UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

TÍTULO: Desarrollo de un Motor de Manipulación de Pirámides Vectoriales para GeoQ.

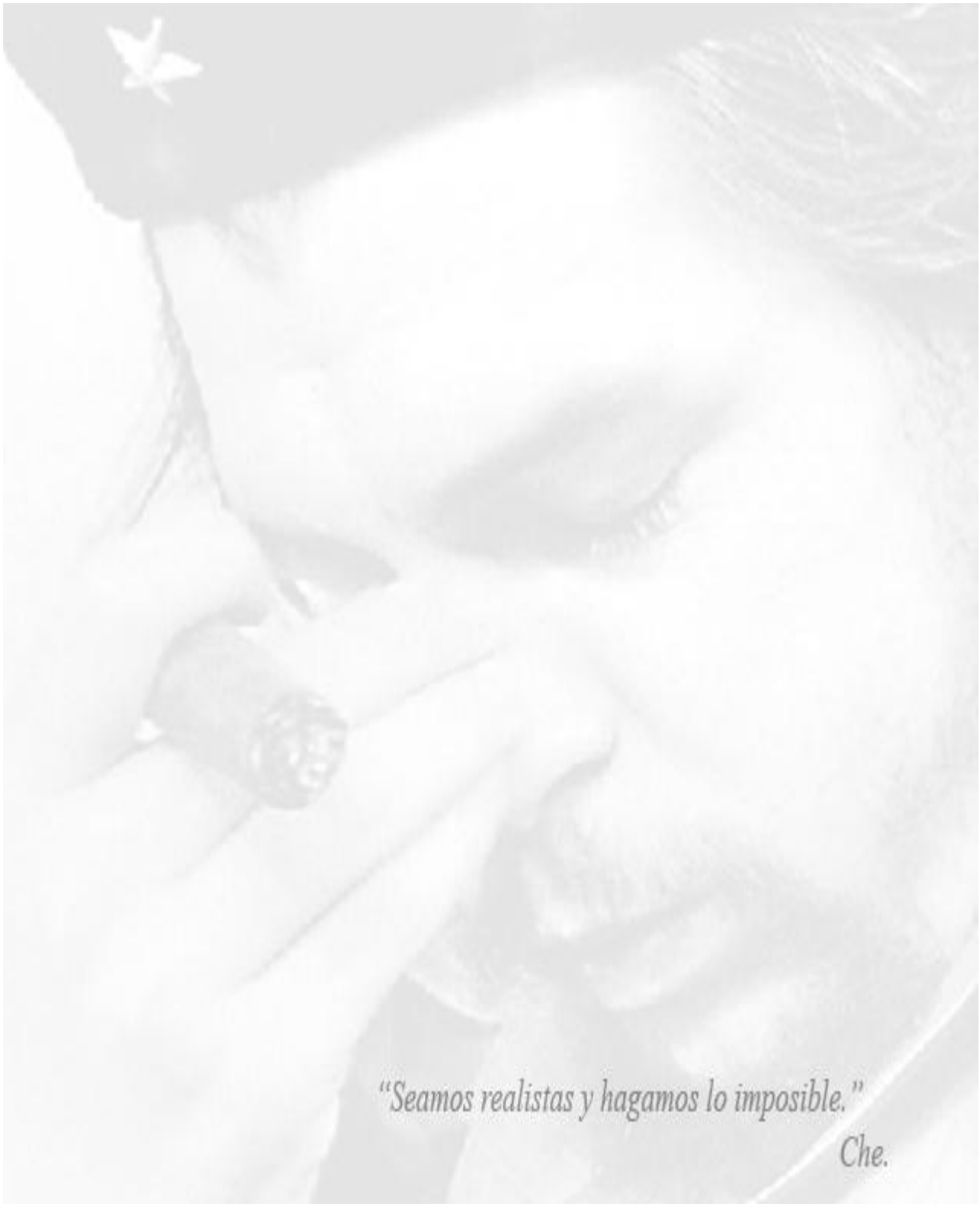
AUTOR: Alexander Pacheco Armas

TUTOR: Ing. Lewis Rodríguez Fuentes

CO-TUTOR: Msc. David Silva Barrera

La Habana, Junio 2012.

Año 54 de la Revolución.


"Seamos realistas y hagamos lo imposible."
Che.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Alexander Pacheco Armas

_____

Lewis Rodríguez Fuentes

_____

DATOS DE CONTACTO

Tutor: Ing. Lewis Rodríguez Fuentes.

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Categoría docente: -

Categoría Científica: -

Años de experiencia en el tema: 1 año.

Año de graduado: 2011.

Correo electrónico: lrfuentes@uci.cu

Co-Tutor: Msc. David Silva Barrera.

Especialidad de graduación: Licenciado en Ciencias de la Computación.

Categoría docente: Profesor Asistente.

Categoría Científica: Máster en Ciencias.

Años de experiencia en el tema: 8 años.

Año de graduado: 2004.

Correo electrónico: dsilva@uci.cu

Agradezco a mis padres, familiares y amigos por todo el apoyo bridado durante la carrera.

A mi Abuelo Nivardo, por el imborrable ejemplo, por mostrarme el camino correcto, por siempre tener una sonrisa en tu rostro y un abrazo listo para cuando lo necesitara. Por haber creado y educado de la mejor manera posible a nuestra familia. Te estaré eternamente agradecido por haber creado este mundo maravilloso, lleno de comprensión y cariño en el que nací. Gracias por cuidarme y estar siempre a mi lado, incluso cuando ya no estás.

Los sistemas de información geográfica constituyen una ciencia relativamente joven, lograr perfeccionarlos y adaptarlos a las más disímiles necesidades es una tarea de todos los sectores de la sociedad, debido al elevado número de servicios que a la misma prestan. Con la presente investigación se pretende incorporar a GeoQ una funcionalidad que permita reducir el tiempo de respuesta a la hora de realizar operaciones con archivos vectoriales de grandes dimensiones. Esta personalización dará al sistema la posibilidad de ser aceptado en el mercado ya que la demora en la manipulación de estos datos es uno de los principales problemas que presenta actualmente.

El documento que se presenta coleccionará los resultados de la investigación desarrollada. Se identificará y analizará exhaustivamente la situación problemática y el dominio donde ocurren los principales procesos del negocio. Se realiza un análisis de las diferentes herramientas y tecnologías actuales candidatas para el modelado y desarrollo del sistema. Posteriormente se enumeran las distintas funcionalidades que debe poseer el sistema y se realiza su diseño para determinar cómo quedará estructurada para su mejor desarrollo, se realiza el diseño del sistema propuesto basado en las funcionalidades planteadas anteriormente, se modela su implementación y finalmente se procede con la construcción del sistema propuesto.

PALABRAS CLAVES: Ingeniería, Modelo, Pirámide, Quantum GIS, SIG

INTRODUCCIÓN

Cada día la informática adquiere más relevancia en la vida de las personas y empresas, pues esta ciencia se aplica a numerosas y variadas áreas como la gestión de negocios, almacenamiento y consulta de información, monitorización y control de procesos, por solo mencionar algunos. Este desarrollo informático le ha dado a la humanidad la posibilidad de solucionar problemas que existen desde hace muchos años. Un sustancial ejemplo de estos problemas fue la gran cantidad de datos que era necesario manipular para la representación de los mapas, coordenadas, direcciones, etc. Con el tiempo se hizo necesario recopilarlos, y una vez recopilados, crear con todos estos un conjunto de símbolos organizados de manera lógica. Con todo ello surge entonces la cartografía, una de las más útiles y antiguas ciencias que ha creado y dominado el hombre, pues es la encargada del estudio y la elaboración de los mapas geográficos.

A medida que se fue desarrollando la cartografía, los mapas se hicieron más complejos a causa de la gran cantidad de símbolos y datos que encerraban en sus formatos, convirtiéndose en un obstáculo para aquellos que no eran expertos en la manipulación de estos. Es entonces cuando aflora la necesidad de agilizar actividades, mejorar la forma de vida y transformar el entorno circundante, dando lugar a la creación de herramientas con características capaces de humanizar los procesos. En ello interviene de manera muy eficaz el uso de Las Tecnologías de la Información y las Comunicaciones (TIC). “Se puede denominar como TIC al conjunto de procesos y productos derivados de las nuevas herramientas (hardware y software), soportes de la información y canales de comunicación relacionados con el almacenamiento, procesamiento y transmisión digitalizados de la información” **(Giraldo Ocampo, 2011).**

Las TIC son tecnologías de gestión e innovación que se basan en sistemas o productos que son capaces de captar información multidimensional, almacenarla, elaborarla, tomar decisiones, transmitirlas, difundirlas y hacerlas inteligibles, accesibles y aplicables en correspondencia con el fenómeno a transformar. Su singularidad radica en la constante innovación que posibilitan y su mayor capacidad de tratamiento de la información.

Dentro del continuo e incesante progreso de las TIC se han ido perfeccionando los Sistemas de Información Geográfica (SIG), cuya necesidad se evidencia desde que el hombre primitivo pintaba en las paredes de las cuevas los animales que cazaba, asociando estos dibujos con trazos lineales. Con el transcurso del tiempo y de acuerdo a las necesidades del hombre, estos sistemas de comunicación fueron cambiando y desarrollándose cada vez más rápido. En 1962 se observó claramente la primera

utilización real de los SIG en el mundo, con el nombre de Sistema de Información Geográfica de Canadá. Este tenía implementadas funcionalidades como superponer capas de información, realizar mediciones y llevar a cabo digitalizaciones y escaneos de datos.

Según David Rhind, "un SIG es un sistema de hardware, software y procedimientos, diseñados para soportar la captura, manejo, manipulación, análisis, modelado y despliegue de datos espacialmente referenciados (geo-referenciados), para la solución de problemas complejos del manejo y planeamiento territorial" **(X Conferencia Internacional sobre SIG, 2005).**

Los SIG han dado una perspectiva totalmente nueva y dinámica de la información, además del importante papel que asumen en la toma de decisiones. Estos tienen la capacidad de capturar información almacenada en planillas o bases de datos que tenga un componente geográfico que permita ver patrones, relaciones y tendencias, que no pueden percibirse en un formato de tabla o lista.

En el mundo existen muchos países con un elevado desarrollo respecto a estos sistemas. En América los líderes son Canadá y EEUU, que cuentan con una gran variedad de aplicaciones de este tipo y con resultados probados. Por la otra parte del mundo se encuentran países asiáticos como La India mientras que Alemania, Rusia y España llevan la punta por Europa.

Internacionalmente se han creado sistemas con excelentes potencialidades; como ejemplo de esto se encuentra el ArcGis y MapInfo, los dos desarrollados bajo licencias privativas, con resultados reconocidos internacionalmente por su robustez, facilidades de uso, funcionalidades y eficiencia. Con un alto nivel de precisión, y prestaciones de alto nivel se encuentran el Kosmo y el GVSIG, que son sistemas de código abierto y que también ofrecen prestaciones de alto nivel de precisión en el tratamiento de datos geográficos.

Estos sistemas tienen implementadas funcionalidades que permiten cargar y manipular archivos cartográficos. Estos archivos se dividen en dos grupos fundamentales, los que contienen datos vectoriales y los que almacenan datos ráster. Los primeros se basan en la representación de imágenes mediante entidades geométricas básicas como puntos, líneas y polígonos. Mientras que los segundos almacenan los datos en forma matricial, donde cada celda consta con un valor único el cual representa un píxel en la pantalla.

Cuba, también tiene sus logros en esta rama, aunque en un menor grado por limitaciones económicas. En este avance juega un papel fundamental la Universidad de las Ciencias Informáticas (UCI)

contribuyendo con el alcance de la soberanía tecnológica de la nación. Dentro de esta universidad se encuentra el centro de desarrollo Geoinformática y Señales Digitales (GEySED) perteneciente a la Facultad 6, donde se ha creado un proyecto de investigación y desarrollo (I+D) llamado SIG-DESKTOP, que tiene como objetivo el desarrollo de Sistemas de Información Geográfica de escritorio. Actualmente en este proyecto se desarrolla un SIG llamado GeoQ, el cual tiene la capacidad de cargar, visualizar y manipular archivos cartográficos, pero muestra un bajo rendimiento en la realización de operaciones con datos vectoriales con grandes dimensiones. Este descenso en el rendimiento viene dado por la gran cantidad de entidades contenidas en estos archivos y que son necesarias procesar para lograr la visualización de las imágenes.

Teniendo en cuenta los planteamientos anteriores se genera el siguiente **problema a resolver:** La carga de gran cantidad de datos en capas vectoriales Shapefile provocan que GeoQ se demore en representar gráficamente la información cartográfica, lo que trae como consecuencia que el usuario no acepte el producto. Como **objeto de estudio** estará la manipulación de grandes volúmenes de datos vectoriales, mientras que el **campo de acción** será la manipulación de grandes volúmenes de datos vectoriales en el sistema GeoQ.

Se propone el siguiente **objetivo general:** Desarrollar un motor de manipulación de pirámides vectoriales para GeoQ, que permita disminuir el tiempo de respuesta del sistema a la hora de manipular datos vectoriales de grandes dimensiones.

