UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

TÍTULO: Componente para visualizar curvas de registro de pozos

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

AUTOR: Rolando Ávila González

TUTOR: Ing. Armando Ortiz Cabrera

Ciudad de la Habana, febrero de 2010

“Año 52 de la Revolución”

DECLARACIÓN DE AUTORÍA

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

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

Rolando Ávila González

_____

(Firma del autor)

Armando Ortiz Cabrera

_____

(Firma del tutor)

DATOS DE CONTACTO

Tutor:

Ing. Armando Ortiz Cabrera
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: aortizc@uci.cu

AGRADECIMIENTOS

A la Revolución y a Fidel por hacer posible una Universidad de excelencia…

A mis padres por su amor infinito, por hacer de mí un hombre de bien, por la educación que en mí inculcaron… por todo.

A mi padrastro por ser como un padre para mí.

A mi hermana por ser otra parte de mi corazón.

A mis abuelos por quererme tanto y estar ahí, siempre en el momento en que los necesito.

A mis amistades por compartir tantos recuerdos….

A mi novia por tener la dedicación y comprensión durante todos estos años.

A mis suegros por acogerme en sus brazos como a un hijo.

A mi tutor Armando Ortiz por su apoyo incansable en todo momento.

A mi tribunal de tesis por hacer posible una tesis mejor.

A todos los que de una manera u otra ayudaron al desarrollo de esta investigación, lleguen mis agradecimientos más sinceros.

DEDICATORIA

A mis padres.

Por haberme dado todo lo que soy como persona, por haberme inculcado los principios, los valores, la perseverancia y el empeño, acompañado siempre de una gran dosis de amor, sin pedir nunca nada a cambio.

A mi novia.

Yanelky, por su paciencia, por su comprensión, por su empeño, por su amor, por ser tal y como es, por permitirme andar en su camino.

A mis familiares.

A mi hermana Disnedys por darme un sobrino especial. A mis abuelos por tanta compresión y cariño que siempre me han dado. A mis tías y primos por quererme demasiado y hacer que mi familia sea feliz. A todos los quiero mucho.

A mis amigos.

A todos mis compañeros de la universidad con los que he compartido todos estos años de clases.

Rolando Ávila González

RESUMEN

La presente investigación surge en el marco de trabajo del Proyecto: “Petrofísica” del polo productivo “Petrosoft” de la facultad 9 de la Universidad de las Ciencias Informáticas (UCI). En la investigación, se realizará la implementación de un componente de software, para la visualización de curvas de registros de pozos. Una vez que se obtenga el componente será integrado al Sistema de Análisis de Registros de Pozos Petroleros (ANPER), surgido como un convenio entre la UCI y el Centro de Investigaciones del Petróleo (CEINPET). La puesta en marcha de dicho sistema, permitirá cambiar la situación existente en la empresa, con respecto a la actividad de análisis petrofísico que allí se desarrolla, mediante software propietarios. Además de contar con una interfaz amigable y servicios adaptados a las necesidades específicas de sus trabajadores. El sistema agrupará las funcionalidades principales de cada software para los procesos de análisis petrofísicos utilizados por el CEINPET y que provienen de empresas internacionales. Esto traería consigo un ahorro en tiempo, mayor usabilidad y que el trabajo sea más cómodo y económico.

PALABRAS CLAVES

Componente,curvas,registro de pozos, interfaz, análisis petrofísicos.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Índice

Introducción

El petróleo es la energía primaria más importante del mundo, prácticamente todas las actividades económicas se sustentan en el como fuente energética. Su gama casi infinita de productos derivados, le convierten en uno de los factores más importantes del desarrollo económico y social. Entre los productos derivados del petróleo se encuentra: la gasolina para el consumo como combustible en motores de combustión interna, el queroseno usado frecuentemente en estufas[1] domésticas y en equipos industriales, el polietileno que constituye la materia prima fundamental para la industria del plástico entre muchos otros.

El Centro de Investigaciones del Petróleo (CEINPET) se encarga de dar respuesta de forma integral a toda la actividad petrolera en el territorio de Cuba, desde la exploración hasta la refinación, en el proceso de investigación – desarrollo – producción. En esta empresa se realizan varias actividades dentro de las que se destacan el análisis petrofísico de los registros de pozo, fundamental para obtener una evaluación que caracterice a cada pozo analizado y poder derivar sus propiedades físicas para su posterior utilización.

Actualmente, la información de los registros de pozos son grabados por empresas especializadas como la Compañía Schlumberger[2] y entregados a Empresas estatales o privadas de petróleo por ejemplo: Petroecuador o Repsol, esta información es entregada en un formato estandarizado a nivel mundial como son los archivos LAS(Log ASCII Standard). En dichos archivos viene toda la información de la corrida de los registros en un pozo perforado incluyendo: datos de temperatura, profundidad, nombre del pozo, ubicación, herramienta usada, características del lodo de perforación y toda una serie de datos llamados de cabecera.

Cuba no cuenta con un software propio capaz de realizar análisis petrofísico. En la actualidad los trabajadores del CEINPET y en especial los petrofísicos, para realizar la visualización de los registros de pozos se auxilian en software existentes producidos por empresas internacionales, conocidas mundialmente por sus aportes hechos en los estudios de las propiedades físicas de este importante producto, como la Compañía Schlumberger. Sin Schlumberger probablemente muchos de estos

[1] Aparato que produce y emite calor destinado a calentar un local.
[2] Compañía de servicios de yacimiento petrolífero que entrega una variedad de servicios y soluciones a la industria de petróleo internacional.

nuevos productores hubieran sido incapaces de sacar el petróleo o el gas de sus subsuelos terrestres o marítimos. [1]

A los petrofísicos del CEINPET les resulta muy difícil el trabajo con estos sistemas e incluso en algunos casos se necesitan de cursos para poder familiarizarlos con los mismos. Por otra parte el CEINPET no cuenta con el recurso monetario suficiente para adquirir y mantener estos sistemas, puesto que la mayoría de ellos son costosos y requieren licencias especiales para poder operar.

Por los motivos expuestos anteriormente es que surge como un convenio entre el CEINPET y la Universidad de las Ciencias Informáticas (UCI), el Sistema de Análisis de Registros de Pozos Petroleros (ANPER), el cual tiene como objetivo lograr la carga, visualización e interpretación de la información contenida en los registros de pozos. Actualmente desarrollado en la UCI por el equipo de proyecto Petrofísica del polo productivo Petrosoft de la facultad 9.

Este trabajo de diploma tiene como objetivo dar solución al módulo de visualización del sistema ANPER para satisfacer las necesidades del país antes expuesta por lo que el **problema** consiste en: ¿Cómo mejorar las condiciones de trabajo de los sistemas de software para la visualización de registros de pozos que cumpla con las necesidades económicas y técnicas del CEINPET?

El **objeto de estudio:** el proceso de visualización de registros de pozos y el **campo de acción**: tipos de visualización de registros de pozos en áreas de exploración y perforación.

Para dar solución al problema científico declarado, esta investigación tiene como **objetivo general**: desarrollar un componente que permita visualizar gráficos de registros de pozos que satisfagan las necesidades económicas y técnicas del CEINPET.

Para dar cumplimiento al objetivo general se han trazado las siguientes **tareas de investigación**:

- ✓ Caracterizar los fundamentos teóricos de los tipos de gráficas, bibliotecas y plataforma .net.
- ✓ Caracterizar las tecnologías a utilizar en desarrollo.
- ✓ Describir el sistema actual, sus deficiencias, funcionalidades y las posibles características a implementar en proyecto de Petrofísica.
- ✓ Desarrollar un componente que permita visualizar la información contenida en los registro de pozos.

El desarrollo exitoso de las tareas expuestas anteriormente contribuirá al cumplimiento a la **hipótesis** que plantea:
Sí: Se desarrolla un componente que permita visualizar gráficos de registro de pozos y que cumpla con las necesidades económicas y técnicas del CEINPET.
Entonces: Se hará posible un mejor análisis de la información de los pozos petroleros.

Para la realización de la investigación se hizo necesario aplicar algunos métodos científicos dentro de los que se incluyen **métodos teóricos y empíricos**, que tienen su sustento en la concepción materialista dialéctica y facilitan la recopilación de la información necesaria para la implementación de un componente para visualizar curvas de registro de pozos. De los teóricos se utilizaron el histórico-lógico, inductivo-deductivo. De los empíricos[3] se utilizaron las entrevistas.

El método **Histórico-Lógico** se aplica para analizar a nivel nacional e internacional, las empresas que utilizan software para el análisis de procesos petrofísicos en pozos petroleros y las características de otros sistemas informáticos para el análisis de procesos petrofísicos similares a los utilizados por el CEINPET, así como investigaciones realizadas anteriormente sobre el tema.

El método **Inductivo-Deductivo**, se aplica para definir el funcionamiento de la empresa en los procesos de análisis petrofísicos desde una perspectiva general hasta llegar a las específicas de cada software utilizado y dentro de éstos a la función de cada módulo en el que esté dividido su trabajo.

Las entrevistas fueron hechas a trabajadores petrofísicos del CEINPET.

Con el desarrollo de este trabajo se espera como **posibles resultados:** la creación de un componente para la visualización de la información de registro de pozos, así como la documentación de los artefactos generados.

El informe está estructurado en tres capítulos, a continuación se presentará el nombre del capítulo y sus objetivos:

[3] Conocimiento basado en la experiencia, todo lo que sabemos y que lo repetimos continuamente sin tener un conocimiento científico.

El Capítulo 1: Fundamentación teórica: En este capítulo se hará una descripción de los métodos utilizados, se abordarán conceptos referentes al dominio del problema y se realizará un estudio del estado del arte.

El Capítulo 2: Tendencias y tecnologías a utilizar: En este capítulo se realizará un estudio de las herramientas, lenguajes de programación, metodologías, que darán soporte al desarrollo del componente.

El Capítulo 3: Propuesta y solución del componente: En este capítulo se definirán los requisitos, tanto funcionales como no funcionales, se realizará el diseño, la implementación y las pruebas del componente.

Capítulo 1“ Fundamentación Teórica”

1.1 Introducción

La perforación y la terminación de pozos es la actividad que consume la mayor cantidad de presupuesto, por lo que la planeación, diseño, ejecución, evaluación y retroalimentación deben cumplir altos niveles de desempeño. Para compensar la declinación de los campos petroleros e incrementar la incorporación de reservas futuras de la industria petrolera, y con la obligación de sustentar la producción, es necesario hacer eficiente la visualización de los registros de pozos.

En este capítulo se caracterizará de manera breve cada software producido por empresas reconocidas mundialmente para el análisis petrofísico y que son utilizados por el CEINPET en vista de capturar sus funcionalidades principales y establecerlas en el diseño del componente para visualizar curvas de registro de pozos. Se tratan cuestiones como: ¿Qué es un componente?, Pista, Curva. Se definirán los conceptos asociados al dominio del problema, identificando las fuentes de información, revisión de materiales y construcción de conceptos.

1.2 ¿Qué es un componente?

Es una unidad de composición de aplicaciones de software[4], que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio. [2]

Un componente de software es una unidad de composición con interfaces contractualmente especificadas y explícitas solo con dependencias dentro de un contexto. Un componente de software puede ser desplegado independientemente y es sujeto a la composición de terceros. [3]

Un componente es una parte modular de un sistema que encapsula implementación y un conjunto de interfaces. Puede ser desplegable y reemplazable.

1.3 Registros de pozos

Proporcionan información in situ[5] del subsuelo, que difícilmente pueden ser obtenidas a través de otros métodos menos costosos. Pueden definirse como mediciones de diversos parámetros y propiedades físicas de un pozo, tomadas a lo largo del mismo y bajo ciertos intervalos. [4].

[4] Se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específicas; en contraposición a los componentes físicos del sistema, llamados hardware.

[5] Son pruebas realizadas para la determinación de las características geotécnicas de un terreno.

Consisten en una serie de mediciones o registros geofísicos, obtenidas por una sonda[6] con varios sensores o antenas transmisoras y receptoras que se introduce en una perforación de barreno para determinar las curvas de cada parámetro que se desea conocer. Con esta técnica se obtiene a diferentes profundidades los parámetros físicos de la formación. Se lleva a cabo para determinar las características físicas de las rocas, de los fluidos que la saturan y de las propiedades de la construcción del pozo. [8].

Un Registro de pozos no es más que una forma de llevar un control o agrupar las mediciones hechas a un pozo determinado de sus propiedades físicas.

1.4 Pista

Es el componente visual que conforma las plantillas y mediante el cual se representan los valores a través de las curvas de los registros de pozos. Contiene una cabecera, donde se establecen las propiedades fundamentales de las curvas que están representadas**.** [9].

La pista es el lugar donde se hace visible las propiedades de una o más curvas, así como el color, la textura y demás características definidas por el usuario.

1.5 Color

El color identifica a una curva determinada en el momento en que esta se representa en la pista. El usuario es quien decide el color de la curva.

1.6 Textura

La textura al igual que el color identifica a la curva cuando está ubicada en la pista. También puede ser definida por el usuario[7].

1.7 Curva

Son las que conforman los registros de pozo, y representan las variaciones de las propiedades de los mismos. Cada curva corresponde a un método diferente aportando cualidades específicas para cada registro y se pueden visualizar a través de las pistas. La curva puede ser clasificada de diferentes formas: [10].

1 *DEPTH*: Profundidad.

1 *GR:* Gamma Ray.

