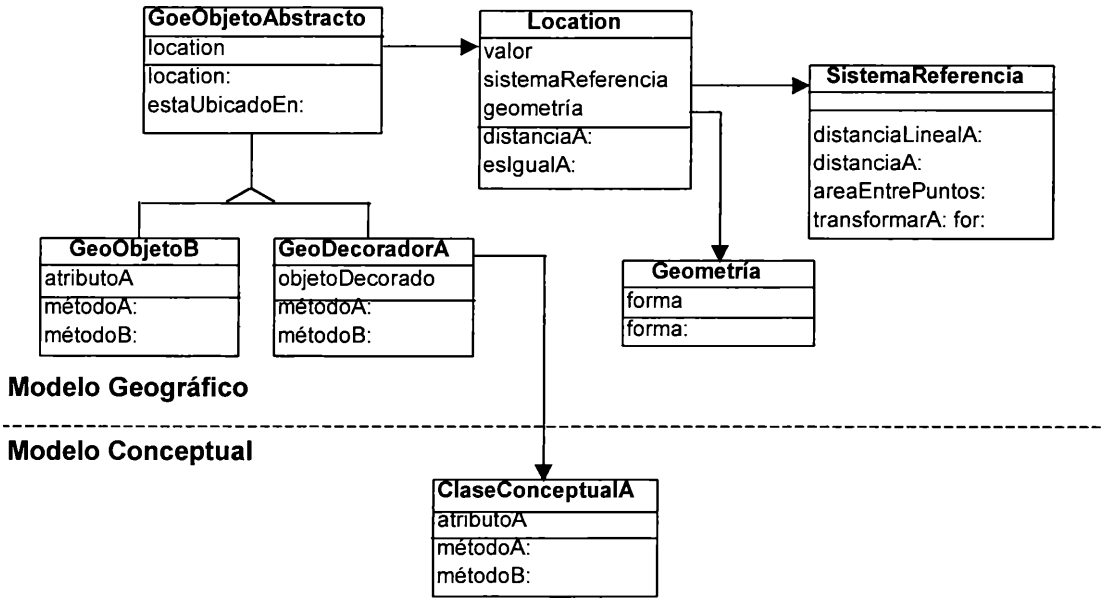


Figura 4.5 Esquema básico para construir aplicaciones geográficas

Esta arquitectura es la que se usará de aquí en adelante para definir datos geográficos. Es decir, asumiremos siempre que un GeoObjeto tiene una Location asociada que define y manipula la posición y, eventualmente la información temporal de ese objeto. La geometría es la que determina todas las características topológicas de los objetos, describirá por ejemplo, a qué topología corresponde (punto, línea, polígono) y las operaciones relacionadas.

4.4 Diseño de patrones para modelar problemas geográficos recurrentes

4.4.1 El patrón de diseño Rol

Los objetos de un modelo geográfico pueden manejar diferentes tipos de información que pueden referirse a intereses completamente diferentes. En el paso de diseño, generalmente se incluye en el objeto toda la información relacionada con él mismo. No es apropiado describir diferentes temas no relacionados en el mismo objeto, como se muestra en el siguiente ejemplo.

Supongamos que estamos modelando un país compuesto por provincias; podríamos definir su comportamiento de manera que permita responder a preguntas tales como: el nombre del país o provincia, todas las provincias que conforman al país, el idioma oficial del país, sistema de gobierno, día de la independencia, feriados nacionales, etc.. La figura 4.6 muestra la estructura básica de las clases País y Provincia.

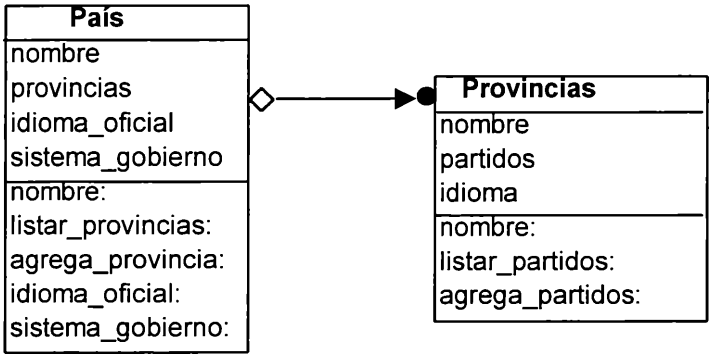


Figura 4.6 Estructura de País y Provincia

Un país también podría necesitar información adicional sobre Demografía (población, promedio de vida, población activa, promedio de ingresos, etc.) e Hidrología (volumen promedio, períodos de inundaciones, etc.). Ya que distintas instancias de País podrían ser analizadas desde diferentes puntos de vista, cada instancia necesitaría mostrar diferentes comportamientos; por ejemplo una instancia de País puede responder sobre sus ríos, mientras otra instancia tiene información sobre su población. Si se incluye toda la información en la misma clase, quedaría una clase compleja difícil de mantener y extender.

De la misma manera, una solución inadecuada sería definir las subclases de País tratando de modelar cada aspecto (ej. Hidrología) como una subclase. La Figura 4.7 muestra la arquitectura resultante, que no permite que el mismo país sea visto desde más de un punto de vista.

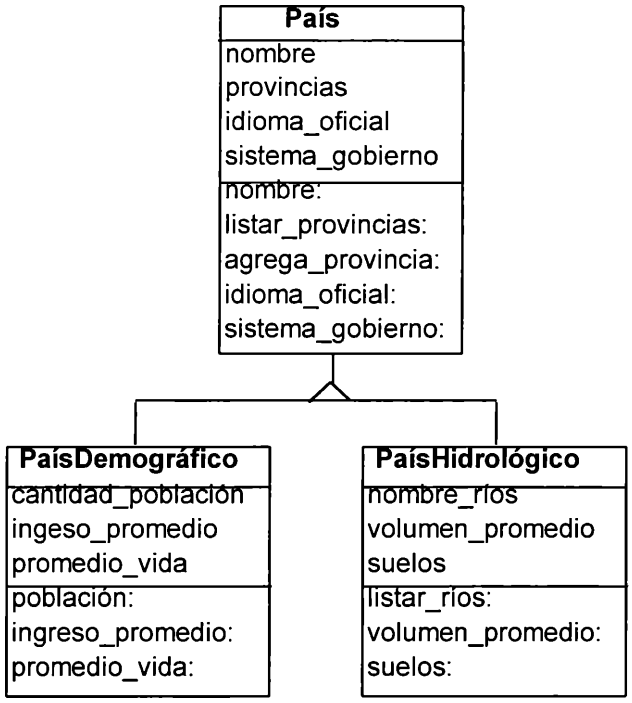


Figura 4.7 Arquitectura resultante con subclases de País

Como se muestra en la figura 4.7, cuando se instancia un País (Demográfico o Hidrológico) se hace muy difícil o imposible para ciertos lenguajes O.O. cambiar las clases de instancia.

Se logra una mejor solución cuando cada tema es definido como un rol; cada rol será un objeto que mantendrá una estrecha relación con el objeto principal. La figura 4.8 muestra un diagrama simple de la relación entre País y sus roles.

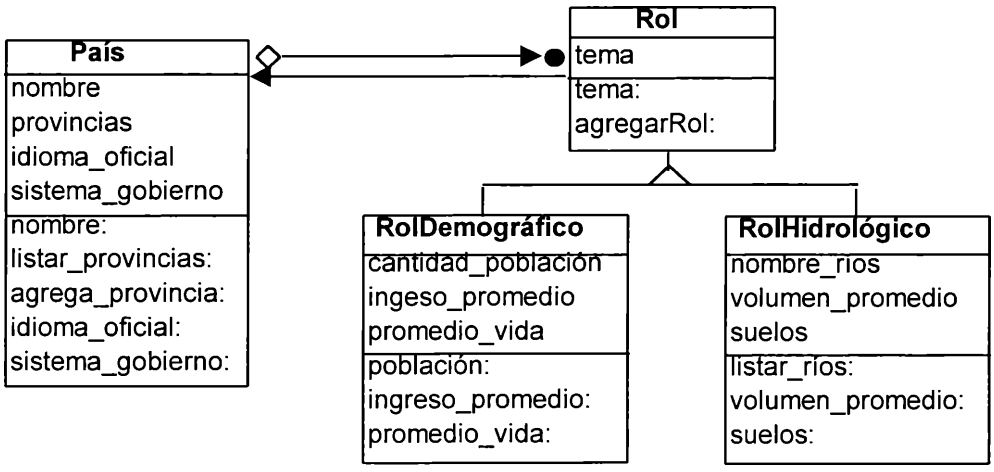


Figura 4.8 Relación entre el País y la jerarquía Rol

Los geoObjetos y sus subclases implementan el comportamiento principal de todos los objetos con características espaciales que pertenezcan al modelo de GIS; por ejemplo, País contiene métodos que proveen su ubicación geográfica, su superficie, etc.. La relación entre un geoObjeto y cada uno de sus roles se implementa con una subclase específica Rol.

Cada rol actúa como una envoltura sobre el objeto que juega ese rol, debido a su interfaz polimórfica. Esta arquitectura permite que un objeto cambie dinámicamente alguno de sus roles e interactúe con ellos para realizar operaciones específicas.

Supongamos que la clase País, además de sus características geográficas, se divide en Provincias. Podría ser necesario que los roles de País colaboren con las provincias del país para calcular algún aspecto geográfico. Por ejemplo, si las provincias tienen un Rol Demográfico, que provee información sobre la población en esa provincia, entonces la población de un país podría ser calculada sumando las poblaciones de todas las provincias. Esto significa que el Rol Demográfico del país debe saber cómo colaborar con el Rol Demográfico de cada provincia, para obtener cada población y calcular el resultado final.

Mientras que el Rol Demográfico aplicado sobre una Provincia es *atómico*, ya que maneja datos concretos, el Rol Demográfico del País es *derivado* ya que debe colaborar con otros para calcular la población. La Figura 4.9 presenta un diagrama de objetos que muestra la relación entre País, Provincia y el Rol Demográfico.

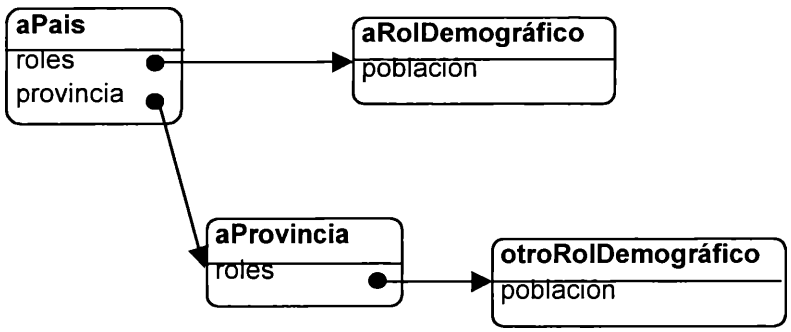


Figura 4.9 Relaciones entre instancias de País, RolDemográfico y Provincia.

Ya que cada rol podría ser caracterizado como derivado o atómico, es necesario abstraer esta condición en una nueva jerarquía, que modele diferentes estrategias usadas para obtener la información deseada. Se puede hacer eso, definiendo una jerarquía EstrategiaAplicada como se muestra en la Figura 4.10.

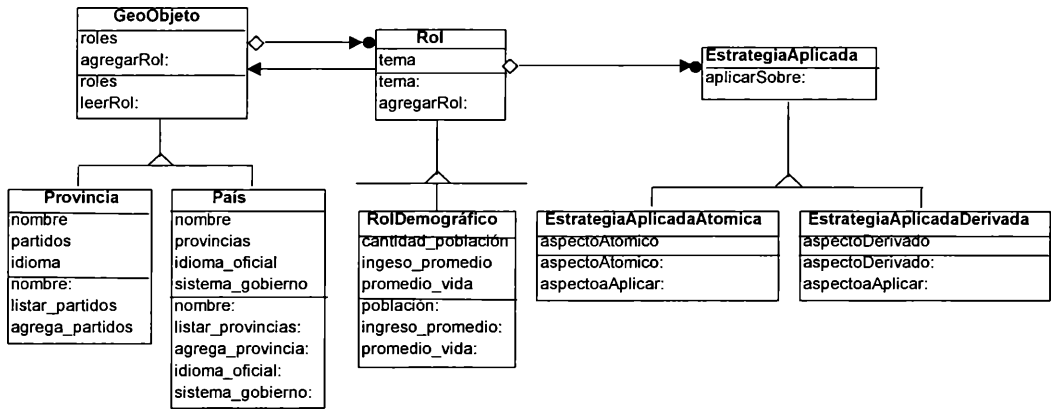


Figura 4.10 Estructura resultante con GeoObject, Rol y jerarquías EstrategiaAplicada

En el ejemplo del país, cuando una instancia de País recibe un pedido de información poblacional, lo satisfará colaborando con todas sus provincias. Se puede modelar la relación entre Provincia, RolDemográfico y País creando un nuevo objeto: la aEstrategiaAplicada, que representa la estrategia para responder o calcular la población. Cuando se crea un objeto correspondiente al rol demográfico, debe asociarse con ese rol la aEstrategiaAplicada específica que será usada para un cálculo particular. La Figura 4.11 muestra la relación resultante entre instancias de País, Provincia, RolDemográfico, EstrategiaAplicadaAtómica y EstrategiaAplicadaDerivada.

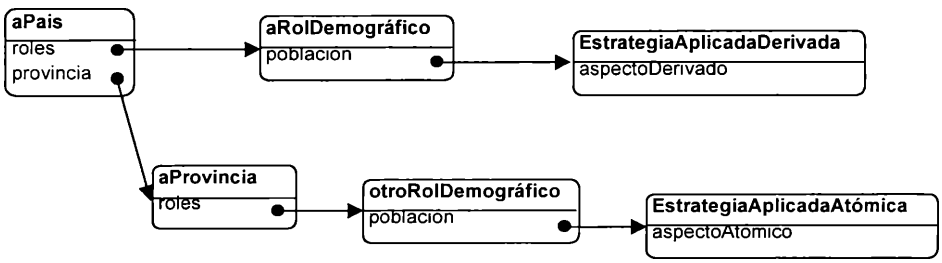


Figura 4.11 Relación entre instancias de País, Provincias, RolDemográfico, Estrategia AplicadaAtómica y EstrategiaAplicadaDerivada

Mientras que aEstrategiaAplicadaAtómica especifica la forma en que una provincia responde a requerimientos demográficos, aEstrategiaAplicadaDerivada

modela e implementa la interacción entre aPaís y aProvincia para producir los datos; y aRolDemográfico es responsable de los cálculos finales. La abstracción Rol también es conocida en otras áreas como modelos de seguridad, aplicaciones financieras, etc.. Dentro del campo de GIS, posibilita asignar dinámicamente distintas visiones a los objetos geográficos. Finalmente permite a los diseñadores de GIS sacar ventaja de la relación 'is divided into', posibilitando construir visiones basadas en el conocimiento y comportamiento de los componentes.

