UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Título: Desarrollo de un componente para Modelación y Visualización en 2D de objetos geológicos.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: **Yosnai Díaz Morrell**

Tutor: **Ing. Yudiel Rodríguez Larrazábal.**

Ciudad Habana, Junio 2011

"Año 53 de la Revolución"

Dedicatoria

Agradecimientos

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al _____ _____de 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 _____.

Yosnai Díaz Morrell
_____

Ing. Yudiel Rodríguez Larrazábal
_____

Datos de Contacto

Tutor:

Nombre y apellidos: Yudiel Rodríguez Larrazábal.

Categoría docente: Adiestrado.

Título de la especialidad de graduado: Ingeniería en Ciencias Informáticas.

Año de graduación: 2009.

Institución donde se graduó: UCI.

Opiniones y Avales

Opinión del Tutor

Resumen

La informática y las tecnologías que están asociadas a ella, forman parte de la cultura tecnológica que nos rodea debido a que contribuyen al desarrollo social imponiendo un nuevo orden mundial a los procesos de gestión, posibilitando ampliar nuestras capacidades físicas y mentales. Por estas razones existen empresas e instituciones actualmente que deciden escoger las alternativas automatizadas para obtener un buen desarrollo de los objetivos que se trazan, logrando una mayor organización, facilidad y disponibilidad de la información externa.

La presente investigación surge en el marco de trabajo del Proyecto Minería perteneciente al Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED) de la Universidad de las Ciencias Informáticas (UCI). Con esta investigación se pretende realizar un componente que permita la modelación y visualización de objetos geológicos en dos dimensiones (2D), dígase puntos, polilíneas, polígonos. La Visualización en 2D tiene su base en la construcción de objetos en dos planos por lo cual los objetos son vistos en los planos XY, YZ, ZX; lo que permitirá tener una mayor visión sobre la representación que se realiza. Dicho componente podrá ser utilizado por el equipo de desarrollo del proyecto Minería en la futura implementación del producto “Geolmin” para el apoyo a la minería.

Palabras Claves:

Componente, Objetos Geológicos, visualización en 2D.

Abstract

The computing science and the technologies associated are part of our technological culture as they contribute to the social development, establishing a new world order to the management processes, thus allowing to extend our physical and mental capacities. For these reasons, there are enterprises and institutions at this time that decide to choose the automated alternatives to obtain a good development of their planned objectives, thus achieving a higher organization, simplicity and availability of the external information.

The present investigation comes out in the environment of the Mining Project work of the Digital Signals and Geo-Computing Development Center from the Computing Sciences University (UCI). The purpose of this investigation is to make a component that allows the modeling and visualization of geological objects in two dimensions (2D), it means, points, poly-lines and polygons. 2D Visualization is based on the construction of objects in two levels, thus, objects are viewed in levels XY, YZ, ZX, which will allow to have a higher vision of the representation that is being made. This component could be used by the development team of the Mining project in the future implementation of the “Geolmin” product to support the mining.

Índice

Índice de Figuras

Índice de Tablas

Introducción

A lo largo del proceso de evolución de la especie humana, el hombre se ha visto obligado a confeccionar sus propios instrumentos de trabajo e irlos perfeccionando con materiales existente en la naturaleza, conllevándolo a determinar el uso eficiente de cada elemento que encontraba. Con el paso del tiempo los instrumentos tuvieron un mejor acabado y fueron realizados con materiales más resistentes. Es por ello que inconscientemente el hombre estaba dando los primeros pasos en lo que más tarde se llamaría Proceso Minero o Minería, la cual es la obtención selectiva de los minerales y otros materiales de la corteza terrestres.

Hoy en día uno de los renglones fundamentales de la economía es la extracción de minerales ya que por su alto valor agregado puede ser empleado en la realización de disímiles artículos, además por el alto costo en el mercado varias empresas, con el interés de obtener riqueza, han comenzado a relacionar un proceso tan antiguo como la minería con el desarrollo actual de las tecnologías y la rama de la informática.

En la actualidad los sistemas computarizados han agilizado y optimizado los procesos de extracción. Es por ello que las principales compañías en esta actividad invierten grandes sumas de dinero en compra de software que ayuden a conocer dónde se encuentra ubicado un mineral, que mejoren la representación y visualización de los datos recolectados en las perforaciones para llegar a tener mejor conocimiento sobre el cuerpo mineral que se desea obtener.

Recientemente han surgido en el mundo diferentes productos de software los cuales realizan visualizaciones del terreno en tres dimensiones (3D), dos dimensiones (2D), modelado de bloque y límites físicos del cuerpo mineral para calcular los recursos y reservas del mismo, así como el diseño de mina; los que sirven de apoyo a las labores de gestión de una mina.

Cuba ha venido incursionándose en el mundo de la Informática aplicada a las labores mineras desde la década del 90, época donde surgieron los primeros software cubanos, los que hasta ese momento no presentaban grandes soluciones pero eran pasos de avance en la rama. Pueden citarse como ejemplo de ello: el sistema MICRONIQ desarrollado por la actual Empresa Geominera de Oriente para las primeras computadora XT que fueron adquiridas por el servicio geológico nacional y Software Integral Minero (SIM)

que permite la modelación de recursos y reservas con introducción de algoritmos de modelación matemática.

Con el reimpulso que aportaron los convenios con firmas extranjeras y la premura de los contratos, el país se vio obligado a utilizar licencias de productos extranjeros como: GEMCOM y más tarde SURPAC, DATAMINE y otros que fueron adquiridos por entidades nacionales o introducidos por los socios extranjeros en las empresas mixtas formadas para el níquel y otros minerales, que si bien respondieron a aquellas urgencias, impusieron una dependencia tecnológica que además del oneroso pago de versiones, licencias, capacitación y servicios nos comprometieron con esas compañías.

La Universidad de las Ciencias Informáticas (UCI) como motor impulsor de la informatización del país ha venido realizando, a lo largo de estos cinco años, software para distintas áreas de trabajo del país. La universidad se encuentra divida en siete facultades las que cuentan con varios Centros de Desarrollo que a su vez son los encargados de guiar a los distintos proyecto productivos. Actualmente el Proyecto Minería, perteneciente al Centro de Desarrollo GEySED[1] de la Facultad VI se encuentra desarrollando una aplicación que apoye el desarrollo de la minería. El equipo de desarrollo se encuentra con el inconveniente de no contar con una herramienta que facilite la modelación y visualización de objetos geológicos.

A partir de la situación descrita se identificó como **problema a resolver:** ¿Cómo desarrollar una herramienta para la modelación y visualización de objetos geológicos en el proyecto Minería? Definiéndose como **objeto de estudio** las técnicas de representación y modelado en dos dimensiones, donde el **campo de acción** lo constituye la informatización de los procesos de modelado y representación de objetos en dos dimensiones.

Teniendo como **Idea a defender:** El componente para la modelación y visualización en dos dimensiones de objetos geológicos proveerá al proyecto Minería de una herramienta informática para el desarrollo del mismo. Para dar solución al problema a resolver planteado, esta investigación tiene como **objetivo general:** Desarrollar un componente que permita modelar y visualizar objetos geológicos en dos dimensiones.

[1]GEySED: Centro de Desarrollo de Geoinformática y Señales Digitales.

A partir de este objetivo general se trazaron los siguientes **objetivos específicos:**

- ✓ Analizar el modelado y visualización de objetos en 2D de objetos geológicos.
- ✓ Especificar los requisitos del componente de software.
- ✓ Diseñar un componente de software que permita la representación de objetos geológicos, a través del modelado y visualización en 2D.
- ✓ Implementar el componente de software diseñado.

Los que se cumplirán a través de las siguientes **tareas de la investigación:**

- ✓ Caracterizar el proceso de modelado y visualización en 2D de objetos geológicos.
- ✓ Caracterizar las tendencias y tecnologías actuales a desarrollar.
- ✓ Elaborar el modelo de dominio del componente.
- ✓ Especificar los requisitos funcionales del software.
- ✓ Elaborar el diagrama de casos de uso (CU) del sistema del componente de modelado y visualización en 2D.
- ✓ Elaborar los diagramas de clase del diseño del componente de modelado y visualización en 2D.
- ✓ Elaborar el diagrama de implementación.
- ✓ Implementar los CU definidos.

Para desarrollar la presente investigación se emplearon diferentes métodos que permitieron obtener información valiosa:

Empírico:

1. **Entrevista:** Se le realizó a los clientes para un mejor entendimiento del dominio del problema.

Teóricos:

- ✓ **Análisis y la síntesis:** Para descomponer el problema de investigación en partes, para un mejor entendimiento de la situación y luego poder sintetizarlos para la confección de la solución propuesta.
- ✓ **Histórico y Lógico:** Para el estudio de los trabajos e investigaciones anteriores y tenerlo como base para esta investigación.
- ✓ **Modelación:** La cual sirvió de apoyo a la hora de realizar los distintos diagramas que a lo largo de la investigación fueron realizados.

Entre los posibles resultados esperados de dicha investigación científica se encuentran**:**

- ✓ Obtener el componente para el modelado y visualización en 2D de objetos geológicos.
- ✓ Obtener la documentación técnica asociada al desarrollo del componente anterior.

El presente trabajo de investigación estará dividido en 4 capítulos los que se resumen a continuación.

Capítulo 1: Estado actual de los software mineros.

En este capítulo se abordan conceptos asociados con el domino del problema. Además se realiza un estudio del estado del arte de los principales sistemas mineros, centrándose en las opciones que brindan estas aplicaciones a las distintas áreas de la minería.

Capítulo 2: Tendencias y tecnologías actuales a desarrollar.

En el presente capítulo se abordará las vías para realizar un componente de software que sirva de apoyo al proyecto Minería. También se analizan las principales técnicas, tecnologías, herramientas y metodologías utilizadas para el desarrollo del componente.

Capítulo 3: Presentación de la solución propuesta.

Se expone el diseño de la solución propuesta y se realiza un estudio de los procesos del negocio, describiéndolos a través del Modelo de Dominio. Se identifican los requisitos funcionales y no funcionales, es diseñado también el Modelo del Sistema. Son diseñados los modelos de análisis y diseño con los que se tendrá el primer acercamiento a la implementación del componente.

Capítulo 4: Construcción de la solución propuesta.

Muestra cómo va estar estructurada la implementación del sistema. Contiene los diagramas de clases de componentes y el modelo de implementación. Se detalla además ejemplos del código de la solución propuesta. Son aplicadas las pruebas de camino mínimo a la solución propuesta y es presentado el componente de visualización.

Capítulo 1: Estado actual de los software mineros.

1.1. Introducción

Para realizar una correcta estimación y recolección de los minerales de la corteza terrestre es preciso realizar un modelado lo más exacto posible de los datos recolectados, este trabajo es realizado por los geólogos y los mineros apoyándose de los distintos software existentes para obtener los minerales que tanta demanda tienen en la actualidad.

Para entender lo antes mencionado se precisa conocer los principales conceptos y definiciones que permiten adentrarse en esta investigación. Precisamente en este capítulo se explicarán los elementos teóricos fundamentales que sustentan el problema. Se profundizará también en las soluciones existentes en la actualidad, teniendo en cuenta los aportes de cada uno de ellos al negocio minero.

1.2. Conceptos asociados con el dominio del problema.

La **geología** es la ciencia y el estudio de la materia física y energía que constituyen la Tierra. El campo de la geología comprende el estudio de la composición, estructura, propiedades, y la historia de la materia física del planeta, los procesos por los que se forma, se trasladó y cambió la historia de la vida en la Tierra, y las interacciones humanas con la Tierra. (Argüero, 2002)

De la definición anterior se puede arribar a la conclusión que un **objeto geológico** es todo compuesto procedente de la tierra el cual puede ser estudiado y llegar a conocer su composición, estructura, propiedades y si tiene alguna utilidad para la sociedad.

Los **sondajes** son perforaciones de pequeño diámetro y gran longitud que se realizan para alcanzar zonas inaccesibles desde la superficie. A través de los sondajes se puede estudiar y analizar zonas de hasta 1.200 m de profundidad por los geólogos. Dichas perforaciones pudieran ser con cierto grado de inclinación el que puede variar a medida que se está perforando. (Leiva Rodríguez, 2007)

Las técnicas más utilizadas actualmente son la perforación con recuperación de testigos o diamantina y la recuperación de detritos o aire reverso. En la primera se utiliza una tubería engastada en diamantes en la punta, obteniéndose un cilindro de roca de un diámetro entre 2 y 5 pulgadas, en tanto que la segunda se

realiza con herramientas que van moliendo la roca, permitiendo obtener sólo trozos de roca de hasta 1 cm. (Leiva Rodríguez, 2007)

Un sondaje tiene como características que cuenta con un identificador (id) que puede ser una cadena de caracteres, cuenta además con una cabeza o boca; una lista de inclinometría la cual es una polilínea con cierto grado de inclinación; cuenta con una lista de muestras de por secciones del pozo. Puede contar con otros campos de interés del cliente. (Leiva Rodríguez, 2007)

Según varios autores la **minería** es el arte de laborar el conjunto de las minas y las explotaciones mineras, es visto también como la obtención selectiva de los minerales y otros materiales de la corteza terrestre, o como la técnica, actividad e industria que se ocupa de la explotación de las minas. (Ministerio de Justicia, 1995)

Existe también la **minería de datos**[2] la cual es proceso no trivial de identificación válida, novedosa, potencialmente útil y entendible de patrones comprensibles que se encuentran ocultos en los datos. Desde otro punto de vista es la integración de un conjunto de áreas que tienen como propósito la identificación de un conocimiento obtenido a partir de las bases de datos la toma de decisión. (Vallejos, 2006) Esta última definición se aleja del tema tratado por lo que es aceptada la primera definición de minería.