2 *CALI o CAL:* Caliper (diámetro de pozo).

[6] Es un objeto de manipulación remota cuya misión es llegar a un objetivo prefijado y realizar algún tipo de acción o mandar información.

[7] Es la persona que utiliza o trabaja con algún objeto o que es destinataria de algún servicio público, privado, empresarial o profesional.

3 *ILM:* Resistividad media por inducción.
4 *ILD:* Resistividad profunda por inducción.
5 *PHIN:* Porosidad Neutrón.
6 *RHOB:* Densidad.
7 *CGR:* Rayos gamma computados.
8 *SGR:* Rayos gamma espectrales.
9 *SP:* Potencial espontáneo.
10 *PEF:* Efecto fotoeléctrico.
11 *POTA:* Potasio.
12 *THOR:* Torio.
13 *URAN:* Uranio.
14 *RX0:* Resistividad de zona labrada.
15 *RLA0…RLA5:* Distintas investigaciones de radio.
16 *NPHI:* Porosidad neutrónica.
17 *DPHI:* Porosidad por densidad.
18 *SPHI:* Porosidad por sónico.
19 *DT:* Tiempo intervalo.
20 *LLD:* Lateroloc profundo.
21 *LLS:* Lateroloc mero (profundidad somera).
22 *MSFL:* Micro dispositivo lateral de poca profundidad.
23 *RT:* Resistividad real.

La clasificación de la curva depende del método empleado para obtener las propiedades físicas de un pozo.

1.8 Tipos de gráficos

Es muy amplia la diversidad de gráficos existentes. La utilización de cada uno de ellos depende de los diferentes conjuntos de datos y del uso para el que estén confinados. A continuación hay una vista general de los tipos de gráficos principales y sus usos más comunes.

1.8.1 Gráfica de barras

Es aquella representación gráfica bidimensional[8] en el que los objetos gráficos elementales son un conjunto de rectángulos dispuestos paralelamente de manera que la extensión de los mismos es proporcional a la magnitud que se quiere representar. Se usan principalmente cuando:

- Se comparan magnitudes entre varias categorías.
- Se desea mostrar la evolución en el tiempo de una determinada magnitud

Los rectángulos o barras pueden estar colocados horizontal o verticalmente, en este último caso reciben también el nombre de gráficos de columna. [5]. A continuación se muestra una gráfica de barras.(Figura 1).

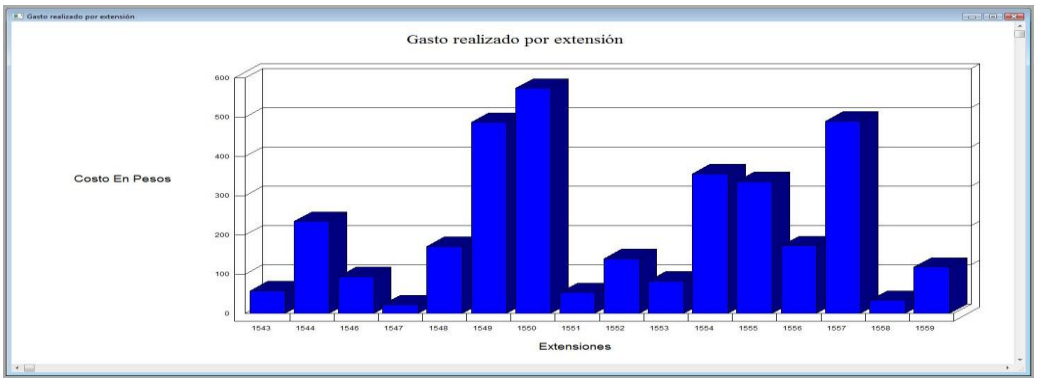

Figura 1. Gráfica de barras[21].

1.8.2 Gráfica de pastel

Es un círculo que se divide o rebana[9] desde su punto central, donde cada rebanada representa la frecuencia proporcional de determinada categoría de una variable como se muestra en la figura 2. Se utiliza para ilustrar la manera en que se distribuye el 100% de un recurso en un período específico. Especialmente útil cuando:

- Se desea transmitir un sentido de equidad, tamaño relativo o desigualdad entre las partes.

[8] Si tiene dos dimensiones, por ejemplo, ancho y largo.
[9] División de un objeto en partes más pequeñas.

- Cuando se desea resaltar las proporciones que representan algunos subconjuntos con respecto al total. [6].

Figura 2. Gráfica de pastel [20].

1.8.3 Gráfica de línea

Los gráficos de líneas muestran una serie como un conjunto de puntos conectados mediante una sola línea. Los gráficos de líneas se usan para representar grandes cantidades de datos que tienen lugar durante un período continuado de tiempo. [7]. Un ejemplo de gráfica de línea se muestra en la figura 3.

- Es utilizada para mostrar tendencias a través del tiempo.
- Con una gráfica de línea es posible hacer una proyección.
- En una gráfica de línea, cada serie de datos es usada para producir una línea, en la cual cada número o dato representa un punto.

Figura 3. Gráfica de líneas[19].

1.9 Objeto de estudio

El objeto de estudio relacionado a la investigación es el proceso de visualización de registros de pozos, este es un proceso[10] que su resultado final estará expresado en un componente para visualizar curvas de registros de pozos y que resolverá las necesidades del CEINPET. Con los estudios realizados a los sistemas de análisis petrofísicos se determinó un conjunto de las mejores funcionalidades que incluyen los sistemas y que podrían formar parte de las funcionalidades del componente a implementar. Además del estudio realizado a los registros de pozos para determinar aspectos importantes como: ¿Qué es una curva?, ¿Qué son los datos de cabecera?.

1.9.1 Descripción General

Todo el proceso de análisis petrofísico se realiza en el CEINPET a través de estudios en laboratorios de los núcleos y registros de pozo. Los núcleos son los fragmentos de roca a los que se le realiza el análisis de sus propiedades y los reservorios son mediciones que se realizan con sensores remotos constituidos por buzo, cable y registrador de superficie, que luego son representadas mediante curvas que expresan las mediciones digitales. Conociendo estas definiciones es importante destacar la realización de la visualización de los registros de pozos para lograr una mejor interpretación de las características de un pozo petrolero y así obtener las propiedades físicas de las rocas como la porosidad, la saturación de hidrocarburos, el índice de permeabilidad, el espesor efectivo, la textura, la estructura y la fracturación. Con el desarrollo de las tecnologías, en el CEINPET se han adquirido software propietarios que garantizan la visualización de estos registros de manera eficiente y rápida. Con ellos es posible la representación de las curvas de los registros originales y los resultados de la interpretación, los gráficos de propiedades cruzadas, los histogramas[11] de frecuencia de los registros y las propiedades, las correlaciones entre pozos, la representación de espesores efectivos y otros criterios.

1.9.2 Descripción del proceso actual.

En la actualidad el CEINPET no cuenta con un software para el análisis petrofísico que sea netamente cubano, compra a empresas reconocidas internacionalmente estos sistemas en vista de darle solución a la actividad petrofísica que aquí se desarrolla y buscando alternativas factibles que propicien un

[10] Es un conjunto de actividades o eventos (coordinados u organizados) que se realizan o suceden.
[11] Un histograma es un resumen gráfico de la variación de un conjunto de datos.

trabajo más hacedero, ya que ninguno de estos sistemas informáticos están diseñados para satisfacer las necesidades específicas de los trabajadores de la empresa.

Dentro de los sistemas comprados más usados se encuentran el Elemental Analysis (ELAN), el Interactive Petrofisic (IP), y el Hidrocarbure Data Systems (HDS). El ELAN es un programa de análisis de registro multimineral capaz de calcular la formación más probable de minerales y volúmenes de fluidos de poros. Es en particular valioso en áreas de litología[12] variada, minerales especiales y sistemas de porosidad duales. [23]. El IP permite ir desde flujos de trabajos sencillos, como el control de calidad de los registros para realizar tareas complejas de varias zonas. Puede ser usado tanto por petrofísicos como por geólogos [24]. Las aplicaciones del HDS están diseñadas para todo tipo de usuarios. Su proceso de desarrollo utiliza las mejores prácticas y tecnologías disponibles para sus tipos de aplicaciones [22].

De ellos se tiene muy poca experiencia de trabajo con el ELAN, que aunque está elaborado en plataforma libre, se hace muy difícil su uso porque se necesita de mucha experiencia y práctica; contradiciendo el propósito de tener un software amigable con el usuario.

Por último en el CEINPET se evidencian conocimientos avanzados para trabajar con el HDS, software con el que se han obtenido resultados exitosos hasta el momento, además del fortalecimiento de las habilidades para su uso.

1.9.3 Situación problemática.

Partiendo de la gran importancia que representa el petróleo a nivel mundial, no solo para la obtención de energía a partir de subproductos combustibles como la gasolina, el gasoil, fueloil, etc., sino también de otros productos para la elaboración de plásticos, fertilizantes, pinturas, productos químicos, fibras sintéticas entre otros, es que el CEINPET está obligado a utilizar software propietarios para realizar el análisis a los Registros de pozos. Estos software luego de ser adquiridos, no permiten realizar actualizaciones y debido a que son propietarios no se puede acceder a su código fuente para una posible mejora del mismo. La mayoría de estos sistemas cuentan con una llave única que restringe su uso a una computadora, a la vez haciendo engorroso el trabajo y despojando al resto de los trabajadores de su utilidad. Estos productos software son elaborados para todo tipo de empresa, de esta forma se aleja de las necesidades específicas de los trabajadores del CEINPET y en muchas ocasiones se necesita de experiencia avanzada para trabajar con los mismos, siendo necesarios establecer cursos avanzados para preparar al personal de trabajo, perdiéndose en tiempo y recursos.

[12] 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.

Por tal motivo es que el equipo de proyecto Petrofísica del polo productivo Petrosoft de la facultad 9 de la UCI, se encuentra desarrollando el sistema ANPER que aún no cuenta con el módulo de visualización.

1.10 Análisis de soluciones existentes

Cuba no cuenta en la actualidad con un software para visualizar los registros de pozos por tal motivo el CEINPET se auxilia en software extranjeros. Dentro de los sistemas que realizan visualización de registros de pozos se destaca el STRATER, una potente e innovadora herramienta para el registro de pozos y el trazado de perforaciones. Con STRATER se puede visualizar gráficamente:

- Profundidad o elevación.
- Notas, comentarios y otros datos de texto.
- Petrología[13], % petrología y descripciones litológicas.
- Potencial espontáneo, rayos gamma, calibrador, neutrónica, DST, densidad aparente, Resistividad, ritmo de perforación, gas total, calidad de los gases y datos sónicos.
- Contabilidad de impactos, número y tipo de muestras, permeabilidad, RQD, lecturas OVM, %recuperación.
- Concentración de contaminantes, contenido de humedad y detalles de construcción de pozos
- Datos de ensayos, petrología de mineralización o alteración, valores BTU y datos de contenido de cenizas.
- Virtualmente cualquier tipo de dato de profundidad o intervalo.

Su avanzada interfaz[14] de usuario permite que el diseño y la visualización de sus datos sean más fáciles que nunca. Strater incluye 13 tipos de registro muy usuales para visualizar sus datos gráficamente: profundidad, línea/símbolo, gráfico cruzado, petrología, barras de zona, barras, porcentajes, postes, postes por clase, gráficos, textos complejos, y registros de construcción de pozos.

Cada uno de los registros puede ser modificado para ajustarse a las necesidades del usuario. Entre algunas de las personalizaciones posibles podemos citar:

- Visualizar registros basados en profundidad o elevación.
- Visualizar profundidad y/o líneas de rejilla variable.

[13] Es la rama de la geología que se ocupa del estudio de las rocas desde el punto de vista genético y de sus relaciones con otras rocas.
[14] Es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora.

- Añadir barras de escala y títulos.
- Configurar diferentes estilos de líneas de contacto entre unidades litológicas.
- Utilizar curvas para crear perfiles litológicos como perfiles de desgaste [12].

Las aplicaciones **HDS** están diseñadas para una máxima flexibilidad, es decir para todo tipo de usuarios. Su proceso de desarrollo utiliza las mejores prácticas y tecnologías disponibles para sus tipos de aplicaciones. Continuamente actualiza las funcionalidades del usuario y las herramientas para facilitar un mejor flujo de trabajo de los procesos para los mismos. HDS se desarrolla en. NET[15], despliega diferentes métodos de prueba, control de calidad de procesos, y es amigable respecto al cliente. [22]. Dentro de las funcionalidades principales del HDS encontramos:

- Visualizar registros que incluye crear, actualizar y guardar plantillas.
- Cambiar colores y escalas.
- Imprimir y guardar registros con encabezamientos.
- Se pueden modelar parámetros para el análisis petrofísico como la profundidad, los indicadores de arcilla de los cuales se puede modificar, insertar y eliminar datos.
- Agregar pistas con las que se van a trabajar.
- Permite agregar, eliminar y cambiar formato de límites.
- Permite insertar y modificar imágenes.

Este software al ser propietario cuenta con una llave única para su uso. Estos restringen su utilización solamente a una computadora a la vez, dejando al resto del personal inactivo. Solamente puede ser usado en varias computadoras en dependencia de la cantidad de licencias que la empresa cuente del software en cuestión, que tienen por característica precios elevadísimos y sin la posibilidad de actualizar sus servicios.