5. OBJETOS COMPUESTOS – RELACIONES DE COMPOSICION

5.1 Objetos Compuestos

5.1.1 Introducción

En los sistemas orientados a objetos, un objeto tiene un conjunto de atributos. El valor de un atributo es otro objeto el que es una instancia del dominio del atributo. Si el dominio de un atributo es una clase primitiva (Integer, String), el valor almacenado para el atributo, es una instancia o un conjunto de instancias del dominio. Si el dominio de un atributo es una clase no primitiva, el valor almacenado para el atributo es el/los identificador/es de la/s instancia/s del dominio. En este caso se dice que un objeto referencia a otros objetos, llamados las instancias de los dominios no primitivos de sus atributos. Los identificadores de las instancias de los dominios no primitivos de sus atributos se llaman *referencias* a esas instancias.

Ya que una referencia a un objeto implica a veces la relación *parte-de* (o consiste-de), generalmente es útil hacer explícita la relación parte-de entre un par de objetos. Se distinguen dos tipos de referencias: débiles y compuestas. Una referencia *débil* es la referencia a un objeto bajo el modelo orientado a objetos, es decir sin las semánticas parte-de. Una referencia *compuesta* es una referencia débil aumentada con la relación parte-de. La semántica de un objeto compuesto es además refinada sobre la base de: si un objeto es una parte de sólo un objeto o de más de uno. Esta consideración lleva a dos tipos de referencias compuestas: **exclusivas y compartidas**. Una referencia compuesta exclusiva de un objeto X a otro objeto Y significa que Y es parte de solamente X; mientras que una referencia compuesta compartida de X a Y significa que Y es parte de X y posiblemente de otros objetos [Kim90].

Se puede refinar más la semántica de una referencia compuesta exclusiva o compartida, sobre la base de: si la existencia de un objeto depende de la existencia de su objeto padre, es decir una referencia compuesta puede ser **dependiente o independiente**. Una referencia compuesta dependiente de X a Y significa que la existencia de Y depende de la existencia de X; mientras que una referencia compuesta independiente no acarrea esta semántica adicional. El borrado de un objeto con referencias dependientes, producirá borrados recursivos de todos los objetos referenciados por el objeto a través de referencias compuestas dependientes (ya sea exclusivas como compartidas).

Resumiendo existen cuatro tipos de referencias compuestas:

1- Referencia compuesta dependiente exclusiva

2- Referencia compuesta independiente exclusiva

3- Referencia compuesta dependiente compartida

4- Referencia compuesta independiente compartida

5.1.2 Semánticas

Un objeto puede tener referencias a varios objetos, sin embargo las referencias no tienen semánticas especiales, sino integridad referecial. Como ya se dijo en la sección anterior, una relación útil que podemos unir a las referencias es la relación *parte_de*. Esta es la base para la construcción de una jerarquía parcial de objetos. Se puede imponer la relación *parte_de* recursivamente sobre todas o sobre algún subconjunto de esas referencias. Una vez que el sistema captura la relación *parte_de* entre una colección de objetos relacionados, tiene la base para permitir consultas sobre jerarquias parciales y para forzar automáticamente las restricciones de integridad que se pueden asociar con las semánticas *parte_de.*

Los objetos relacionados a través de referencias compuestas forman un objeto compuesto; un objeto compuesto es entonces, una jerarquía parcial de objetos componentes. Un tipo de jerarquía parcial, es la jerarquía parcial física en la cual todas las referencias compuestas son exclusivas; un objeto puede ser un componente de a lo sumo un objeto en un ensamblado físico de partes. Un tipo diferente de jerarquía parcial es la jerarquía parcial lógica, la cual puede tener referencias compuestas compartidas; un archivo de texto para un capítulo de un libro puede ser una parte lógica de varios libros.

Un objeto puede tener a lo sumo una referencia compuesta exclusiva a él; y un objeto que tenga una referencia exclusiva a él no puede tener otras referencias compuestas a él. Sin embargo un objeto puede tener cualquier número de referencias convencionales a él, a las que llamamos referencias débiles, sin tener en cuenta si tiene alguna referencia compuesta exclusiva a él. Además, un objeto puede tener cualquier número de referencias compuestas compartidas, dado por supuesto que no tiene referencias compuestas exclusivas.

Un objeto que tiene una referencia compuesta dependiente a él, es borrado si su

padre, es decir el objeto que tiene la referencia dependiente a él, es borrado; o si la referencia misma es borrada. Un objeto con una referencia independiente hacia él no es afectado por el borrado de su objeto padre. Si un objeto tiene una referencia dependiente y una referencia independiente a él, es borrado si el objeto padre con la referencia compuesta dependiente es borrado, o la referencia dependiente es borrada; el objeto con una referencia independiente quedará ahora con una referencia vacía.

Que un objeto x referencie a otro objeto y significa realmente que y es el valor de un atributo A de x. Los objetos x e y son, por supuesto, instancias de algunas clases X e Y respectivamente; y la clase Y es el dominio del atributo A de la clase X. Entonces, la relación parte_de sobre una referencia de un objeto x a un objeto y es una propiedad del atributo de x cuyo valor es y. Esto es, se puede definir un atributo de una clase si tiene la relación parte_de.

Para ver en concreto lo explicado anteriormente, se muestra cómo la sintaxis para el modelo de datos orientado a objetos puede ser extendida para incluir objetos compuestos.

Definición de Clase

Para soportar las semánticas extendidas de una referencia compuesta, extendemos la sintaxis para la especificación de atributos en la definición de clases con cuatro tipos de argumentos claves como sigue [Kim90]:

(NombreAtributo [:**share** SharedValue]
[:**composite** True Or Nil]
[:**exclusive** true Or Nil]
[:**dependent** True Or Nil]

Cuando **composite** es verdadero, especifica que la referencia es compuesta. Si **exclusive** también es verdadero, la referencia compuesta es una referencia compuesta exclusiva. Si **composite** y **dependent** son ambas verdaderas, la referencia es una referencia compuesta dependiente.

Se muestran dos ejemplos de jerarquía parcial. El primero es una jerarquía parcial física (todas las referencias compuestas son exclusivas) y el segundo una jerarquía parcial lógica (puede tener referencias compuestas compartidas).

Ejemplo1:

Considerar una jerarquía compuesta Vehicle. Se requiere que una parte de vehículo pueda ser usada para un solo vehículo al mismo tiempo, sin embargo las partes de vehículo pueden ser usadas por otros vehículos. La definición para la clase Vehicle es como sigue:

```
(class Vehicle
      :superclase nil
:atributes ((Manufacturer      :domain Company)
            (Body              :domain AutoBody
                               :composite true
                               :exclusive true
                               :dependent nil)

            (Divertrain        :domain AutoDrivetrain)
                               :composite true
                               :exclusive true
                               :dependent nil)

            (Tires             :domain ( set-of AutoTire)
                               :composite true
                               :exclusive true
                               :dependent nil)

            (Color             :domain String)))
```

Ya que todos los atributos compuestos en la clase Vehicle son referencias exclusivas, un conjunto de componentes de Vehicle (Body, Drivetain y Tires) puede ser usado para sólo un vehículo. Sin embargo, ya que las referencias exclusivas son independientes, los componentes pueden ser reusados para otros vehículos, si el vehículo que ellos constituyen es desarmado luego. Las componentes de vehicle pueden existir aún si no son parte de algún otro vehicle.

Ejemplo 2:

Considerar ahora documentos electrónicos. Suponemos que un documento consiste de un título, autores y un número de secciones. Una sección está compuesta por párrafos. Un documento puede compartir secciones enteras o párrafos de sección de otros documentos. Pueden agregarse anotaciones a los documentos; sin embargo ellas no se comparten entre los documentos. Los documentos pueden contener imágenes que son extraídas de otros archivos.

Las siguientes son las definiciones de las clases involucradas: Document y Section.

```
(class Document
      :superclase nil
      :atributes ((Título          :domain String)

                  (Author          :domain (set-of string))

                  (Content         :domain (set-of Section)
                                   :composite true
                                   :exclusive nil
                                   :dependent true)

                  (Figures         :domain (set-of Image)
                                   :composite true
                                   :exclusive nil
                                   :dependent nil)

                  (Anotations      :domain ( set-of Paragraph))
                                   :composite true
                                   :exclusive true
                                   :dependent true))

(class Section
      :superclase nil
      :atributes ((Content         :domain (set-of Paragraph)
                                   :composite true
                                   :exclusive nil
                                   :dependent true))
```

El atributo Content, definido como un conjunto, es una referencia compuesta compartida. Otros documentos pueden compartir algún elemento en este conjunto (por ejemplo alguna sección). Una sección existe si pertenece por lo menos a un documento. Similarmente la clase Section tiene una referencia compuesta compartida a Paragraph. Un párrafo puede ser compartido entre diferentes secciones (posiblemente en diferentes documentos). Para que un párrafo exista, debe haber por lo menos una sección conteniéndolo, por lo tanto un documento conteniéndolo. En el caso de Annotation, se asume que una anotación es usada para sólo un documento, de esta manera, la referencia es exclusiva. El atributo Figures es definido como una referencia compuesta independiente, ya que la existencia de Image no depende de los documentos que la contienen.

5.1.3 Implementación

Dado un componente de un objeto compuesto, a veces es necesario determinar sus padres o ancestros. Se necesita mantener en cada componente de un objeto compuesto una lista de referencias compuestas reversas, es decir identificadores de objetos de los objetos padres. Aunque a veces es útil para el sistema poder determinar las raíces de un objeto compuesto para un objeto componente dado, es mejor mantener esa información en cada componente; la razón es que las raíces de un objeto compuesto pueden cambiar cuando éstos son creados en una metodología "bottom up". Además, los punteros reversos deberían ser mantenidos en cada objeto componente en lugar de en una estructura de datos separados. Esta propuesta elimina un nivel de indirección en el acceso a los padres de un componente dado, simplifica el borrado y la migración de objetos; sin embargo hace que se incremente el tamaño del objeto.

El número de referencias compuestas reversas en un objeto componente es igual al número de objetos padres. Una referencia compuesta reversa consiste de un par de flags además del identificador del objeto de un padre. Un flag indica si el objeto es una componente dependiente del padre; mientras que el otro flag indica si el objeto es una componente exclusiva del padre.

5.1.4 Cambios del Esquema

La noción de objeto compuesto genera nuevos cambios en el esquema de una aplicación orientada a objetos. Un tipo de cambio, es la eliminación de un atributo compuesto. El otro cambio, es el agregado o remoción de las semánticas parte_de de un atributo compuesto.

A continuación se detallan los cambios mencionados:

(1) *Sacar un atributo A de una clase C.*

Esta operación causa que todas las instancias de la clase C pierdan sus valores para el atributo A. Si A es un atributo compuesto dependiente, los objetos que son referenciados a través de A son eliminados. El atributo también debe ser eliminado de las subclases que heredan de él.

(2) *Cambiar la herencia de un atributo* (hereda otro atributo con el mismo nombre).

Dependiendo del origen del atributo anterior o nuevo, esta operación puede causar que el atributo anterior sea eliminado. Si el atributo eliminado es compuesto dependiente, esta operación es idéntica a la explicada en (1).

(3) *Remover una clase S como una superclase de la clase C.*

Si esta operación causa que la clase C pierda un atributo compuesto A, los objetos (de otras clases) recursivamente referenciados por instancias de C y sus subclases a través de A son eliminadas de acuerdo a (1).

(4) Eliminar una clase existente C.

Si la clase C tiene uno o más atributos compuestos dependientes, se borran los objetos referenciados por los atributos. Todas las subclases de C se convierten inmediatamente en subclases de las superclases de C.

Ahora se explorará el significado de los cambios de un tipo de atributo a otro diferente. Los cambios pueden ser de dos tipos: remover una restricción de un atributo compuesto y agregar una restricción a un atributo compuesto. Los siguientes cambios remueven una restricción:

(1) Cambiar un atributo compuesto a un atributo no compuesto.

(2) Cambiar un atributo compuesto exclusivo a un atributo compuesto compartido.

(3) Cambiar un atributo compuesto dependiente (exclusivo o compartido) a un atributo compuesto independiente.

(4) Cambiar un atributo compuesto independiente (exclusivo o compartido) a un atributo compuesto dependiente.

Los siguientes cambios agregan una nueva restricción a un atributo compuesto. Se supone que la clase C es el dominio de un atributo A de una clase C' y que A será cambiada.

(1) Cambiar un atributo no compuesto a un atributo compuesto exclusivo. No deben existir referencias compuestas a instancias de la clase C la cual es referenciada por la clase C'.

(2) Cambiar un atributo no compuesto a un atributo compuesto compartido. No deben existir referencias compuestas exclusivas a instancias de la clase C que sean referenciadas por instancias de la clase C'.

(3) Cambiar un atributo compuesto compartido a un atributo compuesto exclusivo. Puede haber a lo sumo una referencia compartida a instancias de la clase C, las cuales son referenciadas por instancias de la clase C'.

5.2 Trabajos Relacionados con Aplicaciones SIG y sus Relaciones de Composición.

5.2.1 Modelo GeoAOO [Kosters et al.95]

5.2.1.1 Introducción

En esta sección se explica la importancia de contar con herramientas que permitan hacer ingeniería de requerimientos (IR) para los desarrollos de software de gran escala en general, y para aplicaciones SIG en particular. Las investigaciones revelan que los métodos IR convencionales no son aptos para modelizar apropiadamente requerimientos específicos de SIG. Para salvar las deficiencias del análisis orientado a objetos (AOO) convencional se introduce el GeoAOO que complementa al AOO con geoprimitivas adecuadas y permite un mejor tratamiento de requerimientos específicos de SIG.

Se discute la tarea de IR para un escenario de aplicación encontrado frecuentemente en proyectos de SIG de la vida real; se refiere a un dominio de aplicación donde los objetos son dispuestos de acuerdo a las relaciones topológicas *parte_de* con ciertas restricciones topológicas. Un ejemplo típico es la relación partition donde las geometrías de las partes son disjuntas y cubren la geometría completa del total. Son partitions por ej. Distritos administrativos como países, provincias y partidos. Consultar las partes de un total dado o modificar una línea de borde son operaciones clásicas asociadas con particiones.

5.2.1.2 Ejemplo de un escenario topológico

Se ejemplificará un escenario topológico con un dominio de difusión que comprende una corporación pública de radio que provee a un país de programas de radio (se utilizará la representación gráfica de Análisis Orientado a Objetos de Coad / Yourdon que se muestra en la figura 5.1).