Una **mina** es una obra resultante del conjunto de excavaciones mineras e instalaciones superficiales y subterráneas que se realizan para la investigación y la explotación de un yacimiento mineral. Las minas son aquellos lugares subterráneos, generalmente ubicados a instancias de zonas montañosas, en los que se hallan principalmente materiales muy valiosos que pueden ser el origen de una cuantiosa riqueza, como ser aluminio, cobre, hierro, plomo, oro, entre otros. (Ministerio de Justicia, 1995)

Los **minerales** son sólidos homogéneos por naturaleza, con una composición química definida y unas propiedades dadas. Normalmente se forma mediante un proceso inorgánico, excepto en algunos casos. Son el sustrato inorgánico de la vida vegetal y animal, formadores del suelo y receptores últimos de la contaminación ambiental que tanto interesa al mundo y en particular, a los biólogos, geólogos, químicos y todos aquellos que se dedican al estudio del medio ambiente. (Ministerio de Justicia, 1995)

[2] *data mining* en inglés

La **Visualización en 2D**, tiene su base en la construcción de objetos en dos planos (X e Y o Ancho y Altura). Un ejemplo de este es la **Figura 1** en la cual se aprecia a través de la interpolación de varios puntos 3 perforaciones realizadas y delimitadas las concentraciones de mineral por colores.

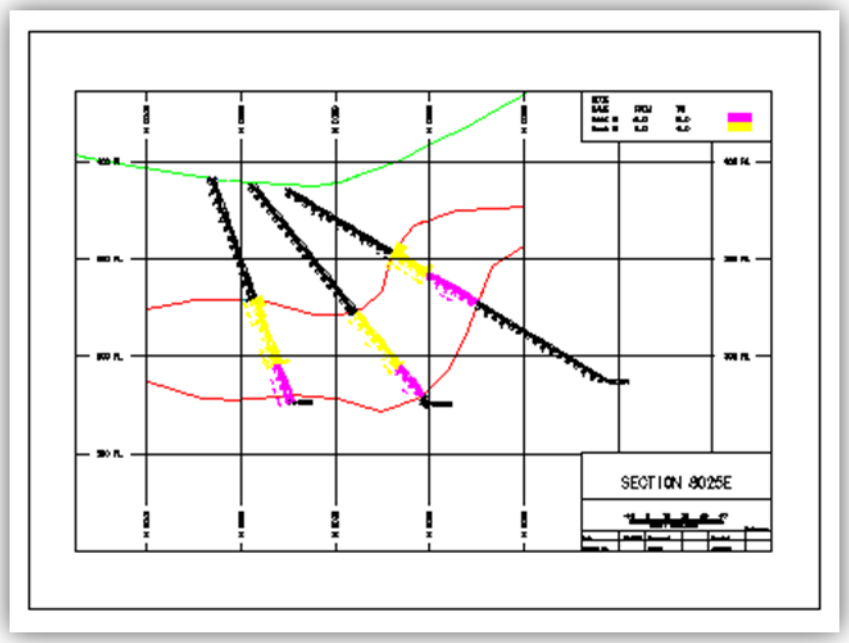

Figura 1 Representación en 2D de un Pozo de Sondeo

Se denomina **interpolación** a la obtención de nuevos puntos partiendo del conocimiento de un conjunto discreto de puntos. En ingeniería y algunas ciencias es frecuente disponer de un cierto número de puntos obtenidos por muestreo o a partir de un experimento y pretender construir una función que los ajuste.

El **punto** es un elemento geométrico adimensional, no tiene ni volumen, ni área ni longitud ni otro análogo dimensional, no es un objeto físico; describe una posición en el espacio, determinada respecto de un sistema de coordenadas preestablecido. (Iznaga Benitez, 2005)

Un punto puede determinarse con diversos sistemas de referencia:

- En el sistema de coordenadas cartesianas, se determina mediante las distancias ortogonales a los ejes principales, que se indican con dos letras o números: (x, y) en el plano; y con tres en el espacio (x, y, z).
- En coordenadas polares, mediante su distancia al centro y la medida angular respecto del eje de referencia: (r, θ).
- En coordenadas esféricas, mediante su distancia al centro y la medida angular respecto de los ejes de referencia: (r, θ, φ)
- En coordenadas cilíndricas, mediante coordenadas radial, acimutal y altura: (ρ, φ, z). (Iznaga Benitez, 2005)

En función de sus posiciones relativas, existen dos tipos de puntos: colineales y coplanarios. Los denominados colineales son aquellos contenidos en una recta, no importa cuántos puntos sean mientras estén alineados y dentro de la recta. Se denominan puntos coplanarios a aquellos que están contenidos en un mismo plano. (Iznaga Benitez, 2005)

En la minería se emplean las coordenadas polares (geológico) por lo que es preciso realizar conversiones a la hora del trabajo con los sistemas de cómputos a coordenadas cartesianas (geométricas) y una vez realizada estas operaciones convertir los ángulos a radianes que la forma en que se trabaja en los sistemas de cómputo. De ahí que:

azimut geológico a geométrico

$$\angle f = \begin{cases} 90 - \angle glsi \angle gl \leq 90 \\ 450 - \angle glsi \angle gl > 90 \end{cases}$$

elevación geológica a geométrica

$$\angle f = 90 - \angle gl$$

ángulos geométricos a radianes

$$\angle \theta = \angle gm \cdot {}^{\pi}/_{180}$$

$\angle f$ se refiere al ángulo resultante, $\angle gl$ se refiere al ángulo geológico, $\angle gm$ se refiere al ángulo geométrico

La **recta o línea recta**, es el ente ideal que se extiende en una misma dirección, contiene infinitos puntos. También se describe como la sucesión continua e indefinida de puntos en una sola dimensión, o sea, no posee principio ni fin. Un caso particular de sucesión infinita de puntos es la **polilínea**, la que se diferencia de la línea recta por extenderse en varias direcciones. (Iznaga Benitez, 2005)

Un **polígono** es una figura geométrica conformada segmentos no alineados, es decir lados de los cuales debe tener 3 o más. En ocasiones se les suele llamar polilínea cerrada. (Iznaga Benitez, 2005)

Una de sus clasificaciones es por su contorno:

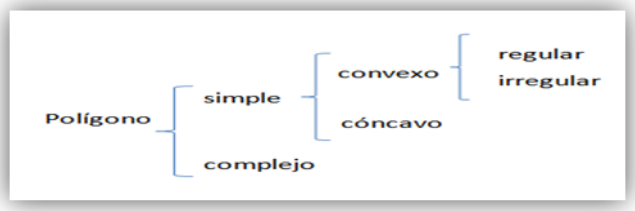

Figura 2 Clasificación de los polígonos según su contorno.

- simple, si dos de sus aristas no consecutivas no se intersecan (cortan).
- complejo, si dos de sus aristas no consecutivas se intersecan.
- convexo, si al atravesarlo una recta lo corta en un máximo de dos puntos.
- cóncavo, si al atravesarlo una recta puede cortarlo en más de dos puntos.
- regular, si tiene sus ángulos y sus lados iguales.
- irregular, si tiene sus ángulos y lados desiguales.

Figura 3 Ejemplo de Polígonos

El **Diseño de Mina** es la concepción de cómo debe quedar la mina al final de la extracción de sus reservas, para iniciar el diseño es necesario contar con leyes de corte que se aplicarán para todo el depósito o para cada una de las partes de depósito. (Geoestadística, 2010)

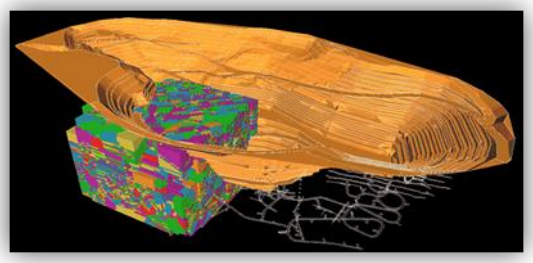

Figura 4 Diseño de una Mina para extraer un cuerpo mineral.

1.3. Soluciones Existentes

En la actualidad las empresas están en constantes cambios para modificar y mejorar las herramientas para minería subterránea y superficial, ya que la mayoría de las empresa están automatizando su información y necesitan mostrar en el software trabajos dinámicos, donde la premisa fundamental es lograr el mayor aprovechamiento de las reservas de la mina, ahorrar tiempo, costos y realizar una mejor gestión de los procesos mineros. A continuación, algunas de las alternativas imperantes en el mercado y las soluciones que ofrecen.

1.3.1. GEMCOM

Gemcom es un software de planificación de minas, apoyo a las operaciones a tajos abiertos y subterráneos. Las soluciones aportadas por esta aplicación las cuales abarcan desde la fase de

exploración hasta la reconciliación y balance metalúrgico a lo largo de la línea de producción hacen que sea un proyecto de exploración empleado en más de 110 países.

Esta aplicación presenta un entorno de trabajo agradable donde puede ser posible la selección de varios espacios de trabajo. Permite el manejo de los datos recolectados de manera manual o importando un fichero con la información, es posible realizar reportes rápidos o usando macros, previamente definidas. Pueden ser creados planos de sondajes y símbolos gráficos con leyendas, los que enriquecen los planos y secciones geométricas.

Todos los ficheros importados por esta aplicación son validados en busca de inconsistencias y duplicidades, permitiendo seguridad en la aplicación. Como mismo son importado ficheros es posible crearlos para el trabajo con otras aplicaciones o simplemente para ser almacenados de forma segura.

El acceso al archivo GCDBaa.mdb [3]contiene todos los tipos diferentes de datos los cuales son agrupados en espacios de trabajo que a su vez están conformados por un conjunto de tablas con campos asociados.

Una tabla puede ser un:

- ✓ Encabezado.
- ✓ Intervalo.
- ✓ Distancia.
- ✓ Punto.
- ✓ Polígono.
- ✓ Usuario.

Cada tipo de espacio de trabajo tiene una estructura mínima única, la cual consiste en un número requerido de tablas y ciertos campos dentro de estas tablas.

1.3.2. SURPAC

Al igual que GEMCOM, esta aplicación ayuda a los usuarios a agregar valor a sus proyectos de recursos minerales y es una guía a la industria de explotación minera.

[3] Archivo de Microsoft Access que contiene las salvas de la base datos.

Esta permite que sea realizada la gestión de un recurso en específico de acuerdo a las especificaciones del usuario y los datos recolectados. Cuenta con una herramienta de visualización, capaz de mostrar el cuerpo mineral durante el proceso de extracción de forma animada. Es posible localizar o ubicar un determinado cuerpo mineral, lo cual es importante a la hora de estimar el tiempo aproximado para su extracción. Esta aplicación permite generar reportes o informes por periodo, polígonos o bancos.

SURPAC presenta la capacidad de trabajar con los formatos nativos, es decir ficheros, además con ficheros de otros sistemas tales como: AutoCAD y Microstation. Para su gestión utiliza Microsoft Excel, aunque pueden hacerse planificaciones con Microsoft Project permitiendo refinar los programas realizados.

Los archivos o tablas de la base de datos son almacenados en un formato binario el cual ahorra espacio en disco y mejora la velocidad de acceso y procesamiento de los datos.

1.3.3. DATAMINE

Es el líder mundial en Software Integrado para la industria de los recursos naturales, con más de 1000 sistemas en diario uso en más de 45 países a través del mundo. Los usos más comunes del sistema son; la captura y análisis de la información, exploración, geología, geoquímica, mecánica de rocas, topografía, modelamiento geológico, diseño de mina a cielo abierto y subterráneas, planeamiento minero, y áreas relacionadas a los estudios ambientales.

Datamine Studio está construido en base a un núcleo central llamado “Core” que provee una excelente administración de datos a través de un sistema de base de datos relacional con completos despliegues gráficos, estadística y administración de datos de sondajes.

Esta aplicación permite la importación de los archivos que contienen la información de la orientación y localización de los sondajes, así como las propiedades de las muestras constituyentes. Esta característica ayuda a un mejor procesamiento de los datos, ya es que posible realizar estadísticas de sondaje, compositar las muestras que no es más que regular el largo o bien de la altura (distancia en la vertical) de las muestras, facilitando poder ver las franjas del mineral desde cualquier dirección. Es posible nombrar y almacenar vistas específicas, de tal forma que el usuario puede regresar a las posiciones anteriormente almacenadas en cualquier minuto.

Cuenta con un sistema de base de datos relacional con completos despliegues gráficos, estadística y administración de datos de sondajes, lo que permite una excelente administración de datos.

Las funciones de manejo de la base de datos y de los archivos de Datamine, le permitirán un completo control sobre sus datos. Datamine ofrece un comprensivo conjunto de herramientas para bases de datos relacional para comparación, actualización, extracción y unión de archivos. Los archivos o tablas de la base de datos son almacenados en un formato binario el cual ahorra espacio en disco y mejora la velocidad de acceso y procesamiento de los datos.

1.3.4. Aporte de las soluciones identificadas

Las soluciones identificadas hasta este momento brindan una visión más detallada de cómo debería ser un software que agílese las labores mineras. En las que se destaca lo siguiente:

- ✓ La interfaz de las aplicaciones es amigable y se encuentra dividida por secciones de trabajo las que se encuentran en relación con las diferentes funcionalidades de las aplicaciones.
- ✓ Permiten la recolección de los datos de forma manual o mediante la importación de ficheros con datos de diversos formatos, es posible realizar reportes rápidos o usando macros, previamente definidas.
- ✓ Permiten exportar o importar ficheros de diferentes formatos y creados por otras aplicaciones mineras.
- ✓ Cuentan con un sistema de base de datos relacional con completos despliegues gráficos, estadísticos y administración de datos de sondajes, los que permiten una excelente administración de datos.
- ✓ Cuenta con herramientas de visualización las que son capaces de mostrar el cuerpo mineral durante el proceso de extracción.
- ✓ Permiten que sea realizada la gestión de un recurso en específico de acuerdo a las especificaciones del usuario y los datos recolectados.

Estas características vistas de algún modo en las soluciones identificadas hasta el momento sirven como guía a la hora de desarrollar el componente de visualización.

1.4. Conclusiones Parciales

Las soluciones identificadas hasta el momento, y específicamente las funcionalidades identificadas anteriormente pueden servir como base a la hora de realizar un software que sirva de apoyo a la minería sin pasar por alto las necesidades del usuario.

Estas aplicaciones fueron creadas con el objetivo de asegurar el éxito del negocio minero. Pero tienen como inconveniente que son software propietario lo cual provoca que sea engorrosa la adquisición de las licencias por su alto costo, además de imposibilitar tener acceso pleno al código fuente. Lo que conllevaría a que no se puedan ajustar a las especificaciones de un determinado cliente o recurso.

Capítulo 2: Tendencias y tecnologías actuales a desarrollar.

2.1. Introducción

En el capítulo anterior fueron identificados algunas aplicaciones que servían de apoyo a la minería, en el presente capítulo se abordaran las vías para realizar un componente de software que sirva de apoyo al proyecto Minería. Son determinadas y justificadas las principales técnicas, tecnologías, herramientas y metodologías que se utilizaran para el desarrollo del componente.