El software **Neuralog** es usado por la comunidad Geocientífica cuando los requerimientos de calidad y el tiempo son críticos. Cuenta con nuevos sistemas de digitalización automatizada que hacen la captura de datos fáciles, rápidos y de inmejorable calidad. Acepta imágenes TIFF a color, escala de grises o blanco y negro. Igualmente permite la importación de archivos formato LAS [13].
Neuralog incluye las siguientes funcionalidades especiales para garantizar la captura de datos:

[15] Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software.

- Definiciones de varios tracks y curvas.
- Corrección de deformación de imágenes (estiramientos, encogimientos, etc.).
- Soporta la vectorización de registros de lodos y Dipmeters.
- Soporta la vectorización de escalas lineales, logarítmicas y multilineales.
- Función para el copiado y pegado de curvas.
- Cálculo automático entre las curvas de resistividad y conductividad.
- Opción para el manejo de fuera de escalas.
- Vectorización de curvas a color.
- Generador de estadísticas de archivos LAS.
- Herramientas para la verificación de la calidad de vectorización [14].

1.11. Conclusiones Parciales

En este capítulo se realizó un estudio de los principales elementos que conforman a los sistemas para visualizar curvas de registros de pozos. Se dió a conocer conceptos fundamentales como: pista,curva entre otros. El estudio realizado anteriormente a los sistemas existentes permitió llegar a la conclusión:

- Los sistemas utilizados por el CEINPET para realizar la visualización de las curvas de los registros de pozos, no se adaptan a las necesidades específicas de los trabajadores.
- Se eligió la gráfica de línea para visualizar las curvas de registros de pozos, debido a que esta gráfica permite mostrar como se comporta un parámetro físico del pozo con respecto a la profundidad en que se halla hecha esta medición.
- Se decidió incluir en el desarrollo del componente para visualizar curvas de registros de pozos las siguientes funcionalidades:

De STRATER:

- Visualizar curvas como: potencial espontáneo, rayos gamma, calibrador, neutrónica, DST, densidad aparente, resistividad, ritmo de perforación, gas total, calidad de los gases y datos sónicos.
- Visualizar registros basados en profundidad o elevación.

De Neuralog:

- Soporte para la vectorización de escalas lineales, logarítmicas y multilineales.

- Vectorización de curvas a color.

De HDS:

- Agregar pistas con las que se van a trabajar.
- Cambiar colores y escalas.

Capítulo 2 "Tendencias y tecnologías a utilizar"

2.1 Introducción

Las tecnologías influyen en el progreso social y económico del hombre. Se presentan cada vez más como una necesidad en el contexto de la sociedad donde los rápidos cambios, el aumento de los conocimientos y las demandas de la información constantemente actualizada se convierten en una exigencia permanente. La principal finalidad de las tecnologías es transformar el entorno, para adaptarlo mejor a las necesidades y deseos humanos. En la actualidad las empresas manifiestan muchos cambios en la apreciación de su efectividad y nivel de competitividad en el mercado. Esto se debe en gran medida al desarrollo de la tecnología informática, que contribuye a la gestión y transferencia de información. Dentro de los logros obtenidos a través de las Tecnologías de la Información y las Comunicaciones (TIC), se encuentran poderosas herramientas que sirven de apoyo en la construcción de software para el desarrollo tanto económico, como social. Aunque no basta para la elaboración de software el uso de herramientas sin aplicar metodología(s) y procesos que guíen todo su desarrollo en vista a cumplir sus objetivos. Es por ello que se hace imprescindible para el desarrollo eficiente de los sistemas informáticos actuales, el uso de las mejores y más apropiadas metodologías, herramientas y procesos para lograr objetivos específicos.

En este capítulo se hará un estudio de las principales herramientas que se usarán en el desarrollo del componente para visualizar curvas de registros de pozo definidas con anterioridad por el arquitecto del proyecto Petrofísica de la Facultad 9.[15]. Además se abordarán aspectos tales como: Las Tecnologías de la Información y las Comunicaciones (TIC), características del rol[16] implementador etc.

2.2 Las Tecnologías de la Información y las Comunicaciones (TIC)

Las Tecnologías de la Información y las Comunicaciones (TIC) son incuestionables, forman parte de la cultura tecnológica que nos rodea y con la que debemos convivir. Amplían nuestras capacidades físicas, mentales y las posibilidades de desarrollo social. Provocan continuas transformaciones en nuestras estructuras económicas, sociales y culturales, e inciden en casi todos los aspectos de la vida: el acceso al mercado de trabajo, la sanidad, la gestión burocrática, la gestión económica, el diseño industrial y artístico, el ocio, la comunicación, la información [25].

En resumen las TIC son aquellas herramientas computacionales e informáticas que procesan, almacenan, sintetizan, recuperan y presentan información representada de la más variada forma. Es

[16] Papel que desempeña un trabajador en el desarrollo de un software.

un conjunto de herramientas, soportes y canales para el tratamiento y acceso a la información. Algunos ejemplos de estas tecnologías son la pizarra digital (ordenador personal + proyector multimedia), los blogs[17], el podcast[18] y, por supuesto, la web. Su gran impacto en todos los ámbitos de nuestra vida hace cada vez más difícil que podamos actuar eficientemente prescindiendo de ellas.

2.3 Metodología de Desarrollo de Software

En un proyecto de desarrollo de software la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. No existe una metodología de software universal, las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable [26].

Existen dos grupos fundamentales de metodologías, las metodologías tradicionales las cuales están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto y las metodologías ágiles que ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente.

Las metodologías tradicionales centran su atención en cumplir con un plan estratégico y organizado del proyecto, definido fundamentalmente en la fase inicial de su desarrollo. Las metodologías tradicionales (formales) se focalizan en documentación, planificación y procesos (Plantillas, técnicas de administración, revisiones, etc.). [58]. Dentro de las metodologías tradicionales se destacan el Rational Unified Process (RUP) y el Microsoft Solution Framework (MSF).

Las metodologías ágiles se basan en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa, esto permite potenciar aún más el desarrollo de software a gran escala. Dentro de sus principales ideas están que los individuos y las interacciones entre ellos, son más importantes que las herramientas y los procesos empleados, es más importante crear un producto software que funcione, que escribir documentación exhaustiva, la colaboración con el cliente debe prevalecer sobre la negociación de contratos y la capacidad de respuesta ante un cambio, es más importante que el seguimiento estricto de un plan. [58]. Algunas de las metodologías ágiles más usadas están Modelo de Extreme Programming (XP), AUP (Agile Unified Process), Scrum e Iconix.

[17] Es un sitio web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores.
[18] Se asemeja a una suscripción a un blog hablado en la que recibimos los programas a través de Internet.

2.3.1 Rational Unified Process (RUP)

RUP es un proceso formal que provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Utiliza UML como lenguaje de notación. [27]. Es un proceso estándar y flexible, al que se le pueden realizar variaciones en dependencia de la aplicación que se desea desarrollar. Esta metodología, además de usarse en el desarrollo de software también se ha usado en otras ramas y procesos industriales. [29]

2.3.1.1 Características de RUP

RUP tiene entre sus principales características que es iterativo e incremental, centrado en la arquitectura y guiado por casos de uso.

La característica de RUP de ser **Iterativo e Incremental** propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en pequeños pedazos o mini-proyectos. Cada mini-proyecto es una iteración que finaliza en un incremento. Las iteraciones se refieren a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto. Para ser más efectivo, las iteraciones deben estar controladas, es decir, deben ser seleccionadas y llevadas a cabo de una manera planeada. La iteración trata con un grupo de casos de uso que en conjunto extienden la usabilidad del producto y trata con los riesgos más importantes. Las iteraciones sucesivas construyen los artefactos del desarrollo a partir del estado en el que fueron dejados en la iteración anterior. [1]

La iteración proporciona un resultado completo, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental. Para ello, cada requisito se debe completar en una única iteración: el equipo debe realizar todas las tareas necesarias para completarlo y que esté preparado para ser entregado al cliente con el mínimo esfuerzo necesario. De esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos. [2]

RUP es una metodología **Dirigida por Casos de Uso** ya que un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor, capturan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la

funcionalidad completa del sistema. Este modelo reemplaza la tradicional especificación funcional del sistema. [1]

Los casos de uso no son solamente una herramienta para especificar los requerimientos del sistema, también dirigen su diseño, implementación y pruebas, esto es, dirigen el proceso de desarrollo. Los casos de uso son desarrollados a la par con la arquitectura del sistema, esto es, los casos de uso dirigen la arquitectura del sistema y la arquitectura del sistema influencia la elección de los casos de uso. Por lo tanto, la arquitectura del sistema y los casos de uso maduran conforme avanza el ciclo de vida. [28]

Para entender el planteamiento de que RUP está **Centrado en la Arquitectura** hay que tener en cuenta que el concepto de arquitectura de software involucra los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura es la vista del diseño completo con las características más importantes hechas más visibles y dejando los detalles de lado. La arquitectura debe proveer espacio para la realización de todos los casos de uso, ambos deben evolucionar en paralelo. [28] La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU[19] relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

En RUP se han definido las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales, los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. A continuación se muestra las fases e iteración de la metodología RUP (figura 4).

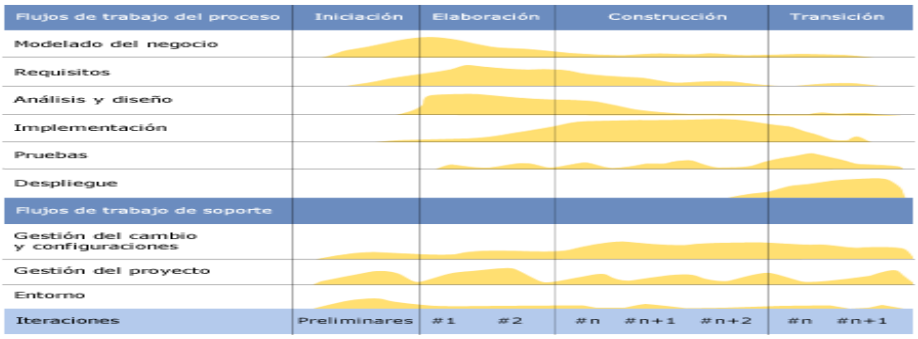

Figura 4. Fases e Iteración de la Metodología RUP [44].

[19] Casos de uso.

En resumen RUP como metodología de desarrollo provee un entorno de proceso de desarrollo configurable, basado en estándares[20], permite tener claro y accesible el proceso que se sigue, además de ser configurado según las necesidades de la organización y del proyecto. Utiliza mejores prácticas probadas en la industria y provee a cada participante con la parte del proceso que le compete directamente, filtrando el resto.

2.3.2 Microsoft Solution Framework (MSF).

MSF es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información. [58]

Los equipos organizados bajo este modelo son pequeños y multidisciplinarios, en los cuales los miembros comparten responsabilidades y balancean las destrezas del equipo para mantenerse enfocados en el proyecto que están desarrollando. Comparten una visión común del proyecto y se enfocan en implementar la solución, con altos estándares de calidad y deseos de aprender. [58]

El modelo de equipos de MSF tiene seis roles que corresponden a las metas principales de un proyecto y son responsables por las mismas. Cada rol puede estar compuestos por una o más personas, la estructura circular del modelo, con óvalos del mismo tamaño para todos los roles, muestra que no es un modelo jerárquico y que todos los roles son igualmente importantes en su aporte al proyecto. Aunque los roles pueden tener diferentes niveles de actividad durante las diversas etapas del proyecto, ninguno puede ser omitido. [58]

La comunicación se pone en el centro del círculo para mostrar que está integrada en la estructura y fluye en todas direcciones. El modelo apoya la comunicación efectiva y es esencial para el funcionamiento del mismo. [58] Todo proyecto es separado en cinco principales fases: Visión y Alcances, Planificación, Desarrollo, Estabilización e Implantación. En la Figura 14 se muestra una representación de la Metodología MSF.

[20] Es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad.

Figura 5. Representación de la Metodología MSF [60].

MSF combina la claridad que planea el modelo en cascada y las ventajas de los puntos de transición del modelo en espiral; incorpora a su equipo de trabajo un rol más en relación a su anterior versión, que es, el Rol de Arquitectura; hace un análisis completo sobre que roles pueden fusionarse, combinarse, es decir que roles pueden ser desarrollados por una sola persona; existe mayor vinculación con el cliente como también orientado al trabajo en equipo y siendo una metodología adaptable al proyecto, le permite ser más flexible que el RUP, garantizando la probabilidad de éxito del proyecto mediante el análisis de riesgo. [59]

Sin embargo esta metodología presenta una gran dificultad que no presenta Rup y es que los precios de licencias, capacitación y soporte por parte de Microsoft son excesivamente caros. [59]

2.4. Programación Orientada a Objeto

En los últimos años la Programación Orientada a Objetos(POO) ha alcanzado un alto grado de popularidad, aceptación y utilización, facilita la creación de software de calidad por sus factores que potencian el mantenimiento, la extensión y la reutilización del software generado bajo sus instancias. La POO trata de amoldarse al modo de pensar del hombre y no al de la máquina. Esto es posible gracias a la forma racional con la que se manejan las abstracciones[21] que representan las entidades del dominio del problema, y las propiedades como la jerarquía[22] o el encapsulamiento.[30]

La Programación Orientada a Objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación. Con la POO hay que

[21] Es un acto mental en el que se aisla conceptualmente un objeto o una propiedad de un objeto.
[22] Designa una forma de organización de diversos elementos de un determinado sistema.

aprender a pensar las cosas de una manera distinta, para escribir los programas en términos de objetos, propiedades, métodos. [31]

Podemos decir entonces, que la POO, intenta simular el mundo real a través del significado de objetos que contienen características y funciones y que facilita el trabajo del programador, puesto que un programa grande siempre será más complicado que la suma de varios programas pequeños, con lo que se considera ventajoso dividir un gran sistema en diversos módulos.