Para darle cumplimiento al objetivo propuesto se definen las siguientes **tareas de la investigación:**

- Caracterizar las diferentes técnicas de reducción de datos vectoriales con pérdida reducida de información visual.
- Realizar el análisis del proceso de representación de datos vectoriales en GeoQ.
- Desarrollar la propuesta de solución.
- Evaluar la calidad del resultado.

Se tiene como **idea a defender:** Si se reduce la cantidad de datos cargados desde las capas vectoriales sin pérdida de información visual, se puede lograr que GeoQ alcance menores tiempos de respuesta, mejorando la aceptación del producto.

Para la realización de este trabajo de diploma se aplicaron diferentes **métodos científicos**:

Métodos Teóricos:

Analítico-Sintético: Donde se estudiará con profundidad de toda la información acerca de las tecnologías, metodologías y herramientas que se pueden utilizar en el desarrollo de las funcionalidades necesarias, lo que permitirá definir con mayor certeza las mismas, además sintetizará sus características y brindará la posibilidad analizar la viabilidad de cada una.

Histórico-Lógico: Con el cual se estudiarán métodos, procedimientos e ideas utilizadas en soluciones anteriores y similares a la que se pretende desarrollar, con el objetivo de extraer elementos que aporten vías de solución.

Modelación: Se utiliza para realizar los modelos correspondientes al ciclo de vida del sistema, esto permite facilidades en el cumplimiento de las tareas de análisis y diseño de los procesos que intervienen en la aplicación.

Métodos Empíricos:

Entrevista: Para definir los posibles requisitos a implementar, así como la obtención de propuestas de soluciones.

Una vez finalizada la investigación se esperan obtener como resultados los enumerados a continuación:

1. Documentación y artefactos generados según la metodología para ciclo completo.
2. Funcionalidades que permitan la partición y representación gráfica de grandes volúmenes de datos vectoriales.
3. Manual de usuario.
4. Documento de informe de la investigación.

El documento se encuentra estructurado de la siguiente manera:

En el **Capítulo 1: "Fundamentación Teórica"** se especifican y explican los principales conceptos asociados al campo de acción definido, se realiza una descripción del objeto de estudio identificado, se detalla la situación problemática actual y se expone el estudio y las principales conclusiones que se obtuvieron luego de que se valoraran las soluciones existentes relacionadas con la temática.

El **Capítulo 2:"Herramientas y Tecnologías"** a utilizar detalla las principales tecnologías, lenguajes de programación y herramientas que se utilizarán para la elaboración de la solución.

En el **Capítulo 3: "Descripción y construcción de la solución propuesta"** se describe la solución propuesta en términos de modelo de dominio, requisitos funcionales y no funcionales y los casos de uso del sistema, todo esto unido a los correspondientes diagramas que lo modelan, además se presenta la construcción propuesta en función de diagramas de clases y estándares del diseño, generalidades y modelo de implementación y los resultados de las pruebas realizadas.

CAPÍTULO 1: Fundamentación Teórica.

1.1 Introducción

En el presente capítulo se abordan conceptos asociados al dominio del problema, así como aspectos y conceptos generales relacionados con el tema de las pirámides vectoriales, además de una descripción del estado del arte del tema a tratar. El objetivo es dejar bien sentadas las bases teóricas para un correcto análisis y posterior implementación del sistema.

1.2 Conceptos asociados al dominio del problema

Cartografía:

Se conoce como cartografía a la ciencia que se dedica al estudio y elaboración de mapas que sirven para la navegación, para la ubicación del ser humano, etc. La palabra cartografía viene del griego grapho y significa 'la escritura de mapas' **(Hiparión, 2011)**.

Es un lenguaje que mediante elementos gráficos ordenados en una secuencia lógica aporta información. Está conformada por varios elementos como la escala, canevá y cuadrícula, leyenda, líneas de marco y sistemas de numeración de hojas. Consiste en la representación, lo más exacta posible, de parte o toda la superficie de la tierra y cualquier cuerpo sobre una superficie plana.

Cartografía digital:

Es la ciencia que se dedica a la elaboración de mapas apoyándose de las TIC. Se ocupa del tratamiento y la representación de datos mediante números o caracteres de un repertorio finito y convierte el complejo arte de la creación de mapas en una labor más sencilla **(Hiparión, 2011)**.

Cada día los mapas se hacen más complejos, por lo que su diseño requiere la utilización de herramientas que permitan simplificar los procesos de elaboración. Las TIC brindan esta posibilidad, ofreciendo innumerables funcionalidades para la representación geográfica. Actualmente este tipo de cartografía constituye la base de muchos sistemas, desde los sistemas de GPS (Global Position System, en inglés), meteorología, ciencias espaciales, topografía, hasta un simple mapa para orientación turística. Todos están basados en la cartografía digital.

Renderización:

La palabra renderización es una adaptación al castellano del vocablo inglés "rendering" y que define un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen o secuencia de imágenes. Habitualmente se utiliza esta nomenclatura para definir el proceso por el cual se pretende imitar un entorno tridimensional, formado por estructuras poligonales, luces, texturas y materiales, simulando ambientes y estructuras físicas verosímiles. También se suele utilizar para procesos 2D que requieren cálculos complejos como la edición de vídeo, la animación o el desarrollo de efectos visuales.

“Es la acción de asignar y calcular todas las propiedades de un objeto antes de mostrarlo en pantalla” **(Guglielmetti, y otros, 2012).**

Escala:

Es la relación existente entre las distancias medidas sobre los mapas y las distancias reales que representan las medidas sobre el terreno. Las escalas se escriben en forma de razón donde el antecedente indica el valor del plano y el consecuente el valor de la realidad. Por ejemplo la escala 1:500, significa que 1u del plano equivale a 500u en la realidad. Ejemplos: 1:1, 1:10, 1:500, 5:1, 50:1. De igual manera la escala gráfica son representadas a través de líneas rectas graduadas, subdivididas en unidades de distancia terrestre (Ver Anexo 1).

Existen tres tipos de escalas:

- Escala natural: Es cuando el tamaño físico del objeto representado en el plano coincide con la realidad, es decir, escala 1:1.
- Escala de reducción: Se utiliza cuando el tamaño físico del plano es menor que la realidad. Esta escala se utiliza para representar mapas físicos de territorios donde la reducción es mucho mayor y pueden ser escalas del orden de E.1:50.000 o E.1:100.000. Para conocer el valor real de una dimensión hay que multiplicar la medida del plano por el valor del denominador.
- Escala de ampliación: En planos muy pequeños se utiliza la escala de ampliación. En este caso el valor del numerador es más alto que el valor del denominador o sea que se deberá dividir por el numerador para conocer el valor real del área representada. Ejemplos de escalas de ampliación son: E.2:1 o E.10:1.

"La escala es la relación matemática que existe entre las dimensiones reales y las del dibujo que representa la realidad sobre un plano o un mapa" **(Scribd, 2011).**

1.3 Representación Vectorial.

1.3.1 Características

En el modelo de datos vectorial los datos geográficos se representan en forma de coordenadas. Las unidades básicas de información geográfica en los datos vectoriales son puntos, líneas (arcos) y polígonos. Cada una de éstas se compone de uno o más pares de coordenadas, por ejemplo, una línea es una colección de puntos interconectados, y un polígono es un conjunto de líneas interconectadas.

A continuación se definen las unidades básicas del modelo vectorial:

Coordenada: Consiste en un par de números, los cuales indican las distancias horizontales en los ejes de coordenadas. Se utiliza para representar localizaciones en la superficie terrestre **(Jiménez, 2007).**

Las coordenadas pueden definirse como un conjunto de números que describen una posición, ya sea a lo largo de una línea, en una superficie o en el espacio. Existen dos coordenadas angulares muy utilizadas en los sistemas de coordenadas geográficas, estas son conocidas como latitud, la cual proporciona la localización de un lugar en dirección norte o sur desde el ecuador y la longitud que mide el ángulo a lo largo del ecuador desde cualquier punto de la tierra.

Punto: Abstracción de un objeto de cero dimensiones representado por un par de coordenadas X, Y. Normalmente un punto representa una entidad geográfica muy pequeña para ser representada como una línea o una superficie **(Jiménez, 2007).**

Los puntos son las unidades básicas de las imágenes vectoriales. Las otras entidades vectoriales están formadas por puntos; por ejemplo las líneas son un conjunto de puntos alineados y con la delimitación del inicio y fin del segmento, mientras que los polígonos están formados por líneas.

Línea o arco: Conjunto de pares de coordenadas ordenados, que representan la forma de entidades geográficas muy finas para ser visualizadas como una superficie a una escala determinada (curvas de nivel, calles, carreteras, líneas ferroviarias o ríos o entidades lineales sin área como los límites administrativos **(Jiménez, 2007).**

Polígono: Entidad utilizada para representar superficies. Un polígono se define por las líneas que forman su contorno y por un punto interno que lo identifica. Los polígonos tienen atributos que describen al elemento geográfico que representan **(Jiménez, 2007).**

En el modelo vectorial hay que hacer una diferenciación clara entre dos conceptos, por un lado tenemos la estructura de datos, es decir la forma en que se guarda la información gráfica. Por otro lado diferenciaremos lo que se denomina topología, o la relación entre los diferentes elementos que componen la cartografía.

Se dice que una estructura de datos es topológica cuando almacena una o más de las siguientes relaciones:

- Conectividad de los arcos en las intersecciones.
- Existencia de conjuntos de arcos formando los límites de los polígonos.
- Relación de contigüidad entre los polígonos.

En el caso en el que ninguna de estas relaciones se forme, se dice que la estructura de datos es exclusivamente cartográfica y no dispone de topología. Muchos programas de CAD[1] tienen características cartográficas, pero no topológicas.

Se entiende por topología la creación de asociaciones entre entidades gráficas. En el caso de los SIG las relaciones se establecen de dos formas:

- Establecimiento de códigos comunes a las entidades.
- Relaciones a través de su posición espacial.

La misión de creación de la topología de una capa o cobertura es establecer estas relaciones y guardarlas en las bases de datos que conforman los ficheros vectoriales estas relaciones serán fundamentales para el tratamiento de muchos de los problemas planteados en los SIG vectoriales, como es el caso:

- Cálculo de áreas: se inicia al crear la tabla de polígonos y se almacena en ella.

[1] Computer Aided Design (Diseño asistido por Computadora).

- Perímetro del polígono: es la suma de las longitudes de los arcos que delimitan el polígono (se extrae de la tabla de arcos y se añade a la de polígonos).
- Camino más corto entre dos puntos de una red.

En función de la forma de almacenamiento de los datos podemos establecer cuatro tipos fundamentales de estructuras:

- Lista de coordenadas "espagueti".
- Diccionario de vértices
- Ficheros DIME ("Dual Independent Map Encoding").
- Arco/Nodo.

Cada una de ellas tiene diferentes ventajas e inconvenientes (Ver Anexo 2).

Las capas de un SIG basado en un modelo vectorial están constituidas por al menos dos ficheros. Uno de ellos contendrá la parte gráfica, basada en general en la estructura arco-nodo vista anteriormente, y otro conteniendo la información de los atributos de los elementos cartográficos, en este caso los ficheros constituirán una base de datos.

La estructura gráfica básica contendrá una cabecera que almacenará los datos de georreferencia, error, tipo de datos representados, unidades de medida, escala y modos de visualización, elementos representados, y asociaciones de elementos. Por otro lado contendrá los identificadores de los elementos, sus coordenadas y asociaciones.

Los modelos basados en estructuras vectoriales son muy apreciados y usados por el empuje que compañías como ESRI[2] han dado a los SIG. Las ventajas que ofrecen estos modelos son:

- Menor tamaño de almacenamiento en proyectos de gran envergadura (uno de sus puntos fuertes inicialmente respecto a los modelos ráster, debido a las características técnicas de los primeros PC).
- Operaciones de análisis espacial sencillas y rápidas
- Posibilidad de creación de cartográficas precisas a diferentes escalas
- La medida de distancias y áreas tiene mayor precisión.

[2] Enviromental Systems Research Institute.

- Permite la gestión individualizada de las entidades geográficas

1.4 Pirámides de Datos.

Se utiliza el término pirámides analógicamente para establecer una relación entre la forma física de estas construcciones con la estructura que se usa para desglosar los datos contenidos en las cartografías usadas por los SIG. La construcción de las pirámides de datos geográficos tiene como principal objetivo la optimización del tiempo de respuesta de un sistema. Cada nivel o capa superior son versiones a menor resolución de la imagen original que permiten la carga rápida de imágenes muy grandes (en un zoom general se mostraría la imagen de menor resolución, mientras que un zoom de detalle cargaría la imagen de mayor resolución). El término pirámides viene dado porque una imagen se representa en varios niveles: el nivel 0 estaría dado por la imagen original, el nivel 1 sería una versión reducida del primero, y así sucesivamente se irá conformando cada parte de la estructura.