2.2. Librerías para la visualización en 2D.

OpenGL y VTK, forman parte de una múltiple variedad de bibliotecas que facilitan el propio desarrollo de modelado 2D. Estas bibliotecas consideradas como unas de las más eficientes y más poderosas, dichas implementaciones, generan no solo la oportunidad de la comprobación, sino de la creación y desarrollo, a diferencia de otras librerías estas son de código abierto e implementación gratuita.

2.2.1. OpenGL

Definido como el primer ambiente de desarrollo para aplicaciones interactivas graficas en 2D, introducido en 1992 y escrito bajo el estándar C[4]. Permite adoptar un desarrollo veloz e innovador incorporando herramientas de renderización, trazado de textura, efectos especiales, cuenta con una alta calidad visual y de desempeño, es estable, puede ser ejecutado tanto en computadoras convencionales como en supercomputadoras, lo que trae como resultado que sea escalable a cualquier máquina. (Moreno Berzal, 2004)

Capacidades de Graficar y Renderizado

OpenGL permite la descripción matemática de objetos a partir de sus primitivas. Permite la organización de estos objetos en el espacio al tiempo que permite modificar el punto de vista y las propiedades de la cámara.

[4]ANSI C: es un estándar publicado por el Instituto Nacional Estadounidense de Estándares (ANSI), para el lenguaje de programación C

Permite calcular el color de los píxeles por asignación directa de color, por cálculo de iluminación, por mapeo de texturas o por una combinación de los tres. Permite convertir la descripción matemática de objetos en formas y colores. Permite también la simulación de "efectos atmosféricos" como cubo de profundidad, transparencia, neblina y renderizado volumétrico. (Moreno Berzal, 2004)

Algunas cualidades generales:

- ✓ Primitivas geométricas, permiten construir descripciones matemáticas de objetos. Las actuales primitivas son: puntos, líneas, polígonos, imágenes y mapas de bits.
- ✓ Mapeado de texturas, que ayuda a traer realismo a los modelos por medio del dibujo de superficies realistas en las caras de los modelos poligonales.
- ✓ Los planos de plantilla, permiten restringir el trazado a ciertas regiones de la pantalla.
- ✓ Las listas de visualización permiten almacenar comandos de dibujo en una lista para un trazado posterior, cuando las listas se usan apropiadamente puedan mejorar mucho el rendimiento de las aplicaciones.
- ✓ Características de feedback[5], selección y elección que ayudan a crear aplicaciones que permiten al usuario seleccionar una región de la pantalla o elegir un objeto dibujado en la misma. El modo de feedback permite al desarrollador obtener los resultados de los cálculos de trazado.
- ✓ Primitivas de *Raster* (bitmaps y rectángulos de píxeles)
- ✓ Operaciones con píxeles (CARRANZA ATHÓ, y otros, 2006)

2.2.2. VTK

Visualization Tool Kit (VTK), es un software distribuido bajo licencia Open Source, su biblioteca está basada en objetos, estrechamente relacionada con el hardware, está desarrollada completamente en C++. Las Aplicaciones pueden escribirse en lenguaje de programación C++, Tcl, Java o Python. La instalación de VTK requiere soporte para OpenGL, lo cual puede ser con una versión comercial.

VTK contiene a una gran variedad de algoritmos de la visualización incluso el métodos escalares, vectoriales, de tensores, de textura, y volumétricos y técnicas de modelado avanzadas como el modelado

[5]Feedback: en español retroalimentación.

implícito, reducción de polígonos, el aplanando de mallas, recorte, contorneando, y triangulación de Delaunay. (Moreno Berzal, 2004)

Capacidades de Graficación y Render

VTK es un sistema real de visualización ya que incluye capacidades para desplegar primitivas geométricas al tiempo que incluye algoritmos para la visualización de campos escalares, campos vectoriales, visualización de tensores, mapeo de texturas, así como métodos volumétricos para representación de superficies y volúmenes. (CARRANZA ATHÓ, y otros, 2006)

La ventaja de esta arquitectura radica en la posibilidad de construir algoritmos eficientes (tanto en procesamiento como en consumo de memoria) en un lenguaje compilado como C++ y mantener al mismo tiempo las facilidades de desarrollo rápido que proporciona los lenguajes interpretados (inexistencia de ciclo de compilación y enlazado, herramienta simple pero a la vez potente y acceso a facilidades para el desarrollo de interfaces gráficas). (Suarez Quirós, y otros, 2009)

Una escena es el resultado de la interacción de varios objetos por ejemplo los "*data objects*[6]".Los *data objects* consisten en una estructura (de puntos y celdas) geométrica y topológica, así como en unos atributos de datos que pueden ser, por ejemplo, escalares o vectores. Los atributos de datos pueden ser asociados con los puntos o celdas del *dataset*. Las celdas son agrupaciones topológicas de puntos, que forman las unidades del *dataset* y se usan para interpolar información entre puntos. Los objetos *dataset* en VTK se muestran en la **Figura 5**. (Moreno Berzal, 2004)

[6] Data objects: Estructura de representación de los datos de un objeto.

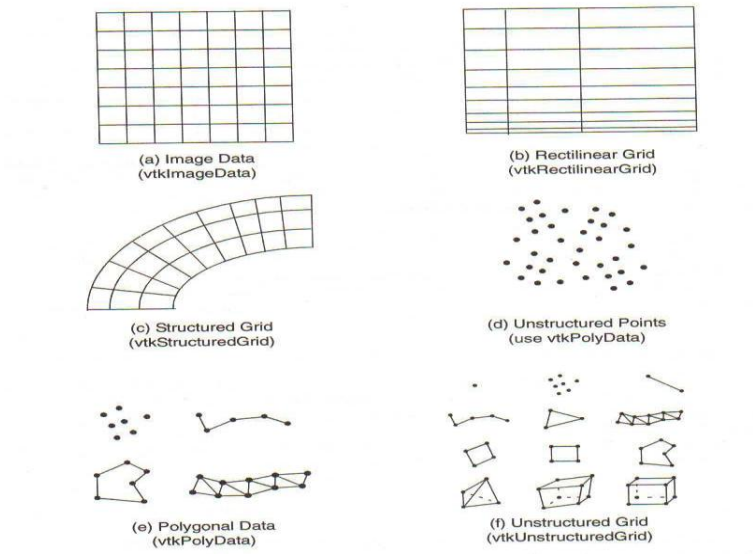

Figura 5 Dataset empleado por VTK.

Para el desarrollo del componente se selecciona la librería VTK debido a que el diseño de esta biblioteca está fuertemente basado en objetos. Además la manera de codificar aplicaciones es más sencilla y rápida que OpenGL, ya que es una biblioteca con rutinas de alto nivel.

2.3. Lenguajes de Programación

Actualmente existen diferentes lenguajes de programación para el desarrollo de aplicaciones de escritorio, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Se han investigado los lenguajes de programación más utilizados actualmente entre los que se cuenta: C++, C# y Java, para seleccionar el más adecuado para implementar en componente.

2.3.1. Lenguaje de programación C++

El C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Las principales características del C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica. Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel: Posibilidad de redefinir los operadores (sobrecarga de operadores) Identificación de tipos en tiempo de ejecución (RTTI).

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje. (Schildt, 2001)

2.3.2. Lenguaje de programación C#

C Sharp, es actualmente uno de los lenguajes de programación más populares en informática y comunicaciones que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. Es diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg. La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. (González Seco, 2001)

Principales características de C#: (Cortijo Bon, 2005; González Seco, 2001)

- ✓ Un programa en C# no necesita de ficheros adicionales al propio código fuente, como los ficheros de cabecera (.h) de C++.
- ✓ Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos.
- ✓ C# tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos.
- ✓ A diferencia de C++, en C# los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada *System.Object*, por lo que dispondrán de los miembros definidos en ésta clase (es decir, serán “objetos”).
- ✓ En principio, en C# el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros.

- ✓ C# mantiene una sintaxis muy similar a C++ o Java que permite, bajo ciertas condiciones, incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes.

2.3.3. Lenguaje de programación Java

Es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras. Sun describe a Java como "simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico". (García de Jalón, y otros, 2000)

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir que programa hecho en Java podrá funcionar en cualquier ordenador del mercado. Esto lo consigue porque se ha creado una Máquina Virtual de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java. (Sun Microsystems, 2010)

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación a redes. De portátiles a centros de datos, de consolas de juegos a súper equipos científicos, de teléfonos móviles a Internet, Java está en todas partes. (Sun Microsystems, 2010)

2.3.4. Lenguaje de Programación a Utilizar

Es seleccionado el lenguaje de programación C++ ya que se demanda que el componente sea desarrollado sobre un lenguaje estandarizado que pueda compilarse en varias plataformas y que sea rápido en cuanto a ejecución.

Las caracterizas descritas anteriormente están presentes en este lenguaje, las cuales lo hacen versátil y uno de los más empleados en la actualidad, ya que es un lenguaje capaz de utilizar características de bajo nivel para realizar implementaciones óptimas logrando que los programas sean más compactos y rápidos. Es esta una de las razones por las que a menudo son utilizados en el desarrollo de video juegos por su

entendimiento con el hardware. Una de las librerías de visualización descrita anteriormente (VTK) fue escrita totalmente en C++, por lo que no existen problemas de integración con ella.

Es un lenguaje que cuenta con gran cantidad de bibliografía y materiales didácticos. Además se cuenta con la experiencia en el desarrollo de este lenguaje por parte del equipo de desarrollo del Proyecto Minería.

2.4. Herramientas de Desarrollo

Un Entorno de Desarrollo Integrado o IDE[7], es un programa informático compuesto por un conjunto de herramientas de programación. Es una herramienta que puede ser exclusiva de un lenguaje o varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE proporcionan un entorno de trabajo amigable.

2.4.1. Herramienta de Desarrollo Qt

Es una amplia plataforma[8] de desarrollo que incluye clases, librerías y herramientas para la producción de aplicaciones de interfaz gráfica en C++ que pueden operar en varias plataformas. Qt dispone de una amplia gama de herramientas que facilitan la creación de formularios, botones y ventanas de dialogo con el uso del ratón. (Thelin, 2007)

Ventajas de Qt

- ✓ Qt es completamente gratuito para aplicaciones de código abierto.
- ✓ Las herramientas, librerías y clases están disponibles para casi todas las plataformas Unix y sus derivados (como Linux, MacOS X, Solaris) como también para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código y la aplicación se verá y actuará mejor que una aplicación nativa.
- ✓ Qt tiene una extensa librería con clases y herramientas para la creación de ricas aplicaciones. Estas librerías y clases están bien documentadas. (Thelin, 2007)

[7] IDE: acrónimo en inglés de *Integrated Development Environment*.
[8] Plataforma: una **plataforma** es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible.

Para el desarrollo de aplicaciones basadas en la plataforma Qt se emplearan varias herramientas dentro de las que se pueden citar:

Para la familia Microsoft Windows

- ✓ Qt 4.0 y MinGW.
- ✓ GNU make.
- ✓ Cualquier editor de texto.
- ✓ Otra opción que lo incluye todo es instalar el entorno de desarrollo Qt SDK.

Para la familia Unix y derivados

- ✓ Qt 4.0.
- ✓ Las utilidades para Desarrollo, normalmente esta es una opción que puede ser seleccionada para ser instalada con tu distribución. (Thelin, 2007)

2.4.2. Herramienta de Desarrollo NetBeans

NetBeans IDE es un excelente entorno de desarrollo, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Este ofrece las funciones de los IDE avanzados, como diseño de interfaz, creación automática de propiedades y clases, asistentes para la conexión con bases de datos. Puede obtener las herramientas que necesite para crear aplicaciones profesionales para el escritorio, la empresa, la web y equipos móviles con el lenguaje Java, C/C++, y Ruby. Muy fácil de instalar y de uso instantáneo y se ejecuta en varias plataformas incluyendo Windows, Linux y Mac OS X y Solaris. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. (Sun Microsystem, 2010)

Ambos productos son de código abierto y gratuito para el uso tanto comercial como no comercial. El código fuente está disponible para su reutilización de acuerdo con la *Common Development and Distribution License* (CDDL). (Sun Microsystem, 2010)

2.4.3. Herramienta de Desarrollo Eclipse

Eclipse provee un conjunto de herramientas para administrar espacios de trabajo; construir, correr y depurar aplicaciones; compartir artefactos con un equipo y hacia versión de código. Es una plataforma que está diseñada para ser infinitamente extendida con cada vez más sofisticadas herramientas. Está construido sobre un mecanismo para el descubrimiento, integración y ejecución de módulos llamados *plugins.* Muchos *plugins*, comúnmente sin relación alguna, pueden ser instalados en una misma instancia de Eclipse, y convivir y cooperar sin problemas para ejecutar una cierta tarea. La clase de producto final incluye aplicaciones IDE, también denominados *rich clients* (clientes ricos), que se benefician del diseño de la plataforma de Eclipse y sus componentes. (The Eclipse Foundation, 2009)

La plataforma Eclipse está habilitada para afrontar las siguientes necesidades:

- ✓ Soportar la construcción de gran variedad de herramientas de desarrollo.
- ✓ Soportar las herramientas proporcionadas por diferentes fabricantes de software independientes.
- ✓ Soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML, y GIF).
- ✓ Facilitar una integración transparente entre las herramientas y tipos de contenidos sin tener en cuenta al proveedor.
- ✓ Proporcionar entornos de desarrollo gráfico (GUI) o no gráficos.
- ✓ Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows y Linux.

2.4.4. Herramientas de Desarrollo a Utilizar

Se selecciona como herramienta de desarrollo Qt, ya que es multiplataforma, se puede integrar con las librerías de VTK las que están bien documentadas ya que fue desarrollada sobre C++, provee a los usuarios de una plataforma fácil y amigable a la hora de trabajar con C++. Qt cuenta con varias clase definidas por él, que enriquecen las librerías del C++.Además presenta algoritmos que facilitan la implementación del negocio minero, los que han sido utilizados en el desarrollo de software de gran reconocimiento y prestigio a nivel mundial en el campo de la minería.

2.5. Metodologías de Desarrollo.

Una Metodología de Desarrollo no es más que un conjunto de técnicas, procedimientos, herramientas y soporte documental que sirve de apoyo a los desarrolladores a la hora de realizar un software. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo (Pressman, 2006). Son variadas las opciones que se pueden tener en cuenta sobre metodología de desarrollo, a continuación se describen algunas.