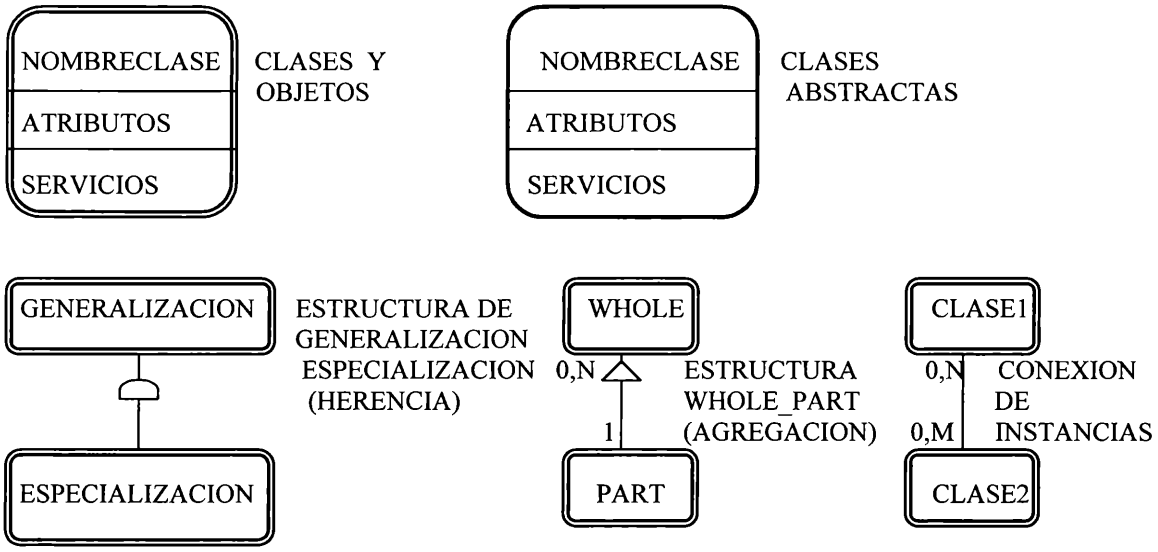


Figura 5.1: Principales primitivas de AOO

El país está particionado en distritos de difusión local, cada uno de ellos manejado por un departamento local de la corporación de radio. La corporación tanto como los departamentos locales, difunden varios programas. Para cada distrito los transmisores garantizan que cualquier programa puede ser recibido en cualquier lugar sin interferencias. La figura 5.2 muestra el dominio del problema. Describe un sector del país particionado en distritos locales, algunos departamentos locales, transmisores y las áreas de propagación de los transmisores dentro del distrito central.

La figura 5.3 muestra un modelo AOO del dominio de difusión. Las clases abstractas *Point* y *Region* generalizan los atributos geométricos comunes y los servicios de los objetos con geometría de punto o bidimensional respectivamente.

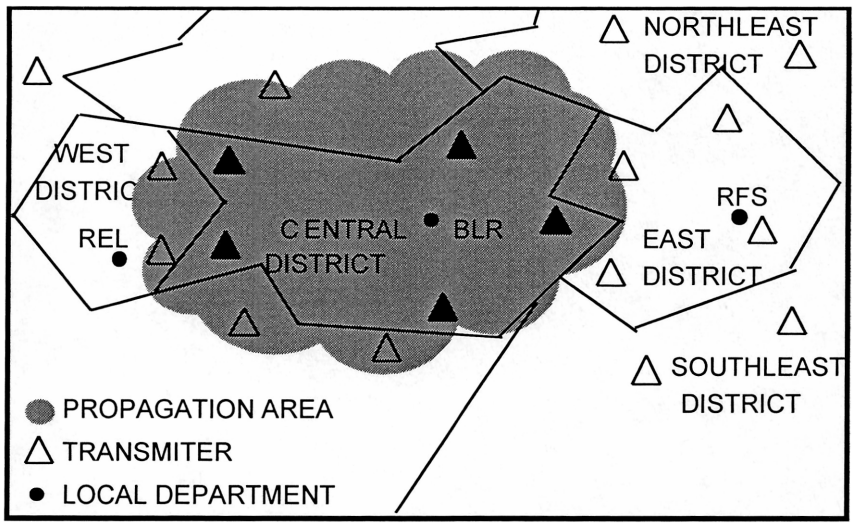


Figura 5.2: Dominio de Difusión

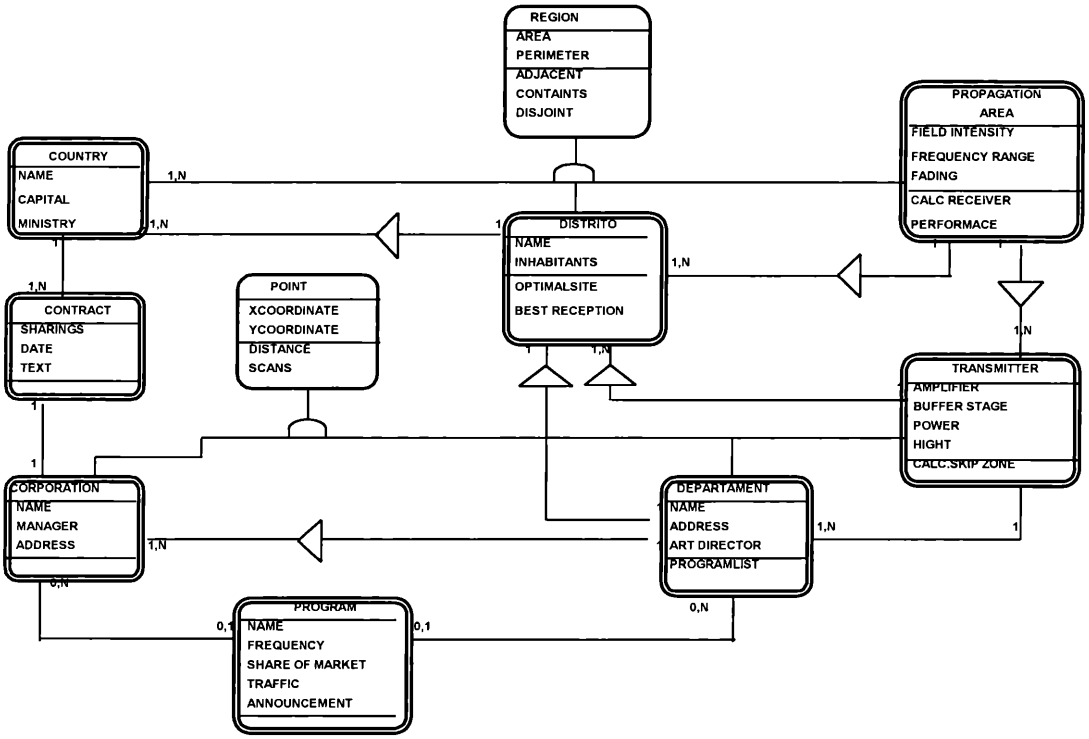


Figura 5.3: AOO. El ejemplo de la difusión

Aunque el ejemplo es pequeño, es necesario mirar detenidamente para darse cuenta que las clases *Contract* y *Program* son clases no espaciales. Más aun, solamente siguiendo las estructuras de herencia, se descubrirá que *Transmitter* y *Department* son

clases especializadas de la clase *Point*. Se ven los mismos problemas con las clases *Propagation area* y *District*, especializadas de *Region*. Para los modelos de AOO de la vida real con más de cientos de clases y relaciones, esta propuesta de modelo encontrará severos problemas.

En lo que respecta a las relaciones topológicas y restricciones, la situación es aún peor. El modelo no provee información sobre las restricciones de inclusión topológica de la relación parte_de entre un distrito, y sus transmisores. No se encuentran las relaciones de partición del país y los distritos como tampoco la relación topológica entre los distritos y las áreas de propagación. Toda esta información esencial no puede expresarse dentro del modelo gráfico y debe ser especificada en la parte textual suplementaria.

Resumimos las principales deficiencias del modelo AOO de la figura 5.3 de la siguiente manera:

- No se provee una distinción clara entre las clases espaciales y no espaciales.
- No es obvio "el tipo" de una clase espacial (punto, línea, región).
- No se hace distinción entre las relaciones topológicas y convencionales.
- Las restricciones topológicas importantes están "ocultas" en las especificaciones textuales.
- Las restricciones topológicas comunes generalmente llevan a especificaciones textuales redundantes.

5.2.1.3 Introducción a GeoAOO

Se introduce GeoAOO que mejora a AOO convencional con algunas primitivas específicas de SIG. El propósito es que el usuario reconozca fácil y naturalmente el dominio del problema.

En general, la identificación de primitivas adecuadas comienza con un análisis cuidadoso del dominio del problema guiado por los siguientes criterios:

- Determinar las características especiales (tipos de datos básicos, relaciones, restricciones, agregados/estructuras) del dominio de aplicación de SIG.
- Para cada una de las características identificadas, determinar si es lo suficientemente relevante y fundamental para ser incorporada como primitiva del modelo. Aquí debe

tomarse en cuenta que demasiadas primitivas decrementan la simplicidad y eventualmente la legibilidad del modelo.

Las primitivas GeoAOO soportan abstracciones de tipo de clases espaciales, estructuras topológicas *parte_de* y estructuras de red. No consideramos aspectos como información de espacio temporal o atributos gráficos. Se apunta al aspecto de geomodelización de GeoAOO, a la potencia expresiva y a la legibilidad de la presentación gráfica, dejando de lado los detalles de la especificación textual complementaria.

5.2.1.4 Tipos de clases

GeoAOO distingue entre un tipo de clase convencional y cuatro tipos de geoclases. Los objetos sin propiedades geométricas son modelados en CLASES CONVENCIONALES, los objetos punto pertenecen a la clase POINT, los objetos unidimensionales a las clase LINE, los objetos bidimensionales a la clase REGION y los objetos scaneados a la clase RASTER. Cada tipo de geoclase provee servicios geométricos standard similares a las operaciones de los correspondientes tipos de datos espaciales de una base de datos espacial. GeoAOO también ofrece restricciones topológicas predefinidas como por ejemplo: si los objetos de una clase pueden intersectarse o no, para mejorar la solidez y precisión de la especificación de clase textual asociada.

La figura 5.4 describe los símbolos de clase de GeoAOO.

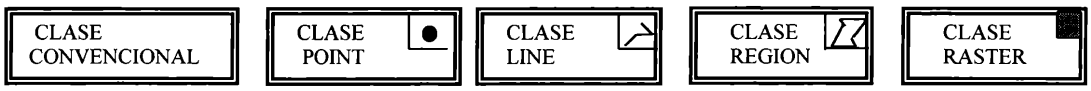


Figura 5.4: Símbolos de clase de GeoAOO

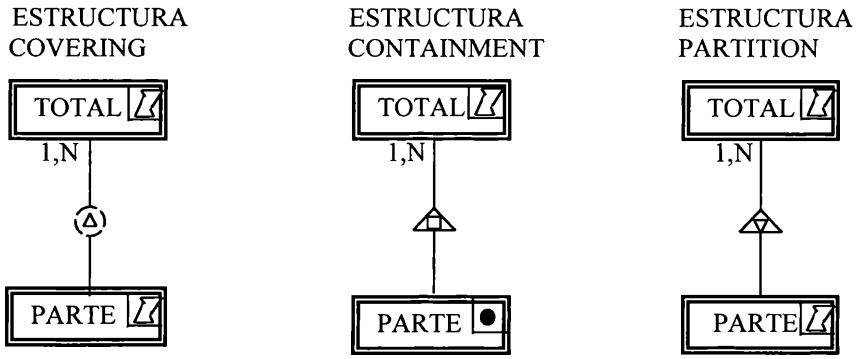


Figura 5.5:Estructuras topológicas parte_de en GeoAOO

5.2.1.5 Estructuras topológicas parte_de

GeoAOO ofrece primitivas topológicas *parte_de* para modelizar relaciones topológicas entre un total y sus partes típicos en los dominios de aplicaciones de SIG. Notar que los objetos convencionales no deben participar en las estructuras topológicas *parte_de*. Las estructuras *parte_de* se dividen en estructuras *covering*, *conteinment* y *partition*. Una propiedad común de todas ellas, es la intersección no vacía de la geometría de cada parte con la geometría del total.

- Para una estructura *covering*, el total y sus partes pertenecen al mismo tipo de geoclase y la geometría del total es cubierta por la unión de las geometrías de sus partes.
- Para una estructura *containment* la geometría del total contiene la geometría de todas sus partes.
- La estructura *partition* es una estructura containment con dos condiciones adicionales: el total y sus partes pertenecen al mismo tipo de geoclase y las geometrías de las partes forman una partición de la geometría del total.

Los símbolos de las tres primitivas se muestran en la figura 5.5. Se proveen las palabras reservadas para la especificación textual de las estructuras topológicas parte_de (figura 5.7). Como en las relaciones convencionales, las estructuras topológicas parte_de proveen servicios standard como acceder o (des)conectar objetos y consultar por objetos vecinos u objetos sobre el borde de una partición.

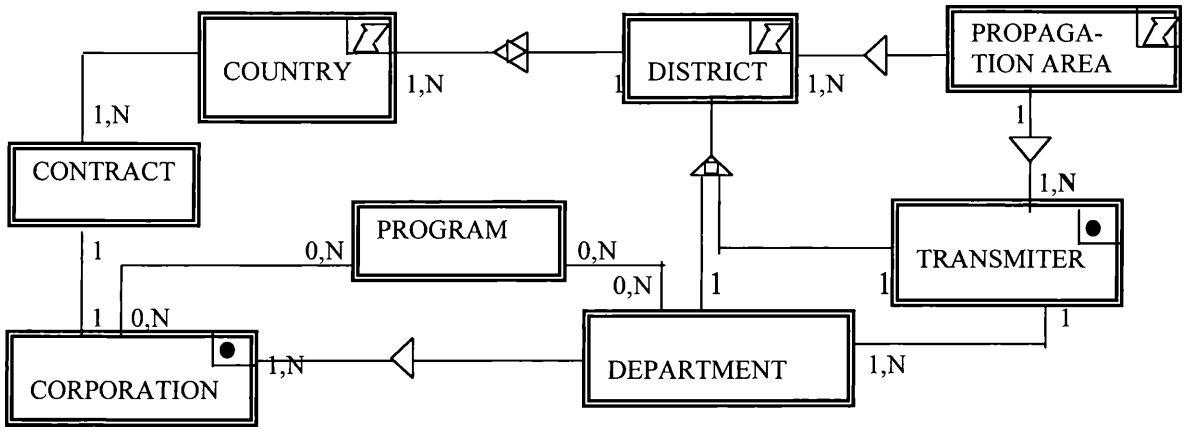


Figura 5.6: GeoAOO: Ejemplo de difusión

Comparando el modelo GeoAOO del ejemplo de difusión de la figura 5.6 con el modelo AOO de la figura 5.3, se ve que el modelo GeoAOO gana claridad y expresividad a través de sus nuevas primitivas. A diferencia del modelo AOO la información espacial relevante es expresada dentro del modelo. Además de una mejor representación gráfica, la especificación textual clarifica a través de las palabras reservadas que se refieren a relaciones topológicas predefinidas y restricciones. La Figura 5.7 muestra la especificación de la clase District.