Ejemplo de ello se evidencia en las pirámides ráster (Figura 1).Los datos ráster se compone de filas y columnas de celdas, cada celda almacena un valor único, el cual representa un píxel en la pantalla. Para acelerar la velocidad de visualización de los documentos ráster con un número elevado de pixeles, es posible crear una pirámide con distintos niveles de resolución. Esto permite que al abrir los documentos que las posean, una capa de nivel de baja resolución se muestre mientras se prepara el conjunto de los datos. Por otra parte cada vez que se haga un zoom, aparecerá el nivel de resolución adecuado, sin necesidad de implicar al conjunto de la imagen.

En ArcCatalog, una de las extensiones ArcGIS, lo implementan de la siguiente manera. Cada vez que se realiza la pre-visualización de un ráster mayor de 1024 por 1024 celdas, el programa pregunta si se desea iniciar el proceso de creación de pirámides, pudiéndose aplazar para más adelante o elegir no realizarlo nunca.

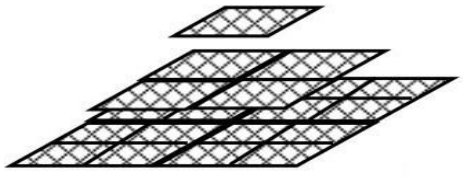


Figura1. Pirámide ráster.

1.4.1 Pirámides Vectoriales

En un SIG, las características geográficas se expresan con frecuencia como vectores, manteniendo las características geométricas de las figuras. Al cargar un archivo vectorial, el sistema reconoce todos los objetos (puntos, líneas, polígonos) que la forman. Luego representa cada uno de ellos para formar la imagen contenida en la capa. Cuando una cartografía contiene una gran cantidad de objetos, el rendimiento del sistema se ve gravemente afectado. Para mejorar la respuesta del sistema es necesario crear un motor para generar pirámides vectoriales (Figura 2). Al igual que las ráster, las pirámides vectoriales estarían formadas por sucesivos niveles de resolución, pero en este caso es necesario reducir los objetos en cada nivel y no las celdas como en las ráster.

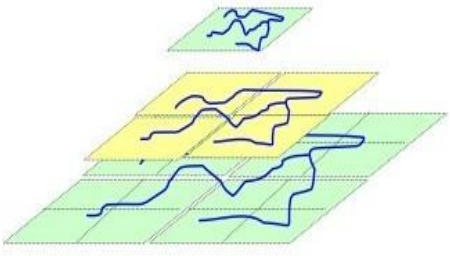


Figura 2. Pirámide vectorial

Básicamente este concepto de pirámides vectoriales consiste en la reducción de los objetos vectoriales que no son significativos en la visualización. Por ejemplo, un usuario carga una cartografía de un país, en un primer momento tendría una vista general. El sistema tiene que procesar y representar todos los objetos que forman la imagen a pesar de que a este nivel de la escala muchos de estos objetos no se ven o son tan pequeños que no aportan ninguna información a la hora de mostrar los datos. A medida que el usuario reduzca el área a mostrar, estas entidades que antes no se veían aumentarán su tamaño y por tanto su importancia visual también aumenta. Por ejemplo, se selecciona un municipio para ser visualizado en una escala mayor, por tanto en el área de visualización solo aparecerá el municipio seleccionado, por lo que los detalles que en la vista anterior no eran significativos (ríos, calles, lagos, etc.) ahora si es necesario mostrarlos.

Se denominará un objeto como no significativo cuando no se vea o sea demasiado pequeño, siendo esta la base de la construcción de las pirámides. El sistema debe reconocer cuando un objeto no es significativo a una determinado zoom mediante cálculos que relacionen la escala con las dimensiones reales del objeto. Si la distancia entre los vértices de un polígono es 0 solo se vería un punto, por tanto esta entidad no es relevante y no debe mostrarse. Al cargar una nueva capa el sistema debe estar preparado para reconocer si es demasiado grande o contiene demasiados objetos. Una vez hecho

esto se debe proponer al usuario la construcción de la pirámide después de haber evaluado si se cumplen alguno de estos criterios.

Criterios a tener en cuenta para comenzar la construcción piramidal.

- **Reducción de objetos:** La construcción de las pirámides comenzaría cuando el sistema reconozca la existencia de un número determinado de objetos que se reduzcan de tal manera que de acuerdo a la escala en la que se encuentre el usuario, no sean significativos en la visualización de la imagen.
- **Cantidad máxima de objetos:** La construcción de las pirámides comenzaría cuando el sistema reconozca un número determinado de objetos a procesar. En este caso no se tiene en cuenta si son significativos o no.
- **Dimensión del área a visualizar:** La construcción de las pirámides comenzaría cuando el sistema reconozca a través del cálculo de las dimensiones del área a visualizar que es mayor que la establecida.

Una vez evaluado el criterio de inicio el usuario tendrá la oportunidad de decidir si comienza o no la construcción.

Si escoge la opción de construir las pirámides el sistema desglosará la imagen en escalones de resolución, uno para cada nivel de la escala. Se eliminarán en cada nivel los objetos que no sean significativos, sacándolos de la lista de entidades que contienen internamente las capas vectoriales. Esto provocará un aligeramiento de la capa, por lo que el tiempo de respuesta del sistema disminuirá considerablemente. El proceso de representación quedará optimizado debido que a medida que se acerca, se dibujan los niveles con mejor resolución, y el rendimiento se mantiene debido a que está dibujando sucesivamente áreas más pequeñas.

Este tema de la reducción de datos vectoriales se puede desarrollar por diferentes vías, siempre y cuando se logre la el objetivo de manera eficiente. La granularidad en una imagen vectorial se refiere al nivel de detalle con que cuente esta, a mayor granularidad mayor definición y detalle mostrará la imagen. En determinados niveles de la escala, por ejemplo al mostrar una vista general de un país, no es necesaria una granularidad alta. Por tato si se logran simplificar los objetos vectoriales de manera tal que no afecte en la visualización, eliminando detalles que a una escala determinada no son significativos, se puede optimizar el proceso de renderización.

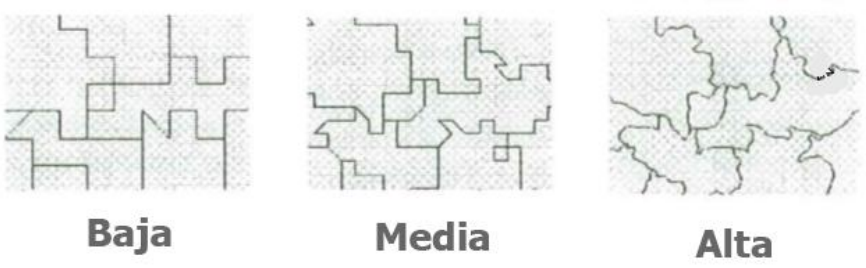


Figura 3. Granularidad

Las formas de reducir los datos para adecuarlos a cada nivel de la escala pueden ser varias, por ejemplo se podrían guardar varias versiones de la imagen en diferentes escalas y así mostrar la adecuada cuando el usuario escoja ese nivel de visualización. Otra forma sería la denominada Vector-Ráster-Vector. De un nivel a otro funciona de la siguiente manera. Convierte la imagen vectorial original en una ráster, y en cada una de las celdas simplificará las entidades existentes eliminando vértices innecesarios en la visualización a este nivel de la escala. Estos métodos están descritos a grandes rasgos; las investigaciones sobre los mismos se encuentran a un nivel teórico por lo que se desconoce la complejidad a la hora de su implementación. Es importante conocer los detalles en este aspecto teniendo en cuenta que con su utilización se busca optimizar el tiempo de respuesta de un sistema, por tanto no tendría sentido utilizar estas vías si entorpecen el desempeño del software.

1.5 Conclusiones Parciales

Una vez analizados los aspectos planteados en este capítulo se llega a la conclusión de que el tema de las pirámides vectoriales es novedoso y a la vez básico para lograr un buen rendimiento en el trabajo con cualquier sistema de información geográfica. En la investigación realizada se estudiaron los SIG ArcGis, MapInfo, GVSIG y Kosmo, sistemas punteros en la rama, y ninguno de los antes mencionados tienen implementadas funcionalidades que permitan la optimización por esta vía. La información acerca del tratamiento de imágenes vectoriales se encuentra dispersa y no es concluyente a la resolución del problema planteado, ya que no existe una fuente que muestre resultados positivos concretos en el tema. Los aspectos recopilados sirven como base para el desarrollo de esta investigación, proporcionando conocimientos necesarios para el entendimiento y posterior desarrollo del sistema.

CAPÍTULO 2: Herramientas y Tecnologías a utilizar.

2.1 Introducción

En el presente capítulo se describen las características de las principales tecnologías, lenguajes de programación y herramientas que se utilizarán para la elaboración de la solución. Se realiza una síntesis del comportamiento de estas tecnologías a nivel mundial y las ventajas inherentes a su utilización, con el objetivo de encontrar una selección del instrumental tecnológico que permita optimizar y ganar en tiempo en el desarrollo del módulo. Se realiza un análisis de las tecnologías que se emplean en el desarrollo y documentación de la aplicación.

2.2 Arquitectura de software

La Arquitectura de Software (AS) puede ser considerada como la base y organización fundamental de todo proyecto de software, que incluye los componentes principales del mismo. Define la conducta de esos componentes, las formas en que interactúan y se coordinan para alcanzar la misión del sistema, los principios que rigen su diseño y su posterior desarrollo. Es una vista abstracta que aporta el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones, según Clements cuando expresa una de las definiciones más reconocidas. “Su encause principal es dotar de conceptos y un lenguaje común a los equipos participantes en un proyecto, y para lograrlo, construir abstracciones materializándolas en forma de diagramas. Se aportan además elementos que ayudan a la toma de decisiones durante el desarrollo del software”.

Una definición la expone la IEEE Std 1471-2000, que plantea: La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución **(IEEE, 2000).**

La arquitectura de software se convierte en un punto clave en el proceso de desarrollo del software. De acuerdo a las características de la AS utilizada en la construcción de GeoQ no se concibe la idea de realizar cambios en la misma, por lo que se continuará trabajando sobre las elegidas por el proyecto SIG-Desktop, que desplegaron su trabajo apoyado en una arquitectura basada en componentes y orientada a objetos.

La Arquitectura Orientada a Objetos (AOO) según Pressman, define a los componentes del sistema como objetos o instancias de los tipos de datos abstractos. Dichos componentes encapsulan datos y operaciones que manipulan dichos datos. Pressman refiere además que: la comunicación y

coordinación entre los componentes se realiza mediante envío de mensajes. Este es un sistema donde se enfatiza el empaquetamiento entre datos y operaciones que permiten manipular y acceder a dichos datos (Pressman, 2005).

Sommerville se refiere a la Arquitectura Orientada a Objetos de la siguiente manera, “Un modelo arquitectónico orientado a objetos estructura el sistema en un conjunto de objetos débilmente acoplados y con interfaces bien definidas. Los objetos realizan llamadas a los servicios ofrecidos por otros objetos. Una organización orientada a objetos está relacionada con las clases de objetos, sus atributos y sus operaciones. Cuando se implementa, los objetos se crean a partir de estas clases y se usan algunos modelos de control para coordinar las operaciones de dichos objetos” **(Sommerville, 2005).**

Después de analizar los conceptos anteriores se concluye que las ventajas del estilo orientado a objetos radican en que como los objetos están débilmente acoplados, la modificación de ellos puede efectuarse sin comprometer a los otros. Los mismos son representaciones de entidades del mundo real, lo que posibilita que la estructura del sistema sea fácilmente comprensible. Permite la reutilización de dichos objetos dado que las entidades en el mundo real se usan en sistemas diferentes.

Arquitectura Basada en Componentes (ABC): Este estilo aboga por el principio de reutilización del software, donde la descomposición de la aplicación en componentes funcionales respalda que los componentes sean implementados de forma que se logre su manipulación sobre otros sistemas. Esta descomposición posibilita la transformación de los componentes pre-existentes en piezas más grandes de un software.

Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado **(SOMASEGAR, y otros, 2009).**

Algunos de los beneficios que posee este estilo son la facilidad de instalación que permite reemplazar una nueva versión del sistema por la existente sin impacto en el mismo o en sus componentes. El uso de componentes de terceros permite distribuir el costo del desarrollo y del mantenimiento, reduciendo el mismo considerablemente. La facilidad del desarrollo constituye otra de las ventajas principales, ya

que al igual que AOO los componentes implementan una interface bien definida lo que permite el desarrollo sin impactar otras partes del sistema.

2.3 Metodología de desarrollo de software

Su necesidad viene dada por la engorrosa tarea de la creación de aplicaciones de software que respondan a las habilidades y restricciones de los estándares del software en el mundo. Ello dificulta gradualmente el trabajo de los desarrolladores, lo que da pie al surgimiento a las metodologías de desarrollo de software. Estas definen Quién debe hacer qué, cuándo y cómo debe hacerlo para obtener los distintos productos en un determinado proceso de software. Su objetivo principal es guiar a los desarrolladores en el diseño e implementación de nuevas aplicaciones de probada calidad. Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo, las cuales se pueden clasificar en dos grupos:

- **Las metodologías pesadas:**

Están orientadas al control de los procesos, estableciendo rigurosamente la definición de roles, herramientas, actividades, artefactos y notaciones para el modelado y documentación detallada. Prestan mayor énfasis a la planificación y control del proyecto, en especificar de forma precisa los requisitos y el modelado.