2.5.1. Programación Extrema (*eXtreme Programming* - XP)

Es una de las metodologías más exitosas en la actualidad y forma parte del grupo de las Metodologías Ágiles, está más orientada a la generación de código con ciclos muy cortos de desarrollo, se dirige a equipos de desarrollo pequeños, hace especial hincapié en aspectos humanos asociados al trabajo en equipo e involucra activamente al cliente en el proceso. XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software. (Pressman, 2006)

Lo aspectos fundamentales en esta metodología son:

- ✓ La simplicidad al desarrollar y codificar los módulos del sistema.
- ✓ La comunicación entre los usuarios y los desarrolladores.
- ✓ La retroalimentación concreta y frecuente del desarrollo, el cliente y los usuarios finales.

2.5.2. Proceso Unificado de Desarrollo (*Rational Unified Process* - RUP)

Es un proceso de desarrollo de software que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. RUP es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. (Pressman, 2006)

Creado por Jacobson, Rumbaugh y Booch. Preparado para desarrollar grandes y complejos proyectos. Unifica criterios sobre elementos de metodologías que la precedieron

Las características principales de RUP son:

- ✓ **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan, es una facilidad que el software debe proveer a sus usuarios. A partir de aquí los casos de uso guían el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- ✓ **Centrado en la arquitectura**: La arquitectura involucra los elementos más significativos del sistema. Surge de las necesidades de la empresa y se refleja en los casos de uso. También se ve influida por otros factores como las plataformas de software, los sistemas operativos, protocolos y requerimientos no funcionales. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML).
- ✓ **Iterativo e incremental:** RUP divide el proceso en cuatro fases, las cuales se desarrollan en iteraciones de las actividades principales básicas de cualquier proceso de desarrollo. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Las iteraciones están controladas y se ejecutan de una forma planificada.

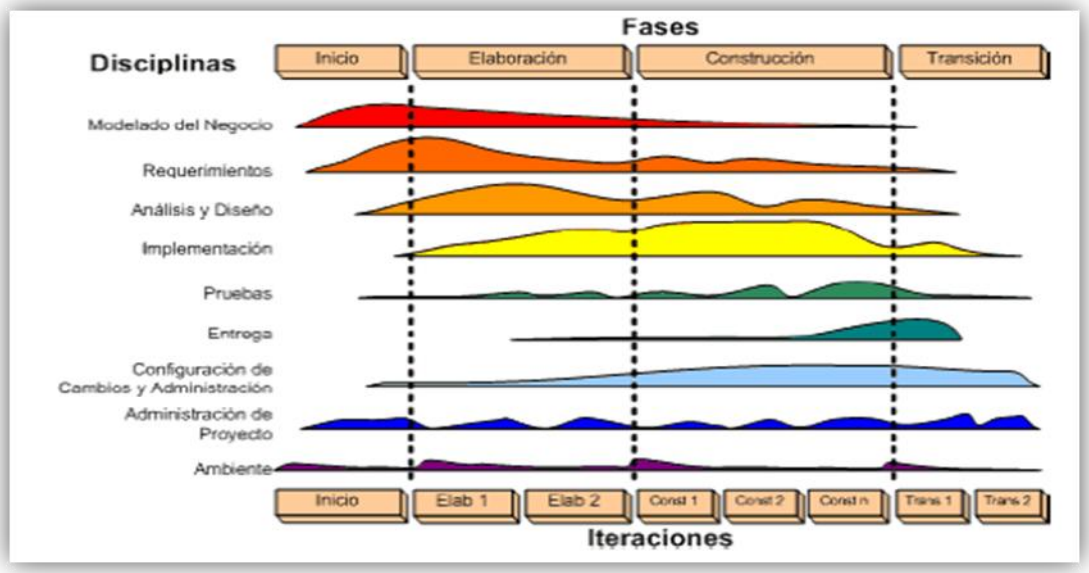

Figura 6 Fases e Iteraciones de la Metodología RUP

2.5.3. Proceso Unificado Abierto (*Open Unified Process- OpenUP*)

OpenUP es un Framework[9] de procesos de desarrollo de software de código abierto. Es un proceso modelo y extensible, dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental, y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. Este proceso de desarrollo unificado está basado en RUP, desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad. Plantea que sólo se debe usar los procesos que sean necesarios, que no se debe usar muchos artefactos, además debe acoplarse a las necesidades del usuario permitiendo ser modificado y extendido. OpenUP está caracterizado por cuatro principios básicos, los cuales son:

- ✓ Colaboración para unificar intereses y compartir conocimientos.
- ✓ Equilibrio de prioridades competentes a maximizar el valor de los involucrados con el resultado del proyecto.
- ✓ Enfoque en la articulación de la arquitectura.
- ✓ Desarrollo continúo para obtener retroalimentación y realizar las mejoras respectivas.

2.5.4. Metodología de Desarrollo a Utilizar

Se selecciona como metodología de desarrollo RUP, ya que provee un entorno de proceso de desarrollo configurable, basado en estándares, permite tener claro y accesible el proceso de desarrollo que se sigue, además de ser configurado según las necesidades de la organización y del desarrollo del componente. Gracias a la característica que es iterativo e incremental permite la incorporación de otras funcionalidades a lo largo de sus ciclos de vida. Ya que se centra en la tecnología Orientada a Objetos la que permitirá la reutilización de clases y componentes. Realiza el proceso de desarrollo de manera entendible ya que genera un gran número de artefactos y documentación correspondiente.

[9] Framework: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas similar.

2.6. Herramientas CASE

Se puede definir a una herramienta CASE (en español, ingeniería de software asistida por computadora, en inglés, *Computer Arded Software Engineering*) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante los pasos del ciclo de vida de desarrollo de un software. (Kendall, y otros, 2006)

Entre las principales ventajas que aporta la utilización de herramientas CASE, se tienen que permiten:

- ✓ El incremento en la velocidad de desarrollo de los sistemas.
- ✓ A los analistas tener más tiempo para el análisis y diseño además de minimizar el tiempo para codificar y probar.
- ✓ Automatizar el dibujo de diagramas.
- ✓ Ayudar en la documentación del sistema.
- ✓ Ayudar en la creación de relaciones en la base de datos.
- ✓ Generar estructuras de código.
- ✓ Aumentar la productividad.

Esto se consigue a través de la automatización de determinadas tareas, como la generación de código y la reutilización de objetos o módulos. Entre las herramientas más utilizadas están:

2.6.1. Rational Rose Enterprise Edition

Es una herramienta CASE basada en UML[10] que permite crear los diagramas que se van generando durante el proceso de ingeniería en el desarrollo del software. Es completamente compatible con la metodología RUP, brinda muchas facilidades en la generación de la documentación del software que se están desarrollando, además posee un gran número de estereotipos predefinidos que viabiliza el proceso de modelación del software. Es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño, pero tiene la limitación de que aún hay varios lenguajes de programación que no

[10] UML: Lenguaje Unificado de Modelado

soporta o que sólo lo hace a medias (Fernández del Monte, 2010; Rational Software Corporation, 2001)

2.6.2. Visual Paradigm

Visual Paradigm es una de las herramientas CASE del mercado, considerada muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite graficar los diferentes diagramas UML, revertir y generar código fuente para Java, C++, PHP, XML Schema, Python y Corba IDL. Es una herramienta que puede ser empleada en varios sistemas operativos.

Visual Paradigm incluye los objetos más recientes de UML además de diagramas de casos de uso, diagramas de clase, diagramas de componentes, ofrece soporte para Rational Rose, integración con Microsoft Visio, además permite generar reportes y documentación en HTML/PDF. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. (Company Headquarters, 2006)

Una de sus principales ventajas es que incorpora el soporte para trabajo en equipo, permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. (Company Headquarters, 2006)

2.6.3. Herramienta CASE a Utilizar

Es seleccionado Visual Paradigm como herramienta de CASE ya que es multiplataforma a diferencia del Rational, es capaz de generar código C++, esta herramienta es partidaria de la filosofía de la estructura de interfaz, dividiendo el trabajo de modelado en interfaces existiendo gran similitud entre estas y las fases de RUP. Visual Paradigm utiliza UML como lenguaje de modelado garantizando que sea fácil de utilizar a la hora de entender y realizar los diversos diagramas.

2.7. Conclusiones Parciales

Con el transcurso del tiempo las tecnologías van cambiando constantemente, es por ello que herramientas utilizadas hasta hace poco tiempo, ya hoy no encuentran espacio entre las existentes. Existe una diversidad de estas herramientas que pueden ser ajustadas a proyectos específicos de acuerdo a las necesidades de cada uno. En la presente investigación será empleado como librería de visualización VTK, como lenguaje de programación C++ sobre el entorno de desarrollo Qt, se modelará sobre Visual Paradigm empleando la metodología RUP el que generara la documentación necesaria del componente.

Por lo que para la elección de las herramientas y tecnologías se tuvo en cuenta que las mismas fueran multiplataforma con lo que se garantizaría que no existieran inconvenientes en el desarrollo del componente, al migrar la universidad a una plataforma soberana y otras características que fueron vistas anteriormente.

Capítulo 3: Presentación de la solución propuesta.

3.1. Introducción

En este capítulo se describe la solución propuesta la que es vista a través de los procesos fundamentales del negocio y el Modelo de Dominio, en este último se tiene en cuenta las interacciones entre los procesos. Son vistos también los requisitos funcionales y no funcionales del componente de modelado y visualización, los que a lo largo del proceso iterativo e incremental son refinados para lograr el diagrama de Casos de Usos del Sistema (CUS) el que contiene actores, casos de uso y sus relaciones y sirve como entrada fundamental para el análisis, diseño del componente.

Es fundamentado también el proceso de análisis y diseño del componente realizado como parte de la propuesta de solución, modelándose los artefactos necesarios que contribuyen a la implementación del sistema. Estos artefactos son el Modelo de Análisis y el Modelo de Diseño, los cuales se encargan de describir en términos de clases cómo deben funcionar los casos de uso del sistema. Aquí también se logra representar la colaboración entre estas clases y se exponen los patrones de diseño y arquitectura que se utilizan en la elaboración de las mismas.

3.2. Procesos del Negocio

El proceso de Modelado y visualización en 2D de objetos geológicos se fundamenta en la utilización de varios subprocesos, pero siempre partiendo sobre la base de cuatro ficheros fundamentales y uno opcional que se describen a continuación:

- ✓ **collars.txt:** Contiene las coordenadas X, Y y Z de los pozos de sondaje.
- ✓ **surveys.txt:** Contiene el azimut e Inclinación de cada muestra del sondaje
- ✓ **assays.txt:** Contiene campos de interés de las muestras tales como la concentración de un mineral determinado.
- ✓ **geology.txt:** Contiene la clasificación geológica (Litológica) de las muestras.
- ✓ **topología.txt (opcional):** contiene información sobre el relieve donde se encuentran ubicados los sondajes.

Estos ficheros cuentan con información valiosa sobre cada sondaje, es por esta razón que una vez importados es posible realizar las transformaciones necesarias para la representación de cada pozo y el trabajo visual o la creación de una escena que cuenta con objetos geológicos. La creación de una escena es posible realizarla de la forma descrita anteriormente o a través de la Interfaz de Comunicación (componente base o núcleo) que proporcionaría un objeto con toda esa información.

Entre las funciones realizadas sobre los objetos ya representados se encuentra el **zoom** la cual no es más que una transformación realizada a la escena tanto acercando o alejando los objetos de esta. El **paneo** consiste en la obtención de un punto de la escena y trasladarla en un espacio determinada de la misma. La **vista de planta** y las **vistas de perfil** son representaciones de la escena en dos ejes de coordenadas las que pueden ser en los ejes X-Y, Y-Z y X-Z. Estas funciones son gestionadas por una entidad principal que es la encargada de conectarse con el componente base.

Una vez vista las interacciones de los distintos procesos se puede arribar a la creación de un Modelo de Dominio o Dominio del Problema, como suele ser abordado por algunos autores, el cual es un esquema bastante general de cómo opera el negocio.

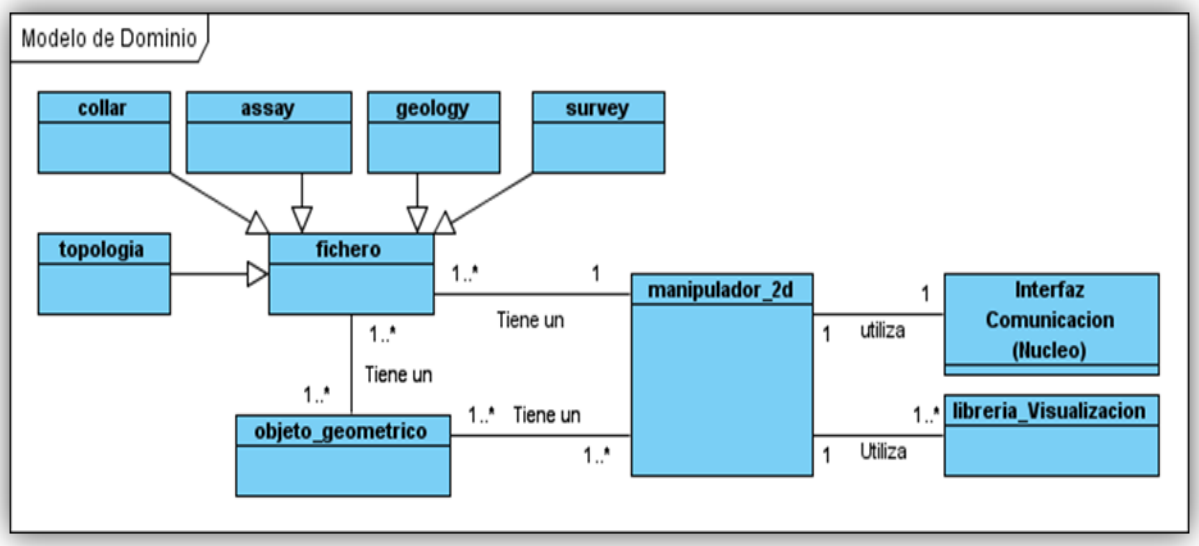

Figura 7 Modelo de Dominio del Componente de Modelado y Visualización en 2D.