2.5 Lenguajes de Programación

Los lenguajes de programación son herramientas que nos permiten crear programas y software. Una computadora funciona bajo el control de uno de esos programas que debe ser entendido por esta a través del código o lenguaje de máquinas. Es decir, este programa debe ser codificado utilizando específicamente el lenguaje de máquina.

No obstante lo anterior, los lenguajes de programación en código de máquina son verdaderamente difíciles de entender para una persona, ya que están compuestos de códigos numéricos sin sentido nemotécnico[23] . De ahí la importancia de la mayoría de los demás lenguajes de programación, los cuales facilitan la tarea, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar, representan en forma simbólica y en manera de un texto los códigos que podrán ser leídos por los seres humanos.[30]

Existen estrategias que permiten ejecutar en una computadora un programa realizado en un lenguaje de programación simbólico. Los procesadores del lenguaje son los programas que permiten el tratamiento de la información en forma de texto, representada en los lenguajes de programación simbólicos. Los lenguajes de programación pueden ser interpretados (que usan un intérprete) o compilados (que usan un compilador[24]). La ejecución de un programa con compilador requiere de dos etapas:

- Traducir el programa simbólico a código máquina.
- Ejecución y procesamiento de los datos.

Otros lenguajes de programación utilizan un programa intérprete o traductor, el cual analiza directamente la descripción simbólica del programa fuente y realiza las instrucciones dadas.

[23] La nemotécnica es una técnica que trata de fortalecer la memoria a través de la utilización de asociación de ideas, esquemas.

[24] Es un programa que se encarga de traducir los programas escritos por el programador en lenguaje de alto nivel, es decir más cercano al lenguaje humano.

El intérprete en los lenguajes de programación simula una máquina virtual , donde el lenguaje de máquina es similar al lenguaje fuente. La ventaja del proceso intérprete es que no necesita de dos fases para ejecutar el programa, sin embargo su inconveniente radica en la velocidad de ejecución debido a que es más lenta al analizar e interpretar las instrucciones contenidas en el programa fuente.[30]
Entre los lenguajes de programación más conocidos y utilizados en la actualidad se pueden citar: Delphi, Visual Basic, Pascal, C#, Java, etc.

2.5.1Lenguaje de Programación C#

C# es un lenguaje propuesto por Microsoft para satisfacer las necesidades actuales y de un futuro cercano. Es, por tanto, una herramienta, como todos los lenguajes de programación, pero adaptada al trabajo actual.
El objetivo de Microsoft[25] ha sido la creación del primer lenguaje orientado a componentes, al estilo de Visual Basic, pero con la flexibilidad y potencia de C++ y sin muchas de sus complejidades. C# ha sido diseñado para una plataforma, la plataforma Microsoft .NET, en la que los servicios son ofrecidos en forma de componentes. Cuenta con construcciones sintácticas nativas para la definición, implementación y consumo de propiedades, métodos y eventos, lo cual le diferencia claramente de C++ y Java.

Para construir un componente no es necesario hacer nada especial aparte de crear una nueva clase. Dicho de otro modo, cualquier clase de objeto C# es un componente, sin necesidad de crear GUID (Globally Unique Identifier) para interfaces y clases de componentes.

Es un lenguaje completamente orientado a objetos con una gran cantidad de características y mejoras sobre sus predecesores. Al igual que Java, trabaja en un entorno manipulado de memoria, aislando al programador de los engorrosos detalles del hardware[26]. A pesar de esto, el programador tendrá la posibilidad de renunciar a esta característica y lidiar por sí mismo con la manipulación de la memoria [32].

Su competir más cercano es Java, lenguaje con el que guarda un enorme parecido. En este aspecto, es importante señalar que C# incorpora muchos elementos de los que Java carece como: el

[25] Es una empresa multinacional estadounidense, fundada en 1975 por Bill Gates y Paul Allen, dedicada al sector de la informática.
[26] Corresponde a todas las partes físicas y tangibles de una computadora.

rendimiento, el cual es mejor, soporta más tipos primitivos, incluyendo tipos numéricos sin signo, compilación condicional, aplicaciones multi-hilo simplificadas; y otros [33].

Las principales características que definen al lenguaje son:

- **Sencillez de uso:** Elimina muchos elementos añadidos por otros lenguajes y que facilitan su uso y comprensión, como por ejemplo ficheros de cabecera, o ficheros fuentes IDL.
- **Modernidad:** Al ser un lenguaje de última generación, incorpora elementos que se ha demostrado a lo largo del tiempo que son muy útiles para el programador, como tipos decimales o booleanos, así como una instrucción que permita recorrer colecciones con facilidad (instrucción foreach). Estos elementos hay que simularlos en otros lenguajes como C++ o Java.
- **Orientado a objetos:** Como lenguaje de última generación es orientado a objetos. Además, soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo.
- **Orientado a componentes:** La propia sintaxis incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular.
- **Recolección de basura:** Todo lenguaje incluido en la plataforma .NET tiene a su disposición el recolector de basura.
- **Eficiente:** Todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros.
- **Compatible:** Para facilitar la migración de programadores de C++ o Java a C#, no sólo se mantiene una sintaxis muy similar a la de los dos anteriores lenguajes, sino que también ofrece la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos, tales como las DLLs[27] de la API de Win32.

2.6 Lenguaje de Modelado UML

UML es un conjunto de herramientas, que permite modelar (analizar y diseñar) sistemas orientados a objetos, se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un

[27] Son bibliotecas de enlace dinámico.

lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.[40]

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar. UML no puede compararse con la programación estructurada, pues significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento[28]. [40]

Puede conectarse con lenguajes de programación (Ingeniería directa e indirecta), permite documentar todos los artefactos[29] de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.), cubre las cuestiones relacionadas con el tamaño propio de los sistemas complejos y críticos, es un lenguaje muy expresivo que abarca todas las vistas necesarias para desarrollar y luego desplegar los sistemas. Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar, brinda la posibilidad de establecer conceptos y artefactos ejecutables. Es considerado como el mejor soporte a la planeación y al control de proyectos y permite una alta reutilización y minimización de costos. [41]

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. Utiliza técnicas orientadas a objetos (OO). [40]

Cabe resaltar que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación que ha puesto fin a las llamadas “guerras de métodos” pues constituye una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

28 Es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio.
29 Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

2.7 Herramientas de Desarrollo

En el presente epígrafe se describen las herramientas a usar en el desarrollo del componente para visualizar curvas de registro de pozos.

2.7.1Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. [34]

Se integra con varios ambientes de desarrollo integrados (IDE) lo cual posibilita pasar del código al modelado y viceversa. Establece interoperabilidad con otras aplicaciones como el Visio[30] y el Rational Rose. Disponible en múltiples lenguajes y plataformas: Microsoft Windows (98, 2000, XP, o Vista) Linux, Mac OS X, Solaris o Java. [35]

Presenta como ventajas fundamentales el hecho de ser una herramienta multiplataforma de modelado visual UML y una herramienta CASE muy potente y fácil de utilizar. Además VP-UML [31]aporta a los desarrolladores de software una plataforma de desarrollo puntera para construir aplicaciones de calidad mejores y más baratas con rapidez. Aporta una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos IDE líderes del mercado. [36]

2.7.2 Visual Studio 2008

.NET es una tecnología desarrollada por Microsoft con los objetivos principales de mejorar los sistemas operativos [32]y de obtener un entorno diseñado para el desarrollo y ejecución del software en forma de servicios que puedan ser accedidos a través de Internet de forma independiente al lenguaje de programación, sistema operativo y hardware utilizados tanto para desarrollarlos como para publicarlos.

Visual Studio .NET es la herramienta de desarrollo multilenguaje más completa para construir e integrar rápidamente aplicaciones y servicios Web XML (Extensible Markup Language, Lenguaje de Marcas Ampliable). Aumenta de un modo extraordinario la productividad de los desarrolladores y crea nuevas oportunidades de negocio. En su diseño se han integrado a fondo los estándares y protocolos

30 Es un software de dibujo vectorial para Microsoft Windows.
31 Visual Paradigm for UML.
32 Es un software que actúa de interfaz entre los dispositivos de hardware y los programas usados por el usuario para manejar un computador.

de Internet, como XML y SOAP (Simple Object Access Protocol), por lo que Visual Studio .NET simplifica considerablemente el ciclo de vida del desarrollo de aplicaciones [38].
Visual Studio 2008 tiene un elevado aumento de la productividad al escribir aplicaciones dirigidas a la nueva versión de .NET Framework, esto incluye la ampliación de tipos de proyectos, siempre en evolución y los aspectos de equipo orientados a la ingeniería de software [39].

A las mejoras de desempeño, escalabilidad y seguridad con respecto a la versión anterior, se agregan entre otras, las siguientes novedades:

- Mejora en las capacidades de Pruebas Unitarias**:** Son ejecutas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos.
- Visual Studio Tools for Office (VSTO): Integrado con Visual Studio 2008 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project.
- LINQ (Language Integrated Query): Nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos.
- Soluciones multiplataforma**:** Visual Studio 2008 ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework.

Visual Studio 2008 como IDE de desarrollo ofrece la visión de las aplicaciones clientes inteligentes, al permitir a los desarrolladores con avanzadas herramientas de desarrollo, y otras características innovadoras, la creación de aplicaciones de manera rápida a través de diversas plataformas.

2.8 Framework 3.0

El framework 3.0 Net, es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. Su diseño está enfocado a cumplir los objetivos siguientes:[15]

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, pero distribuida en Internet[33] o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.[37]

2.9 Conclusiones Parciales

Para el desarrollo del componente para visualizar curvas de registros de pozos se decidió utilizar:

- El lenguaje de programación C-Sharp por ser simple, eficaz y orientado a objetos. Además para proveer al CEINPET una solución inmediata a sus necesidades, aprovechando el conocimiento de los programadores y la facilidad que tiene C# como lenguaje de programación, para construir aplicaciones rápidamente.
- La metodología tradicional por conocer claramente los requerimientos, tanto funcionales como no funcionales del sistema y obligar al cliente a tomar decisiones al inicio del proyecto basándose en normas provenientes de estándares seguidos por el entorno de desarrollo. Además las metodologías ágiles pese a que permiten disminuir costos y brindar flexibilidad a los proyectos de software, se deben aplicar en proyectos donde exista mucha incertidumbre, donde el entorno es volátil, donde los requisitos no se conocen con exactitud.
- La metodología tradicional RUP por estar integrado con toda la suite Rational de herramientas y poder ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Además para poder realizar una evaluación en cada fase, establecer cambios de

[33] Es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.

objetivos y por tener MSF precios de licencias, capacitación y soporte por parte de Microsoft excesivamente caros.

- La herramienta UML Visual Paradigm por representar una colección premiada de herramientas que facilita las organizaciones visuales y el diagrama de diseño. Además de poseer la ventaja de ser una herramienta de libre uso y multiplataforma, lo que permitirá migrar en un futuro el componente a software libre.

Capítulo 3: "Propuesta y solución del componente".

3.1 Introducción

En este capítulo se hace referencia a las capacidades que el componente para visualizar curvas de registros de pozo debe cumplir (Requerimientos funcionales), así como las cualidades que debe tener (Requerimientos no funcionales). Se definirán los casos de uso a implementar y se describirán a los actores que interactuarán con el componente. Además se realizará el diseño teniendo en cuenta la tecnología de implementación, pues de no ser así se estaría incurriendo en un gasto de tiempo en vano. Se especifica el modelo de implementación y las pruebas realizadas al componente en vísperas de optener un producto con la mejor calidad posible.

3.2 Especificación de los requerimientos del software a implementar

Para modelar el sistema, se identifican sus requisitos, tanto funcionales como no funcionales, y se modelan los funcionales en términos de casos de uso del sistema.

3.2.1 Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir [45].

RF1 Editar la Representación del Registro de Pozo

El sistema debe permitir que el Petrofísico pueda editar o modificar la Representación del Registro de Pozo, realizando cambios en sus propiedades a voluntad del cliente.

RF1.1 El sistema debe permitir que el Petrofísico pueda modificar el número de una pista.

RF1.2 El sistema debe permitir que el Petrofísico pueda seleccionar la visibilidad de la pista.

RF1.3 El sistema debe permitir que el Petrofísico pueda seleccionar la visibilidad del grid[34].

RF1.4 El sistema debe permitir que el Petrofísico pueda seleccionar el tipo de escala (lineal, logarítmica) para la visibilidad de la curva.

RF1.5 El sistema debe permitir en caso de ser escala lineal que la curva en la pista se represente de forma lineal.

RF1.6 El sistema debe permitir en caso de ser escala logarítmica que la curva en la pista se represente de forma logarítmica.

[34] Es una delimitación del área de graficación para entender mejor el significado de un gráfico.

RF1.7 El sistema debe permitir que el Petrofísico pueda visualizar el registro completo de una curva.

RF1.8 El sistema debe permitir que el Petrofísico pueda seleccionar la visibilidad de la curva de una pista determinada.