- **Las metodologías ágiles:**

Proporcionan una serie de pautas, principios y técnicas pragmáticas que aseguran la entrega del proyecto con menos complicación y satisfactoriamente. Esto lo cumple a través de muestra de versiones parcialmente funcionales del software al consumidor en cortos periodos de tiempo para que se pueda evaluar y sugerir cambios en el producto según se va desarrollando. “Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software.”(Carrillo Pérez, y otros, 2008)

2.3.1 Proceso Unificado de Desarrollo de Software (RUP)

RUP (Rational Unified Process según sus siglas en inglés) es una metodología de desarrollo de software pesada, la cual se centra en la definición detallada de los procesos y tareas a realizar y las herramientas a utilizar. Pretende prever todo el desarrollo del producto de antemano por lo que

requiere una extensa documentación. Jacobson, Booch y Rumbaugh esta metodología de la siguiente manera. “Se puede definir como un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos **(JACOBSON, y otros, 2000).**

Su ciclo de vida se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y ser iterativo e incremental. Está constituido por 9 flujos de trabajo, los 6 primeros son conocidos como flujos de ingeniería y los 3 últimos de apoyo; todos tienen lugar sobre 4 fases de desarrollo. Ello asegura que RUP proporcione un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo de software, que se defina, para cada etapa y cada disciplina, el flujo de trabajo, los trabajadores que intervienen, las actividades que realizan y los artefactos que se necesitan o producen.

Debido a la complejidad y alcance de la solución de software a realizar, la investigación se desarrollará sobre la metodología de desarrollo RUP ya que es la metodología más acorde, dada la magnitud del proyecto, además de ser una metodología bien documentada y ejemplificada.

2.4 Lenguaje Unificado de Modelado Versión 2.0.

Por sus siglas en inglés UML, es un lenguaje que permite visualizar, especificar, construir, documentar, detallar y describir los métodos o procesos de un sistema de software. Es conocido en la industria y soportado por diversas herramientas y metodologías de desarrollo, siendo adoptado como notación estándar por empresas productoras de aplicaciones informáticas a nivel mundial. Posee una notación gráfica muy expresiva que posibilita representar todas las fases de un proyecto informático: desde el diseño y análisis de la aplicación hasta la implementación y configuración de la misma. Se convierte en una gran ventaja la posibilidad de que puede ser usado en todas las etapas de desarrollo de un sistema y su representación gráfica permite ser usada para la comunicación con los usuarios. Permite además, realizar una verificación y validación del modelo realizado y modelar estructuras complejas.

Como rasgo a destacar está que es absolutamente independiente del lenguaje de implementación, de forma tal que los diseños logrados usando UML se pueden realizar en cualquier lenguaje orientado a objetos. “UML ayuda a los usuarios a entender la realidad desde un punto de vista de la tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades de dinero en proyectos que no estén seguros en su desarrollo, reduciendo el costo y el tiempo empleado en la construcción de los módulos que construirán el software” **(JACOBSON, y otros, 2000).**

2.5 Herramienta CASE

Una herramienta CASE es concebida como una tecnología para automatizar el desarrollo y mantenimiento del software combinando metodologías de desarrollos y herramientas de software. Las herramientas CASE están destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de la misma en términos de tiempo y dinero. Son las aplicaciones en las que se apoyan los ingenieros y arquitectos de software para realizar las tareas de diseño de un proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, entre otras funcionalidades. Entre los beneficios ofrecidos por la tecnología CASE son notables: la facilidad que brindan para la revisión de aplicaciones, el soporte que poseen para el desarrollo de prototipos de sistemas, la generación de código, la mejora en la habilidad para satisfacer los requerimientos del usuario y el soporte interactivo para el proceso de desarrollo.

Visual Paradigm

Visual Paradigm (VP) es una herramienta profesional capaz de soportar todo el ciclo de vida del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Está orientada a UML que proporciona apoyo al modelado visual, el cual ayuda a una más rápida construcción de aplicaciones con mejor calidad y a un menor coste. Es definido por los usuarios como necesaria para la captura de requisitos, planificación de software, modelado de clases entre otras funcionalidades. Permite dibujar mayoritariamente todos los diagramas de clases además de admitir código inverso, generación de código desde diagramas y generación de la documentación. Entre las características más notables se pueden encontrar:

- Editor de Detalles de Casos de Uso: entorno para la especificación de los detalles.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Soporte ORM (Mapeo de objetos a la base de datos): generación de objetos Java desde la base de datos.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Generación de bases de datos.
- Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Distribución automática de diagramas.
- Reorganización de las figuras y conectores de los diagramas UML.

Otra característica a destacar es que es multiplataforma, lo que permite y facilita el trabajo en varios escenarios; de ahí la selección de esta herramienta CASE para la representación de todos los modelos a realizar en la presente investigación.

2.6 Lenguaje de programación

Los lenguajes de programación constituyen idiomas diseñados para que sirvan de instrumentos que posibilitan implementar programas y software. Están compuestos por un conjunto de símbolos y reglas sintácticas y semánticas que definen el significado de sus elementos además de su estructura. Ello facilita la labor de programación, pues estos lenguajes disponen de formas adecuadas que permiten ser leídas y escritas por personas. Resultan independientes del prototipo de ordenador a utilizar.

"Un lenguaje de programación es una técnica de comunicación estandarizada para expresar las instrucciones a una computadora. Se trata de un conjunto de reglas sintácticas y semánticas utilizadas para definir los programas de ordenador" **(Informática, 2006).**

Se pueden definir entonces como la técnica que permite al programador especificar con precisión los datos que una computadora usará para la toma de una decisión. También cómo estos datos serán almacenados o transmitidos y precisamente las acciones a tomar en diversas circunstancias. Son, sin duda, la vía de comunicación entre el hombre y la máquina.

2.6.1 Lenguaje de programación C++.

C++ precisamente es el lenguaje seleccionado para la implementación de la solución a proponer por la versatilidad y potencia que refiere. El mismo proviene del lenguaje C, del cual mantiene sus ventajas en cuanto a riqueza de operadores y expresiones y elimina algunas de las dificultades y limitaciones que poseía el anterior. Este nuevo lenguaje está diseñado con la intención de extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos llegando a ser un lenguaje híbrido.

Se muestra como lenguaje imperativo, orientado a objetos y que se basa en una filosofía que exige del programador un completo cambio de mentalidad con respecto a C. Mantiene una considerable potencia para programación a bajo nivel y se le ha añadido elementos que lo dotan de un estilo de programación con alto nivel de abstracción. Es multiparadigma, poseedor de programación genérica,

que se suma a los otros dos paradigmas ya existentes para cuando surge: programación estructurada y programación orientada a objetos.

Con el surgimiento de este potente lenguaje se agrega un concepto llamado sobrecarga de operadores lo que brinda la posibilidad de redefinir los operadores y poder crear nuevos tipos que se comporten como tipos fundamentales.

2.7 Framework de Desarrollo

Se puede definir como una estructura tecnológica de soporte, definido con artefactos o módulos de software concretos, con el objetivo de crear las bases para la mejor organización y desarrollo de los proyectos de software. Los framework de desarrollo en la informática han marcado un hito significativo, pues poseen significativas ventajas que simplifican el proceso de implementación y estandarización de los procesos en alguna medida. Refiere ser toda una tecnología o modelo de programación que contiene máquinas virtuales, compiladores, bibliotecas de administración de recursos en tiempo de ejecución y especificaciones de lenguajes, además de incluir una biblioteca de componentes reutilizables. Fueron diseñados con el fin de facilitar el desarrollo de software posibilitando a los proyectos identificar los requerimientos y prestar menos atención a los detalles de programación, quizás de bajo nivel, para proveer un sistema funcional.

Luego del estudio de las aplicaciones prácticas que desarrolla y las diferentes conceptualizaciones del término, se pueden precisar como las principales ventajas de la utilización de un framework:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- La reutilización de componentes de software. Los framework son los paradigmas de la reutilización.
- El uso y la programación de componentes que siguen una política de diseño uniforme.
- Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que genera como resultado bibliotecas más fáciles de aprender a usar.

2.7.1 QT

Es un framework multiplataforma utilizado para el desarrollo de aplicaciones informáticas y está implementado con C++. Extiende al lenguaje con el que está implementado a través de macros y meta información, algunas de estas características son los bucle foreach, la sentencia forever y la introspección. Su principal propósito es posibilitar a los desarrolladores la construcción de aplicaciones multiplataforma con una alta productividad a partir de una misma base de código, mediante el ofrecimiento de herramientas potentes y sencillas.

Está compuesto por un conjunto de módulos que proveen funcionalidades específicas a través de bibliotecas de clases multiplataforma. Algunos de estos módulos son:

- Bases de Datos: Qt SQL
- Core: Qt Core
- Comunicación en red: Qt Network
- Interfaz Gráfica de usuario: Qt GUI
- Multimedia: Phonon, Qt Multimedia
- Quick: Qt Declarative, QML
- Webkit: Qt Webkit
- XML: Qt XML

Está disponible bajo la licencia GPL: los cambios realizados al código fuente de Qt deben ser compartidos con la comunidad; LGPL: es posible crear aplicaciones de código cerrado, los cambios realizados al código fuente de Qt deben ser compartidos con la comunidad; y Comercial: es posible crear aplicaciones de código cerrado, los cambios realizados al código fuente de Qt pueden mantenerse cerrados.

Aunque utiliza a C++ de manera nativa, ofrece soporte para otros lenguajes como Python mediante PyQt, Java mediante QtJambi, o C# mediante Qyoto y ofrece una suite de aplicaciones para facilitar y agilizar las tareas de desarrollo.

Qt también provee poderosas herramientas de desarrollo, entre ellas destaca un completo entorno de desarrollo, llamado Qt-Creator, que incluye un editor de texto con autocompletado, diseñador de interfaces gráficas, gestión de proyectos, sistema de depuración, integración con sistemas de control de versiones y entre otras características.

2.8 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE, siglas en ingles), es un ambiente de desarrollo para escribir la lógica y el diseño de interfaces de aplicaciones de software. Están compuestos por un conjunto de herramientas que le facilitan al programador desarrollar una determinada aplicación de software. Estos entornos consisten en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Su objetivo principal es acortar la brecha entre el usuario y el lenguaje de programación proporcionando un marco de trabajo amigable. Pueden dedicarse en exclusiva a un sólo lenguaje de programación o bien para varios y han de ser aplicaciones por sí solas o parte de otros sistemas.

2.8.1 Qt Creator

Como se expresa en el acápite anterior, Qt Creator es el entorno de desarrollo por excelencia para Qt. Es un IDE multiplataforma, soportado principalmente por Windows XP y Vista, Linux y Mac OS X, que permite desarrollar aplicaciones implementadas con el lenguaje de programación C++ de manera sencilla y rápida. Se caracteriza por ser abierto, gratuito y muy eficiente. Proporciona herramientas para el diseño y desarrollo de aplicaciones complejas, entre sus principales características se distinguen:

- Editor avanzado para C++ y JavaScript.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador Visual.
- Soporte para el control de versiones.
- Simulador para interfaces de usuarios móviles.
- Soporte para objetivos de escritorios y portátiles.

Qt Creator constituye una multiplataforma que se ajusta a las necesidades de los desarrolladores. Se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt. **(Corporation, 2011).**

Este IDE está basado completamente en la librería Qt y todo su código fuente está protegido por la licencia GNU GPL, de manera que los proyectos implementados en este entorno deberán ser desarrollados bajo la misma licencia. Qt Creator permite crear aplicaciones de escritorio y plataformas de dispositivos móviles para diversos sistemas operativos haciendo uso del CMAKE. CMAKE constituye un conjunto de herramientas diseñadas para construir, probar y empaquetar software

independientemente de la plataforma en la que se desarrolle; y se utiliza para generar y/o automatizar código. Para lograr esto, genera ficheros makefiles nativos y espacios de trabajo que pueden usarse en el entorno de desarrollo deseado.

2.9 Conclusiones Parciales

En el presente capítulo se han descrito la metodología de desarrollo, el lenguaje de programación, la herramienta CASE y el entorno de desarrollo integrado con el propósito de definirlos como las tecnologías y herramientas adecuadas para el diseño e implementación del trabajo de diploma. Se concluye así la segunda de las tareas de la investigación con la cual se obtuvieron las bases para el desarrollo del presente trabajo:

El lenguaje de representación y modelado seleccionado es UML y la herramienta CASE Visual Paradigm. Como lenguaje de programación se escoge a C++ para el desarrollo de la funcionalidad en su totalidad. El entorno de desarrollo a usar es QT-Creator, poseedor del CMAKE que permite crear aplicaciones multiplataforma. Todo el proceso estará guiado por la metodología RUP.

Se concluye que las bases para comenzar la implementación de funcionalidades de la aplicación, en términos ingenieriles, están debidamente planteadas.

CAPÍTULO 3: Descripción y construcción de la solución propuesta.