3.3. Especificación de Requisitos

La Especificación de Requisitos de Software es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye requisitos funcionales (RF) que describen las funciones y servicios que se espera que la aplicación ofrezca a los usuarios finales. También contiene requisitos no funcionales (RNF) que imponen restricciones en el diseño e implementación de las funcionalidades ofrecidas. En este apartado, se registran los requisitos capturados, y se especifican las capacidades operacionales y funcionales que el sistema deberá tener con el mayor detalle posible.

3.3.1. Requisitos Funcionales

- RF1- Visualizar Vista Aérea o de Planta.
- RF2- Visualizar las Vistas de Secciones o Perfiles
- RF3- Obtener el *Bounding box.*
- RF4- Representar Objetos Geométricos.
    - RF4.1 Representar Puntos.
    - RF4.2 Representar Polilínea.
    - RF4.1 Representar Polígono.
- RF5- Permitir la realización de Zoom a las vistas
    - RF5.1- Escalar Modelo.
- RF6- Permitir la realización de Paneo a las vistas.
    - RF6.1- Trasladar Modelo.
- RF7- Seleccionar Objetos.
- RF8- Eliminar Objetos.
- RF9- Definir Plano.
- RF10- Actualizar Vistas.

3.3.2. Requisitos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Son esas características que posibilitan que el producto sea atractivo, usable, rápido, confiable, etc. Usualmente, están vinculados a requerimientos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Requerimientos de Apariencia o Interfaz externa:

- RNF1: El diseño del Sistema debe ser sencillo y funcional, de fácil manejo y reconocimiento visual a través de elementos visibles que identifiquen cada una de sus acciones.

Restricciones en el diseño e implementación:

- RNF2: El sistema estará implementado en lenguaje C++, utilizando como IDE de desarrollo Qt. Para la modelación de la arquitectura se utilizará el *Visual Paradigm*.

Requerimientos de Seguridad:

- RNF3: El Sistema debe garantizar que la información sea registrada, visualizada, eliminada y actualizada de forma segura.

Requerimientos de Usabilidad:

- RNF4: El componente luego de instalada, deberá ejecutarse con calidad en cualquier plataforma.

Requerimientos de Rendimiento:

- RNF5: Se debe contar con un rápido procesamiento de los datos y con un tiempo de respuesta mínimo.

Requerimientos de Soporte:

- RNF6: El Sistema debe ser:
  - ✓ Fácil instalación, configuración y puesta en marcha.
  - ✓ De arquitectura abierta y distribuida, modular, de capacidad escalable.
  - ✓ Programado orientado a objeto.

Requerimientos de Portabilidad:

- RNF7: El Sistema debe ser multiplataforma, debe ejecutarse de manera óptima sin importar el Sistema operativo que utilice el usuario.

3.4. Modelado del sistema

El sistema propuesto permitirá el desarrollo del componente de modelado y visualización en 2D al proyecto minería, el cual agilizará las labores de modelado de los pozos de sondaje y servirá para la toma de decisiones. A continuación se expone el diagrama de casos de uso del sistema para ofrecer una visión general de las principales funcionalidades que serán implementadas en el mismo.

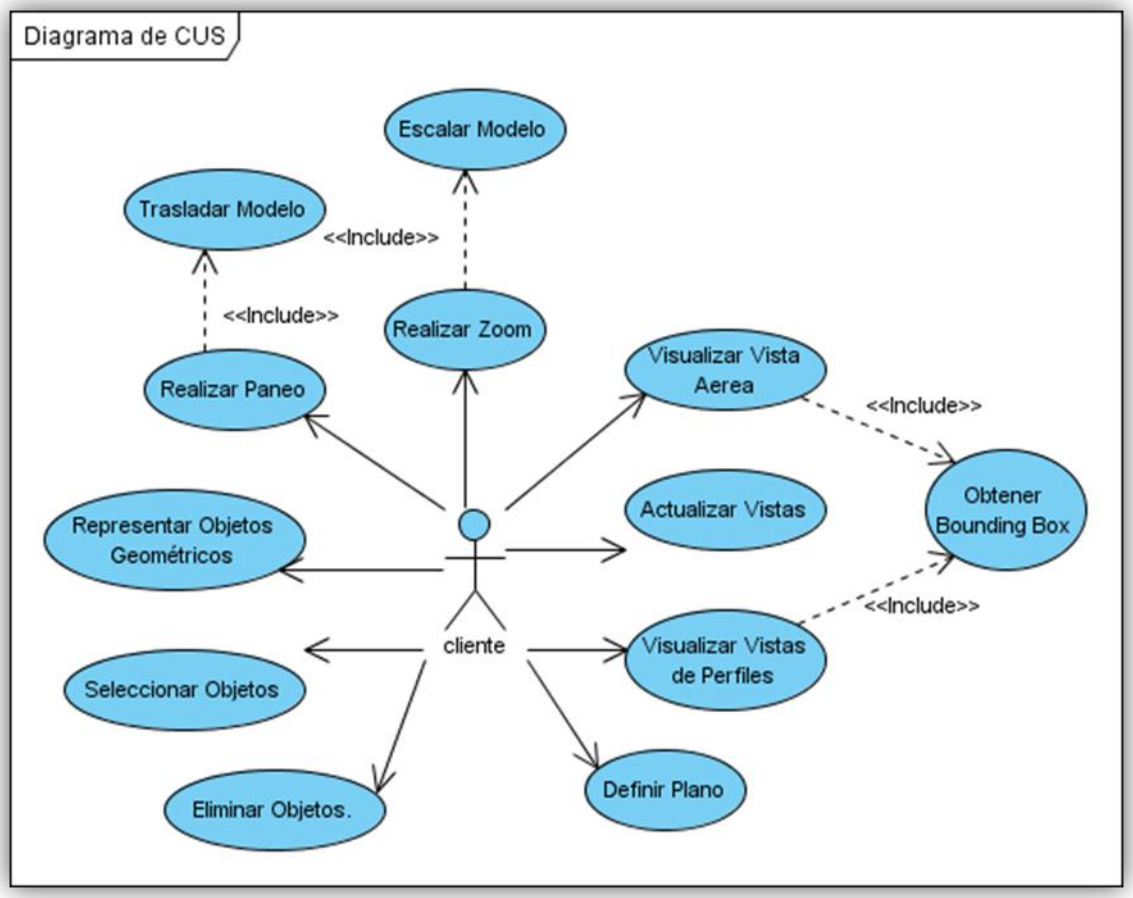

Figura 8 Diagrama de CUS del Componente de Modelado y Visualización en 2D.

En la figura anterior se muestra el Diagrama de CUS donde se tienen los Casos de Usos el actor que los inicializa y el tipo de relación que se establece entre ellos. Donde los Casos de Usos Visualizar Vista Aérea, Vista de Perfiles y Representar Objetos Geométricos son los Casos de Usos Arquitectónicamente Significativos (CUAS).

3.4.1. Actores del Sistema

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema. (JACOBSON, y otros, 2000)

Actor	Descripción
Cliente	Es el especialista encargado de interactuar con el sistema, el que le brinda al sistema los datos necesarios para que el sistema pueda gestionar las diferentes vistas, generar reportes, así como es el encargado de hacer zoom y paneo. En el momento y región deseado

Tabla 1 Actores del Sistema

3.4.2. Descripción de los Casos de Uso del Sistema

Las descripción de los caso de uso representan los artefactos narrativos que describen el comportamiento del sistema desde el punto de vista del usuario, son las respuestas a los requerimientos funcionales.

A continuación se brinda una breve descripción de los CUS: Visualizar Vista Aérea, Vista de Perfil, Representar Objetos Geométricos y Obtener Bounding Box.

3.4.2.1. Descripción del Casos de Uso Visualizar Vista Aérea

Caso de Uso	Visualizar Vista Aérea
Actores:	Cliente
Resumen:	El Caso de Uso se inicializa cuando el cliente selecciona el menú Vista Aérea y el sistema muestra en un espacio del área de trabajo en el que se visualizan los objetos representados.
Precondiciones:	Debe haberse realizado el RF1 o contar el sistema con datos para visualizar.
Referencias:	RF1
CU Asociados:	Obtener Bounding Box (incluido)
Prioridad:	Crítico
Flujo Normal	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona el menú Vista Aérea	1.1 Busca los objetos existentes para representarlos. 1.2 Se Invoca el CU Bounding Box. 1.3 Realiza las transformaciones necesarias para

	representar la información recibida. 1.4 Muestra en la interfaz los objetos geométricos representados.

Tabla 2 Descripción del CU Visualizar Vista Aérea.

3.4.2.2. Descripción del Casos de Uso Visualizar Vista de Perfil

Caso de Uso	Visualizar Vista de Perfil
Actores:	Cliente
Resumen:	El Caso de Uso se inicializa cuando el cliente selecciona el menú Vista de Perfil y el sistema muestra en un espacio del área de trabajo en el que se visualizan los objetos representados en los planos YZ y ZX.
Precondiciones:	Debe haberse realizado el RF1 o contar el sistema con datos para visualizar.
Referencias:	RF2
CU Asociados:	Obtener Bounding Box (incluido)
Prioridad:	Crítico
Flujo Normal	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona el menú Vista de Perfil	1.1 Busca los objetos existentes para representarlos. 1.2 Se Invoca el CU Bounding Box. 1.3 Realiza las transformaciones necesarias para representar la información recibida. 1.4 Muestra en la interfaz los objetos geométricos representados.

Tabla 3 Descripción del CU Visualizar Vista de Perfil.

3.4.2.3. Descripción del Casos de Uso Obtener Bounding Box.

Caso de Uso	CU Obtener Bounding Box
Actores:	CU Visualizar Vista Aérea o de Planta, CU Visualizar las Vistas de Secciones o Perfiles.
Resumen:	El caso de uso delimita el área donde se encuentran los objetos geológicos, se

	inicia al recibir la información a delimitar proporcionada por el CU Visualizar Vista Aérea o el CU Visualizar las Vistas de Secciones o Perfiles, devolviéndoles a estos un objeto.
Precondiciones:	Debe haberse realizado el CU Representar elementos geométricos
Referencias:	RF3
CU Asociados:	-
Prioridad:	Secundario

Flujo Normal	
Acción del Actor	**Respuesta del Sistema**
1. Brinda la información de los objetos geológicos a representar.	1.1 Busca los puntos que contengan el valor de mínimos y máximos en ambos ejes de coordenadas. 1.2 Devuelve un objeto con los datos recibidos y los obtenidos.

Tabla 4 Descripción del CU Obtener Bounding Box

3.4.2.4. Descripción del Casos de Uso Representar Objetos Geométricos.

Caso de Uso	CU Representar Objetos Geométricos
Actores:	Cliente
Resumen:	El CU se inicializa cuando el cliente selecciona el menú Representar Objetos Geométricos, entre los que se encuentran los puntos, polilíneas y polígonos.
Precondiciones	Deben existir puntos representados para poder representar polilíneas y polígonos.
Referencias:	RF4
CU Asociados:	-
Prioridad:	Crítico

Flujo Normal	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el menú Objetos Geométricos • Punto (sección A) • Polilínea (sección B)	1.1 Realiza las acciones definidas en cada una de las secciones.

• Polígono (sección C)	
Sección A	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción representar punto.	
2. Comienza a dar clic en el área de representación.	2.1 Captura las coordenadas y las conserva en una estructura temporal. 2.2 Se invoca el CU Vista.
Sección B	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción representar polilínea.	1.1 Se crea un objeto de tipo polilínea con los puntos representados. 1.2 Se invoca el CU Vista Aérea.
Sección C	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción representar polilínea.	1.1 Se crea un objeto de tipo polígono con los puntos representados. 1.2 Se invoca el CU Vista Aérea.

Tabla 5 Descripción del CU Representar Objetos Geométricos

3.5. Modelo de Análisis

El modelo del análisis es un artefacto que usualmente se genera para entender claramente los requisitos y realizar mejor el diseño. Es usado para representar la estructura global del sistema, describe la realización de casos de uso y sirve como una abstracción del Modelo de Diseño y puede contener: las clases y paquetes de análisis, las realizaciones de los casos de uso, las relaciones y los diagramas.

3.5.1. Diagramas de Clases del Análisis

En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Las clases del análisis se clasifican en Interfaz, de Control o Entidad.

- ✓ **Clase Interfaz:** Modela la interacción entre el sistema y sus actores.

- ✓ **Clase Control:** Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.
- ✓ **Clase Entidad:** Modelan información que posee larga vida y que es a menudo persistente. (Pressman, 2006)

La siguiente imagen muestra el Diagrama de Clases del Análisis del CU Visualizar Vista Aérea o de Planta. En la que se tiene una visión más detallada de la interacción existente entre cada una de las clases del análisis. Para consultar los diagramas de los demás casos de uso consultar el documento **"Modelo del Análisis".**

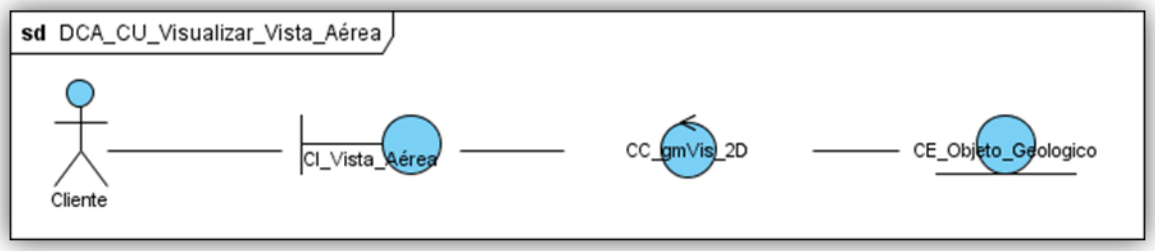

Figura 9 Diagrama de Clases del Análisis del CU Visualizar Vista Aérea**.**

3.5.2. Diagramas de Interacción

Los diagramas de interacción del análisis capturan el comportamiento de los casos de uso y tienen dos formas de expresarse: mediante los diagramas de secuencia y los diagramas de colaboración. (Pressman, 2006) A continuación se detalla el comportamiento del CU Visualizar Vista Aérea o de Planta mediante el diagrama de colaboración el que Muestran las relaciones entre los objetos y los mensajes que intercambian. Para consultar los diagramas de los demás casos de uso consultar el documento **"Modelo del Análisis".**

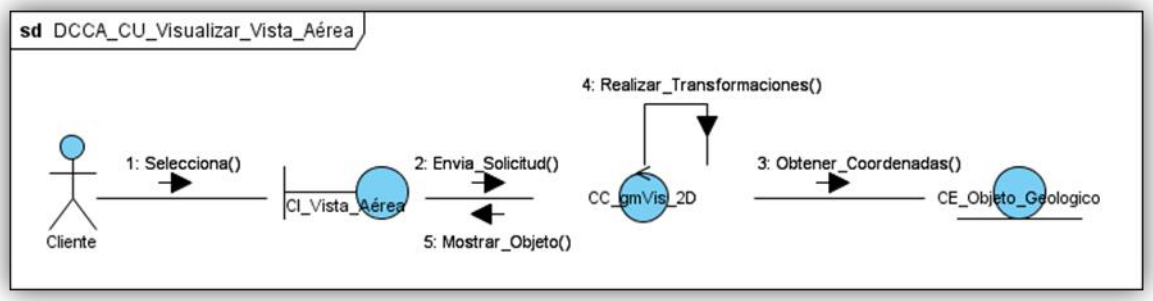

Figura 10 Diagrama de Colaboración de las Clases del Análisis de CU Visualizar Vista Aérea.