RF1.9 El sistema debe permitir que el Petrofísico pueda modificar el color de la curva en una pista determinada.

RF1.10 El sistema debe permitir que el Petrofísico pueda modificar el grosor de la curva en una pista determinada.

RF1.11 El sistema debe permitir que el Petrofísico pueda modificar el estilo de la curva (línea continua, línea discontinua) en una pista determinada.

RF1.12 El sistema debe permitir que el Petrofísico pueda modificar el nombre de la curva en una pista determinada.

RF2 Componente de Gestión:

El sistema debe permitir adicionar, eliminar y guardar una pista.

RF2.1 El sistema debe permitir que el Petrofísico pueda adicionar una pista.

RF2.2 El sistema debe permitir que el Petrofísico pueda eliminar una pista.

RF2.3 El sistema debe permitir que el Petrofísico pueda guardar una pista.

3.2.2 Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. Son los aspectos del sistema visibles para el usuario, que no están relacionados de forma directa con el comportamiento funcional del sistema. Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, esto puede marcar la diferencia entre un producto bien aceptado y uno con poca aceptación [45].

RNF Usabilidad

Será usable por cualquier tipo de usuario con experiencia, básica, media o avanzada en los procesos de análisis petrofísicos.

Se mostrará la información de forma lógica y correctamente estructurada.

El sistema debe ser fácil de instalar y debe mostrar el avance de la instalación para guiar al usuario del tiempo de instalación que necesita y el que se va ejecutando.

RNF Apariencia o interfaz externa

El sistema tendrá un menú de herramientas para que el usuario pueda disponer de él en la medida de sus necesidades, visible todo el tiempo para facilitar el fácil acceso a las funcionalidades del software.
El sistema tendrá un ambiente agradable y sencillo, combinado con colores y una estructura amigable que permita que el usuario se adapte con facilidad.

El sistema permitirá claridad en los servicios que brinda, con términos asociados a los procesos de análisis petrofísicos, mejorando la comprensión del sistema.

El sistema tendrá una buena navegabilidad, propiciando varias opciones para acceder a cada servicio brindado.

RNF Rendimiento

El sistema debe garantizar un tiempo de respuesta de 3 a 5 segundos en dependencia de la complejidad del servicio.

RNF Seguridad

Confidencialidad: La información manejada por el sistema debe estar protegida de acceso no autorizado y divulgación. Los usuarios del sistema deben autenticarse antes de realizar cualquier actividad.
Integridad: La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, además de verificar las acciones irreversibles; es decir, se le solicitará al usuario la confirmación al realizar operaciones como la eliminación. Se aplicarán las reglas de la “programación segura”, mediante el tratamiento de excepciones.

Disponibilidad: Al personal autorizado se le garantizará el acceso al sistema y a la información en todo momento, mientras no haya un fallo (fluido electrico, conexión de red). El sistema debe tener habilitada la protección contra fallos en caso de errores.

RNF Hardware

El ordenador donde se instale el componente deberá cumplir con las siguientes características:
Se podrá contar con los periféricos Teclado y Mouse.
Requisito Mínimo: Procesador: Celeron con CPU 1.8 GHz, Memoria: 256 Mb, Disco Duro: 60 Gb.
Requisitos Recomendados: Procesador: Pentium IV 2.66 GHz, Memoria: 512 Mb, Disco Duro: 80 Gb.

RNF Software

Se utilizará como lenguaje de programación C#.

Se utilizará como Sistema Operativo, Windows 2000/XP/Vista, Linux, Unix, Mac OSX.

Se utilizará para el modelado la herramienta Visual Paradigm for UML.

Se utilizarán los IDE Visual Studio 2008.

Se utilizará para el servidor de control de versiones Subversion.

RNF Reusabilidad

Las funcionalidades del sistema se desarrollarán en forma de componentes con el objetivo de la reutilización de estos en futuras versiones del sistema y en el desarrollo de nuevos sistemas y aplicaciones.

La documentación de la ayuda y del sistema en general, deberá estar estructurada de tal manera que pueda ser reutilizada y permita la creación de una familia o línea de productos dentro del polo.
El sistema debe permitir posteriores modificaciones y actualizaciones, así como modificar elementos ya existentes.

El sistema estará dividido por módulos funcionales independientes.

El sistema permitirá la integración de nuevas funcionalidades mediante la incorporación de componentes desarrollados por cualquier empresa.

RNF Portabilidad

El sistema debe ser multiplataforma, siendo compatible con cualquier tipo de sistema operativo (SO GNU/Linux, SO Microsoft Windows).

3.3 Definición de los casos de uso del sistema.

A continuación se darán a conocer los actores que interactuarán con el componente para visualizar curvas de registros de pozos y los casos de uso a implementar.

3.3.1 Descripción de los actores.

Los 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. [51]. La descripción de los actores del sistema se encuentra a continuación en la Tabla 1.

Actores	Descripción
Petrofísico	Son las personas encargadas de realizar las operaciones para los procesos de análisis petrofísicos. Son las personas encargadas de obtener como resultado de la entrada del archivo .LAS en el sistema, la visualización de los registros de pozos y poder realizar informes valorativos.

Tabla 1. Descripción de los actores del sistema.

3.3.2 Diagrama de Casos de Uso del Sistema

Un diagrama de casos de uso del sistema es un modelo de las funciones deseadas para el sistema y su entorno, y sirve como contrato entre el cliente y los desarrolladores. Se utiliza como entrada esencial para las actividades de análisis y diseño. [51]. En la figura 6 se muestra el diagrama de casos de uso del sistema.

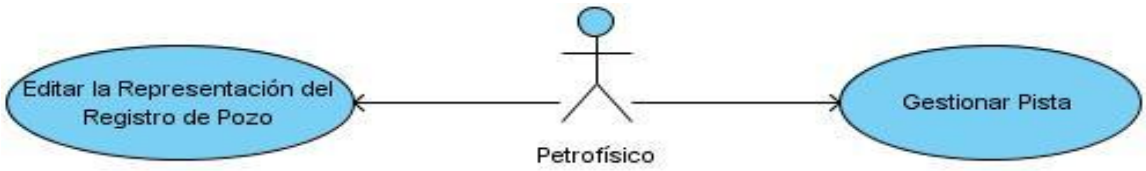

Figura 6. Diagrama casos de uso del sistema.

3.3.3 Casos de Uso del Sistema

Los casos de uso del sistema son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las capacidades o condiciones (requisitos funcionales) que debe cumplir el sistema. [51]. A continuación se muestra en la Tabla 2 la descripción

del caso de uso “Editar la Representación del Registro de Pozo por Criterios”. La descripción del caso de uso “Gestionar Pista” se encuentra en la Tabla 3 (Anexo 1).

Editar la Representación del Registro de Pozo Por Criterios.

Caso de Uso del sistema:	Editar la Representación del Registro de Pozo Por Criterios
Actores:	Petrofísico
Propósito	Realizar el estudio de la representación de los Registros de Pozos en vista a obtener las Evaluaciones de las Formaciones.
Resumen:	El Caso de Uso se inicia cuando el Petrofísico solicita editar la representación de los registros de pozo de forma personalizada y finaliza cuando se obtiene el informe valorativo de las formaciones.
Precondiciones:	Debe existir al menos una representación de registros de pozos en el área de trabajo.
Referencias	RF1, RF1.1, RF1.2, RF1.3, RF1.4, RF1.5, RF1.6, RF1.7, RF1.8, RF1.9, RF1.10, RF1.11, RF1.12.
Prioridad	Crítico

Flujo normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. El Petrofísico solicita editar la representación.	2. El sistema le muestra una ventana con los posibles criterios de modificación (modificar el número de la pista, modificar el nombre de la curva [que pueden ser DEPTH, GR, CALI o CAL, ILM, LD, PHIN, RHOB, CGR, SGR, SP, PEF, POTA, THOR, URAN, RX0, RLA0…RLA5, NPHI, DPHI, SPHI, DT, LLD, LLS, MSFL, RT], modificar el valor de la escala mínima, modificar el valor de la escala máxima, seleccionar el tipo de escala [que puede ser lineal o logarítmica], modificar el color de la curva, modificar el grosor de la curva, modificar el estilo de la curva [que puede ser línea continua,

	línea discontinua]. El sistema muestra además las opciones aceptar, cancelar, agregar, eliminar en cuatro botones.
3. El Petrofísico llena los campos requeridos para editar la representación del registro de pozo y selecciona la opción deseada.	4. El sistema Gestiona la Selección Si selecciona la opción Aceptar el sistema procede a visualizar la representación editada. Si selecciona la opción Cancelar se cierra la ventana sin guardar los cambios efectuados. Si selecciona la opción Agregar el sistema permite editar nuevos criterios de modificación. Si selecciona la opción Eliminar el sistema elimina un criterio de representación.
	5. El sistema actualiza las propiedades.

Prototipo de Interfaz *"Editar Representación"*

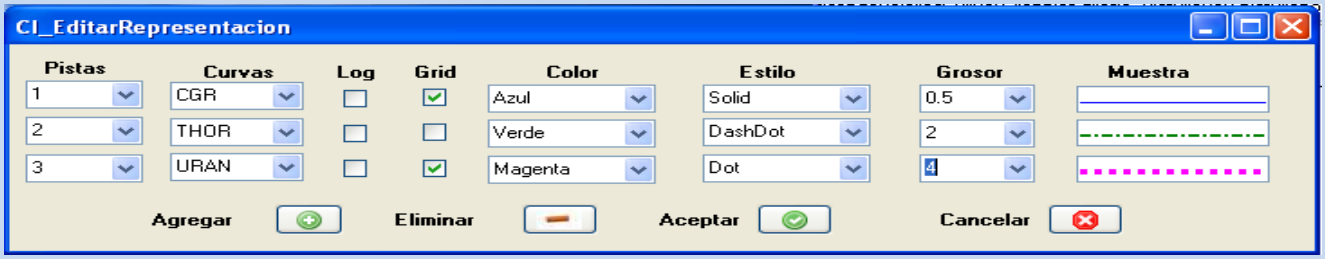

Figura 7. Prototipo de interfaz: Editar Representación.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones	Se logra realizar el estudio a través de la Edición de la Representación del Registro de Pozo por Criterios.

Tabla 2. Descripción del caso de uso del sistema "Editar la Representación del Registro de Pozo por Criterios".

3.4 Estilo Arquitectónico

Un estilo arquitectónico define una codificación particular de elementos de diseño. El estilo limita los tipos de elementos de diseño y su acuerdo formal. Un estilo arquitectónico restringe tanto los elementos de diseño como las relaciones formales entre ellos [48].

3.4.1 Arquitectura basada en componentes

El objetivo de la tecnología de componentes software es construir aplicaciones complejas mediante ensamblado de módulos (componentes) que han sido previamente diseñados por otras personas a fin de ser rehusados en múltiples aplicaciones. La ingeniería de programación que sigue esta estrategia de diseño es actualmente una de las más prometedoras para incrementar la calidad del software, abreviar los tiempos de acceso al mercado y gestionar el continuo incremento de su complejidad.

La arquitectura software de una aplicación basada en componentes consiste en uno o un número pequeño de componentes específicos de la aplicación (que se diseñan específicamente para ella), que hacen uso de otros muchos componentes prefabricados que se ensamblan entre sí para proporcionar los servicios que se necesitan en la aplicación.[51]

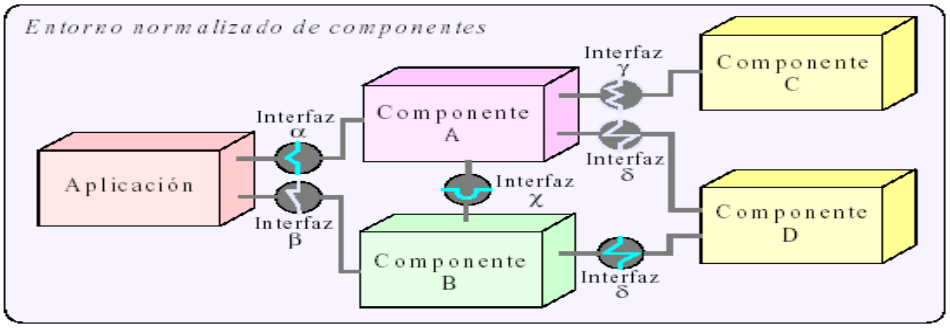

Figura 8. Arquitectura basada en componentes[51]

En la tecnología de componentes la interfaz constituye el elemento básico de interconectividad[35]. Cada componente debe describir de forma completa las interfaces que ofrece, así como las interfaces que requiere para su operación. Y debe operar correctamente con independencia de los mecanismos internos que utilice para soportar la funcionalidad de la interfaz.

Características muy relevantes de la tecnología de programación basada en componentes son la modularidad y la rehusabilidad y en todos ellos coincide con la tecnología orientada a objetos de la que se puede considerar una evolución. Sin embargo, en la tecnología basada en componentes también se requiere robustez ya que los componentes han de operar en entornos mucho más heterogéneos y diversos.[51]

[35] Posibilidad del usuario de pasar de un espacio de información a otro, en el momento en que su propia voluntad se lo indique, convirtiéndose en su propio intermediario para la selección de la información.