3.1 Introducción

En este capítulo se comienza la descripción de la solución propuesta en términos de modelo de dominio, seleccionado como indicado para dar solución al problema planteado en el diseño metodológico. Se especifican los requisitos funcionales y no funcionales para definir las funcionalidades y cualidades que debe tener la solución propuesta. Se definen también los actores del sistema, el diagrama de casos de uso del sistema y la descripción de cada uno de estos.

Se presenta además la documentación del proceso de implementación que materializará la propuesta de solución. Esto se hace en términos de diagramas de clases asociado a los casos de usos identificados, así como otros artefactos como los modelos de implementación y despliegue. Se formula además el examen de validación de la solución propuesta.

3.2 Modelo de Dominio

Esta investigación no se desarrolla con el objetivo de satisfacer las necesidades de un cliente específico ni tiene por objetivo la creación de una aplicación independiente en la que deban identificarse procesos a automatizar sino, en optimizarla que ya existe a través de la incorporación de un motor para la manipulación de pirámides vectoriales. Teniendo en cuenta esto, no es necesario definir un Modelo del Negocio por lo que se procederá a trabajar en el desarrollo del Modelo del Dominio.

El modelo del dominio (Figura 4) se crea con el objetivo de documentar los conceptos dominantes y vocabulario del sistema. Básicamente es un modelo conceptual, que describe las entidades implicadas del software y sus relaciones. Este modelo proporciona una vista estructural que describe y obliga el alcance del producto final. Puede ser tomado por ende como el punto de partida para el diseño y es definido en algunos casos como una primera versión de dicho sistema.

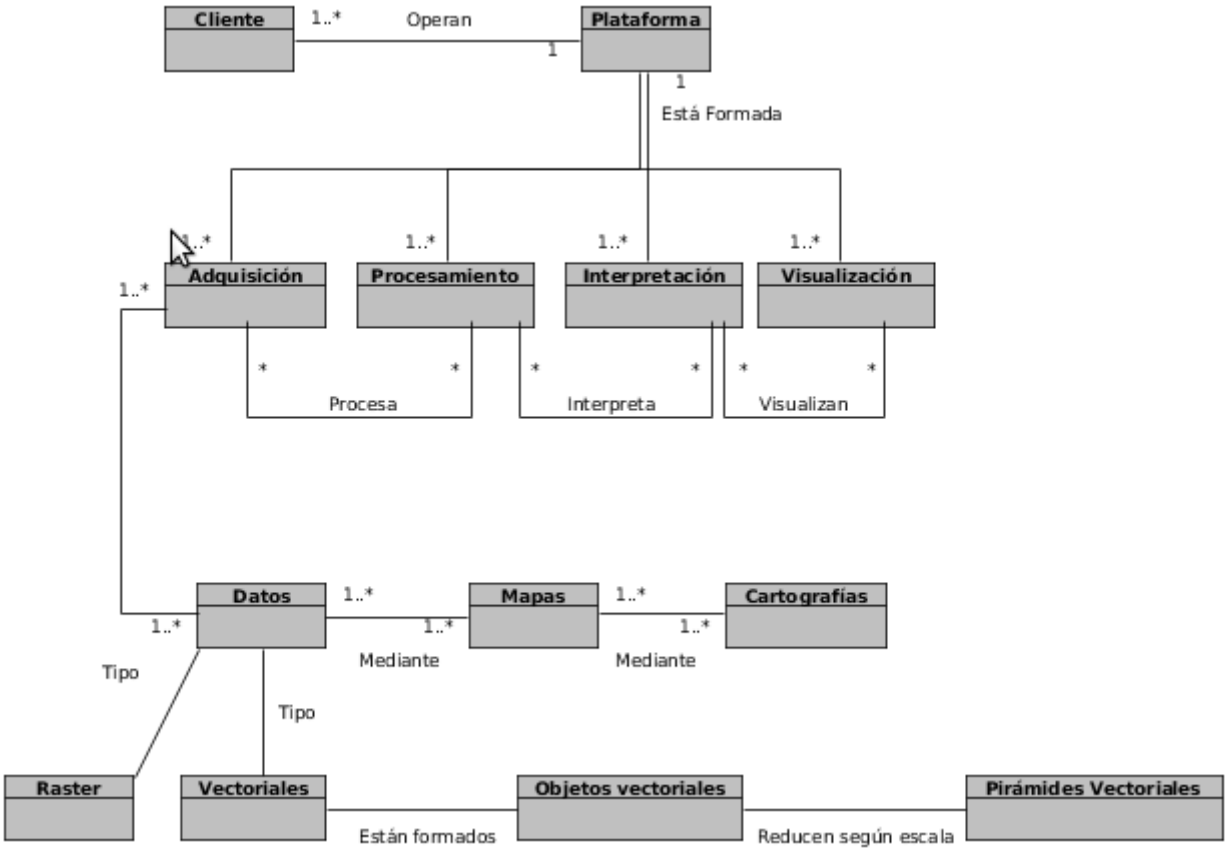


Figura 4: Diagrama de dominio.

3.3 Descripción del Modelo de Dominio

El cliente es el usuario que interactúa con la plataforma, esta responde a las necesidades del mismo realizando las operaciones básicas de un SIG: visualización, interpretación, procesamiento y adquisición de los datos contenidos en mapas y cartografías. Estos datos pueden aparecer en dos tipos, ráster o vectoriales, los de tipo vectorial están formados por objeto (puntos, líneas, polígonos). La reducción de los objetos se realizaría atendiendo al nivel de la escala que elija el usuario mediante las pirámides vectoriales.

3.4 Requisitos funcionales

RF 1: Identificar construcción de pirámide vectorial.

El sistema debe ser capaz de identificar luego de cargar una capa si es necesario crear una pirámide para los datos cargados.

RF 2: Construir pirámide vectorial.

El sistema debe ser capaz de construir las diferentes capas pertenecientes a la pirámide vectorial a partir de los objetos obtenidos de las capas cargadas.

RF 3: Representar datos.

El sistema debe ser capaz de representar los datos del mapa de acuerdo a la pirámide creada.

3.5 Requerimientos No Funcionales

Los requisitos no funcionales son aquellas cualidades o propiedades que el software debe poseer. Son estas propiedades las que hacen que el software sea confiable, eficiente, ágil y atractivo a los ojos del usuario.

RNF 1. Usabilidad

- El sistema podrá ser utilizado por usuarios con conocimientos básicos en el manejo de la aplicación GeoQ

RNF 2. Apariencia o interfaz externa

- La nueva interfaz que se incluirá en la aplicación GeoQ, encargada de interactuar con el usuario, deberá ser amigable, intuitiva y de fácil comprensión para el usuario, facilitando en todo momento la comunicación de este con el sistema y siguiendo la misma línea de diseño del GeoQ.

RNF 3. Portabilidad y operatividad

- El sistema debe ser compatible con los Sistemas Operativos Microsoft Windows 2000 NT, Microsoft Windows XP o superior y en plataformas Unix/Linux.

RNF 4. Software.

- Microsoft Windows XP Profesional o superior.
- Linux de la familia Debian (Debian, Ubuntu, Mint, Nova).

RNF 5. Hardware.

Las computadoras que utilizarán la aplicación deberán contar con un microprocesador con velocidad de procesamiento superior a un 1 GHz y memoria RAM de 512 MB o superior.

3.6 Descripción del sistema propuesto.

El Modelo de Casos de Uso ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios. Cada tipo de usuario se representa mediante un actor. Los actores utilizan el sistema al interactuar con los Casos de Uso. (Jacobson, 2000)

3.6.1 Descripción de actores

Actores	Descripción
Usuario	Usuario del sistema.
Sistema	Sistema que inicia automáticamente los casos de uso necesarios para la construcción de la pirámide vectorial.

Tabla 1: Descripción de actores del sistema

3.6.2 Casos de uso del sistema

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores.

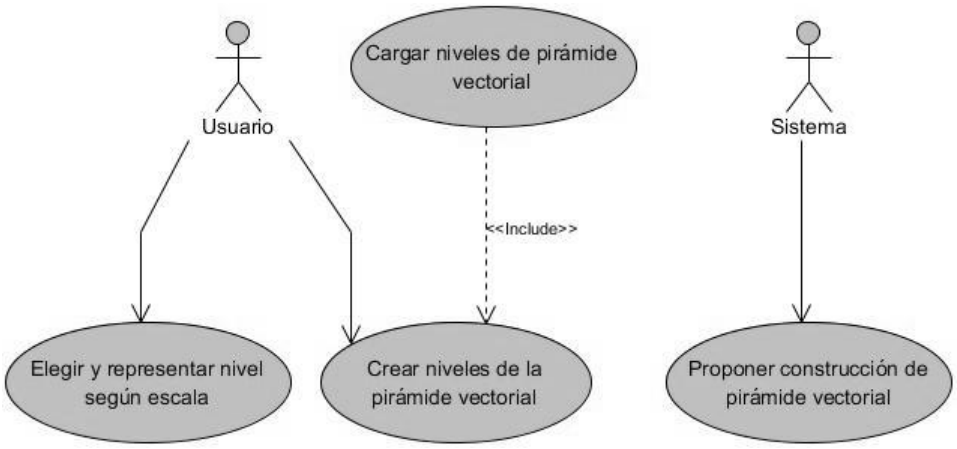


Figura 5: Diagrama de casos de uso del sistema

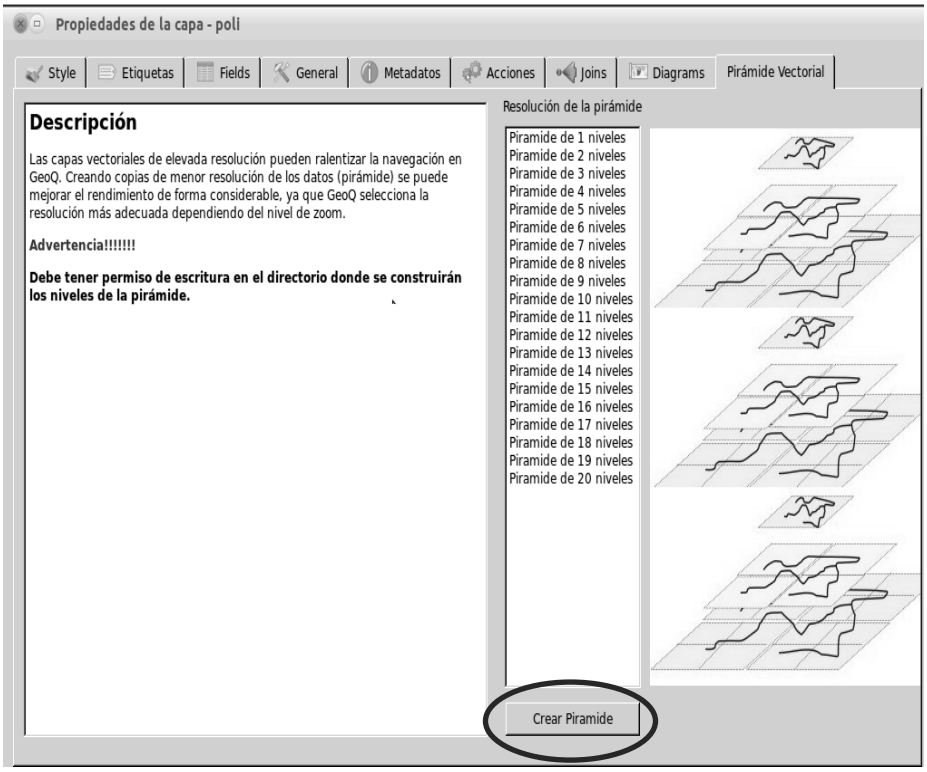
3.6.3 Descripción textual de los casos de uso del sistema.

A continuación se presenta la descripción textual de los casos de uso del sistema, por su importancia para lograr un mayor avance en la construcción del software es recomendable identificar los casos de uso arquitectónicamente significativos, que tienen una prioridad de crítico. En la Tabla 2 se muestra la descripción textual del Caso de Uso Crear niveles de la pirámide vectorial, la descripción de los casos de uso restantes puede encontrarse en el Anexo3 de la presente investigación.

Caso de Uso:	Crear niveles de la pirámide vectorial
Actores:	Usuario del sistema
Resumen:	El caso de uso se inicia cuando el usuario acepta la opción de construir la pirámide vectorial. Se comienzan a crear los niveles de la pirámide vectorial. El caso de uso termina cuando se han construido todos los niveles necesarios de la pirámide vectorial.
Precondiciones:	Debe existir al menos una capa vectorial cargada.
Referencias	RF 2
Prioridad	Crítico
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El usuario acepta la construcción de la pirámide vectorial.	2. El sistema crea una copia de la capa.
	3. El sistema elimina los objetos no pertenecientes a la copia creada según el nivel al que pertenece.
	4. Se ejecuta el paso 2 y 3 hasta tanto no se creen los niveles necesarios para la pirámide vectorial y termina el caso de uso.

Prototipo de Interfaz



Pos-condición	Ninguna

Tabla 2. Descripción textual del caso de uso del sistema: Crear pirámide vectorial.