CLASS: DISTRICT
TYPE: REGION
ATRIBUTTES
NAME:
Identification of a DISRTICT - *object*
DOMAIN: TEXT;
INHABITANTS:
Actual number of of residents
DOMAIN: NATURAL;
...
SERVICES
OPTIMAL-SITE:
IN transmitter: TRANSMITTER
Calculates the optimal location for the indicated transmitter
OUT *location:* POINT
BEST_RECEPTION:
IN *position:* POINT, *frecuency* : REAL
Calculates the transmiter providing the best reception of the specifed frecuency at the indicated position
OUT *transmitter* : TRANSMITTER
...
RELATIONSHIPS
PROPAGATION:

self IS COVERED BY 1,N PROPAGATION AREA-*objects*
STATIONS:
self CONTAINS 1,N TRANSMITTER – *objects*
STUDIOS:
self CONTAINS 1 DEPARTMENT – *objects*
ADMINISTRATION:
self IS PARTITION-PART OF COUNTRY– *objects*
INVARIANTS
I.) *self*.NO_HOLES ^ *self*.CONNECTED

END DISTRCT

Figura 7: GeoAOO. Especificación de la clase DISTRICT

5.2.2 Relaciones Parte_de usuales en un SIG

Algunos elementos geográficos que aparecen en el mundo real se pueden ver como agregaciones de otros objetos. De acuerdo a la situación que estamos modelando, encontramos diferentes casos que pueden tener diferentes restricciones acerca de la semántica de composición. A veces la existencia de un total depende de la existencia de sus partes, esta restricción podría ser inadecuada en otros casos. Definir y entender la semántica exacta de la composición es muy importante ya que permite establecer un diseño apropiado y restricciones de implementación.

Se definen tres relaciones parte_de que usualmente aparecen en un SIG [Gordillo et al.97]. Estas relaciones definen restricciones acerca de su cardinalidad y acerca del tiempo de vida que involucra a los objetos.

- **Has_a**: A Has_a B, significa que una instancia de A puede tener una o más instancias de B como sus partes. Esta relación no tiene restricciones particulares sobre A y B. Los tiempos de vida de las instancias son independientes.

  Por ejemplo, considerar que A es un país y B es un río. El río se puede secar y el país sigue existiendo, y el país se puede disolver (administrativamente) y el río sigue existiendo.

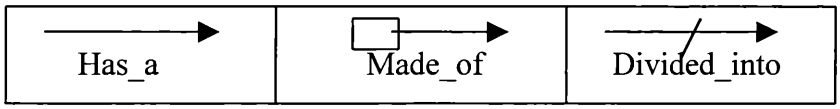
- **Made_of**: A Made_of B, si para cada instancia de A existe al menos una instancia de B. El tiempo de vida de las instancias de A termina en el momento que ellas no tienen instancias de B.

  Por ejemplo, considerar que A es una cadena de montañas y B montañas. No tendría sentido que una instancia de A existiera sin ninguna instancia de B.

- **Divided_into**: Si A divided_into B, entonces cada instancia de B debe estar asociada con una instancia de A, si no la instancia de B no existe. El tiempo de vida de las instancias de B está limitado por su relación con una instancia de A.

  Por ejemplo, un país esta dividido en provincias y no existen provincias fuera de un país.

Para manejar todas estas relaciones de composición, se define una notación adicional a las primitivas OMT, para distinguir cada una de ellas.



6. COMPOSICIONES SEMÁNTICAS PARA SIG

6.1 Introducción

En los últimos años, hay una gran tendencia en el uso de conceptos orientados a objetos para diseñar estructuras abstractas de aplicaciones geográficas; pero dada la complejidad de los dominios y la gran variedad de datos que caracterizan a tales aplicaciones, el proceso de diseño de estos tipos de sistemas resulta dificultoso y complejo.

Usualmente la semántica del dominio, es difícil de expresar y se deben agregar anotaciones textuales al modelo resultante. La mayoría de las veces, los métodos de diseño de los SIGs orientados a objetos son extensiones de otro tipo de aplicaciones. Generalmente las semánticas de estas extensiones son especificaciones pobres.

El objetivo principal de esta tesis es definir relaciones semánticas parte_de entre objetos referenciados geográficamente, utilizando el Modelo de Diseño Orientado a Objetos para aplicaciones SIG tratado en [Gordillo et al.97]

Como primer paso, investigamos algunas de las semánticas existentes de las relaciones parte_de [Kim90] [Kosters et al.95] [Gordillo et al.97], analizando las restricciones que definen cada una de las relaciones, mostrando la aplicación de estas relaciones mediante un ejemplo en un dominio SIG.

Luego definimos nuevas relaciones agregando además, restricciones topológicas entre el total y las partes. Finalmente encontramos equivalencias entre las relaciones definidas en el modelo conceptual y geográfico.

6.2 Analizando restricciones de relaciones parte_de existentes

Veremos el significado de las restricciones de dependencia e independencia, exclusivas y compartidas, luego las analizaremos en términos de las restricciones impuestas por la cardinalidad y el tiempo de vida de los objetos. Estas restricciones son muy importantes cuando nos referimos a las relaciones parte_de que usualmente aparecen en un SIG.

Dependiente/Independiente – Exclusiva/Compartida

Dentro del contexto de Base de Datos Orientada a Objetos, en [Kim90] se propone distinguir las relaciones parte_de de otras relaciones de referencia. Su propósito

es proveer a las partes con semánticas adicionales. Un link parte_de (referencia compuesta) desde un objeto **t** a un objeto **p** puede ser clasificado como *exclusivo* o *compartido* y ortogonal a esto puede ser *dependiente* o *independiente.*

Una referencia parte_de exclusiva desde t a p significa que p es parte de sólo t, mientras que una referencia parte_de compartida desde t a p significa que p es parte de t y posiblemente de otros objetos.

Las semánticas de las referencias parte_de exclusivas o compartidas pueden refinarse aún más sobre la base de que una parte depende de la existencia de su objeto padre. En estos términos, una referencia parte_de puede ser dependiente o independiente. Una referencia parte_de dependiente desde t a p significa que la existencia de p depende de la existencia de t, mientras que una referencia parte_de independiente no acarrea esta semántica.

De acuerdo a la definición dada en [Kim90] podemos dividir a la semántica de la relación parte_de en cuatro categorías:

1) Referencia compuesta dependiente exclusiva: Referencia compuesta dependiente de X a Y significa que la existencia de Y depende de la existencia de X e Y es solo parte de X.

Ejemplo: un documento compuesto al que se pueden agregar anotaciones. La existencia de las anotaciones depende de la existencia del documento y una anotación es sólo parte de un documento.

2) Referencia compuesta independiente exclusiva: Referencia compuesta independiente X de Y significa que la existencia de Y no depende de la existencia de X, pero Y puede ser sólo parte de X.

Ejemplo: un auto y sus componentes, consideremos sus ruedas. La existencia de las ruedas no depende de la existencia del auto, pero las ruedas pueden ser parte de sólo un auto a la vez.

3) Referencia compuesta dependiente compartida: Referencia compuesta dependiente de X a Y significa que la existencia de Y depende de la existencia de X e Y es parte de X y posiblemente de otros objetos.

Ejemplo: un documento compuesto por secciones. La existencia de la sección depende de la existencia del documento y además una sección puede ser compartida por varios documentos.

4) Referencia compuesta independiente compartida: Referencia compuesta independiente X de Y significa que la existencia de Y no depende de la existencia de X, e Y es parte de X y posiblemente de otros objetos.

Ejemplo: un documento conteniendo imágenes. La existencia de la imagen no depende de la existencia del documento y la imagen puede ser parte de varios documentos.

Cardinalidad

En particular la dependencia de una parte sobre un objeto total tendrá una cardinalidad mínima de uno, ya que una parte **p** es parte de al menos un total **t** (**p** y **t** son instancias de una clase **P** y **T** respectivamente).

De esta manera, la dependencia implica una cardinalidad mínima de uno sobre el lado del total.

Las restricciones de cardinalidad pueden ser representadas como un rango de valores de la forma [min val : max val]. Si en alguno de los casos, no existe un valor mínimo o máximo se representará con dos puntos, indicando que esta información es irrelevante como mostramos en la siguiente figura:

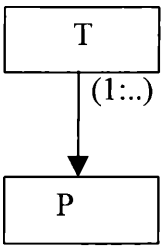


Figura A

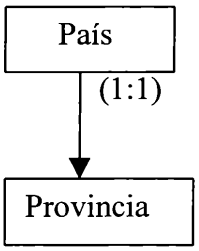


Figura B

En la figura A, se ve que una parte P dependiente de un total T implica que P es parte_de un total T, la cardinalidad mínima debe ser 1 del lado del total y no hay restricciones sobre la cardinalidad máxima.

En la figura B, se considera como ejemplo un país y una provincia. La provincia es parte_de dependiente del país, la cardinalidad mínima será 1 sobre el total (país) y la

cardinalidad máxima también es 1, ya que una provincia puede ser parte de sólo un país a la vez.

En una relación parte_de exclusiva entre **t** y **p**, la cardinalidad máxima del lado de t (total) debe ser 1, ya que p (parte) deberá ser parte de a lo sumo un t (total). La cardinalidad mínima es irrelevante (Figura C).

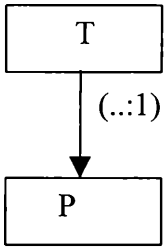


Figura C

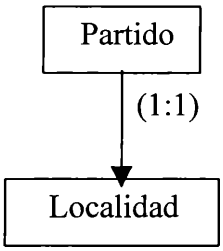


Figura D

En la figura D, se considera como ejemplo un partido y una localidad. La localidad es parte_de exclusiva de un partido, la cardinalidad máxima es 1 sobre el partido, ya que una localidad puede pertenecer a un solo partido.

Las relaciones parte_de independientes y compartidas, no definen restricciones acerca de la cardinalidad de los objetos.

Tiempo de Vida

Con respecto al tiempo de vida, vamos a analizar las restricciones de dependencia sobre los objetos partes.

1) Cualquier parte p dependiente debe ser creada después o concurrentemente con sus objetos padres t.

2) Una parte p dependiente debe ser borrada antes o concurrentemente con su objeto padre t. El borrado de un objeto padre t implica el borrado de sus partes p dependientes.

3) Una parte_de dependiente no puede ser desconectada, ya que las partes p no pueden existir separadas de su objeto padre t.

Por lo dicho anteriormente, podemos concluir que la dependencia impone restricciones sobre la secuencia de eventos, en este caso, sobre el tiempo de vida de las partes con relación a los totales.

Formalmente, las restricciones de dependencia pueden ser dadas por la siguiente implicación:

Si existe una referencia parte_de dependiente desde **t** a **p** entonces:

1) (crear (t) <= crear (p) y

2) ((borrar (p) <= borrar (t)) y

3) El link parte_de (p,t) no puede ser desconectado luego de crear(p)

6.3 Relaciones semánticas usuales en un Dominio de SIG

Como se mencionó en la sección 5.2.2, las relaciones que usualmente aparecen en un SIG definidas en [Gordillo et al.97] definen restricciones acerca de la cardinalidad y el tiempo de vida. Analizaremos estas restricciones en forma detallada en cada una de las relaciones.

Has_a: A Has_a B, significa que una instancia de A puede tener una o más instancias de B como sus partes. Esta relación no tiene restricciones particulares sobre A y B. Los tiempos de vida de las instancias son independientes.

Notamos que en esta relación no hay restricciones acerca de la cardinalidad y queda claro que tanto para el total como para sus partes el tiempo de vida es independiente.

Made_of: A Made_of B, si para cada instancia de A existe al menos una instancia de B. El tiempo de vida de las instancias de A termina en el momento que ellas no tienen instancias de B.

En esta relación, la cardinalidad mínima de A es 1 y el tiempo de vida del total depende del tiempo de vida de sus partes.

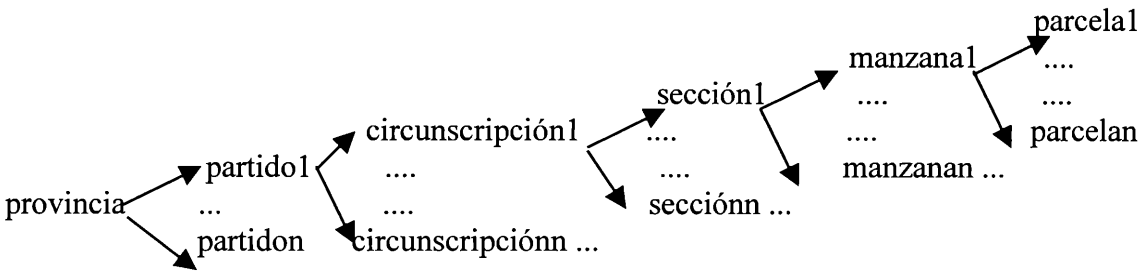
Divided_into: Si A divided_into B, entonces cada instancia de B debe estar asociada con una instancia de A, si no la instancia de B no existe. El tiempo de vida de las instancias de B está limitado por su relación con una instancia de A.

La cardinalidad mínima de A es 1 y el tiempo de vida de las partes depende del total.

6.4 Representando relaciones parte_de para una aplicación SIG

Basándonos en el Modelo Orientado a Objetos para SIG [Gordillo et al.97], diseñaremos una aplicación SIG, utilizando las relaciones vistas anteriormente para el modelo conceptual y las relaciones semánticas entre objetos con características geográficas (referenciadas espacialmente) vistas en la sección 5.2.1 para el modelo geográfico [Kosters et al.95].

El problema a resolver es la recaudación impositiva en el ámbito de una provincia. Para registrar la información referente a las tierras y sus propietarios se usará la siguiente representación catastral:



con las consideraciones:

- La provincia está compuesta por al menos un partido.
- Cada partido está compuesto por al menos una circunscripción.
- Cada circunscripción está compuesta por al menos una sección.
- Cada sección está compuesta por al menos una manzana.
- Cada manzana puede o no contener parcelas.

6.4.1 - Modelo Conceptual

6.4.1.1 Diagrama de clases y relaciones:

La figura 6.1 muestra el diagrama de clases y relaciones de la aplicación que estamos considerando, utilizando la notación OMT [Rumbaugh et al.91]. Para la representación de las relaciones de composición utlizamos la notación adicional de las primitivas OMT [Gordillo et al.97].