Ventajas:

- Reutilización del software. Lleva a alcanzar un mayor nivel de reutilización de software.
- Simplifica las pruebas. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- Simplifica el mantenimiento del sistema. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- Mayor calidad. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

Desventajas:

- Si no existen los componentes, hay que desarrollarlos y se puede perder mucho tiempo, así como en que estos componentes pueden tener conflictos si de estos sale una nueva versión es posible que haya que re- implementar estos componentes.
- Las actualizaciones de los componentes adquiridos no están en manos de los desarrolladores del sistema.[50]

En resumen se puede decir que el desarrollo de software basado en componentes es la evolución natural de la ingeniería software para mejorar la calidad, disminuir los tiempos de desarrollo y gestionar la creciente complejidad de los sistemas. Sin embargo, disponer de componentes no es suficiente tampoco, a menos que seamos capaces de reutilizarlos, y reutilizar un componente no significa usarlo más de una vez, sino que implica la capacidad del mismo de ser utilizado en contextos distintos a aquellos para los que fue diseñado.

3.5 Patrones utilizados

En el diseño del componente se utilizaron los siguientes patrones para dar solución a las diferentes problemáticas.

Patrones Grasp[36]

Experto: Patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con el no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. Indica que el objeto con la información necesaria para desempeñar la responsabilidad debería encargarse de ella. [52]

Bajo Acoplamiento: Es un principio que debemos recordar durante las decisiones de diseño: es la meta principal que es preciso tener presente siempre. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. [52].

Alta cohesión: Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.[52].

3.6. Modelo de Diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida. En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además, impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema. [53].

3.6.1 Diagrama de clases del Diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. [55]. El diseño se utiliza para que el programador en cuestión pueda realizar una correcta implementación del sistema a desarrollar.

Diagrama de clases del diseño “Editar la Representación del Registro de Pozo por Criterios”.

[36] Son patrones generales de software para asignación de responsabilidades.

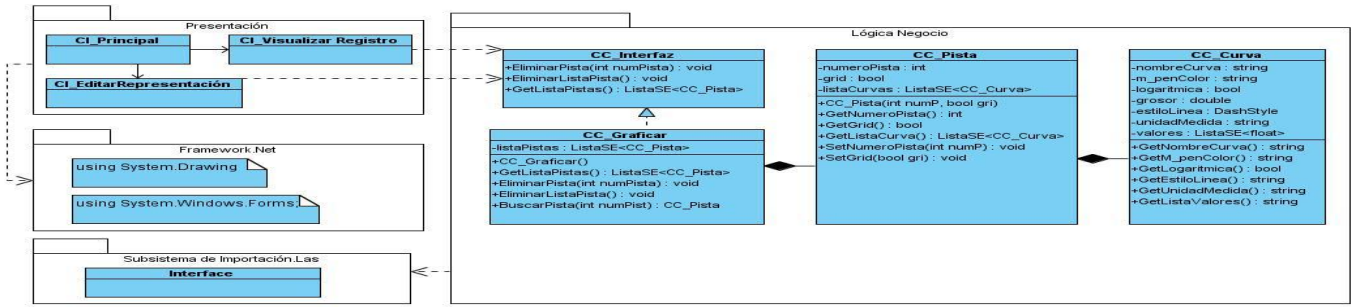

Figura 9. Diagrama de clases del diseño "Editar la Representación del Registro de Pozo por Criterios".

Diagrama de clases del diseño "Gestionar Pista".

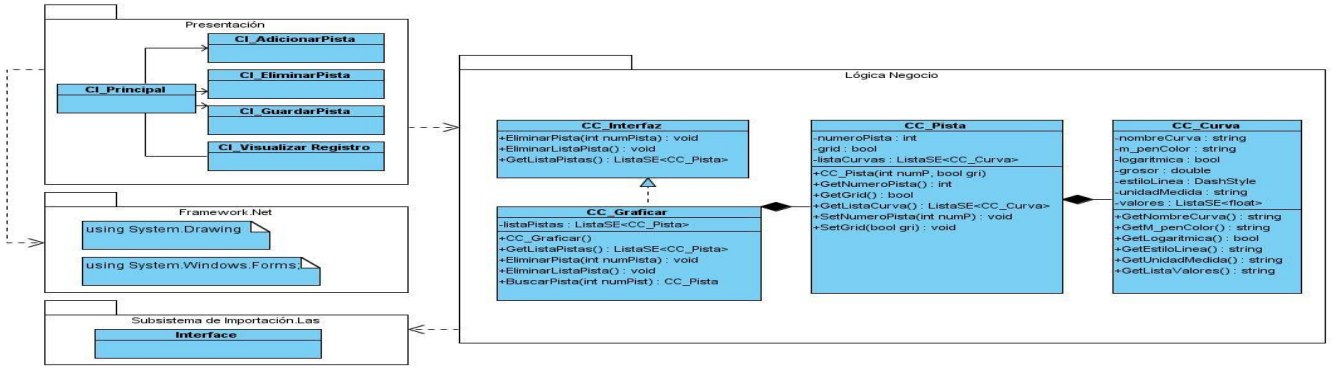

Figura 10. Diagrama de clases del diseño "Gestionar Pista".

3.6.2 Descripciones de las clases.

Nombre: CI_Principal	
Tipo de clase: Interfaz	
Atributo	**Tipo**
menuStrip1	MenuStrip
pictureBox1	PictureBox
button1	Button
button2	Button
button3	Button
button4	Button
button5	Button
button6	Button
Para cada responsabilidad:	
Nombre:	editarRepresentaciónToolStripMenuItem2_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_EditarRepresentacion.
Nombre:	button2_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_EditarRepresentacion.
Nombre:	salirToolStripMenuItem2_Click(object sender, EventArgs e)

Descripción:	Cierra el componente.
Nombre:	importarFicheroToolStripMenuItem1_Click(object sender, EventArgs e)
Descripción:	Importa el fichero a visualizar.
Nombre:	eliminarPistaToolStripMenuItem2_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_EliminarPista.
Nombre:	adicionarPistaToolStripMenuItem2_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_AdicionarPista.
Nombre:	button3_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_AdicionarPista.
Nombre:	button4_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_EliminarPista.
Nombre:	button5_Click(object sender, EventArgs e)
Descripción:	Cierra el componente.
Nombre:	button1_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_VisualizarRegistro.
Nombre:	guardarPistaToolStripMenuItem2_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_GuardarPista.
Nombre:	button6_Click(object sender, EventArgs e)
Descripción:	Habre la interfaz CI_GuardarPista.
Nombre:	CI_Principal()
Descripción:	Constructor de la forma.

Nombre: CI_VisualizarRegistro	
Tipo de clase: Interfaz	
Atributo	**Tipo**
grafic	Graphics
posicionX	int
posicionY	int
coorXMin	int
coorYMin	int
coordXMax	int
coordYMax	int
coordNombreCurvaX	int
coordNombreCurvaY	int
cordYGred	int
cordAuxYGred	int
listCoord	ListaSE<int>

Para cada responsabilidad:	
Nombre:	ValorMaximoCoordenadas()
Descripción:	Devuelve la última posición del eje y donde se dibujo.
Nombre:	DevolverLista()
Descripción:	Devuelve un listado de los pictureboxs donde se dibuja.
Nombre:	CI_VisualizarRegistro(Interfaz cc_graf,CImpExpRegistrosPozo control)
Descripción:	Constructor de la forma.
Nombre:	NormalizarProfundidad()
Descripción:	Devuelve un listado de la profundidad del registro de pozo adaptada al entorno de graficación.
Nombre:	NormalizarCurva(CC_Curva curva)
Descripción:	Devuelve un listado de las mediciones de la curva adaptada al entorno de graficación.

Nombre:	DibujarGrid(Graphics grafic, PictureBox picturebox, CC_Pista pista)
Descripción:	Dibuja el grid.
Nombre:	GenerarPista(int numPista,int x,int y)
Descripción:	Genera dinámicamente el picturebox donde se va a graficar la pista.
Nombre:	DibujarLineaHorizontal(Graphics grafic, PictureBox picturebox,int x,int y)
Descripción:	Dibuja una línea horizontal.
Nombre:	MayorPicturebox()
Descripción:	Devuelve la posición del picturbox que dibuja la mayor cantidad de curvas.
Nombre:	CurvaProfundidad()
Descripción:	Devuelve una lista con los valores de la profundidad.
Nombre:	PistaProfundidad()
Descripción:	Crea el picturebox donde se va a representar la profundidad.
Nombre:	DibujarParte1Pista(Graphics grafic,PictureBox picturebox,int numeroPista)
Descripción:	Dibuja el nombre de la pista en el picturebox(**Parte1**).
Nombre:	DibujaEdicionCurva(Graphics grafic, PictureBox picturebox, CC_Pista pista)
Descripción:	Dibuja la forma en que se va a mostrar las curvas en la pista(**Parte2**).
Nombre:	GraficandoValoresCurva(Graphics grafic, PictureBox picturebox, CC_Pista pista)
Descripción:	Representa en el picturebox los valores de la curva(**Parte3**).
Nombre:	VisualizarRegistroPozo()
Descripción:	Une las tres partes del gráfico.
Nombre:	CI_VisualizarRegistro_Load(object sender, EventArgs e)
Descripción:	Visualiza la representación al abrise la interfaz.
Nombre: CI_GuardarPista	
Tipo de clase: Interfaz	
Atributo	**Tipo**
comboBox1	ComboBox
label1	Label
label2	Label
button1	Button
label3	Label
button2	Button
saveFileDialog1	SaveFileDialog
Para cada responsabilidad:	
Nombre:	GetImageFormat()
Descripción:	Devuelve el formato en que se va a guardar la pista.
Nombre:	PictureBoxUnionImagenes(PictureBox profundidad,PictureBox pista)
Descripción:	Devuelve un picturebox en el que se dibuja la imagen de la pista de profundidad con la pista a salvar.
Nombre:	CI_GuardarPista_Load(object sender, EventArgs e)
Descripción:	Muestra en un combobox las pistas creadas en la edición.
Nombre:	button1_Click(object sender, EventArgs e)
Descripción:	Guarda la representación elegida por el usuario.
Nombre:	button2_Click(object sender, EventArgs e)
Descripción:	Cierra la interfaz CI_GuardarPista.
Nombre:	CI_GuardarPista(Interfaz cc_graf, CImpExpRegistrosPozo control)

Descripción:	Constructor de la clase.
Nombre: CI_EliminarPista	
Tipo de clase: Interfaz	
Atributo	**Tipo**
listView1	ListView
button1	Button
columnHeader1	ColumnHeader
columnHeader2	ColumnHeader
columnHeader3	ColumnHeader
label1	Label
textBox1	TextBox
label2	Label
label3	Label
button2	Button
Para cada responsabilidad:	
Nombre:	CI_EliminarPista(Interfaz cc_graf)
Descripción:	Constructor de la clase.
Nombre:	button1_Click(object sender, EventArgs e)
Descripción:	Elimina la pista elegida por el usuario.
Nombre:	ActualizarListview()
Descripción:	Representa en el listview las pistas editadas.
Nombre:	CI_EliminarPista_Load(object sender, EventArgs e)
Descripción:	Actualiza el listview al abrise la interfaz CI_EliminarPista.
Nombre:	button2_Click(object sender, EventArgs e)
Descripción:	Cierra la interfaz CI_EliminarPista.
Nombre: CI_EditarRepresentacion	
Tipo de clase: Interfaz	
Atributo	**Tipo**
button5	Button
label16	Label
button6	Button
label15	Label
button7	Button
label14	Label
button8	Button
label13	Label
checkBox5	CheckBox
checkBox6	CheckBox
checkBox3	CheckBox
checkBox4	CheckBox
checkBox2	CheckBox
checkBox1	CheckBox
textBox7	TextBox
comboBox11	ComboBox
comboBox12	ComboBox
comboBox13	ComboBox
comboBox14	ComboBox
comboBox15	ComboBox
textBox4	TextBox
comboBox6	ComboBox

comboBox7	ComboBox
comboBox8	ComboBox
comboBox9	ComboBox
comboBox10	ComboBox
textBox3	TextBox
comboBox5	ComboBox
comboBox4	ComboBox
comboBox3	ComboBox
comboBox2	ComboBox
comboBox1	ComboBox
label10	Label
label9	Label
label8	Label
label7	Label
label6	Label
label5	Label
label2	Label
label1	Label
Para cada responsabilidad:	
Nombre:	CI_EditarRepresentacion(Interfaz graf, CImpExpRegistrosPozo control)
Descripción:	Constructor de la clase.
Nombre:	DibujarMuestra()
Descripción:	Dibuja en los texbox ”Muestra”la forma en que será representada la curva.
Nombre:	GenerarControles()
Descripción:	Crea controles dinámicos para editar los datos de las pistas.
Nombre:	button4_Click(object sender, EventArgs e)
Descripción:	Cierra la interfaz CI_EditarRepresentación.
Nombre:	CI_EditarRepresentacion_MouseMove(object sender, MouseEventArgs e)
Descripción:	Dibuja la muestra al mover el mouse sobre la forma.
Nombre:	button8_Click(object sender, EventArgs e)
Descripción:	Crea los controles dinámicos.
Nombre:	button7_Click(object sender, EventArgs e)
Descripción:	Elimina la última representación editada por el usuario.
Nombre:	button6_Click(object sender, EventArgs e)
Descripción:	Almacena los datos editados por el usuario.
Nombre:	button5_Click(object sender, EventArgs e)
Descripción:	Cierra la interfaz CI_EditarRepresentación.
Nombre: CC_Interfaz	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	void EliminarPista(int numPista);
Descripción:	Elimina una pista.
Nombre:	void AdicionarPista(int numP,bool gri,string nomCurv, string m_penC,bool log, double gros, string estiloL, CImpExpRegistrosPozo controladora);
Descripción:	Adiciona una pista.
Nombre:	void EliminarListaPista();
Descripción:	Elimina la lista de pistas.
Nombre:	ListaSE<CC_Pista> GetListaPistas();
Descripción:	Devuelve la lista de pistas editadas por el usuario.