3.7 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Con el paso del tiempo el uso de estos se ha ido intensificando, pues permiten la producción de software más resistente a los cambios. Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, usando las prácticas ya empleadas por muchos y tomando las mejores propuestas. Clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo de software. Por tanto, están basados en la recopilación del conocimiento de los expertos en la producción de sistemas informáticos.

Para el desarrollo del sistema se usaron algunos patrones GRASP (General Responsibility Assignment Software Patterns), estos patrones describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

- **Creador.**

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. El uso de este patrón se evidencia en la clase QgsVectorLayerProperties, que es la encargada de crear instancias de la clase QgisApp para el desarrollo de las principales funcionalidades del sistema.

- **Bajo acoplamiento.**

El término "acoplamiento" hace alusión al grado de dependencia que tienen dos unidades de software y el objetivo del patrón Bajo Acoplamiento es reducir al máximo el acoplamiento entre estas. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

Una de las principales pautas a seguir en el diseño de la aplicación fue el uso de este patrón entre las unidades de software. Ello aumenta la reutilización de las mismas y evita el efecto onda, ya que un

defecto en una unidad puede propagarse a otras, haciendo incluso más difícil de detectar dónde está el problema y minimiza el riesgo de cambiar múltiples unidades de software cuando una sea afectada.

- **Alta cohesión.**

El término "cohesión" es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. El patrón Alta cohesión plantea que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. Se tuvo como objetivo tener una alta cohesión ya que mejora la claridad y facilidad con que se entiende el diseño, se simplifica el mantenimiento y las mejoras de funcionalidad, a menudo se genera un bajo acoplamiento y soporta mayor capacidad de reutilización.

- **Controlador.**

Asigna la responsabilidad del manejo de los mensajes de los eventos de un sistema a una clase que represente una de las siguientes opciones: el "sistema" global (controlador de fachada); la empresa u organización global (controlador de fachada), algo en el mundo real que es activo (por ejemplo; el papel de una persona) y que pueda participar en la tarea (controlador de tareas); un manejador artificial de todos los eventos del sistema de un caso de uso(controlador de casos de uso) (LARMAN 1999).

Se asocian con operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos. Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad y por ende no realiza mucho trabajo por sí mismo.

3.8 Modelo de Diseño

Este modelo se desarrolla en el flujo de trabajo de diseño el cual tiene como objetivo la construcción del modelo de diseño que permite describir la realización física de los casos de uso traduciendo los requisitos funcionales y no funcionales a una representación del sistema. Estas acciones se realizan para crear una entrada apropiada y un punto de partida para las actividades de implementación. Durante esta etapa se generan todas las especificaciones para la programación del sistema.

El modelo de diseño constituye en sí un modelo de objetos que describe la realización física de los Casos de Uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas en el entorno de la implementación, tienen impacto en el sistema a considerar. Pretende crear un plano del modelo de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases. Procura lograr la descomposición de los trabajos de implementación en partes manejables a través de una comprensión de los aspectos relacionados con los requisitos funcionales y no funcionales unidos a las restricciones de los lenguajes de programación, sistemas operativos y tecnologías de distribución.

3.8.1 Diagrama de Clases del Diseño

Este diagrama que se muestra en la Figura 6 se elabora con el objetivo de tener en cuenta los detalles concretos de la implementación del sistema, describiendo la realización física de los casos de uso que este engloba. Este tipo de diagrama presenta una estructura estática que muestra las relaciones entre las clases del sistema y las relaciones existentes entre ellas. Se obtienen como resultado del refinamiento del modelo conceptual.

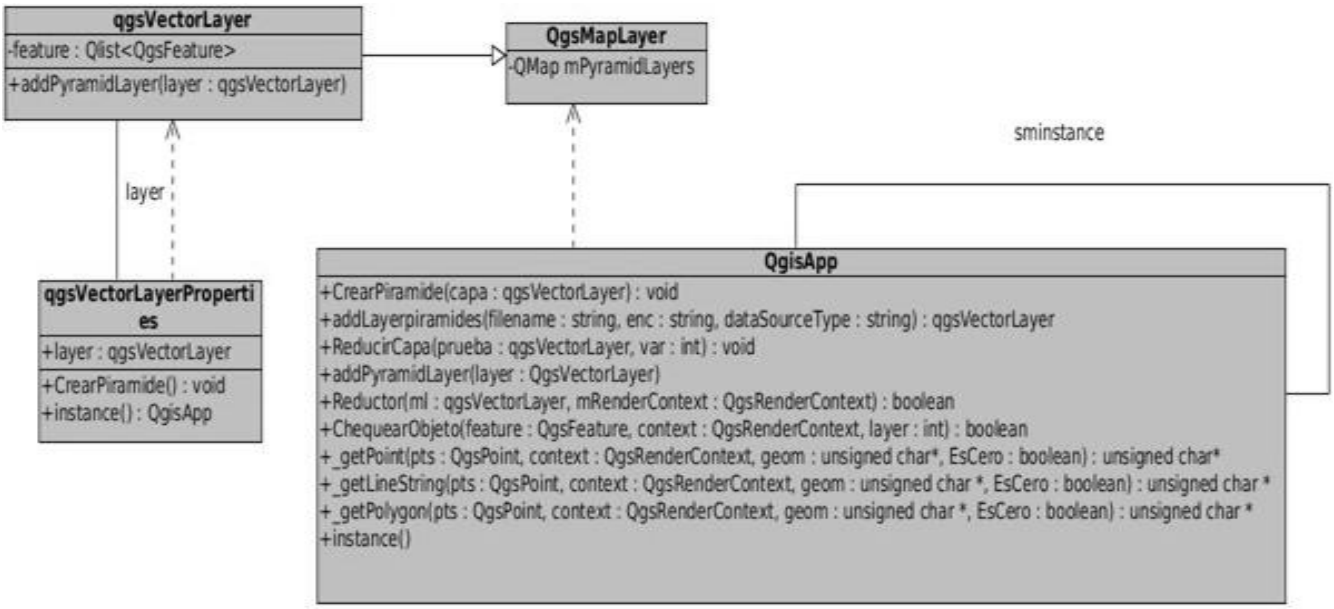


Figura 6: Diagrama de Clases del Diseño del CU Crear Niveles de Pirámide Vectorial.

3.8.2 Descripción del Diagrama de Clases del Diseño.

En este diagrama aparecen las clases implicadas en la realización de CU Crear Niveles de la Pirámide Vectorial, donde se identifican la clase qgsMapLayer que posee una relación de dependencia con la

clase controladora QgisApp, en la cual se realizan todas las operaciones de la aplicación, la clase qgsVectorLayer que tiene una relación de generalización con qgsMapLayer y la clase QgsVectorLayerProperties que posee una relación de dependencia con qgsVectorLayer, ésta es la clase donde comienza el proceso de construcción de la pirámide vectorial.

3.9 Generalidades de la Implementación

3.9.1 Modelo de Despliegue

El diagrama de despliegue muestra la disposición física de los distintos nodos que componen el sistema y el reparto de los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene algo de memoria y, a menudo, capacidad de procesamiento. En este caso solo es necesario un nodo, que representa la PC cliente donde se ejecuta el sistema.

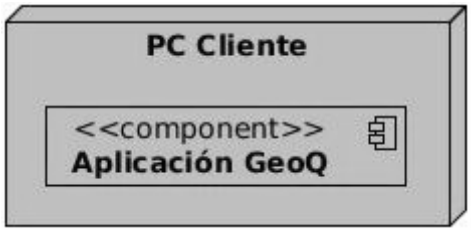


Figura 7: Diagrama Despliegue

3.9.2 Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros **(Sommerville, 2005).**

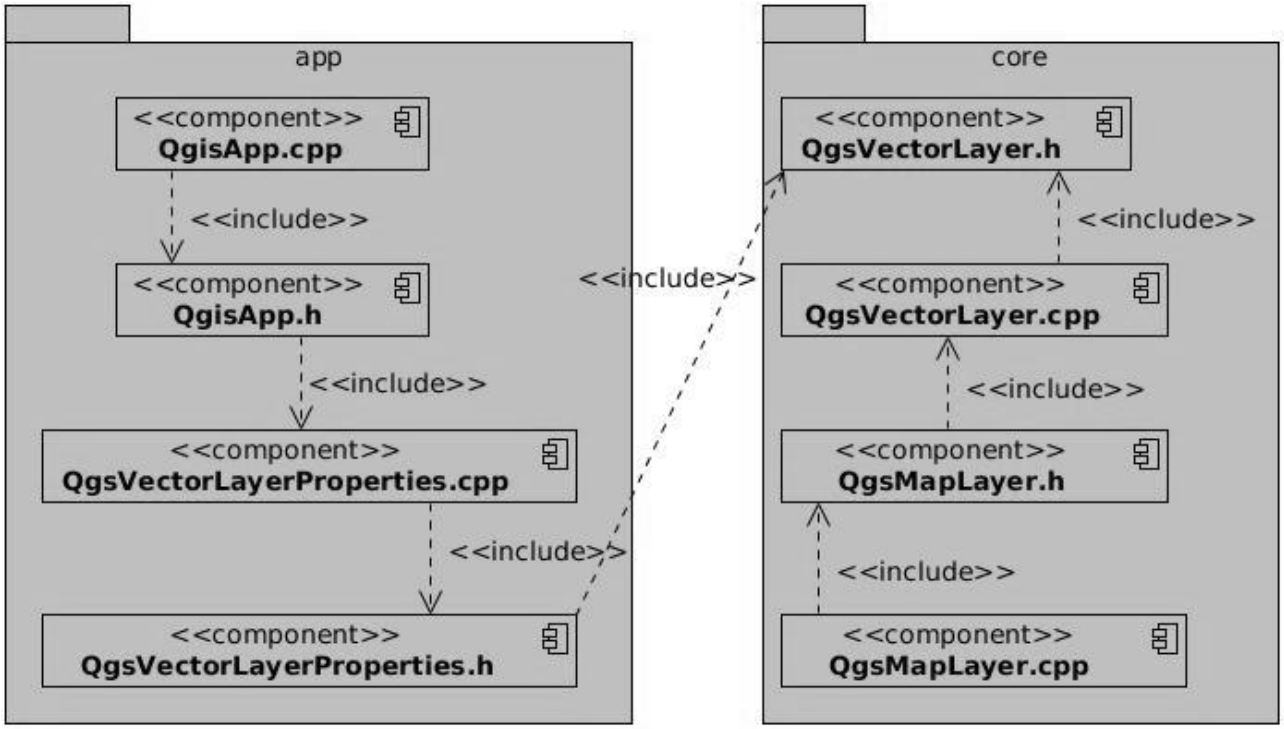


Figura 8: Diagrama de Componentes para CU Crear Niveles de Pirámide Vectorial

3.10 Pruebas del Sistema Propuesto

Durante el proceso de desarrollo de software se pueden cometer disímiles errores, desde la misma concepción del proyecto hasta su fase final de desarrollo, por lo que se hace necesario técnicas o pruebas de validación del software. Ello se debe fundamentalmente a que la identificación de requisitos descritos de manera errónea o el mal estado del diseño de clases pueden provocar faltas graves en el buen funcionamiento de una aplicación, de ahí la importancia de realizar dichas pruebas. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. Pressman la define como el proceso de ejecución de un programa con la intención de descubrir un error. Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y la codificación. La creciente percepción del software como un elemento del sistema y la importancia de los costes asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien detalladas **(Pressman, 2005).**

Disímiles son los tipos de pruebas existentes, por lo que se hace necesario una buena selección del prototipo de prueba apropiado a determinado negocio según las características del mismo, y a lo que se desee probar. Entre las diferentes clasificaciones se tienen:

- **Pruebas de Verificación:** Se comprueba el cumplimiento de las especificaciones del diseño.
- **Pruebas de Validación:** Se encargan de velar por el cumplimiento de los requisitos del análisis.
- **Pruebas de Caja Blanca:** Se conoce el código y se trata de ejecutar cada uno de los elementos del mismo.
- **Pruebas de Caja Negra:** Solamente se conoce la interfaz y se trata de probar cada uno de los elementos que componen a la misma.

3.11 Pruebas de Caja Blanca.

Son las pruebas que comprueban los caminos lógicos del software proponiendo casos de pruebas que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos, para determinar si el estado real coincide con el esperado o mencionado. Por tanto, estas requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código.

Técnica de Camino Básico.

Esta técnica permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba. Los pasos a seguir para realizar esta prueba son:

1. Se realiza el grafo de flujo, a partir del código fuente a probar.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

3.11.1 Pruebas al CU Crear Niveles de Pirámide Vectorial.

A continuación se muestra en la Figuras 9 el código del método Crear Pirámide. Este método es el más importante ya que es el encargado de crear todos los niveles de la pirámide, recibiendo como parámetro la capa original.