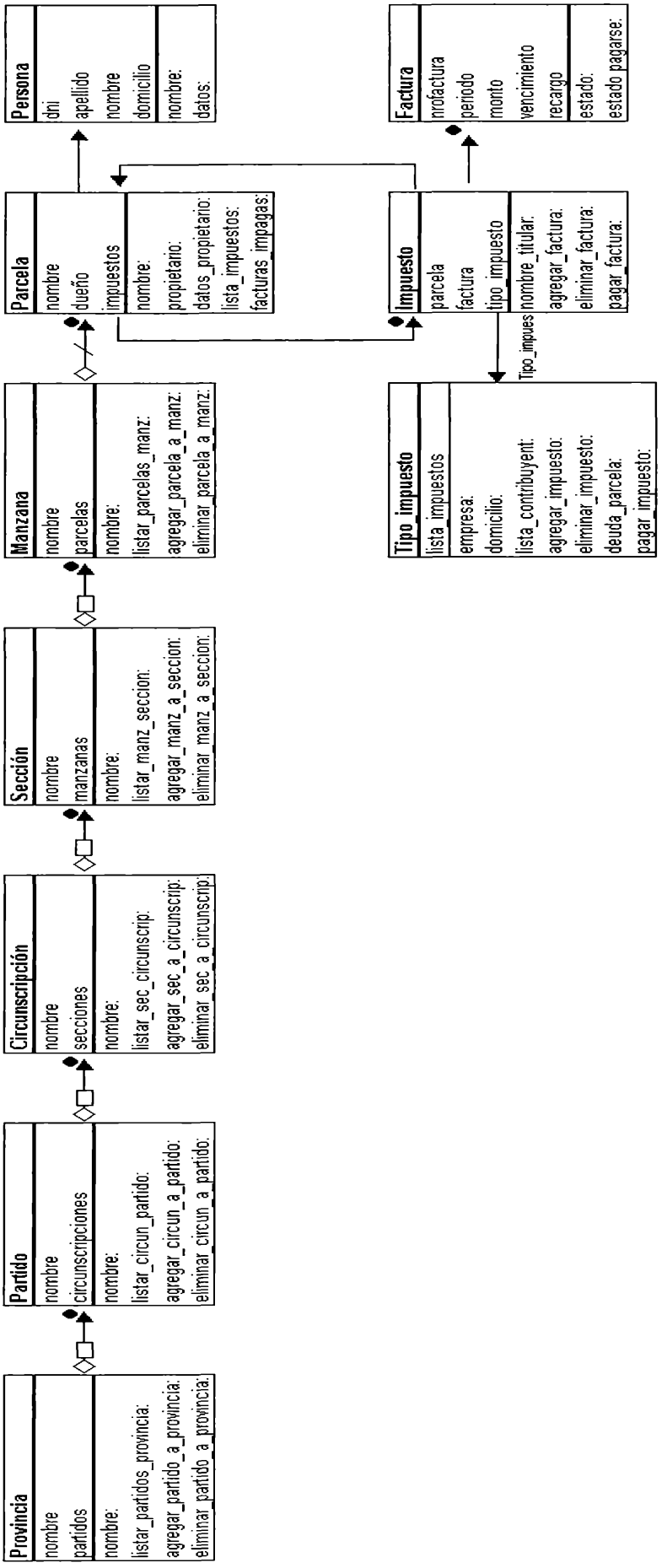


Figura 6.1: Modelo Conceptual

6.4.1.2 Descripción de las Operaciones

Clase: TIPO_IMPUESTO

empresa: devuelve el nombre de la empresa.

domicilio: devuelve el domicilio de la empresa.

lista_contribuyent(tipo_imp,lista_tit): devuelve en lista_tit el nombre y apellido de todos los titulares de tipo_imp.

agregar_impuesto: agrega un impuesto correspondiente a una parcela.

eliminar_impuesto: elimina un impuesto correspondiente a una parcela.

deuda_parcela (id , parcela) : Dada una parcela devuelve todas las facturas impagas de la misma.

pagar_impuesto(id, parcela, nrofactura): Dada una parcela y el número de factura, la misma cambiará su estado como paga.

Clase: IMPUESTO

nombre_titular(parcela): Dada una parcela devuelve el nombre de su dueño.

agregar_factura(nrofactura): Agrega una factura para esa parcela.

eliminar_factura (nrofactura): Elimina una factura para esa parcela.

pagar_factura(nrofactura): Pone la factura pasada como parámetro como paga.

Clase: FACTURA

Estado(est): devuelve en est el estado en que se encuentra la factura (paga o impaga).

estado_pagarse: si la factura está impaga cambia su estado a paga, de lo contrario se informará que ya ha sido paga.

Clase: PROVINCIA

nombre: retorna el nombre de la provincia

listar_partidos_provincia: lista todos los partidos pertenecientes a la provincia.

agregar_partido_a_provincia: agrega un partido a la provincia.

eliminar_partido_a_provincia: elimina un partido de la provincia.

Clase: PARTIDO

nombre: retorna el nombre del partido.

listar_circun_partido: lista todas las circunscripciones pertenecientes al partido.

agregar_circun_a_partido: agrega una circunscrpción al partido.

eliminar_circun_a_partido: elimina una circunscrpción del partido.

Clase: CIRCUNSCRIPCION

nombre: retorna el nombre de la circunscripción

listar_sec_circunscrip: lista todas las secciones pertenecientes a la circunscripción.

agregar_sec_a_circunscrip: agrega una sección a la circunscripción.

eliminar_sec_a_circunscrip: elimina una sección de la circunscripción.

Clase: SECCION

nombre: retorna el nombre de la sección.

listar_manz_sección: lista todas las manzanas pertenecientes a la sección.

agregar_manz_a_sección: agrega una manzana a la sección.

eliminar_manz_a_sección: elimina una manzana de la sección.

Clase: MANZANA

nombre: retorna el nombre de la manzana.

listar_parcelas_manz: lista todas las parcelas pertenecientes a la manzana.

agregar_parcela_a_manz: agrega una parcela a la manzana.

eliminar_parcela_a_manz: elimina una parcela de la manzana.

Clase: PARCELA

nombre: Retorna el nombre de la parcela

propietario: devuelve el nombre del dueño de la parcela .

datos_propietario: devuelve los datos del propietario de la parcela.

lista_impuestos: devuelve todos los impuestos que paga esa parcela.

facturas_impagas(impuesto,lista_facturas): dado un tipo de impuesto devuelve todas las facturas impagas del mismo.

Utilizamos el patrón TYPE OBJECT [Johnson et al.97] en las clases Tipo_Impuesto e Impuesto, ya que a priori no conocemos todos los tipos de impuestos, ni la cantidad total de impuestos que se cobrarán para cada tipo. Entonces,

Tipo_Impuesto e Impuesto y la relación es_instancia_de entre ellos es una aplicación del patrón antes mencionado.

6.4.1.3 Descripción de las Relaciones Semánticas

Desde el punto de vista de las definiciones de [Kim90], todas las relaciones semánticas del modelo son dependientes exclusivas. Dependientes porque si existe un partido, tiene que existir una provincia a la cual pertenece; si existe una circunscripción tiene que existir un partido al cual pertenece; si existe una sección tiene que existir una circunscripción a la cual pertenece; si existe una manzana tiene que existir una sección a la cual pertenece; y si existe una parcela tiene que existir una manzana a la cual pertenece. Son exclusivas porque un partido pertenece sólo a una provincia; una circunscripción pertenece sólo a un partido; una sección pertenece sólo a una circunscripción; una manzana pertenece sólo a una sección; y una parcela pertenece sólo a una manzana.

Según [Gordillo et al.97], las relaciones entre provincia y partido, partido y circunscripción, circunscripción y sección, y sección y manzana son "made_of", ya que como se dijo en la especificación del problema, una provincia está compuesta por al menos un partido, un partido está compuesto por al menos una circunscripción, etc..

La relación entre manzana y parcela es "divided_into", ya que una manzana puede o no contener parcelas.

6.4.2 - Modelo Geográfico

6.4.2.1 Diagrama de clases y relaciones

Las clases que contienen características espaciales son las siguientes: PROVINCIA, PARTIDO, CIRCUNSCRIPCIÓN, SECCIÓN, MANZANA y PARCELA.

Cada una de ellas se decora con una nueva clase que además de englobarla, contiene la información geográfica.

La figura 6.2 muestra la primera aproximación para el modelo geográfico, utilizando para la representación de las relaciones de composición la notación de [Kosters et al.95].

6.4.2.2 Descripción de las Operaciones

Clase: GEOPROVINCIA

nombre: Retorna el nombre de la provincia

perímetro: Retorna el perímetro de la provincia

superficie: Retorna la superficie de la provincia

Las clases GEOPARTIDO, GEOCIRCUNSCRIPCIÓN, GEOSECCIÓN y GEOMANZANA, tienen las mismas operaciones que la clase GEOPROVINCIA.

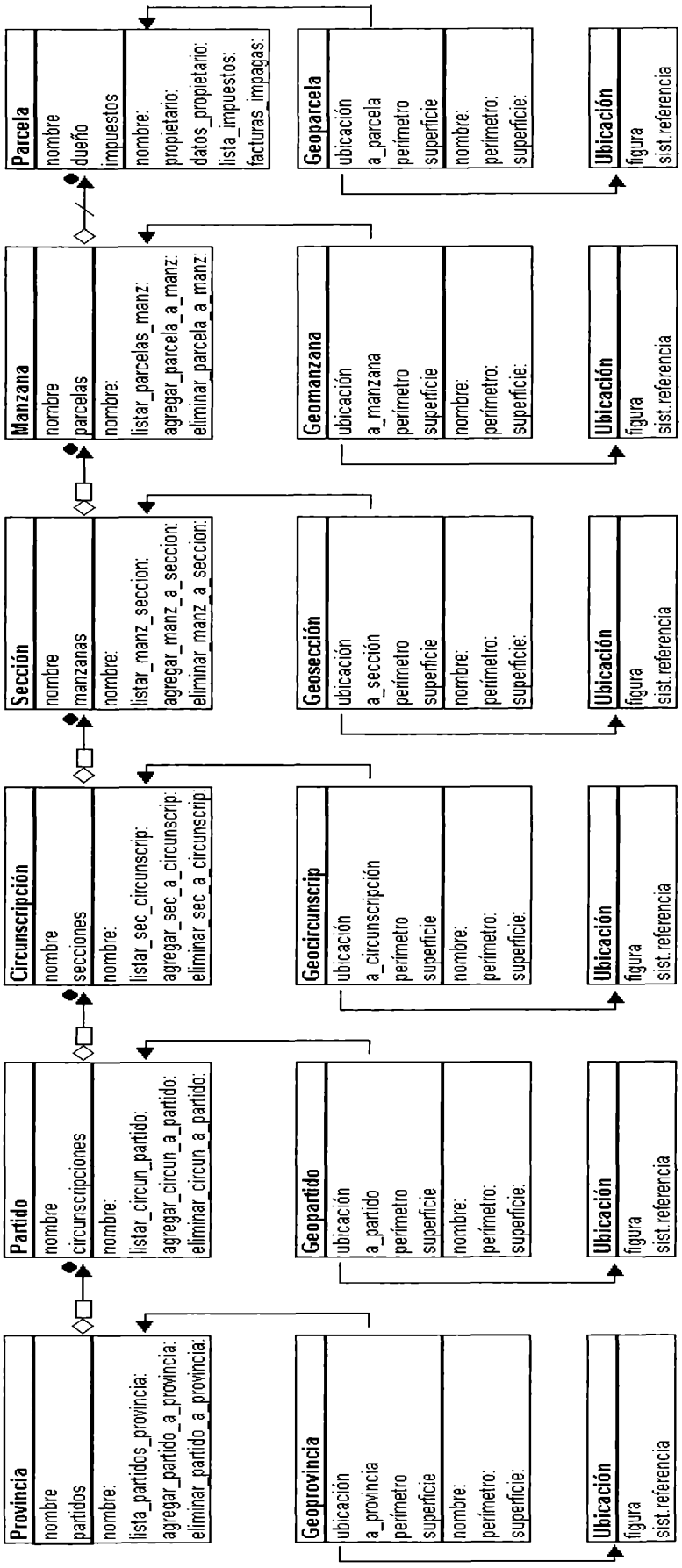


Figura 6.2: Modelo Geográfico

6.4.2.3 Descripción de las Relaciones Semánticas

Este modelo presenta la siguiente dificultad. El proceso para representar geográficamente los partidos de una provincia es engorroso. Con la operación lista_partidos de una instancia de la clase provincia, se recuperan todas las instancias de la clase Partido de dicha provincia. Para cada elemento de esa lista, se debe recorrer las instancias de Geopartido hasta encontrar la que haga referencia a ese elemento, y así se podrá representar geográficamente. Se presenta la misma situación cuando se quiere recuperar todas las circunscripciones de un partido, todas las secciones de una circunscripción, todas las manzanas de una sección, y todas las parcelas de una manzana.

Estos procesos se pueden simplificar agregando nuevas relaciones semánticas entre las geoclases, como se muestra en la segunda aproximación (Figura 6.3).

Se definen las relaciones semánticas entre los objetos con características geográficas según [Kosters et al.95].

Las relaciones entre Geoprovincia y Geopartido, entre Geopartido y Geocircunscripción, Geocircunscripción y Geosección, y Geosección y Geomanzana son "covering". Ya que, por la especificación del problema, la unión de los partidos que pertenecen a una provincia, cubre el total de la misma; las circunscripciones que pertenecen a un partido cubren el total del mismo; etc. La misma relación se cumple entre geomanzana y geoparcela, si la manzana ya está dividida en parcelas.

La simplificación a la que hicimos referencia anteriormente consiste en que desde la instancia de Geoprovincia podemos acceder directamente a las instancias de los Geopartidos que la forman y así poder representarlos.