3.6. Modelo de Diseño

3.6.1. Diagramas de Clases del Diseño

Los diagramas de clases del diseño son los diagramas principales en el flujo de trabajo análisis y diseño para obtener un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones. A continuación se muestra el Diagrama de Clases del Diseño del CU Visualizar Vista Aérea o de Planta. Para consultar los diagramas de los demás casos de uso consultar el documento **"Modelo de Diseño"**.

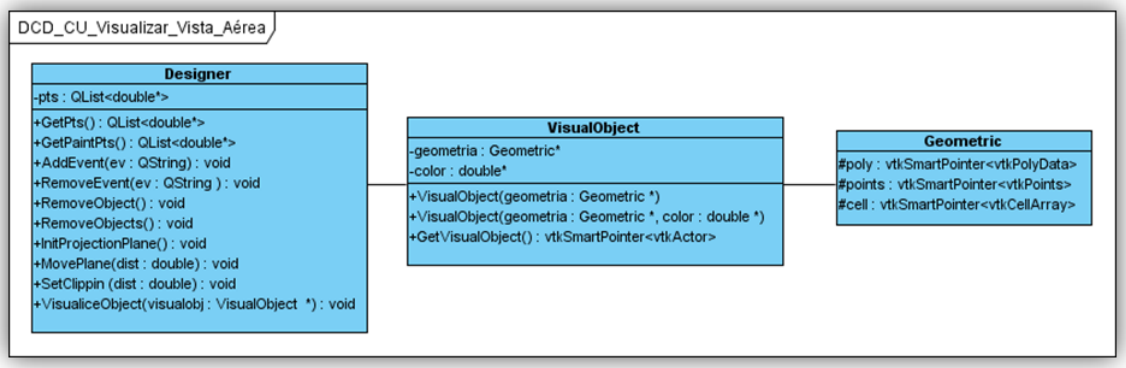

Figura 11 Diagrama de Clases del Diseño del CU Visualizar Vista Aérea.

3.7. Arquitectura del Componente

La arquitectura empleada para el desarrollo del componente está basada en capas donde cada una de ellas proporciona un conjunto de servicios y procesos. Se puede contar con la capa de presentación y la de lógica de aplicación.

Esta arquitectura se caracteriza por:

- ✓ Describir la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.
- ✓ Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.
- ✓ Este modelo ha sido descrito como una “pirámide invertida de re-uso” donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella.

Donde los principios fundamentales estarán dados por la alta cohesión y la reusabilidad, a continuación se explican más detalladamente cada uno de ellos.

- ✓ Alta cohesión. Cada capa contiene funcionalidad directamente relacionas con la tarea de dicha capa.
- ✓ Reutilizable. Las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios.

La siguiente figura muestra cómo se evidencia la arquitectura de capas en el componente.

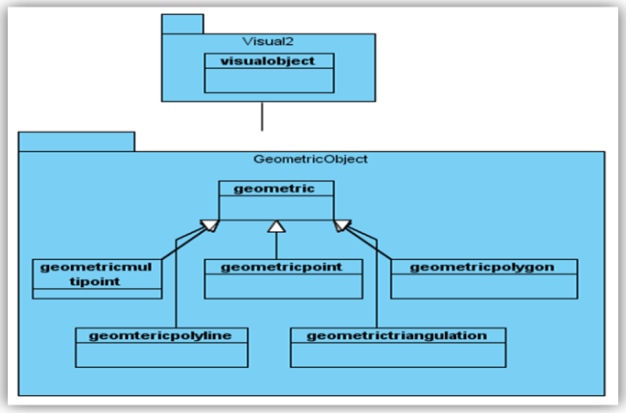

Figura 12 Arquitectura del Componente de Visualización

3.8. Patrones de Diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia. En el desarrollo de multitud de aplicaciones hay problemas de diseños que se repiten o que son análogos, es decir, que responden a un cierto patrón. Con el uso de patrones los diseños serán mucho más flexibles, modulares y reutilizables. (Estrugo Lahera, y otros, 2010)

Para el resultado de la investigación se emplean un grupo de patrones relacionados con el diseño de software, conocidos como patrones GRASP (General Responsibility Asignment Software Patterns): Experto, Alta Cohesión y Controlador, los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Este grupo de patrones está muy relacionado con los problemas básicos del diseño.

El patrón **experto** se evidencia en la clase *designer.h* ya que ésta posee los datos necesarios y contiene la información que se precisa para realizar cualquier representación en la interfaz de visualización. Para manejar los eventos de entrada externa que recibe el componente se empleó el patrón **controlador** el que para evitar saturarse con demasiadas responsabilidades le asigna las mismas a otras clases; se evidencia en la clase *designerInteractor.h.* El patrón **alta cohesión** se aprecia en la clase *visualobject.h* ya que es la que cuenta con la responsabilidad de moderar en un área funcional y colaborar con otras clases para llevar a cabo las tareas de visualización.

3.9. Conclusiones Parciales

En este capítulo se tuvo el primer acercamiento a lo que más tarde se convertirá en el modelo de implementación, entre las tareas vistas se contó con el levantamiento de requisitos funcionales y no funcionales dejando claro como el sistema resuelve el problema planteado. Para tener una mayor comprensión del funcionamiento del componente se realizó la traducción de los requisitos funcionales a los Casos de Usos del Sistema, a los que posteriormente se les realizaron el modelado del análisis y el diseño, arribando a los diagramas principales que sirven de base para la fase de implementación del componente.

Capítulo 4: Construcción de la solución propuesta.

4.1. Introducción

El propósito esencial de este capítulo es definir cómo desarrollar la arquitectura comenzando con el resultado de la etapa de diseño e implementando las clases definidas en el capítulo anterior en términos de componentes. Se modela el diagrama de componentes conformando el modelo de implementación del sistema, dando una visión de cómo quedará construida y distribuida la aplicación. Son vistos ejemplo de código fuente y realizadas las pruebas para determinar su eficiencia, así como la interfaz de usuario y mostrados datos reales del proceso de visualización.

4.2. Diagrama de Componentes

Los diagramas pueden ser usados para modelar y documentar cualquier arquitectura de sistema, los elementos de modelado dentro de un diagrama de componentes son componentes y paquetes. Los diagramas de componentes muestran los componentes software que constituyen una parte reusable, sus interfaces, y sus interrelaciones. Un paquete en un diagrama de componentes representa una división física del sistema.

Para el desarrollo del componente se cuenta con tres archivos fundamentales los que están divididos de la siguiente manera, el archivo “Data” en que contendrá las clase referentes a las transformaciones de los datos del pozo de sondaje, el archivo “Visual” tendrá las clases necesarias para realizar las transformaciones visuales, mientras “View” las clases de interacción con los diferentes objetos. Además se contara con la librería de visualización de VTK, específicamente la libvtk-5.2, y con la versión 4.6 de Qt.

A continuación el Diagrama de Componentes propuesto:

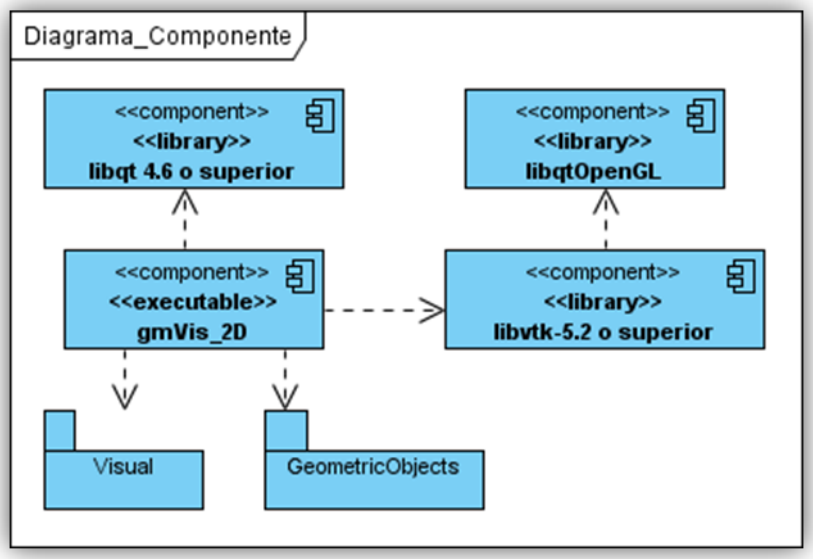

Figura 13 Diagrama de Componentes del Sistema

4.3. Código Fuente

Para realizar una representación exitosa de coordenada cartesiana es preciso transformarlas a coordenadas polares pero esto no es suficiente ya que la librería de visualización de VTK exige que sean tratados a un vtkPolyData, que no es más que una estructura que contienen un identificador, y las coordenadas en un plano y en el espacio. En la siguiente figura se aprecia la explicación anterior.

```
vtkSmartPointer<vtkPolyData> GeometricPoint::getGeometric()
{
    vtkIdType pid[1];
    pid[0] = points->InsertNextPoint(getPunto());
    cell->InsertNextCell ( 1,pid );

    //Cambiar los puntos y los vÃ©rtices para crear la geometrÃ-a y la topologÃ-a del polydata
    poly->SetPoints(points);
    poly->SetVerts ( cell );

    return poly;
}
```

Figura 14 Transformar Puntos y Vértices a vtkPolyData

Una vez obtenido el vtkPolyData se pasa a insertar esta estructura en el *mapper*[11], y posteriormente crear el *actor*[12] que será adicionado en el render para ser visualizado en el *widget*[13]. Las figuras siguientes evidencias dicha explicación.

```
vtkSmartPointer<vtkActor> VisualObject::GetVisualObject()
{
    vtkSmartPointer<vtkPolyDataMapper> mapper = vtkSmartPointer<vtkPolyDataMapper>::New();
    vtkSmartPointer<vtkActor> actor = vtkSmartPointer<vtkActor>::New();
    if(GeometricTriangulation *temp = dynamic_cast<GeometricTriangulation *> (geometria))
    {
        vtkSmartPointer<vtkDelaunay2D> data = temp->getTriangulation();
        mapper->SetInputConnection(data->GetOutputPort());

        if(temp->IsCurvature())
        mapper->SetScalarRange(temp->getGeometricCurvatures()->GetOutput()->GetScalarRange());
    }
    if(GeometricPolyline *temp = dynamic_cast<GeometricPolyline *> (geometria))
    {
        if(temp->IsRibbon())
        {
            vtkSmartPointer<vtkRibbonFilter> data = temp->getGeometricRibbon();
            mapper->SetInputConnection(data->GetOutputPort());
        }
        else
        {
            vtkSmartPointer<vtkPolyData> data = geometria->getGeometric();
            mapper->SetInput(data);
        }
    }
    else
    {
        vtkSmartPointer<vtkPolyData> data = geometria->getGeometric();
        mapper->SetInput(data);
    }
    actor->SetMapper(mapper);
    actor->GetProperty()->SetColor(color[0],color[1],color[2]);
    actor->GetProperty()->SetPointSize(6.0);
    actor->GetProperty()->SetLineWidth(2.0);
    return actor;
}
```

Figura 15 Código Fuente del método GetVisualObject() responsable de devolver un actor.

[11]Mapper: El dispositivo de mapeo sirve como un marco genérico para asignar un dispositivo de un bloque a otro.
[12]Actor: Componentes de VTK que permiten el ajuste y control de las propiedades de las manifestaciones físicas de los datos según la represente en la pantalla. El término "actor" viene de la analogía con el escenario - El actor es una representación física de los datos, cuya apariencia se puede modificar a través de la iluminación, el maquillaje, el vestuario, etc.
[13]Widget: Pequeña aplicación o programa que son ejecutados por un motor de *widgets* o *Widget Engine*. Entre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

```
void ViewerBase::VisualiceObject(VisualObject *visualobj)
{
    render->AddActor(visualobj->GetVisualObject());
    render->Render();
    render->ResetCamera();
}
```

Figura 16 Código Fuente del método VisualiceObject() responsable de adicionar el actor al render.

Finalmente es preciso realizar una llamada en la sentencia principal del componente para poder obtener una representación de los datos en 2D. En la siguiente figura se aprecia un ejemplo en el que se muestra una línea a la que se le hacen todas las transformaciones explicadas anteriormente.