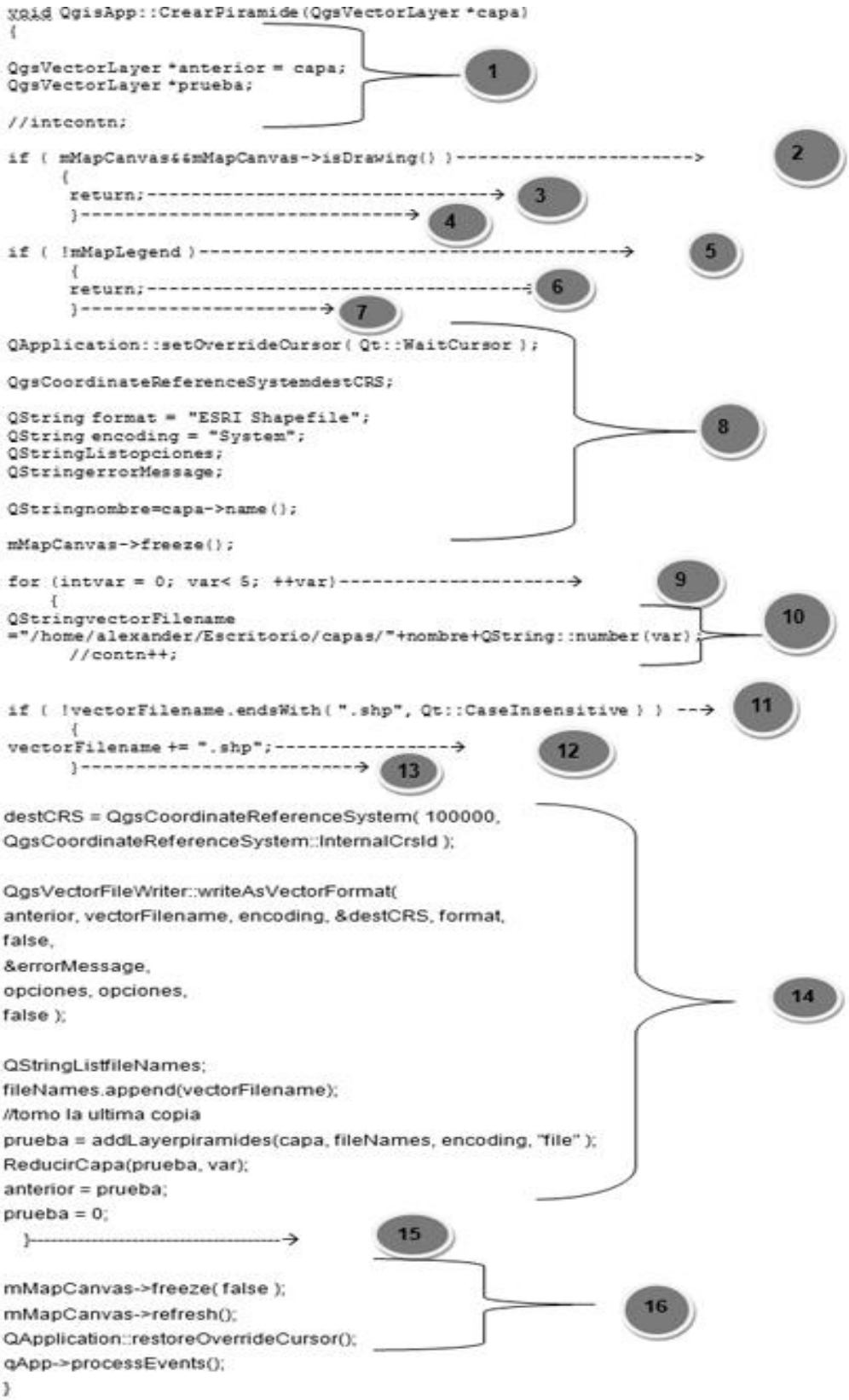


Figura 9 Código del método Crear Pirámide.

Paso 1: Grafo de flujo asociado al código fuente.

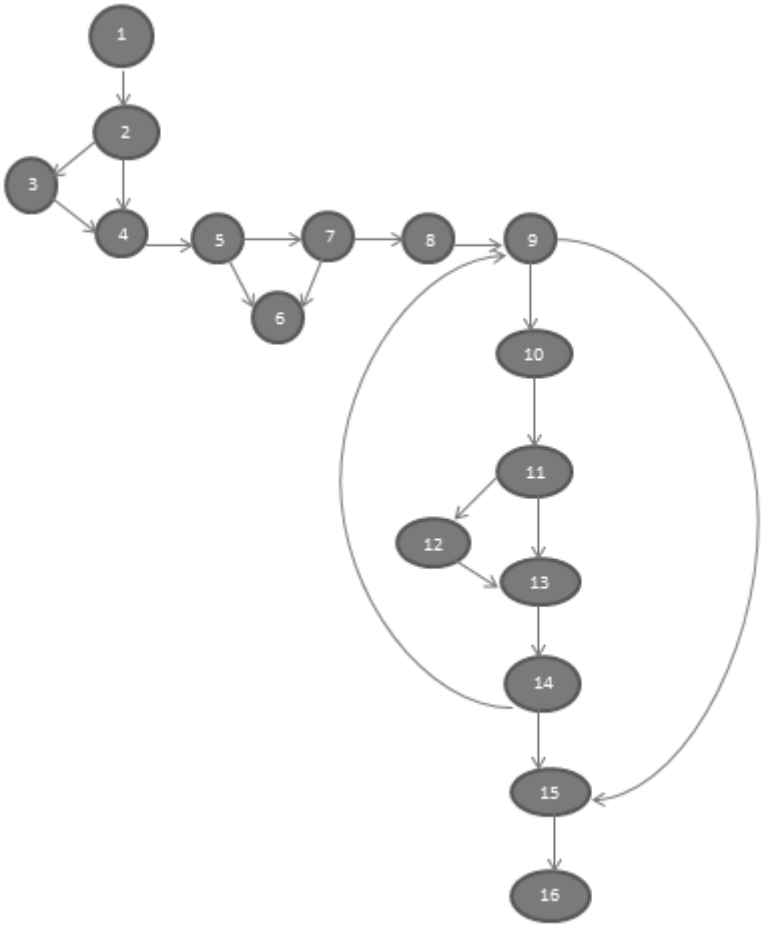


Figura 10 Grafo de flujo para CU Crear niveles de pirámide vectorial

Paso 2: Cálculo de la complejidad ciclomática.

La complejidad ciclomática es una de las métricas de software que proporciona una medida cuantitativa de la complejidad lógica de un programa. El valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa, y proporciona un número superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado por lo menos una vez.

V (G)= E – N + 2; donde E es el número de aristas, y N, el número de nodos de la gráfica de flujo.

V (G)= 19 – 16 + 2

V (G)= 5

La complejidad ciclomática dio como resultado cinco, lo que significa que existen seis posibles caminos por donde el flujo puede circular, lo que indica que el número total de casos de prueba a realizar. En la siguiente tabla se muestran los caminos escogidos para realizar los casos de prueba.

Paso 3: Conjunto básico de caminos independientes.

Números	Caminos básicos
1	1, 2 , 3, 4, 5, 6, 7, 8, 9, 15, 16
2	1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 9, 15, 16
3	1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 13, 14, 9, 15, 16
4	1, 2, 4, 5, 6, 7, 8, 9, 16
5	1, 2, 4, 5, 7, 8, 9, 16

Luego de tener elaborado el grafo de flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos.

Paso 4: Casos de prueba para el conjunto de caminos básicos.

Caso de Prueba para el Camino básico 1	
Caminos	1-2-3-5-6-12-13-14-15-16-18
Entrada	Una capa QgsVectorLayer. (capa original)
Resultado de la prueba	Satisfactorio

Caso de Prueba para el Camino básico 2	
Caminos	1-2-3-5-6-12-13-14-15-17-18
Entrada	Una capa QgsVectorLayer. (capa original)
Resultado de la prueba	Satisfactorio

Caso de Prueba para el Camino básico 3	
Caminos	1-2-3-5-6-7-8-10-11-6-12-13-14-15-17-18
Entrada	Una capa QgsVectorLayer. (capa original)
Resultado de la prueba	Satisfactorio

Caso de Prueba para el Camino básico 4	
Caminos	1-2-3-5-6-7-8-9-8-10-11-6-12-13-14-15-17-18
Entrada	Una capa QgsVectorLayer. (capa original)
Resultado de la prueba	Satisfactorio

Caso de Prueba para el Camino básico 5

Caminos	1-2-3-4-3-5-6-12-13-14-15-17-18
Entrada	Una capa QgsVectorLayer. (capa original)
Resultado de la prueba	Satisfactorio

Resultados de las pruebas realizada.

Teniendo en cuenta que todos los casos arrojaron un resultado satisfactorio, se concluye que las funcionalidades implementadas dan cumplimiento correctamente a los requisitos funcionales planteados para el desarrollo del sistema.

3.12 Conclusiones Parciales

En este capítulo se expusieron y detallaron los argumentos facilitarán la materialización de los casos de usos descritos. Sé aseguró de este modo el cumplimiento del objetivo primario de esta investigación al lograr que la herramienta GeoQ construyera pirámides de datos para capas vectoriales permitiendo una disminución del tiempo de respuesta del sistema.

CONCLUSIONES

Terminado el proceso de incorporación de nuevas funcionalidades a la aplicación SIG de escritorio GeoQ, objetivo primario de la presente investigación, se arribó a las siguientes conclusiones:

Permanecen reflejados en la documentación los conceptos fundamentales que permiten entender la lógica y funcionamiento del entorno de trabajo donde se desarrolló la misma.

Las herramientas y tecnologías seleccionadas para la incorporación de las nuevas funcionalidades a GeoQ propiciaron la correcta materialización del objetivo de la investigación.

Los requisitos funcionales y no funcionales identificados durante el levantamiento de requisitos fueron implementados en su totalidad y debidamente probados.

El resultado de la investigación alcanza mayor relevancia teniendo en cuenta que en ninguno de los SIG estudiados se encontraron soluciones similares a la desarrollada.

RECOMENDACIONES

Con el objetivo de perfeccionar y ampliar las funcionalidades de la herramienta desarrollada se propone al proyecto SIG- Desktop las siguientes recomendaciones:

- Evaluar la idoneidad del diseño propuesto para su uso en otros Sistemas de Información Geográfica que adolecen de las funcionalidades descritas en el presente trabajo.
- Incluir en el sistema desarrollado funcionalidades que permitan actualizar los niveles de la pirámide vectorial luego que una capa sea editada.
- Continuar trabajando en el establecimiento de dependencias entre las clases planteadas en el diseño y las clases concretas proporcionadas por el GeoQ que faciliten a los desarrolladores soluciones de menor complejidad.

REFERENCIAS BIBLIOGRÁFICAS

- **Carrillo Pérez, Isaias y Pérez Gonzáles, Rodrigo. 2008**.Metodologia de Desarrollo de Software. Mexico : s.n., 2008.
- **Changxiu Cheng, Feng Lu, Mingbo Zhang. 2003**. A Multi-Granularity Approach Adaptive Mass Vector Dataset Access and Transmission. Beijing : s.n., 2003.
- **IEEE, 2000**.IEEE Recommended Practice for Architectural Description of Software-Intensive SystemsCorporation, 2011.
- **Corporation, 2011.** Qt Creator IDE and tools 2011.
- **ESRI. 2010**.Manual ArcView. 2010.
- **SOMASEGAR y otros, 2009**.Microsoft. Application Architecture Guide. patterns & practices (2nd edition)
- **Giraldo Ocampo, Ing. Julián Darío.**Corporación Universitaria Remington (Universidad de San Buenaventura Medellin). http://juliangiraldo.wordpress.com.
- **Gracia, Joaquin. 2005**.IngenieroSoftware. [Online] Mayo 27, 2005. [Cited: enero 20, 2012.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php. IBNH/.
- **Hyparion.**Hyparion. [Online] [Cited: 10 26, 2011.] www.hyparion.com.
- **Jacobson, Ivar. 2000**. 2000.
- **Jiménez, Miguel Ramírez. 2007**.Construcción de un SIG para la Gestión de Explotación de Recursos del Subsuelo. Catalunya : s.n., 2007.
- **Pressman, Roger S. 2005**.Software Engineering. New York : The MacGraw-Hill, 2005
- **Rendering Definition**. [Online] [Cited: 03 8, 2012.] http://www.renderingdefinition.net/.
- **LARMAN, 1999**.C. Applying UML and Pattern. An Introduction to Object-Oriented Analysis and Design,
- **X Conferencia Internacional sobreSIG. Rhind, David. 2005**. San Juan, Puerto Rico : s.n., 2005.