Nombre: CC_Graficar	
Tipo de clase: Controladora	
Atributo	**Tipo**
listaPistas	ListaSE<CC_Pista>
Para cada responsabilidad:	
Nombre:	CC_Graficar()
Descripción:	Constructor de la clase.
Nombre:	GetListaPistas()
Descripción:	Devuelve el listado de las pistas editadas.
Nombre:	AdicionarPista(int numP,bool gri,string nomCurv, string m_penC,bool log, double gros, string estiloL, CImpExpRegistrosPozo controladora)
Descripción:	Adiciona una pista a la lista de pistas de la clase.
Nombre:	EliminarPista(int numPista)
Descripción:	Elimina una pista de la lista de pistas.
Nombre:	EliminarListaPista()
Descripción:	Elimina la lista de pistas editadas.
Nombre:	BuscarPista(int numPist)
Descripción:	Devuelve la pista.
Nombre: CC_Pista	
Tipo de clase: Controladora	
Atributo	**Tipo**
numeroPista	int
grid	bool
listaCurvas	ListaSE<CC_Curva>
Para cada responsabilidad:	
Nombre:	CC_Pista(int numP,bool gri)
Descripción:	Constructor de la clase.
Nombre:	GetNumeroPista()
Descripción:	Devuelve el número de la pista.
Nombre:	GetGrid()
Descripción:	Devuelve sila pista tiene o no grid.
Nombre:	GetListaCurva()
Descripción:	Devuelve el listado de las curvas correspondiente a la pista.
Nombre:	SetNumeroPista(int numP)
Descripción:	Cambia el número de la pista.
Nombre:	SetGrid(bool gri)
Descripción:	Cambia la disponibilidad del grid.
Nombre:	AdicionarCurvaPista(string nomCurv, string m_penC,bool log, double gros, string estiloL, CImpExpRegistrosPozo controladora)
Descripción:	Adiciona una curva a la pista.
Nombre:	EliminarCurvaPista(CC_Curva curva)
Descripción:	Elimina la curva correspondiente de la pista.
Nombre: CC_Curva	
Tipo de clase: Controladora	
Atributo	**Tipo**
nombreCurva	string
m_penColor	string
logaritmica	bool
grosor	double

estiloLinea	string
unidadMedida	string
valores	ListaSE<float>
Para cada responsabilidad:	
Nombre:	CC_Curva(string nomCurv, string m_penC,bool log, double gros, string estiloL, string unidadM, ListaSE<float> vals)
Descripción:	Constructor de la clase.
Nombre:	GetNombreCurva()
Descripción:	Devuelve el nombre de la curva.
Nombre:	GetM_penColor()
Descripción:	Devuelve el color de la curva.
Nombre:	GetLogaritmica()
Descripción:	Devuelve si la curva es logarítmica o no.
Nombre:	GetEstiloLinea()
Descripción:	Devuleve el estilo de la línea.
Nombre:	GetUnidadMedida()
Descripción:	Devuleve la unidad de medida.
Nombre:	GetListaValores()
Descripción:	Devulve la lista de valores de la curva.
Nombre:	SetNombreCurva(string nombreC)
Descripción:	Cambia el nombre de la curva.
Nombre:	SetM_PenColor(string m_penC)
Descripción:	Cambia el color de la curva.
Nombre:	SetLogaritmica(bool log)
Descripción:	Cambia la forma en que se va a representar la curva.
Nombre:	SetGrosor(double gros)
Descripción:	Cambia el grosor de la curva.
Nombre:	SetEstiloLinea(string estiloL)
Descripción:	Cambia el estilo de línea de la curva.
Nombre:	MenorValor()
Descripción:	Devuelve el menor valor de la lista de valores de la curva.
Nombre:	MayorValor()
Descripción:	Devuelve el mayor valor de la lista de valores de la curva.

Tabla 3. Descripción de las clases.

3.6.3 Diagramas de Interacción del Diseño (Secuencia).

Los diagramas de interacción incluyen la interacción de los mensajes entre los objetos que se definen en el modelo conceptual y otras clases de objetos [47].

Un diagrama de secuencia muestra una interacción que está organizada como una secuencia temporal. En particular, muestra los objetos que participan en la interacción mediante sus líneas de vida y mediante los mensajes que intercambian, organizados en forma de una secuencia temporal. [56].

Diagrama de secuencia del diseño “Editar Representación de Registro de pozo”.

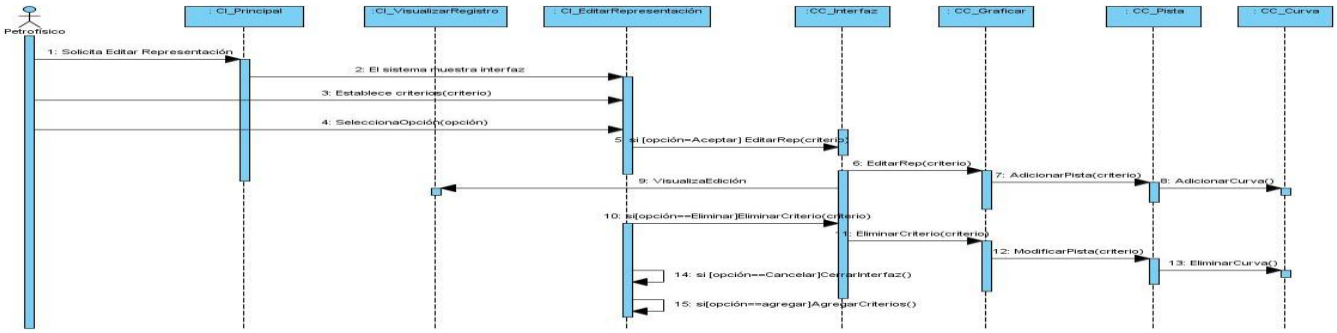

Figura 11. Diagrama de secuencia del diseño "Editar Representación de Registro de pozo".

Diagrama de secuencia del diseño"Adicionar Pista"

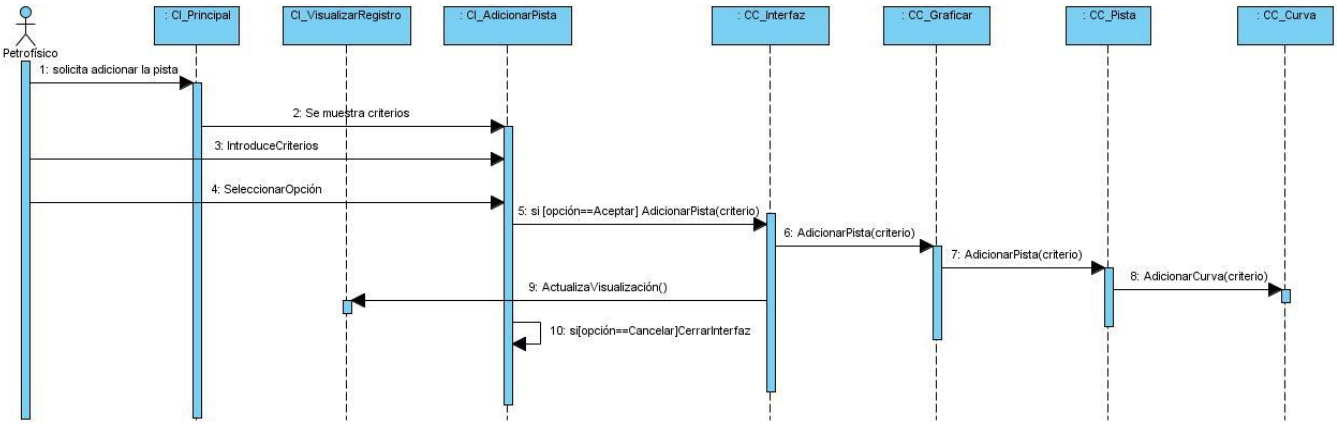

Figura 12. Diagrama de secuencia del diseño"Adicionar Pista".

Diagrama de secuencia del diseño"Eliminar Pista"

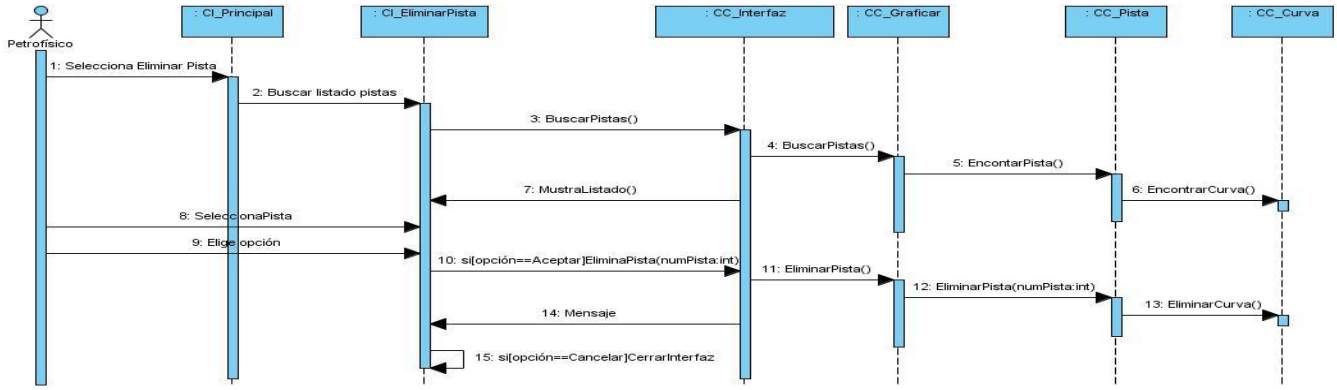

Figura 13. Diagrama de secuencia del diseño"Eliminar Pista".

Diagrama de secuencia del diseño"Guardar Pista"

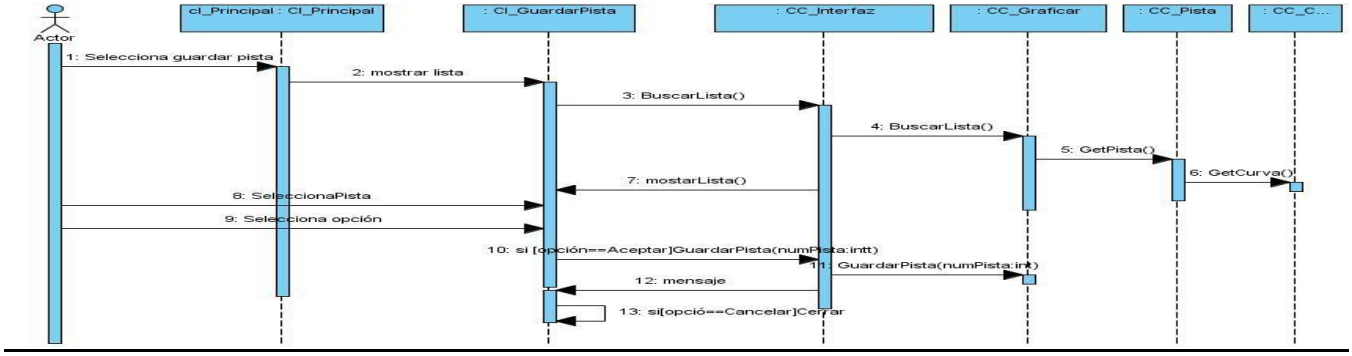

Figura 14. Diagrama de secuencia del diseño"Guardar Pista".

3.7 Implementación

En la implementación empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

3.7.1 Modelo de Implementación

El Modelo de Implementación describe como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El Modelo de Implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración y modelación disponible en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes uno de otros.

3.7.2 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. [37]. Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos. Los componentes representados pueden ser datos, archivos, ejecutables, código fuente y directorios. A continuación se muestra el correspondiente al sistema desarrollado.

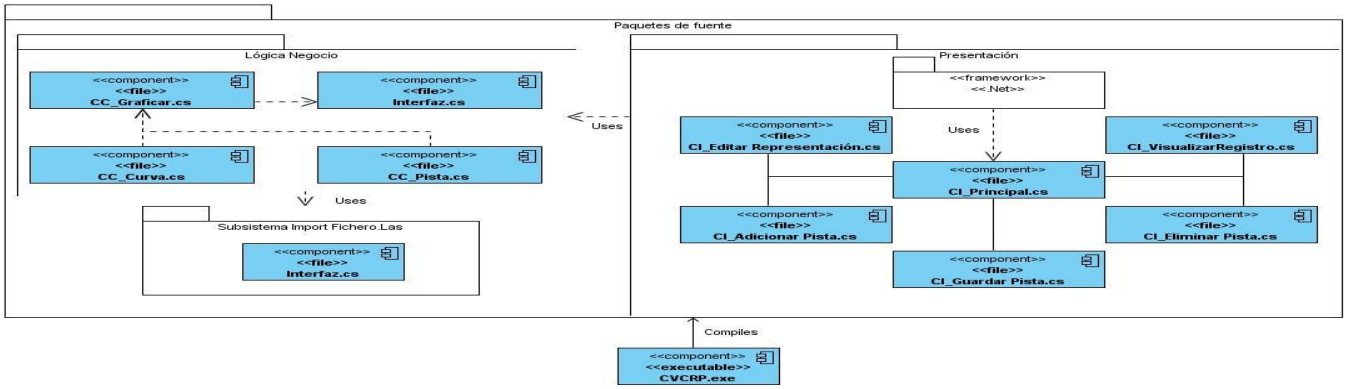

Figura 15. Diagrama de componentes.