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    designer = new Designer(ui->qvtkWidget);

    double pt1[3] = {0.0,0.0,0.0};
    double pt2[3] = {1.0,1.0,1.0};
    QList<double *> pts;
    pts.append(pt1);
    pts.append(pt2);
    double color[3] = {1.0,0.5,0.5};

    Geometric *geo = new GeometricPolyline(pts,false);
    VisualObject *obj = new VisualObject(geo,color);

    designer->VisualiceObject(obj);
    designer->InitProjectionPlane();
}
```

Figura 17 Ejemplo de llamada al método VisualiceObject() para obtener una Vista.

4.4. Pantallas principales de la aplicación

A continuación son mostrados algunos pantallazos donde se visualiza un cuerpo desde sus etapas de diseño inicial y hasta el diseño final el que se visualiza en 2D.

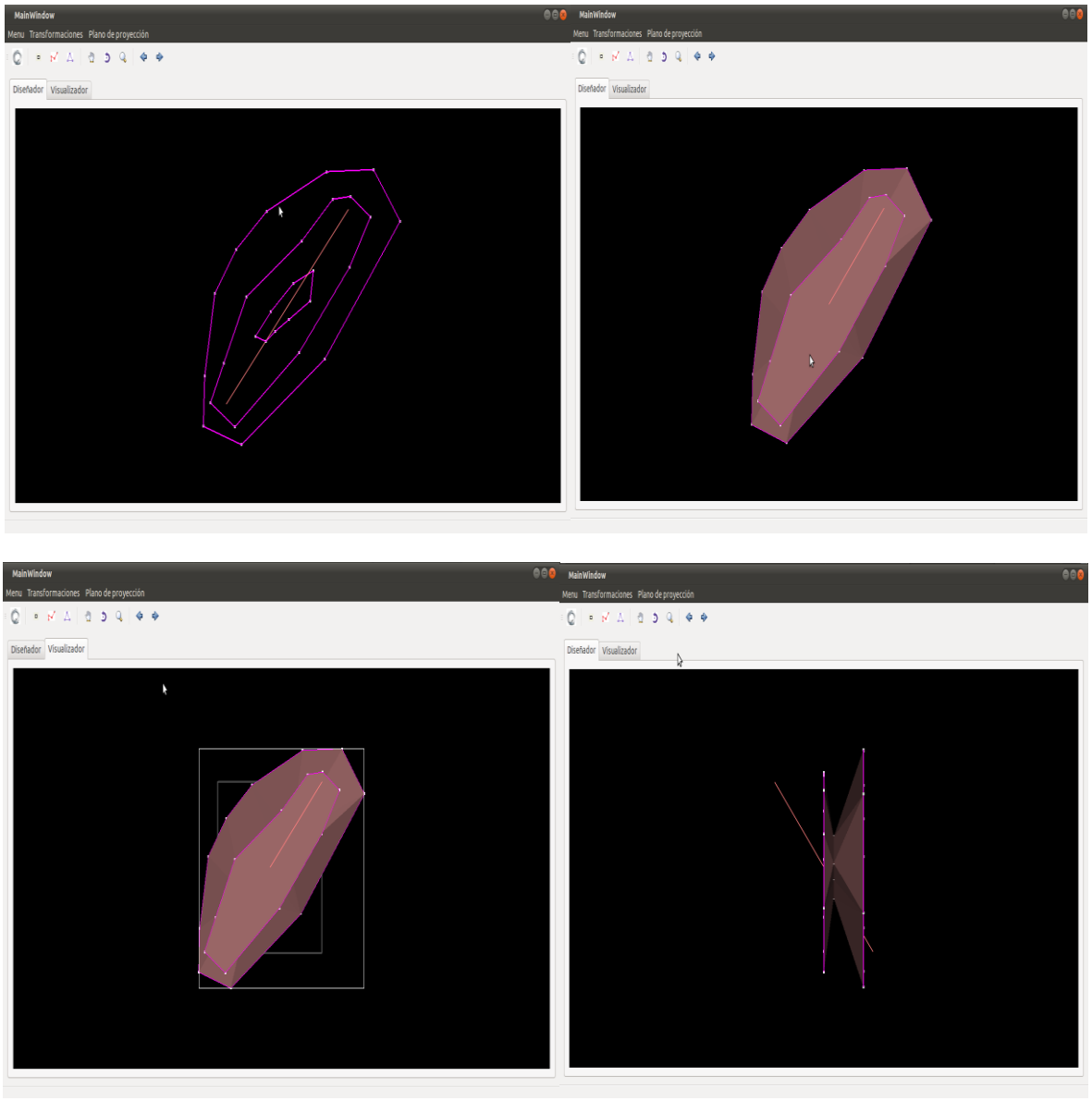

Figura 18 Pantallazos del componente de visualización en 2D

4.5. Validación del sistema

En la realización de todo sistema o componente de software es preciso realizar diversas pruebas para determinar si el sistema se comporta como realmente debería hacerlo, es por esta razón que han sido definidas un conjuntos de pruebas englobadas en dos áreas fundamentales las pruebas de Caja Blanca o código fuente y las pruebas de Caja Negra o interfaz de usuario.

En las pruebas de caja negra solamente se conoce la interfaz y se trata de probar cada uno de los elementos que componen a la misma. Por esta razón son definidos varios casos de prueba de caja negra que a continuación son mostrados.

Caso de Prueba: Representar Objetos Geométricos**.**

Descripción General: Permite adicionar puntos, polilíneas, polígonos a las Vistas.

Condiciones de Ejecución: El cliente debe haber representado puntos en el plano para los casos de adicionar polilíneas y polígonos.

Sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1 Insertar Punto	EC 1.1. El usuario selecciona la opción insertar punto.	El sistema habilita la interfaz visual para que el cliente pueda dar clic en la posición que desee.	✓ Componente ✓ Gestión Objetos ✓ Representar Punto
	EC 1.2. El usuario no selecciona la opción insertar punto.	El sistema mantiene Deshabilita la interfaz visual.	✓ Componente ✓ Gestión Objetos ✓ Representar Punto
SC 2 Insertar Polilínea	EC 2.1. El usuario selecciona la opción insertar polilínea.	El sistema muestra una polilínea desde la primera posición en que el cliente dio clic hasta la última incluyendo los puntos intermedios.	✓ Componente ✓ Gestión Objetos ✓ Representar Polilínea
	EC 2.2. El usuario no selecciona la opción insertar polilínea.	El sistema muestra los puntos que el cliente allá representado o la interfaz deshabilitada en caso que no allá representado algún punto.	✓ Componente ✓ Gestión Objetos ✓ Representar Polilínea
SC 3 Insertar Polígono	EC 3.1. El usuario selecciona la opción insertar polígono.	El sistema muestra un polígono desde la primera posición en que el cliente dio clic incluyendo los puntos intermedios hasta el punto inicial.	✓ Componente ✓ Gestión Objetos ✓ Representar Polígono
	EC 3.1. El usuario no selecciona la opción insertar polígono.	El sistema muestra los puntos que el cliente allá representado o la interfaz deshabilitada en caso que no allá	✓ Componente ✓ Gestión Objetos ✓ Representar Polígono

representado algún punto.
Nota: el sistema visualiza en todas los puntos, polilíneas y polígonos que el cliente represento en otro momento, superponiendo unos sobre otros.

Tabla 6 Definición de las secciones del Caso de Prueba Representar Objetos Geométricos

Escenario de la sección	Evento	Respuesta del Sistema	Resultado de la Prueba
EC 1.1. El usuario selecciona la opción insertar punto.	Dar clic izquierdo sobre la interfaz	El sistema muestra los puntos en la interfaz	Satisfactoria
EC 1.2. El usuario no selecciona la opción insertar punto.	N/A	Mantiene su estado actual	Satisfactoria
EC 2.1. El usuario selecciona la opción insertar polilínea.	N/A	El sistema muestra la polilínea en la interfaz, de los puntos representados con anterioridad.	Satisfactoria
EC 2.2. El usuario no selecciona la opción insertar polilínea.	N/A	Mantiene su estado actual	Satisfactoria
EC 3.1. El usuario selecciona la opción insertar polígono.	N/A	El sistema muestra el polígono en la interfaz, de los puntos representados con anterioridad.	Satisfactoria
EC 3.1. El usuario no selecciona la opción insertar polígono.	N/A	Mantiene su estado actual	Satisfactoria

Tabla 7 Prueba realiza a cada sección del Caso de Prueba Representar Objetos Geométricos.

Por su importancia y resultadas aportados se realizaron las pruebas de Caja Blanca y específicamente las pruebas camino mínimo las que son un tipo de pruebas en las que se analiza una porción de código que pueda ser analizada de manera aislada, como por ejemplo funciones y métodos. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. A los que después de pasarle unos parámetros de entrada se deben obtener otros parámetros de salida claramente definidos o son las encargadas de chequear la correcta creación de objetos.

A continuación es aplicada la prueba al método GetVisualObject() responsable de devolver objeto geométrico para que sea visualizado, este método es el responsable de darle cumplimiento a al CU Visualizar Vista Aérea o de Planta.

```
vtkSmartPointer<vtkActor> VisualObject::GetVisualObject()
{
    vtkSmartPointer<vtkPolyDataMapper> mapper = vtkSmartPointer<vtkPolyDataMapper>::New();
    vtkSmartPointer<vtkActor> actor = vtkSmartPointer<vtkActor>::New();
    if(GeometricTriangulation *temp = dynamic_cast<GeometricTriangulation *> (geometria))
    {
        vtkSmartPointer<vtkDelaunay2D> data = temp->getTriangulation();
        mapper->SetInputConnection(data->GetOutputPort());