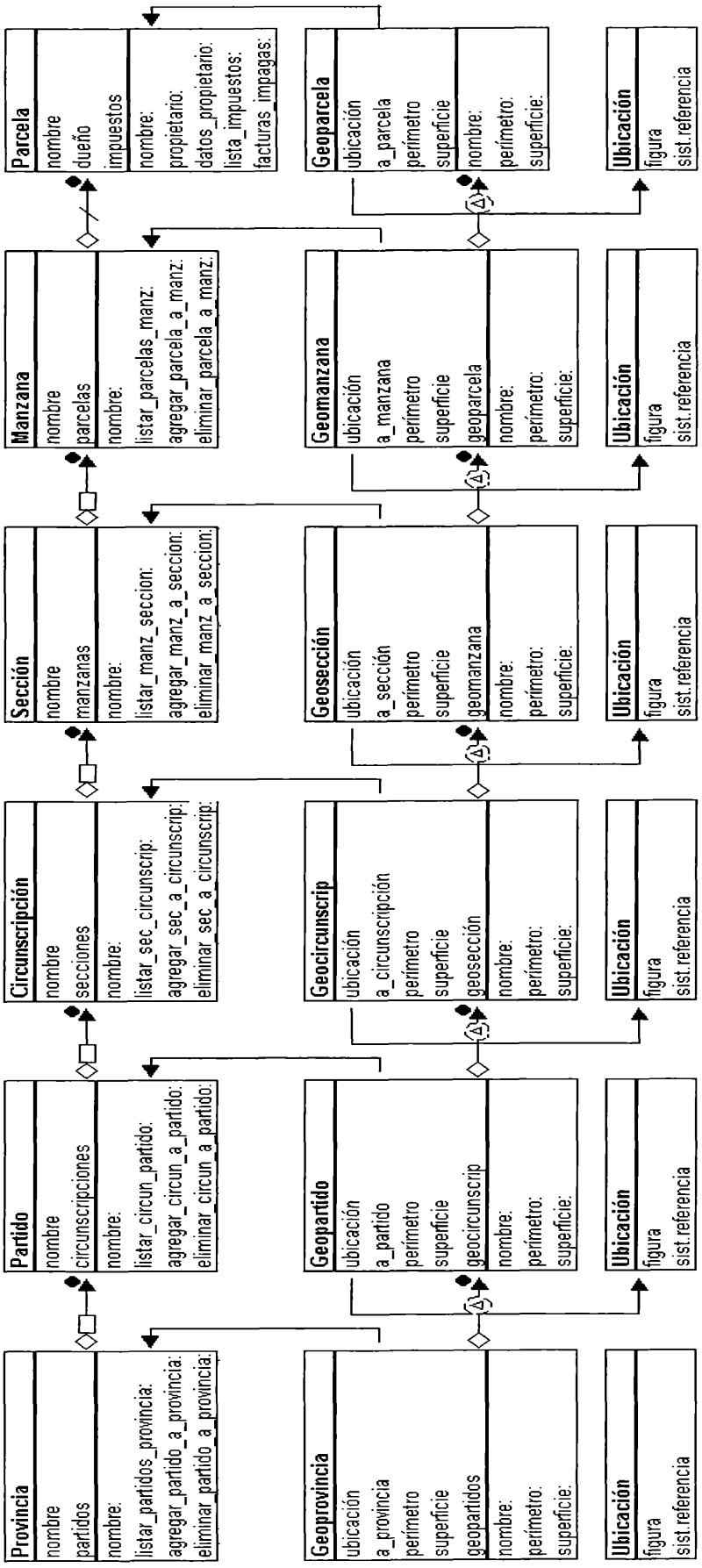


Figura 6.3: Modelo Geográfico

6.5 Definiendo nuevas relaciones Semánticas

En esta sección se definen nuevas relaciones semánticas parte_de entre objetos convencionales, como así también entre objetos georeferenciados, es decir nuevas relaciones para ser utilizadas en el diseño del modelo conceptual y el modelo geográfico. Lo que se pretende es salvar ambigüedades en las relaciones existentes, como por ejemplo en la relación "divided_into", no queda explícitamente definido el tiempo de vida del total, y en la relación "made_of" no queda explícito el tiempo de vida de las partes. Además, es de fundamental importancia que la semántica de cada relación contemple todas las restricciones: dependencia/independencia, exclusivo/compartido, cardinalidad y tiempo de vida.

Lo mismo ocurre en el modelo geográfico. Las semánticas de las relaciones de los objetos georeferenciados que se utilizan en este modelo, deben contemplar restricciones topológicas, dentro de estas restricciones analizamos las siguientes:

- **Inclusión** de las partes con respecto al total. Se analiza si las partes están incluidas completa o parcialmente en el total.
- **Disyunción** entre las partes. Se analiza si las partes se superponen o son disjuntas dentro de un total.
- **Cobertura** de las partes con respecto al total. Se analiza si la unión de las partes cubre el total parcial o completamente.

Notar que no deben existir relaciones topológicas entre objetos convencionales.

Relaciones del Modelo Conceptual

Tiene_un_compartido: A tiene_un_compartido B significa que una instancia de A puede tener una o más instancias de B como sus partes. Cada instancia de B puede ser parte de una instancia de A y posiblemente de otras instancias de A. El tiempo de vida de las instancias de A y B es independiente.

Ejemplo: Considerar que A es una provincia y B un río, una instancia de B podría ser parte de dos instancias de A distintas (un río puede ser parte de dos provincias).

Tiene_un_exclusivo: A tiene_un_exclusivo B significa que una instancia de A puede tener un o más instancias de B como sus partes, y cada instancia de B puede ser parte solamente de una instancia de A. El tiempo de vida de las instancias de A y B es independiente.

Ejemplo: A es una ciudad y B una plaza, una instancia de B puede ser parte solamente de una instancia de A.

Hecho_por_compartido: A Hecho_por_compartido B si para cada instancia de A existe al menos una instancia de B, y cada instancia de B puede ser parte de otras instancias de A. El tiempo de vida de las instancias de A terminan en el momento que ellas no tienen instancias de B y el tiempo de vida de B es independiente.

Ejemplo: A es una provincia y B es un límite político. Para que la provincia exista tiene que haber al menos un límite, y el límite puede pertencer a varias provincias.

Hecho_por_exclusivo: A Hecho_por_exclusivo B si para cada instancia de A existe al menos una instancia de B, y cada instancia de B puede ser parte solamente de una instancia de A. El tiempo de vida de las instancias de A terminan en el momento que ellas no tienen instancias de B y el tiempo de vida de B es independiente.

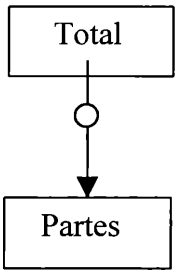
Ejemplo: A es una cadena montañosa y B es una montaña. Para que A exista tiene que existir al menos una instancia de B, y cada instancia de B puede ser parte solamente de una instancia de A.

Dividido_en_compartido: A dividido_en_compartido B si cada instancia de B está asociada con una instancia de A, si no la instancia de B no existe, y además B puede ser parte de otras instancias de A. El tiempo de vida de las instancias de B está limitado por su relación con una instancia de A y todas las instancias de B forman el total. El tiempo de vida de A es independiente de las instancias de B.

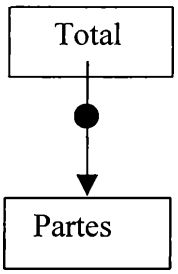
Dividido_en_exclusivo: A dividido_en_exclusivo B si cada instancia de B está asociada con una instancia de A, si no la instancia de B no existe, y además B puede ser parte solamente de una instancia de A. El tiempo de vida de las instancias de B está limitado por su relación con una instancia de A y todas las instancias de B forman el total. El tiempo de vida de A es independiente de las instancias de B.

Ejemplo: A es un país y B una provincia. Una instancia de B puede ser parte solamente de una instancia de A. Una instancia de B no puede existir sin estar asociada a una instancia de A.

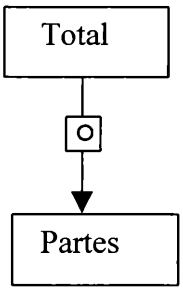
Para representar todas estas relaciones de composición , definimos una extensión de la notación para primitivas OMT [Rumbaugh et al.91], para distinguir cada una ellas.



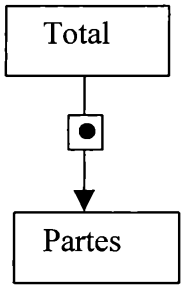
Tiene_un_compartido



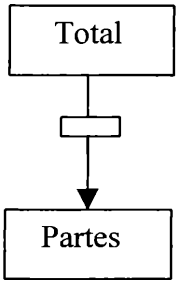
Tiene_un_exclusivo



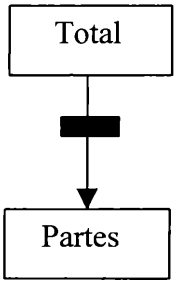
Hecho_por_compartido



Hecho_por_exclusivo



Dividido_en_ compartido

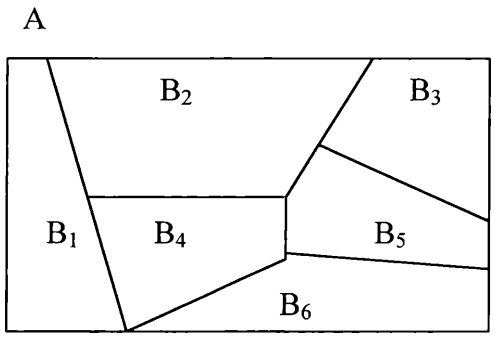


Dividido_en_exclusivo

Relaciones en el Modelo Geográfico

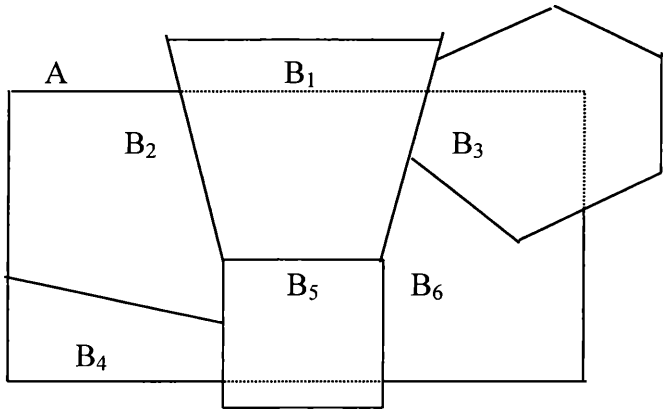
Cubierto_Total: La geometría del total es cubierta totalmente por la unión de las geometrías de las partes, las geometrías de las partes están totalmente incluidas en el total y las geometrías de las partes son disjuntas.

Ejemplo: Considerar un país (A) como el total y sus provincias (B_i) como sus partes. La unión de las provincias cubren el país y ninguna provincia se superpone con otra, es decir, son disjuntas.



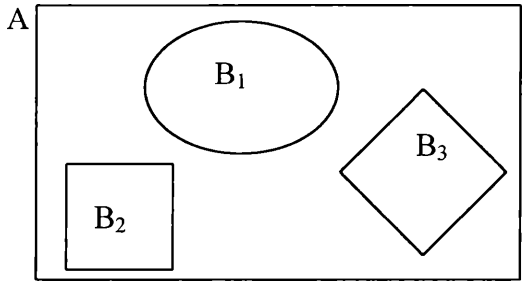
Cubierto_Parcial: La geometría del total es cubierta totalmente por la unión de las geometrías de las partes, las geometrías de las partes pueden estar incluidas parcialmente en la geometría del total y las geometrías de las partes son disjuntas.

Ejemplo: Considerar una provincia (A) como el total y un grupo de llanuras (B_i) como sus partes. La unión de las llanuras podría cubrir toda la provincia y ellas no estar totalmente incluidas dentro de la provincia, es decir estar incluidas parcialmente.



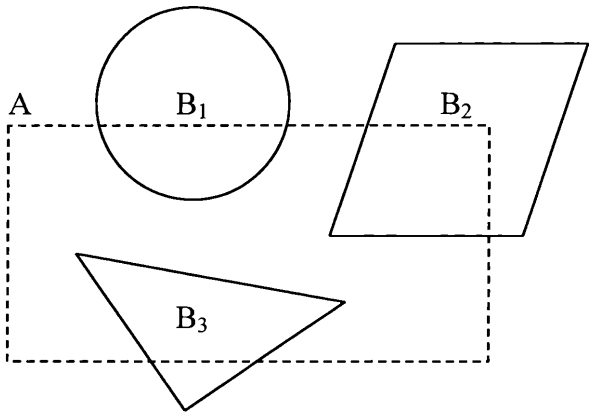
Contiene_Total: La geometría del total es cubierta parcialmente por las geometrías de sus partes, las geometrías de las partes están incluidas totalmente en la geometría del total y las geometrías de las partes son disjuntas.

Ejemplo: Considerar una ciudad (A) como el total y sus plazas (B_i) como sus partes. Las plazas están contenidas en la ciudad, pero no la cubren por completo, además no se superponen entre ellas y cada plaza está completamente incluida en la ciudad.

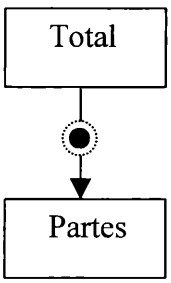


Contiene_Parcial: La geometría del total es cubierta parcialmente por las geometrías de sus partes, las geometrías de las partes pueden estar incluidas parcialmente en la geometría del total y las geometrías de las partes son disjuntas.

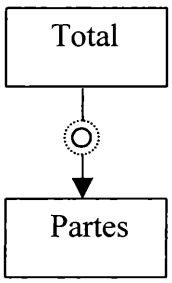
Ejemplo: Considerar una provincia (A) como el total y sus ríos (B_i) como las partes. Los ríos pueden atravesar otras provincias, es decir pueden no estar totalmente incluidos en una provincia.



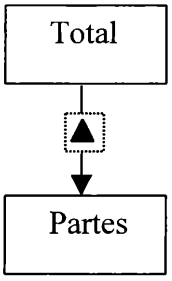
Notación Gráfica



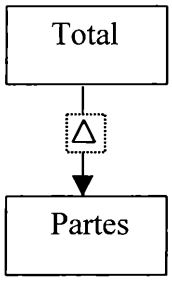
Cubierto_Total



Cubierto_Parcial



Contiene_Total



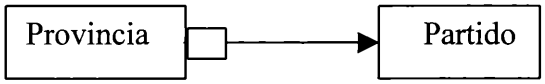
Contiene_Parcial

Representando las nuevas relaciones para SIG

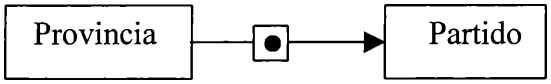
Una vez definidas las nuevas relaciones, ejemplificaremos su aplicación en el problema de recaudación impositiva tratado anteriormente:

Modelo Conceptual

- La relación entre provincia y partido, fue definida como "Made_of":

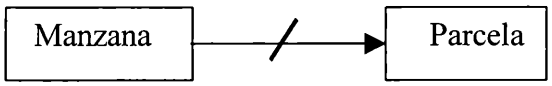


Ahora la podemos definir como "Hecho_por_exclusivo".

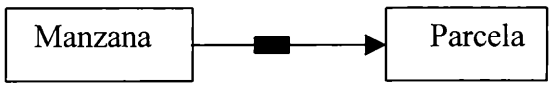


Con la nueva relación definida, se agrega la restricción de exclusividad, es decir, un partido sólo puede ser parte de una provincia. Lo mismo sucede con las relaciones entre partido y circunscripción, circunscripción y sección y sección y manzana

- La relación entre manzana y parcela fue definida como " Divided_into" :



Ahora la podemos definir como "Dividido_en_exclusivo".