BIBLIOGRAFÍA CONSULTADA

- **BUSCHMANN Frank, R. M., ROHNERT Hans, SOMMERLAD Peter y STAL Michael**. Pattern-Oriented Software Architecture – A System of Patterns. 1996. p.
- **CASTILLA,I.C.U.d.,** Prácticas de la ingeniería de software, Una Herramienta CASE para ADOO. 2007.
- **ÖVERGAARD Gunnar, PALMKVIST Karin.** Use Cases Patterns and Blueprints. Addison Wesley Professional, 2004. ISBN 0-13-145134-0.
- **SALAZAR GÓMEZ, Lisset; DUEÑAS NARANJO, Yarely**. Sistema automatizado para la captura de información referente al Balance Nacional de Recursos y Reservas de Petróleo de la Oficina Nacional de Recursos Minerales. Universidad de las Ciencias Informáticas. Ciudad de la Habana, 2008.
- **STAIR, Ralph M., et al (2003).** Principles of Information Systems, Sixth Edition. Thomson Learning, Inc., pp. 132. ISBN 0-619-06489-7
- **GAMMA Erich, HELM Richard, JOHNSON Ralph, VLISSIDES John.** Elements of Reuseable Object-Oriented Software, New York, 1997.
- **Chaves Pérez, Juan Manuel. 2010.** GESTION INFORMÁTICA CON SOFTWARE LIBRE. Cádiz : s.n., 2010.
- **Garcia de Jalon, Javier and Brazales, Alfonso. 1998.** Aprenda C++ como si estuviera en primero. España : s.n., 1998.
- **Gracia, Joaquin. 2005.** IngenieroSoftware. [Online] Mayo 27, 2005. [Cited: enero 20, 2012.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php. IBNH/.
- **Iglesia, Carlos. 2011**. sensagent. sensagent. [En línea] 2011. [Citado el: 20 de Noviembre de 2011.] http://dictionary.sensagent.com/postgis/es-es/.
- **Pockin. 2008**. Modelado de Sistemas com UML. 2008.
- **Nicanor, Lisandro Damián. 2007**. Introducción al desarrollo con Qt. 2007.
- **Olaya, Víctor. 2010.** Sistemas de Información Geográfica. Girona : s.n., 2010
- **Pressman, Roger S. 2005.** Software Engineering. New York : The MacGraw-Hill, 2005. 978-0-07-337598-7.
- **Reynoso, Carlos. 2004.** Introducción a la arquitectura de software. Buenos Aires : s.n., 2004.
- **Rumbaugh, J., Booch, G. y Jacobson, L. 2000.** El Proceso Unificado de Desarrollo. 2000.
- **Tobarra Narro, Manuel. 2003.** CMM y RUP: Una perspectiva común. Albacete : s.n., 2003.

ANEXOS

Anexo 1. Representación de la escala.

1.1 Representación de la escala en forma numérica.

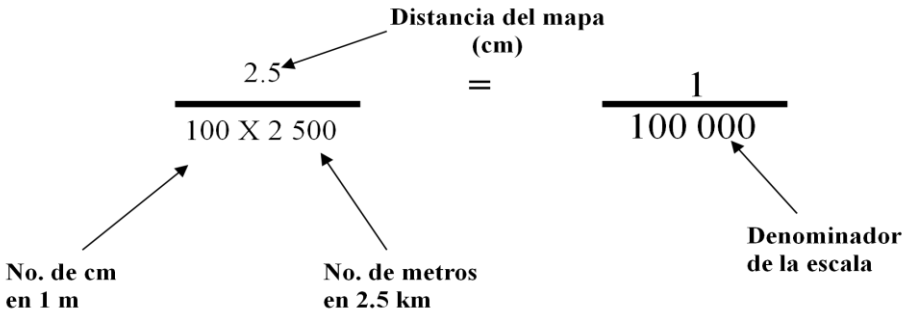


Figura 11 Representación de la escala en forma numérica.

1.2 Representación de la escala en forma gráfica.

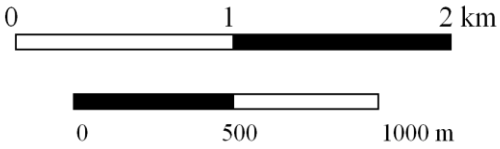


Figura 12Representación de la escala en forma gráfica.

Anexo 2.Representación vectorial.

2.1 Lista de coordenadas "espagueti"

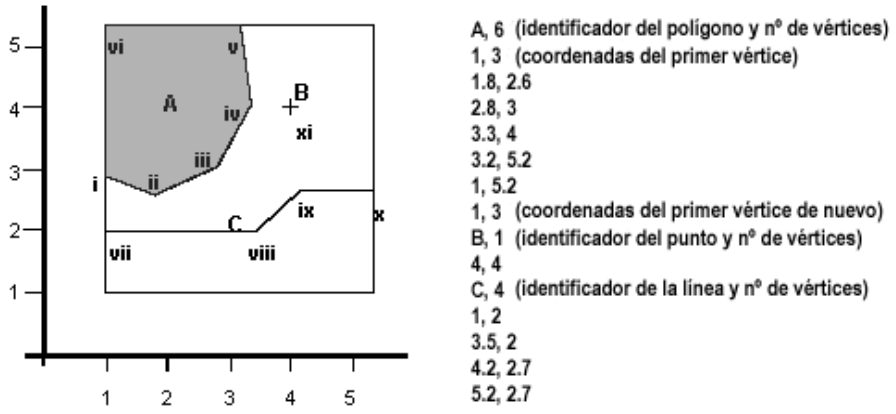


Figura 13Lista de coordenadas "espagueti".

Ventajas:

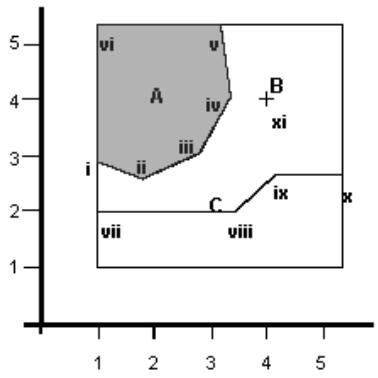
- Sencilla.

- Fácil de operar.
- Muy utilizada en la cartografía automática.

Desventajas:

- No almacena topología.
- Muchas operaciones redundantes, que implican espacios de almacenamiento mayores.

2.2 Diccionario de vértices



fichero 1		
vértice	X	Y
i	1	3
ii	1.8	2.6
iii	2.8	3
iv	3.3	4
v	3.2	5.2
vi	1	5.2
vii	1	2
viii	3.5	2
ix	4.2	2.7
x	5.2	2.7
xi	4	4

fichero 2

polígono A: i, ii, iii, iv, v, vi
punto B: xi
línea C: vii, viii, ix, x

Figura 14 Diccionario de vértices

Ventaja:

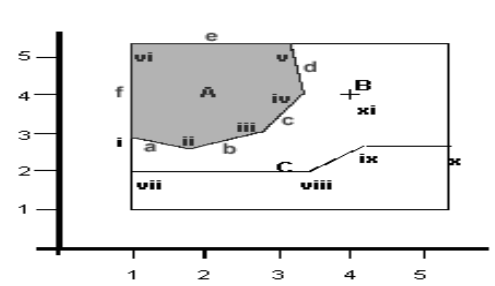
- No hay operaciones redundantes

Desventajas:

- No almacena la topología

2.3 Ficheros DIME (“Dual Independent Map Encoding”)

En los ficheros DIME los nodos (intersecciones de líneas) son identificados con códigos, Se le es asignado un código direccional de la forma "fromnode" (nodo origen) y "tonode" (nodo final) y tanto las direcciones de las calles como las coordenadas UTM se definen de forma explícita para cada vínculo.



fichero 1

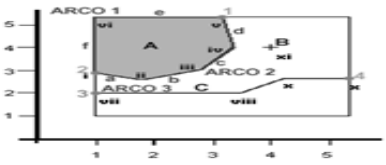
vértice	X	Y
i	1	3
ii	1.8	2.6
iii	2.8	3
iv	3.3	4
v	3.2	5.2
vi	1	5.2
vii	1	2
viii	3.5	2
ix	4.2	2.7
x	5.2	2.7
xi	4	4

segmento	polig. dcha.	polig. izq.	vértice origen	vértice final
a	externo	A	i	ii
b	externo	A	ii	iii
c	externo	A	iii	iv
d	externo	A	iv	v
e	externo	A	v	vi
f	externo	A	vi	i
g	externo	externo	vii	viii
h	externo	externo	viii	ix
i	externo	externo	ix	x

polígono	segmentos
A	a, b, c, d, e, f

Figura 15Ficheros DIME.

2.4 Arco / Nodo



Fichero 1. Coordenadas de los nodos y vértices de cada arco

ARCO	Nodo origen	Vértices intermedios	Nodo final
1	3.2, 5.2	1, 5.2	1,3
2	1,3	1.8,2.6 2.8,3 3.3,4	3.2, 5.2
3	1,2	3.5,2 4.2,2.7	5.2,2.7

Fichero 3. Topología de polígonos

Polígono	Arcos
A	1, 2

Fichero 2. Topología de arcos

ARCO	Nodo origen	Nodo final	Polígono derecha	Polígono izquierda
1	1	2	Externo	A
2	2	1	A	Externo
3	3	4	Externo	Externo

Fichero 4. Topología de los nodos

Nodo	Arcos
1	1,2
2	1,2
3	3
4	4
5	5

Figura 16Arco / Nodo.

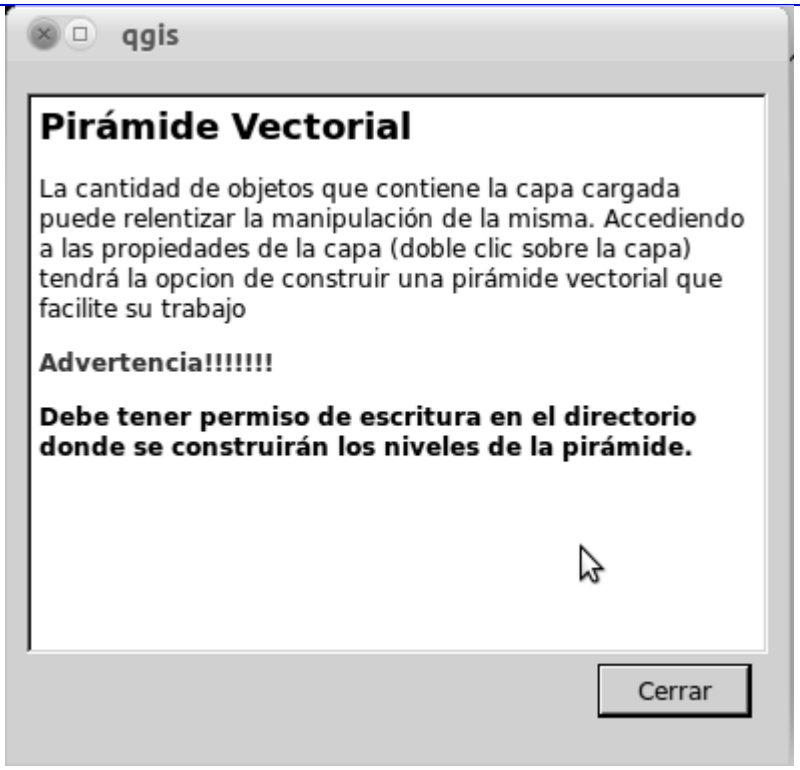
Anexo 3. Descripción expandida de los casos de usos del sistema.

3.1 Descripción del Caso de Uso Proponer construcción de Pirámide Vectorial.

Caso de Uso:	Proponer Construcción de Pirámide Vectorial
Actores:	Sistema
Resumen:	El caso de uso se inicia cuando el sistema determina

	que es necesaria la construcción de la pirámide vectorial. Envía un mensaje de dialogo al usuario brindándole la posibilidad de comenzar o no dicho proceso. Termina cuando el usuario acepta o cancela el cuadro de dialogo.
Precondiciones:	Debe existir al menos una capa vectorial cargada.
Referencias	RF 1
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema calcula la cantidad de objetos que forman la capa.
	2. El sistema muestra un mensaje de dialogo dando la opción de comenzar o no la construcción de la pirámide vectorial.
3. El usuario del sistema acepta la construcción de la pirámide vectorial y termina el caso de uso.	
Prototipo de Interfaz	



Postcondición	Ninguna.

3.2 Descripción del Caso de Uso Cargar niveles de la pirámide vectorial.

Caso de Uso:	Cargar niveles de pirámide vectorial
Actores:	Usuario
Resumen:	El caso de uso comienza cuando el sistema termina la construcción de todos los niveles de la pirámide. El sistema carga los niveles creados y termina el caso de uso.
Precondiciones:	Debe estar creada la pirámide vectorial.
Referencias	RF 3
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema accede a la dirección de origen de los niveles de la pirámide vectorial.
	2. El sistema carga los todos los niveles de la pirámide vectorial y termina el caso de uso.
Poscondición	Ninguna.

3.3 Descripción del Caso de Uso Elegir y representar nivel según escala.

Caso de Uso:	Elegir y representar nivel según escala
Actores:	Usuario del sistema
Resumen:	El caso de uso comienza cuando el sistema ha terminado de cargar todos los niveles de la pirámide. El sistema elige que nivel se visualizará según la escala en que se encuentre el usuario del sistema.
Precondiciones:	Debe existir al menos una capa vectorial cargada.
Referencias	RF 3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario del sistema ejecuta	1. El sistema elige de acuerdo

acción de zoom sobre el mapa.	al zoom en que se encuentre el usuario del sistema el nivel de la pirámide.
	2. El sistema representa el nivel de la pirámide elegido y termina el caso de uso.
Poscondición	Ninguna.