        if(temp->IsCurvature())
        mapper->SetScalarRange(temp->getGeometricCurvatures()->GetOutput()->GetScalarRange());
    }
    if(GeometricPolyline *temp = dynamic_cast<GeometricPolyline *> (geometria))
    {
        if(temp->IsRibbon())
        {
            vtkSmartPointer<vtkRibbonFilter> data = temp->getGeometricRibbon();
            mapper->SetInputConnection(data->GetOutputPort());
        }
        else
        {
            vtkSmartPointer<vtkPolyData> data = geometria->getGeometric();
            mapper->SetInput(data);
        }
    }
    else
    {
        vtkSmartPointer<vtkPolyData> data = geometria->getGeometric();
        mapper->SetInput(data);
    }
    actor->SetMapper(mapper);
    actor->GetProperty()->SetColor(color[0],color[1],color[2]);
    actor->GetProperty()->SetPointSize(6.0);
    actor->GetProperty()->SetLineWidth(2.0);
    return actor;
}
```

Figura 19 Código Fuente del método GetVisualObject().

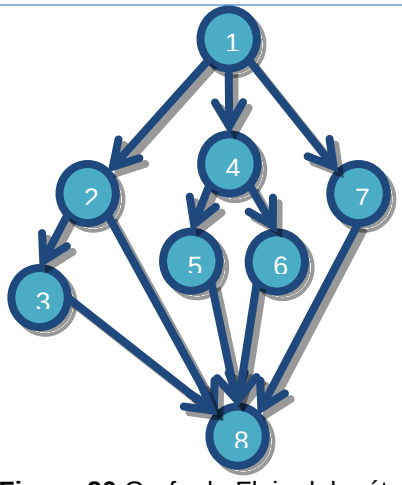

Figura 20 Grafo de Flujo del método GetVisualObject().

V(G)= N aristas – N nodos + 2
V(G)= 11- 8 + 2
V(G)= 5

Caminos Básicos

- ✓ CB1: 1-2-8
- ✓ CB2: 1-7-8
- ✓ CB3: 1-2-3-8
- ✓ CB4: 1-4-5-8
- ✓ CB5: 1-4-6-8

CB2: 1-2-8

Caso de prueba: Visualizar Vista Aérea o de Planta

Entrada: Lista de punto.

Resultado esperado: Devuelve el objeto que contiene la información de los puntos geométricos, que serán utilizados para visualizar las vistas.

Resultado de la Prueba: Satisfactoria

Tabla 8 Caso de Prueba del CU Visualizar Vista Aérea.

Los Casos de Prueba definidos tanto los de caja negra como de caja blanca arrojaron resultados satisfactorios ya que los casos de usos a los cuales se le realizaron las pruebas se comportaron como fueron definidos en las descripciones textuales, es decir, se hizo un correcta traducción de las descripciones textuales de los CU a la implementación de los mismos.

4.6. Conclusiones parciales

En este capítulo fue visto el diagrama de componente del sistema con el que se tuvo un mejor acercamiento a al producto que se quiere obtener, además se logró implementar los casos de usos definidos en la Fase de Análisis y Diseño, así como realizadas las pruebas pertinentes, tanto de Caja Negra como de Caja Blanca, las cuales fueran satisfactorias por lo que el componente se encuentra en condiciones de pasar a la Fase de Despliegue.

Conclusiones

Para darle cumplimiento a la situación problemática mencionada al inicio de la investigación se realizó un análisis exhaustivo de las diferentes herramientas existentes, trazando las pautas para desarrollar el sistema concebido, satisfaciendo las necesidades del cliente y garantizando la construcción de una aplicación estable que se pudiera mantener y actualizar. Con la Implantación del patrón arquitectónico basado en capas se logró obtener la estrategia necesaria para las etapas de diseño e implementación garantizando que el componente fuese flexible y escalable.

La metodología de desarrollo de software definida (RUP) proporciono la artefactos necesarios que son de gran importancia para el proyecto Minería. El desarrollo de componentes permitió disminuir el tiemplo empleado en la codificación y estandarizar la propuesta con el resto de los módulos que componen el proyecto.

Los Casos de Prueba diseñados permiten detectar errores tempranos, agilizar y simplificar el flujo de trabajo de pruebas, concluyendo que el componente cumple con los requisitos planteados. Con la puesta en marcha del sistema se logró cumplir con el objetivo general trazado al inicio del proceso de desarrollo, dándole solución a los problemas existentes en el proyecto Minería.

Bibliografía Referenciada

Argüero, Graciela L. 2002. Los Recursos suelo y agua. Libro de Texto para el Trayecto Ciencia de la Tierra, del Programa de Postitulación en Ciencias Naturales. s.l. : F.C.E.F y Naturales de la U.N.Cba., 2002, Vol. Versión Actualizada.

CARRANZA ATHÓ, FREDY y FLORIAN CRUZ, LAURA. 2006. OPENGL Y VTK. Trujillo,Perú : s.n., 2006, Vol. COMPUTACIÓN GRÁFICA II.

Company Headquarters. 2006. Visual Parading. 10 Reasons to Choose Visual Parading. [En línea] 2006. [Citado el: 14 de Enero de 2011.] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

Cortijo Bon, Francisco. 2005. Desarrollo Profesional de Aplicaciones. Introducción al lenguaje de programación C#. [En línea] 2005. [Citado el: 16 de Enero de 2011.] http://decsai.ugr.es/~cb/CSharp/languaje/intro.xml..

Estrugo Lahera, Evelyn y Cuellar Rodríguez, Dany. 2010. Diseño e Implementación del Portal Web para la Direccion General de Prevención del Delito. Habana : s.n., 2010.

Fernández del Monte, Ing. Yusleydi. 2010. Manual para Modelo del Negocio Visual Paradigm. [En línea] 10 de Octubre de 2010. [Citado el: 15 de Enero de 2011.] http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_05/Laboratorio_1/Materiales_complementarios/Manual_para_Modelo_del_Negocio_Visual_Paradigm_.pdf.

García de Jalón, Javier, y otros. 2000. Aprenda Java como si estuviera en primero. San Sebastián : s.n., 2000. pág. 143.

Geoestadística. 2010. Geoestadística.com. Diseño de Minas. [En línea] 2010. [Citado el: 23 de Noviembre de 2010.] http://geoestadistica.com/diseno_minas.htm.

González Seco, José Antonio. 2001. El lenguaje de programación C#. 2001.

Iznaga Benitez, Arsenio Miguel. 2005. Inter-nos. Gráfico por Computadora. [En línea] Telestudio, UCI, 2005. [Citado el: 18 de Diciembre de 2010.] http://inter-nos.uci.cu/Teleclases/tc_conferencias.asp?id_asig=graficos_por_computadoras_04-05.

JACOBSON, Ivar, BOOCH, Grady y RUMBAUGH, James. 2000. El Proceso Unificado de Desarrollo de Software. Addison Wesley. : s.n., 2000, capítulos 7,8, págs. 125-163, 187-202.

Kendall, Kenneth y Kendall, Julie. 2006. Análisis y Diseño de Sistemas. s.l. : Prentice-Hall, 2006. 978-970-26-0577-5.

Leiva Rodríguez, Alejandro David. 2007. Simulación Geoestadística incorporando un campo de Direcciones Variables. Facultad de Ciencias Físicas y Matemáticas, Departamento de Ingeniría Minera. [En línea] Enero de 2007. [Citado el: 17 de Diciembre de 2010.] http://www.cybertesis.cl/tesis/uchile/2007/leiva_a/sources/leiva_a.pdf.

Ministerio de Justicia. 1995. Ley No.76. Ley de Mina. Ordinaria. La Habana : Gaceta Oficial, 1995. págs. 33-48. ISSN 1682-7511.

Moreno Berzal, Ignacio. 2004. Desarrollo de algoritmos de procesamiento de imágenes con VTK. Madrid : s.n., 2004. GVA-ELAI-UPM®PFC0081-2004.

Pressman, Roger S. 2006. Ingeniería de Software, un enfoque práctico. Sexta edición. 2006. ISBN-970-10-5473-3.

Rational Software Corporation. 2001. Rational Suite Tutorial. 2001A.04.00. 2001. 800-024444-000.

Schildt, Herbert. 2001. C++. Guia de autoensefianza. Aravaca : s.n., 2001. ISBN: 84-481-3203-3.

Suarez Quirós, J, y otros. 2009. Desarrollo de un entorno de visualizacion 2D y 3D multiplataforma. Asturias,Gijón : Universidad de Oviedo, 2009.

Sun Microsystem. 2010. NetBeans. Welcome to the NetBeans Community. [En línea] 2010. [Citado el: 17 de Enero de 2011.] http://www.netbeans.org/about/index.html.

Sun Microsystems. 2010. Java. Conozca más sobre la tecnología Java. [En línea] 2010. [Citado el: 15 de Enero de 2011.] http://www.java.com/es/about/.

The Eclipse Foundation. 2009. Eclipsepedia. The Official Eclipse FAQs. [En línea] 2009. [Citado el: 17 de Enero de 2011.] http://wiki.eclipse.org/The_Official_Eclipse_FAQs.

Thelin, Johan. 2007. Qt Development. s.l. : Foundations of Qt Development, 2007. ISBN-13 (pbk): 978-1-59059-831-3.

Vallejos, Sofia J. 2006. Minería de Datos. Trabajo de Adscripción. [En línea] 2006. [Citado el: 20 de Enero de 2011.] http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/Mineria_Datos_Vallejos.pdf.

Bibliografía Consultada

Argüero, Graciela L. 2002. Los Recursos suelo y agua. Libro de Texto para el Trayecto Ciencia de la Tierra, del Programa de Postitulación en Ciencias Naturales. s.l. : F.C.E.F y Naturales de la U.N.Cba., 2002, Vol. Versión Actualizada.

CARRANZA ATHÓ, FREDY y FLORIAN CRUZ, LAURA. 2006. OPENGL Y VTK. Trujillo,Perú : s.n., 2006, Vol. COMPUTACIÓN GRÁFICA II.

Ceballos Sierra, Javier. 2001. El Lenguaje De Programación C#. s.l. : Ra-ma, 2001.

Company Headquarters. 2006. Visual Parading. 10 Reasons to Choose Visual Parading. [En línea] 2006. [Citado el: 14 de Enero de 2011.] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

Cortijo Bon, Francisco. 2005. Desarrollo Profesional de Aplicaciones. Introducción al lenguaje de programación C#. [En línea] 2005. [Citado el: 16 de Enero de 2011.] http://decsai.ugr.es/~cb/CSharp/languaje/intro.xml.

Estrugo Lahera, Evelyn y Cuellar Rodríguez, Dany. 2010. Diseño e Implementación del Portal Web para la Direccion General de Prevención del Delito. Habana : s.n., 2010.

Fernández del Monte, Ing. Yusleydi. 2010. Manual para Modelo del Negocio Visual Paradigm. [En línea] 10 de Octubre de 2010. [Citado el: 15 de Enero de 2011.] http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_05/Laboratorio_1/Materiales_complementarios/Manual_para_Modelo_del_Negocio_Visual_Paradigm_.pdf.

García de Jalón, Javier, y otros. 2000. Aprenda Java como si estuviera en primero. San Sebastián : s.n., 2000. pág. 143.

Geoestadística. 2010. Geoestadística.com. Diseño de Minas. [En línea] 2010. [Citado el: 23 de Noviembre de 2010.] http://geoestadistica.com/diseno_minas.htm.

González Seco, José Antonio. 2001. El lenguaje de programación C#. 2001.

Hall, Prentice. 2008. C++ GUI Programming with Qt4. 2da Edición. 2008.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2002. El Paradigma Cuantitativo de la Investigacion Cientifica. Ciudad de la Habana : EDUNIV Editorial Universitaria, 2002.

2010. IBM España s.a. Rational Rose Enterprise. [En línea] 2010. [Citado el: 14 de Enero de 2011.] http://www-142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=M221280M46834Z27.

2010. Informática Aplicada a la Minería. OFERTA UNICA DE MANUALES O TUTORIALES SOFTWARE MINERO. [En línea] 2010. [Citado el: 23 de Noviembre de 2010.] http://informaticaminera.blogspot.com/2008/08/el-mercado-de-los-software-mineros.html.

Iznaga Benitez, Arsenio Miguel. 2005. Inter-nos. Gráfico por Computadora. [En línea] Telestudio, UCI, 2005. [Citado el: 18 de Diciembre de 2010.] http://inter-nos.uci.cu/Teleclases/tc_conferencias.asp?id_asig=graficos_por_computadoras_04-05.

JACOBSON, Ivar, BOOCH, Grady y RUMBAUGH, James. 2000. El Proceso Unificado de Desarrollo de Software. Addison Wesley. : s.n., 2000, capítulos 7,8, págs. 125-163, 187-202.

Kendall, Kenneth y Kendall, Julie. 2006. Análisis y Diseño de Sistemas. s.l. : Prentice-Hall, 2006. 978-970-26-0577-5.

Leiva Rodríguez, Alejandro David. 2007. Simulación Geoestadística incorporando un campo de Direcciones Variables. Facultad de Ciencias Físicas y Matemáticas, Departamento de Ingeniría Minera. [En línea] Enero de 2007. [Citado el: 17 de Diciembre de 2010.] http://www.cybertesis.cl/tesis/uchile/2007/leiva_a/sources/leiva_a.pdf.

López Palenzuela, Filiberto y Vallejo Mursulí, Luis Javier. 2009. Implementación de un componente de software para la visualización tridimensional de Neuroimágenes. [En línea] Junio de 2009. [Citado el: 30 de Noviembre de 2010.] http://bibliodoc.uci.cu/TD/TD_2102_09.pdf.

Ministerio de Justicia. 1995. Ley No.76. Ley de Mina. Ordinaria. La Habana : Gaceta Oficial, 1995. págs. 33-48. ISSN 1682-7511.

Moreno Berzal, Ignacio. 2004. Desarrollo de algoritmos de procesamiento de imágenes con VTK. Madrid : s.n., 2004. GVA-ELAI-UPM®PFC0081-2004.

Pressman, Roger S. 2006. Ingeniería de Software, un enfoque práctico. Sexta edición. 2006. ISBN-970-10-5473-3.

Rational Software Corporation. 2001. Rational Suite Tutorial. 2001A.04.00. 2001. 800-024444-000.

Schildt, Herbert. 2001. C++. Guia de autoensefianza. Aravaca : s.n., 2001. ISBN: 84-481-3203-3.

Suarez Quirós, J, y otros. 2009. Desarrollo de un entorno de visualizacion 2D y 3D multiplataforma. Asturias,Gijón : Universidad de Oviedo, 2009.

Sun Microsystem. 2010. NetBeans. Welcome to the NetBeans Community. [En línea] 2010. [Citado el: 17 de Enero de 2011.] http://www.netbeans.org/about/index.html.

Sun Microsystems. 2010. Java. Conozca más sobre la tecnología Java. [En línea] 2010. [Citado el: 15 de Enero de 2011.] http://www.java.com/es/about/.

The Eclipse Foundation. 2009. Eclipsepedia. The Official Eclipse FAQs. [En línea] 2009. [Citado el: 17 de Enero de 2011.] http://wiki.eclipse.org/The_Official_Eclipse_FAQs.

Thelin, Johan. 2007. Qt Development. s.l. : Foundations of Qt Development, 2007. ISBN-13 (pbk): 978-1-59059-831-3.

Valle Martínez, Yusnier. 2009. Modelación y visualización de superficies. [En línea] Julio de 2009. [Citado el: 20 de Diciembre de 2010.] http://bibliodoc.uci.cu/TD/TD_2800_09.pdf.

Vallejos, Sofia J. 2006. Minería de Datos. Trabajo de Adscripción. [En línea] 2006. [Citado el: 20 de Enero de 2011.] http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/Mineria_Datos_Vallejos.pdf.

Glosario de Términos

Acimut o azimut: Es el ángulo de una dirección contado en el sentido de las agujas del reloj a partir del norte geográfico.

Renderización: (*render* en inglés) es un término usado en informática para referirse al proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en programas de diseño.

Píxeles: (acrónimo del inglés *picture element*, "elemento de imagen") es la menor unidad homogénea en color que forma parte de una imagen digital.

Ráster: Una **imagen rasterizada**, también llamada ***mapa de bits***, ***imagen matricial*** o ***bitmap***, es una estructura o fichero de datos que representa una rejilla rectangular de píxeles o puntos de color, denominada ***raster***, que se puede visualizar en un monitor de ordenador, papel u otro dispositivo de representación.

Litología: es la parte de la geología que trata de las rocas, especialmente de su tamaño de grano, del tamaño de las partículas y de sus características físicas y químicas.

Bounding box: caja delimitadora en español, rectángulo que envuelve todas las figuras de una escena.

Anexos

Anexo 1 Descripción de CU Realizar Zoom

Caso de Uso	Realizar Zoom
Actores:	Cliente
Resumen:	El CU se inicializa cuando el cliente selecciona los menú zoom más (+) o zoom menos (-) y el sistema muestra la escena con la transformación solicitada, existe el zoom extendido que retornar la escena a un forma original.
Precondiciones:	Debe haberse realizado una de las vistas.
Referencias:	RF5
CU Asociados:	Escalar Modelo(incluido)
Prioridad:	Crítico

Flujo Normal	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona uno de los menús (+) o (-)	1.1 Busca información de la escena, toma valor actual del factor de escala y la acción a realizar 1.2 Se invoca el CU Escalar Modelo el cual devuelve el nuevo factor de escala. 1.3 Muestra la escena transformada según opción empleada por el usuario. • (+) • (-)
2. El usuario selecciona uno de los menús (extendido)	2.1 Restaura la escena a su posición inicial antes de haberle realizado cualquier transformación.
Poscondiciones:	

Anexo 2 Descripción de CU Escalar Modelo

Caso de Uso	Escalar Modelo

Actores:	CU Realizar Zoom
Resumen:	El CU se inicializa por el actor, el cual envía los datos de la escena, el factor de escala actual y el valor de la acción a realizar, zoom (+) o (-).
Precondiciones:	Debe haberse el CU Realizar Zoom a las Vistas
Referencias:	RF5.1
CU Asociados:	-
Prioridad:	Secundario

Flujo Normal	
Acción del Actor	**Respuesta del Sistema**
1. Envía los datos de la escena, factor de escala actual, valor de la acción.	1.1 Calcula el nuevo factor de escala para la escena. 1.2 Devuelve el valor.
Poscondiciones:	

Anexo 3 Descripción del CU Realizar Paneo

Caso de Uso	Realizar Paneo
Actores:	Cliente
Resumen:	El Cliente determina la posición original de escena y la posición a la cual desea trasladarla. Se apoya del CU Trasladar modelo, una vez obtenido las trasformaciones necesarias determina que objetos de la escena mostrar.
Precondiciones:	Debe haberse realizado una de las vistas.
Referencias:	RF6
CU Asociados:	Trasladar Modelo(incluido)
Prioridad:	Crítico

Flujo Normal	
Acción del Actor	**Respuesta del Sistema**

1. Proporciona posición origen y destino.	1.1 Se invoca el CU Trasladar Modelo, el que devuelve una escena traslada. 1.2 Determina que objetos mostrar según características del área de trabajo.

Anexo 4 Descripción del CU Trasladar Modelo

Caso de Uso	Trasladar Modelo
Actores:	CU Realizar Paneo a las Vistas
Resumen:	El CU inicia cuando recibe el punto origen y el destino de la traslación realizada por el usuario. El CU traslada la escena hacia la posición destino.
Precondiciones:	Debe haberse el CU Realizar Paneo a las Vistas
Referencias:	RF5.1
CU Asociados:	-
Prioridad:	Secundario
Flujo Normal	
Acción del Actor	**Respuesta del Sistema**
1. Envía los datos de la escena, punto origen y punto destino.	1.1 Realiza las transformaciones necesarias para trasladar la escena. 1.2 Devuelve la escena trasformada

Anexo 5 Diagrama Clases de Secuencia del Caso de Uso Visualizar Vista Aérea