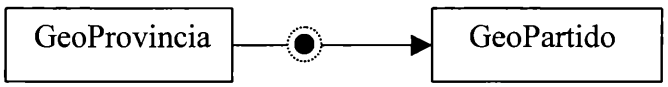


Con la nueva relación definida, queda explícito que el tiempo de vida del total es independiente del tiempo de vida de las partes, es decir una manzana puede existir aunque no esté dividida en parcelas. Además se agrega la restricción de cardinalidad.

Modelo Geográfico

- La relación entre Geoprovincia y Geopartido fue definida como "Covering".

Ahora la definimos como "Cubierto_Total"



Con la nueva relación definida, se agrega la restricción de disyunción entre las partes, además de que las partes estén totalmente incluidas en el total. Lo mismo sucede con las relaciones entre Geopartido y Geocircunscripción, Geocircunscripción y Geosección, Geosección y Geomanzana y Geomanzana y Geoparcela.

6.6 Restricciones Geométricas

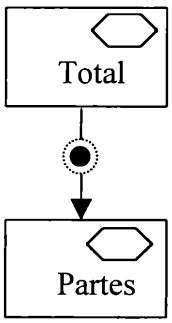
Podríamos refinar aún más las relaciones definidas para el modelo geográfico, agregándoles restricciones geométricas. Como estamos tratando relaciones entre geoobjetos, es decir, objetos con características espaciales, sería conveniente saber el "tipo" de la clase espacial del total y de sus partes (por ejemplo, si son Líneas, Puntos o Polígonos)

Para todas las relaciones definidas para el Modelo Geográfico, analizaremos de qué "tipo" debe ser el total y de qué "tipo" deben ser sus partes.

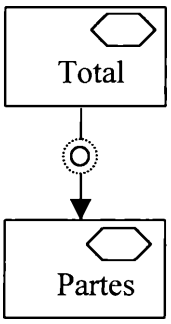
- Para las relaciones *Cubierto_Total* y *Cubierto_Parcial*, como las partes deben cubrir completamente el total, ambos deben ser polígonos.
- Para la relación *Contiene_Total*, como el total puede ser cubierto parcialmente y las partes pueden estar totalmente incluidas, el total puede ser polígono y línea y las partes pueden ser punto, línea y polígono.

- Para la relación *Contiene_Parcial,* como el total puede ser cubierto parcialmente por las partes y éstas pueden estar parcialmente incluidas, el total puede ser polígono y línea y las partes pueden ser punto, línea y polígono.

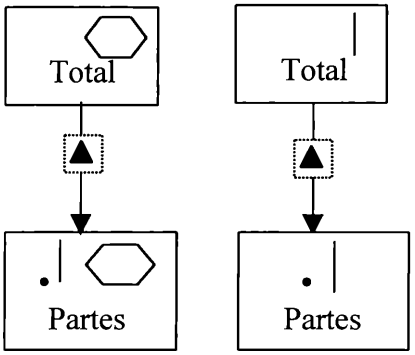
Notación Gráfica



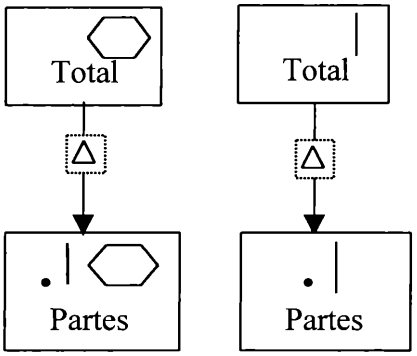
Cubierto_Total



Cubierto_Parcial



Contiene_Total



Contiene_Parcial

6.7 Equivalencias entre el Modelo Conceptual y el Modelo Geográfico

Analizamos si existen equivalencias entre las relaciones definidas en el Modelo Conceptual y el Modelo Geográfico. Si dos clases A y B son decoradas con clases A' y B' respectivamente, y existe una relación entre instancias de clases de A y B, entonces podemos deducir que existe una relación equivalente entre A' y B':

- Si la relación entre A y B es *Tiene_un_Compartido*, entonces la relación entre A' y B' será *Cubierto_Parcial* o *Contiene_Parcial.*

  No podemos afirmar que la relación entre A'y B' sea Cubierto_Total o Contiene_Total, ya que en la relación Tiene_un_compartido existe la restricción de que cada instancia de B puede ser parte de una instancia de A y posiblemente de

otras instancias de A. En cambio en las relaciones Contiene_total y Cubierto_total, las geometrías de las partes están totalmente incluidas en la geometría del total, entonces cada instancia de B' será parte de sólo una instancia de A'.

- Si la relación entre A y B es *Tiene_un_Exclusivo*, entonces la relación entre A' y B' será *Cubierto_Total* y *Contiene_Total*.

No podemos afirmar que la relación entre A' y B' sea Cubierto_Parcial o Contiene_Parcial, ya que en la relación Tiene_un_exclusivo existe la restricción e que cada instancia de B puede ser parte solamente de una instancia de A. En cambio en las relaciones Cubierto_parcial y Contiene_parcial, las geometrías de las partes pueden estar incluidas parcialmente en la geometría del total (una parte puede pertenecer a varios totales), entonces cada instancia de B' puede ser parte de varias instancias de A'.

- Si la relación entre A y B es *Dividido_en_Exclusivo*, entonces la relación entre A' y B' será *Cubierto_Total*.

No podemos afirmar que la relación entre A' y B' sea Cubierto_parcial o Contiene_parcial, ya que en la relación Dividido_un_exclusivo existe la restricción de que cada instancia de B puede ser parte solamente de una instancia de A. En cambio en las relaciones Cubierto_parcial y Contiene_parcial, las geometrías de las partes pueden estar incluidas parcialmente en la geometría del total (una parte puede pertenecer a varios totales), entonces cada instancia de B' puede ser parte de varias instancias de A'. Tampoco podemos afirmar que la relación entre A' y B' es Contiene_Total, ya que la relación Dividido_en_exclusivo, también tiene la restricción de que todas las instancias de B forman el total de A. En cambio en la relación Contiene_total la geometría del total es cubierta parcialmente por las geometrías de las partes.

- Si la relación entre A y B es *Hecho_por_Exclusivo*, entonces la relación entre A' y B' será *Cubierto_Total* y *Contiene_Total*.

No podemos afirmar que la relación entre A' y B' sea Cubierto_parcial o Contiene_parcial, ya que en la relación Hecho_por_exclusivo existe la restricción de que cada instancia de B puede ser parte solamente de una instancia de A. En cambio

en las relaciones Cubierto_parcial y Contiene_parcial, las geometrías de las partes pueden estar incluidas parcialmente en la geometría del total (una parte puede ser parte de varios totales), entonces cada instancia de B' puede ser parte de varias instancias de A'.

- Si la relación entre A y B es Hecho_*por_Compartido*, entonces la relación entre A' y B' será *Cubierto_Parcial* o *Contiene_Parcial*.

  No podemos afirmar que la relación entre A' y B' sea Contiene_total o Cubierto_total, ya que en la relación Hecho_por_compartido existe la restricción de que cada instancia de B puede ser parte de una instancia de A y posiblemente de otras instancias de A. En cambio en las relaciones Contiene_total y Cubierto_total, las geometrías de las partes están totalmente incluidas en la geometría del total, entonces cada instancia de B' será parte de solo una instancia de A'.

- Si la relación entre A y B es *Dividido_en_compartido*, entonces la relación entre A' y B' será *Cubierto_Parcial*.

  No podemos afirmar que la relación entre A' y B' sea Cubierto_total o Contiene_total, ya que en la relación Dividido_un_compartido, existe la restricción de que cada instancia de B puede ser parte de una instancia de A y posiblemente de otras instancias de A. En cambio en las relaciones Contiene_total y Cubierto_total, las geometrías de las partes están totalmente incluidas en la geometría del total, entonces cada instancia de B' será parte de solo una instancia de A'. Tampoco podemos afirmar que la relación entre A' y B' es Contiene_parcial, ya que la relación Dividido_en_compartido también tiene la restricción de que todas las instancias de B forman el total de A. En cambio en la relación Contiene_parcial la geometría del total es cubierta parcialmente por las geometrías de las partes.

6.8 Diseño de una aplicación SIG

En esta sección se mostrará como utilizar las relaciones semánticas parte_de definidas, en una aplicación SIG, encontrada en proyectos de la vida real.

Por ejemplo supongamos que se quiere desarrollar una aplicación para representar la hidrografía y el relieve de una país. En la parte hidrográfica se considera los ríos y lagos, y en el relieve se consideran las cadenas montañosas, las llanuras y mesetas.

Se tiene en cuenta:

- El país está compuesto por provincias.
- Un río puede atravesar varias provincias.
- Un lago pertenece a una provincia solamente.
- Una cadena montañosa está compuesta por al menos una montaña.
- Una cadena montañosa puede atravesar varias provincias.
- Una llanura puede atravesar varias provincias.
- Una meseta puede atravesar varias provincias.

6.8.1 Modelo Conceptual

6.8.1.1 Diagrama de clases y relaciones:

La figura 6.4 muestra el diagrama de clases y relaciones de la aplicación que estamos considerando, utilizando la notación OMT [Rumbaugh et al.91]. Para la representación de las relaciones de composición utlizamos la notación adicional de las primitivas OMT explicadas en la sección 6.4

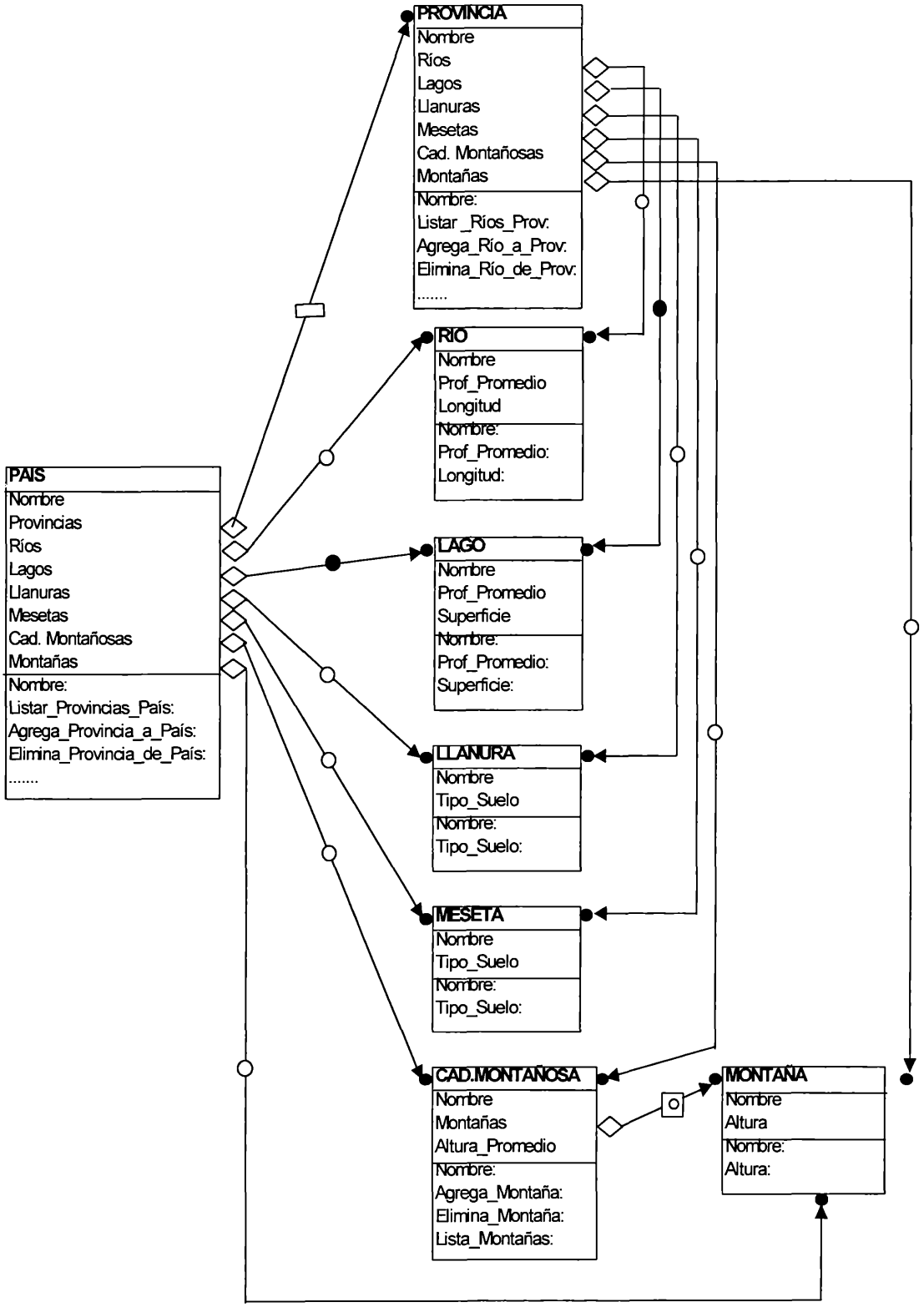


Figura 6.4: Modelo Conceptual

6.8.1.2 Descripción de las Operaciones

Clase: PAIS

Descripción: Esta clase representa un país con las provincias que lo forman y algunos aspectos hidrológicos y de relieve. Un PAIS es una estructura que contiene provincias, ríos, lagos, mesetas, llanuras, cadenas montañosas y montañas.

Nombre: Devuelve el nombre del país.

Listar_Provincias_País(L_Prov): Devuelve en la lista L_Prov los nombres de las provincias que forman parte del país.

Agrega_Provincia_a_País(Prov): Agrega la provincia Prov al país.

Elimina_Provincia _de_País(Prov): Elimina la provincia Prov del país.

Listar_Ríos_País (L_Río): Devuelve en la lista L_Río los nombres de los ríos que forman parte del país.

Agrega_Río_a_País(aRío): Agrega el río aRío al país.

Elimina_Río_de_País (aRío): Elimina el río aRío del país.

Se repiten las tres últimas operaciones para lago, llanura, meseta, cadena montañosa y montaña.

Clase: PROVINCIA

Descripción: Esta clase representa algunos aspectos hidrológicos y de relieve de una provincia. Una Provincia es una estructura que contiene ríos, lagos, mesetas, llanuras, cadenas montañosas y montañas.

Nombre: Devuelve el nombre de una provincia.

Listar_Ríos_Prov(L_Río): Devuelve en la lista L_Río los nombres de los ríos que atraviesan la provincia.

Agrega_Río_a_Prov(a_Río): Agrega el río a_Río a la provincia.

Elimina_Río_de_Prov(a_Río): Elimina el río a_Río de la provincia.

Se repiten las tres últimas operaciones para lago, llanura, meseta, cadena montañosa y montaña.

Clase: RIO

Descripción: Esta clase representa un río con sus características.

Nombre: Devuelve el nombre del río.

Prof_Promedio: Devuelve la profundidad promedio del río.

Longitud: Devuelve la longitud del río.

Clase: LAGO

Descripción: Esta clase representa un lago con sus características.

Nombre: Devuelve el nombre del lago.

Prof_Promedio: Devuelve la profundidad promedio del lago.

Superficie: Devuelve la superficie del lago.

Clase: LLANURA

Descripción: Esta clase representa un llanura con sus características.

Nombre: Devuelve el nombre de la llanura.

Tipo_suelo: Devuelve el tipo de suelo de la llanura.

Clase: MESETA

Descripción: Esta clase representa una meseta con sus características.

Nombre: Devuelve el nombre de la lmeseta.

Tipo_suelo: Devuelve el tipo de suelo de la meseta.

Clase: CADENA MONTAÑOSA

Descripción: Esta clase representa una CADENA MONTAÑOSA. Una CADENA MONTAÑOSA es una estructura que contiene las montañas que la forman.

Nombre: Devuelve el nombre de la cadena montañosa.

Agrega_Montaña (a_Mont): Agrega la montaña a_Mont a la cadena montañosa.

Elimina_Montaña (a_Mont): Elimina la montaña a_Mont a la cadena montañosa.

Lista_Montañas (L_Mont): Devuelve en la lista L_Mont los nombres de las montañas que forman la cadena montañosa

Clase: MONTAÑA

Descripción: Esta clase representa una montaña con sus características.

Nombre: Devuelve el nombre de la montaña.

Altura: Devuelve la altura de la montaña.

6.8.1.3 Descripción de las Relaciones semánticas

- La relación entre país y provincia es *Dividido_en_Exclusivo*, ya que una provincia debe ser parte de sólo un país y una provincia no puede existir sin estar asociada a un país.
- La relación entre país y río, país y llanura, país y meseta, país y cadena montañosa, y país y montaña es *Tiene_un_compartido*, ya que un río, una llanura, una meseta, una cadena montañosa y una montaña pueden atravesar varios países, un río existe aunque el país se disuelva, administrativamente por ejemplo, y un país puede seguir existiendo aunque el río se seque.
- La relación entre país y lago es *Tiene_un_exclusivo*, ya que, por la definición del problema, un lago puede pertenecer sólo a una provincia, y como una provincia puede ser parte de sólo un país, entonces un lago puede ser parte de sólo un país.
- La relación entre cadena montañosa y montaña es *Hecho_por_exclusivo*, ya que para que una cadena montañosa exista debe existir al menos una montaña que la forme, y una montaña puede ser parte solamente de una cadena montañosa.
- La relación entre provincia y río, provincia y llanura, provincia y meseta, provincia y cadena montañosa, y provincia y montaña es *Tiene_un_compartido*.
- La relación entre provincia y lago es T*iene_un_exclusivo*, ya que un lago puede pertenecer sólo a una provincia.

6.8.2 Modelo Geográfico

6.8.2.1 Diagrama de clases y relaciones:

La figura 6.5 muestra el diagrama de clases y relaciones de la aplicación considerada, utilizando la notación OMT [Rumbaugh et al.91]. Para la representación de las relaciones de composición utlizamos la notación adicional de las primitivas OMT explicadas en la sección 6.4.

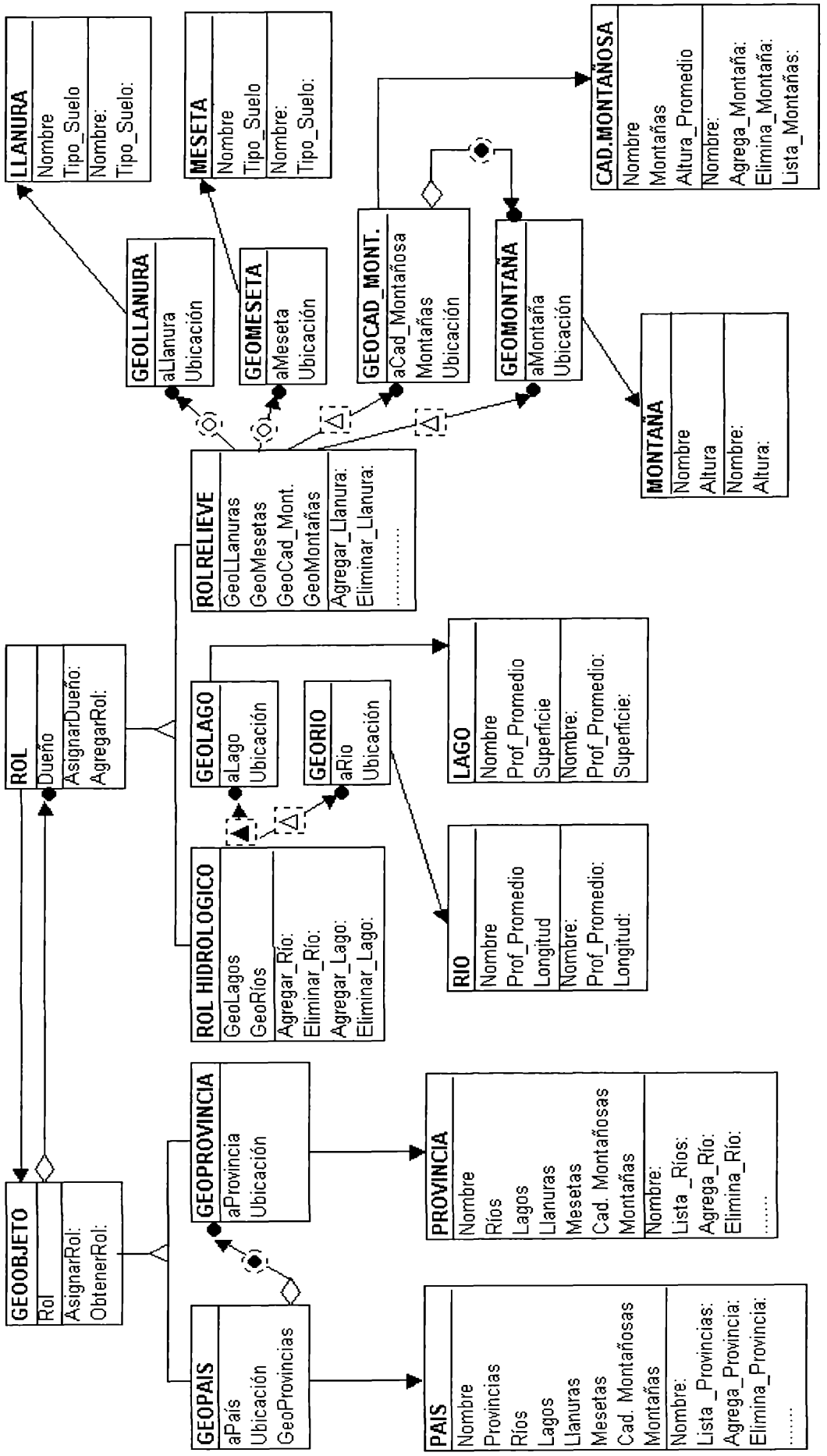


Figura 6.5: Modelo Geográfico

6.8.2.2 Descripción de las Relaciones semánticas

- La relación entre Geopaís y Geoprovincia es *Cubierto_Total,* ya que la unión de las Geoprovincias cubren al Geopaís y las Geoprovincias son disjuntas.
- La relación entre Geopaís y Geolago es *Contiene_Total*, ya que los Geolagos están contenidos completamente en los Geopaíses, pero no lo cubren totalmente y además los Geolagos son disjuntos.
- La relación entre Geopaís y Georío es *Contiene_Parcial,* ya que los Georíos pueden ser parte de varios Geopaíses.
- La relación entre Geopaís y Geollanura es *Cubierto_Parcial*, ya que la unión de las Geollanuras cubren todo el Geopaís y pueden no estar totalmente incluidas en el Geopaís.
- La relación entre Geopaís y Geomeseta es *Cubierto_Parcial,* ya que la unión de las Geomesetas cubren todo el Geopaís y pueden no estar totalmente incluidas en el Geopaís.
- La relación entre Geopaís y Geocad_mont es *Contiene_Parcial,* ya que las Geocad_mont pueden ser parte de varios Geopaíses.
- La relación entre Geopaís y Geomontaña es *Contiene_Parcial* ya que las Geomontaña pueden ser parte de varios Geopaíses.
- La relación entre Geocad_mont y Geomontaña es *Cubierto_Total,* ya que la unión de las Geomontañas cubren a la Geocad_mont y las Geomontañas son disjuntas.
- La relación entre Geoprovincia y Geolago es *Contiene_Total* , ya que los Geolagos están contenidos completamente en las Geoprovincias, per no lo cubren totalmente y además los Geolagos son disjuntos.
- La relación entre Geoprovincia y Georío es *Contiene_Parcial* ya que los Georíos pueden ser parte de varias Geoprovincias.
- La relación entre Geoprovincia y Geollanura es *Cubierto_Parcial* ya que la unión de las Geollanuras cubren toda la Geoprovincia y pueden no estar totalmente incluidas en la Geoprovincia.
- La relación entre Geoprovincia y Geomeseta es *Cubierto_Parcial* ya que la unión de las Geomesetas cubren toda la Geoprovincia y pueden no estar totalmente incluidas

en las Geoprovincias.

- La relación entre Geoprovincia y Geocad_mont es *Contiene_Parcial* ya que las Geocad_mont pueden ser parte de varias Geoprovincias.
- La relación entre Geoprovincia y Geomontaña es *Contiene_Parcial* ya que las Geomontaña pueden ser parte de varias Geoprovincias.

7. CONCLUSIONES

En esta tesis estudiamos los conceptos fundamentales de los SIGs y en particular los modelos de diseño orientados a objetos para representar este tipo de aplicaciones. A partir del modelo de diseño definido en el capítulo 4, analizamos la importancia de las relaciones parte_de para representar restricciones semánticas en la aplicación. Estas relaciones resultan de gran importancia, tanto desde el punto de vista del diseño como de la implementación.

El modelo utilizado define dos niveles de desarrollo: el nivel conceptual, donde se definen las clases sin tener en cuenta las características espaciales; y el nivel geográfico, en el cual se decoran los objetos del nivel anterior con geoobjetos, que encapsulan la información geográfica.

El aspecto fundamental de esta tesis es definir relaciones semánticas parte_de para ambos niveles de diseño, teniendo en cuenta en cada caso las semánticas correspondientes. En el diseño de las relaciones definidas para el modelo conceptual, se tuvieron en cuenta restricciones de tiempo de vida, cardinalidad, exclusividad y dependencia entre los objetos; para las del modelo geográfico, en cambio, se tuvieron en cuenta restricciones topológicas de inclusión, disyunción y cobertura entre los geoobjetos.

Luego de haber definido las relaciones en los dos modelos, se buscó una equivalencia entre ambos, es decir, para cada una de las relaciones del modelo conceptual ,existe al menos una relación equivalente para los geoobjetos en el modelo geográfico. Este estudio se realizó de manera de poder establecer en el futuro conversiones automáticas de un modelo a otro basándose en un conjunto de restricciones definidas a partir de esta transformación. Es importante destacar, que si bien se han estudiado y definido diferentes semánticas para este tipo de relaciones utilizando el modelo orientado a objetos y que existen algunas definiciones para aplicaciones geográficas específicas, nunca se ha estudiado cuál es el resultado cuando se debe combinar la utilización de relaciones parte_de en el nivel conceptual y en el geográfico.

Como hemos visto, el análisis debe ser realizado desde puntos de vista muy diferentes, ya que los elementos a tener en cuenta en uno y otro nivel (tiempo de vida,

exclusividad/dependencia, etc. en el nivel conceptual; geometría en el nivel geográfico) son diferentes y, por lo tanto, es necesario establecer una compatibilidad entre ambos.

Como consecuencia de lo anterior, encontramos que las definiciones existentes no eran suficientes para establecer esta compatibilidad. Es por ello que se refinaron algunas definiciones que fueron tomadas como base para lograr el objetivo.

8. BIBLIOGRAFIA

[Alexander et al.77] K. Alexander, S. Ishikama, M. Silverstein, "A Pattern Language", Oxford University Press, 1977.

[Aronoff91] S. Aronoff, "Geographic Information System: A Management Perspective",WDL Publications, Otawa, canada, ISBN:0-921804-91.

[Balaguer et al.97] F. Balaguer, S. Gordillo, F. Das Neves: "Patterns for GIS Applications Design". In the proceedings of Patterns Language of Programming 1997.

[Fowler97] Fowler, M: "Patterns: Reusable Object Model". Addison Wesley, 1997.

[Gamma et al.94] E. Gamma, R. Helm, R Johnson, J. Vlissides, "Design Patterns. Elements of reusable Object Oriented Sofware", Addison Wesley Professional Computing Series, 1994.

[Gordillo et al.97] S. Gordillo, F. Balaguer, F. Das Neves, C. Mostaccio: "Developing GIS with Objects. A Desing Patterns Approach". ACM GIS'97. Advances in Geographic Information Systems, 1997

[Johnson et al.97] Jhonson Ralph and Wool Bobby. " The Type Object" . Pattern Languages of Program Desing. Ed by Robert Martin, Dirk Riehle, Frank Buschmann, 1997.

[Kim90]Won Kim "Introduction to Object-Oriented Database". Ed. MIT Press, 1990.

[Kosters et al.95] G. Kosters, B. Pagel, H. Six, "Object-Oriented Requeriments Engineering for Gis Aplications", in proceedings of ACM GIS'95: 3th. International Workshop on Advances in Geographic Information Systems, R. De Laurini, P. Bergougnoux K. Makkiand, N. Pissinou, november1995, PP 61-68.

[Medeiros94 et al.] C Medeiros, M. A. Casanova and G. Camara: "The Domus project. Building an OODB GIS for environmental control"Proceedings of IGIS'94. International Worshop on Advanced Research in GIS, Spinger Verlag LNCS, N. 884, pp 45-54.

[Open GIS] Consortium (OGC), 1996B, The Open Gis Guide-A Guide to Interoperable Geo-processing, Available at http://ogis.or/guide/guide1.htm.

[Postmesil97] Postmesil M:"Maps Alive:Viewing Geospacial Information on the WWW" Proceedings of the six International Worl Wide Web Conference, 1997.

Available at http: //www6.nttlabs.com/Hypernews/get/PAPER130.htm.

[Rumbaugh et al.91] Rumbaugh, M. Blaha, M.Premerlani and W. Lorensen: " Object-Oriented Modeling and Desing" .Prentice Hall, Englewoods Cliff, New Jersey, 1991.

[Tryfona et al.95] Tryfona N. and Hadzilacos T., "Geographic Applications Development: Models and Tools for the Conceptual level", in Proc of the 3rd ACM International Workshop on Advances in Geographic Information Systems, PP 10-28, 1995.