3.8 Prueba

Las pruebas es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.[57]. Las pruebas son técnicas de validación predominantes o procesos de ejecución de un programa con la intención de descubrir errores de una aplicación que antes no se habían descubierto.

3.8.1 Prueba de Caja Negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. [57]

3.8.1.1. Casos de Prueba.

Un caso de prueba permite detallar la forma en que se va a probar el sistema, incluyendo los datos de entrada con las que se realizará la prueba correspondiente, las condiciones de ejecución y resultados obtenidos. [57]

Deben verificar:

- Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.

- Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

3.8.1.1.1 Caso de prueba para el caso de uso del sistema "Editar la Representación del Registro de Pozo".

Entrada	Resultados	Condiciones
El usuario accede a la opción "Editar Representación" .	El sistema notifica al usuario que es necesario que importe el registro de pozo. Ejemplo: "Usted debe importar el registro de pozo".	No se importe el registro de pozo a visualizar.
El usuario accede a la opción "Editar Representación" y se muestra un formulario con los datos requeridos en la edición. Este selecciona la opción "Aceptar" dejando algunos campos vacíos, o simplemente dejándolos todos vacíos.	El sistema notifica al usuario que es necesario que llene los datos solicitados. Ejemplo**:** "Usted debe llenar todos los datos".	No se registren valores vacíos en los campos de entrada de datos.
El usuario accede a la opción "Editar Representación" y se muestra un formulario con los datos requeridos en la edición. El usuario selecciona la opción "Agregar".	El sistema muestra una nueva edición.	No debe haber excedido el límite de 20 pistas.
El usuario accede a la opción "Editar Representación" y se muestra un formulario con los datos requeridos en la edición. El usuario selecciona la opción "Eliminar".	El sistema muestra una ventana de advertencia para eliminar la pista, desde la cual el usuario puede "Aceptar" o "Cancelar".	Debe haber más de una edición.
El usuario accede a la opción "Editar Representación" y se muestra un formulario con los datos requeridos en la edición. El usuario selecciona la opción "Cancelar".	El sistema procede a cancelar la edición y cierra la opción "Editar Representación".	En cualquier condición.

El usuario accede a la opción “Editar Representación” y se muestra un formulario con los datos requeridos en la edición. Este introduce los siguientes datos: Pistas: “1” Curva: “THOR” Log“” Grid:“” Color: “Azul” Estilo: “Solid” Grosor:“1” Pistas: “2” Curva:“POTA” Log: “” Grid:“” Color: “Rojo” Estilo:“Dash” Grosor: “2” Pistas: “3” Curva: “SGR” Log:“” Grid: “x” Color: “Amarillo” Estilo: “Custom” Grosor: “3” Este selecciona la opción “Aceptar”.	El sistema representa en el campo “Muestra” la forma en que será mostrada la curva. El sistema procede a visualizar la edición.	No se registren valores vacíos en los campos de entrada de datos.

Tabla 4. Caso de prueba para el caso de uso del sistema: “Editar la Representación del Registro de Pozo”.

3.8.1.1.2 Caso de prueba para el caso de uso del sistema“Gestionar Pista”.

Sección “Adicionar Pista”.		
Entrada	**Resultados**	**Condiciones**
El usuario accede a la opción “Adicionar Pista”**.**	El sistema notifica al usuario que es necesario que importe el registro de pozo. Ejemplo: “Usted debe importar el registro de pozo”.	No se importe el registro de pozo a visualizar.

El usuario accede a la opción “Adicionar Pista” y se muestra un formulario con los datos requeridos en la edición. Este selecciona la opción “Adicionar pista” dejando algunos campos vacíos, o simplemente dejándolos todos vacíos.	El sistema notifica al usuario que es necesario que llene los datos solicitados. Ejemplo**:** “Usted debe llenar todos los datos”.	No se registren valores vacíos en los campos de entrada de datos.
El usuario accede a la opción “Adicionar Pista” y se muestra un formulario con los datos requeridos en la edición. Este introduce los siguientes datos: Marcar nueva pista: “1” Curva: “THOR” Log: “” Grid:“x” Color: “Azul” Estilo: “Solid” Grosor: “1” El usuario selecciona la opción “Adicionar pista”.	El sistema representa en el campo “Muestra” la forma en que será mostrada la curva. El sistema procede a visualizar la edición.	No se registren valores vacíos en los campos de entrada de datos.
El usuario accede a la opción “Adicionar Pista” y se muestra un formulario con los datos requeridos en la edición. El usuario selecciona la opcion“Agregar curvas”.	El sistema muestra una nueva edición.	No debe haber excedido el límite de 5 pistas.
El usuario accede a la opción “Adicionar Pista” y se muestra un formulario con los datos requeridos en la edición. El usuario selecciona la opción “Eliminar curvas”.	El sistema muestra una ventana de advertencia para eliminar la pista, desde la cual el usuario puede “Aceptar” o “Cancelar”.	Debe haber más de una edición.

El usuario accede a la opción “Adicionar Pista” y se muestra un formulario con los datos requeridos en la edición. El usuario selecciona la opción “Cancelar”.	El sistema procede a cancelar la adición y cierra la opción “Adicionar Pista”.	En cualquier condición.

Sección “Eliminar Pista”.		
Entrada	**Resultados**	**Condiciones**
El usuario accede a la opción “Eliminar Pista”**.**	El sistema notifica al usuario que es necesario que importe el registro de pozo. Ejemplo: “Usted debe importar el registro de pozo”.	No se importe el registro de pozo a visualizar.
El usuario accede a la opción “Eliminar Pista” sin antes haber editado la representación o adicionado una pista.	El sistema notifica al usuario que es necesario que edite la representación de la pista. Ejemplo: "Usted debe editar la representación".	No exista una representación.
El usuario accede a la opción “Eliminar Pista” existiendo al menos una representación. Se muestra un listado con las pista editadas. El usuario selecciona la opción “Cancelar”.	El sistema procede a cancelar y cierra la opción “Eliminar Pista”.	En cualquier condición.
El usuario accede a la opción “Eliminar Pista” existiendo al menos una representación. Se muestra un listado con las pista editadas. El usuario selecciona la opción “Eliminar pista” dejando el campo “Pista a eliminar” vacío.	El sistema notifica al usuario que es necesario que llene el campo “Pista a eliminar” con el número de la pista a eliminar. Ejemplo: "Usted debe llenar el campo Pista a eliminar con el número de la pista".	No se edite el campo “Pista a eliminar”.

El usuario accede a la opción “Eliminar Pista” existiendo al menos una representación. Se muestra un listado con las pista editadas. El usuario introduce la pista en el campo “Pista a eliminar” y selecciona la opción ‘‘Eliminar pista”.	El sistema muestra una ventana de advertencia para eliminar la pista, desde la cual el usuario puede “Aceptar” o “Cancelar”.	Debe haber más de una edición.
Sección “Guardar Pista”.		
Entrada	**Resultados**	**Condiciones**
El usuario accede a la opción “Guardar Pista”**.**	El sistema notifica al usuario que es necesario que importe el registro de pozo. Ejemplo: “Usted debe importar el registro de pozo”.	No se importe el registro de pozo a visualizar.
El usuario accede a la opción “Guardar Pista” sin antes haber editado la representación o adicionado una pista.	El sistema notifica al usuario que es necesario que edite la representación. Ejemplo: "Usted debe editar la representación".	No exista una representación.
El usuario accede a la opción “Guardar Pista”. Se muestra un formulario con los datos necesarios para guardar una pista. El usuario selecciona la opción “Cancelar”.	El sistema procede a cancelar y cierra la opción “Guardar Pista”.	En cualquier condición.
El usuario accede a la opción “Guardar Pista”. Se muestra un formulario con los datos necesarios para guardar una pista. El usuario selecciona la opción “Guardar”.	El sistema notifica al usuario que es necesario que llene el campo “Pista a guardar” con el número de la pista a eliminar. Ejemplo: "Usted debe llenar el campo Pista a eliminar".	No se edite el campo “Pista a guardar”.

El usuario accede a la opción “Guardar Pista”. Se muestra un formulario con los datos necesarios para guardar una pista. El usuario introduce el número de la pista a guardar y selecciona la opción “Guardar”.	El sistema procede a guardar la pista elegida por el usuario. El sistema muestra una notificación de que la pista ha sido guardada satisfactoriamente. Ejemplo: "Pista guardada satisfactoriamente.".	Se marca la pista a guardar.

Tabla 5. Caso de prueba para el caso de uso del sistema: “Gestionar Pista”.

3.9 Conclusiones.

Luego del desarrollo de las actividades definidas para este capítulo de la tesis, haberse definido los requerimientos, diseño, implementación y prueba se arribó a las siguientes conclusiones:

- Se logró trasformar los requerimientos en un diseño que propició la implementación del componente.
- Con la implementación de la solución propuesta se obtuvo un componente para visualizar curvas de registros de pozos, como medio de apoyo a las actividades de análisis petrofísico que en el CEINPET se desarrolla.
- Se desarrolló un componente que permite una navegación sencilla a través de diferentes secciones de trabajo, con una interfaz cómoda, amigable, de fácil entendimiento y sobre todo de facilidad de uso por los usuarios.
- La prueba realizada demostró que el componente responde aceptadamente a la interacción del usuario.

Conclusiones Generales

Con el desarrollo del componente para visualizar curvas de registro de pozo, guiándose por el Proceso Unificado de Desarrollo de Software (RUP) y luego de la realización de las tareas de investigación científica, se ha arribado a las siguientes conclusiones:

- Las aplicaciones informáticas encargadas de llevar el proceso de visualización de registros de pozos son desarrolladas principalmente por países capitalistas desarrollados, y los países del tercer mundo, para acceder a estas tecnologías, deben pagar altos precios.

- Las herramientas, metodología de desarrollo y lenguajes utilizados, brindaron el soporte necesario para lograr un producto con los requerimientos deseados, además de proporcionarle al mismo una calidad y rendimiento acordes a las exigencias planteadas por el cliente.

- El componente para visualizar curvas de registros de pozos se encuentra listo para ser integrado al Sistema de Análisis de Registros de Pozos Petroleros actualmente desarrollado por el equipo de desarrollo del proyecto Petrofísica.

Recomendaciones

A partir de los resultados o beneficios que proporciona este trabajo de diploma, se proponen las siguientes recomendaciones:

- Que este material sea tomado como material de consulta por el personal (dígase técnicos o profesionales) que vayan a desarrollar un sistema similar.
- Continuar desarrollando el sistema con el objetivo de hacer su uso más flexible, posibilitando su aplicación en otras instituciones del país.
- Implementar el componente de software en otros lenguajes de programación que sean multiplataforma.
- Migrar el componente a software libre.

Bibliografía Referenciada

[1] 2008. Visitado en el Sitio El Boomeran (g) (blog literario en español.). [Disponible en: http://www.elboomeran.com/blog-post/18/3074/andres-ortega/schlumberger/] [Consultado el día 25 de enero del 2010].

[2]Reinoso, Yordamy. Diseño e Implementación de Componentes Reutilizables para la capa de Interfaz del Proyecto Telebanca 2.0. Trabajo de titulación (Ingeniero en Ciencias Informáticas). Ciudad Habana, Universidad de las Ciencias Informáticas (UCI), 2007. 4p.

[3]Clemens Szyperski. Component Software Component Software Beyond Object – Oriented Programming, 1998. [Citado el: 26 de enero de 2010.]

[4]Landa, Andrés. Diseño de un algoritmo para la visualización y procesamiento automatizado de registros de pozo. Trabajo de titulación (Ingeniero Geofísico). Sartenejas, Universidad de Simón Bolívar, 2004. 1p.

[5] 2008. Visitado en el Sitio InfoVis.Net (Revista.). [Disponible en: http://www.infovis.net/printMag.php?num=157&lang=1] [Consultado el día 25 de enero del 2010].

[6] La Coctelera. [En línea] 07 de 06 de 2008. [Citado el: 17 de 03 de 2010.] http://uvaq.espacioblog.com/post/2008/06/07/estadistica-mkt-1.

[7] http://msdn.microsoft.com/es-es/library/ms159640.aspx [Consultado el día 15 de enero del 2010].

[8] http://www.geotem.com.mx/contenido/metodos/m_reg_geo_p.html [Consultado el día 20 de enero del 2010].

[9] Neira, Yusleidy. Análisis y Diseño del Sistema para el análisis petrofísico. Trabajo de titulación (Ingeniero en Ciencias Informáticas). Ciudad Habana, Universidad de las Ciencias Informáticas (UCI), 2010. 198p.

[10] Neira, Yusleidy. Análisis y Diseño del Sistema para el análisis petrofísico. Trabajo de titulación (Ingeniero en Ciencias Informáticas). Ciudad Habana, Universidad de las Ciencias Informáticas (UCI), 2010. 198p.

[12] http://www.addlink.es/productos.asp?pid=430 [Consultado el día 16 de noviembre del 2009].

[13] http://www.neuralog.com/www_backup_files/espanol/log.htm[Consultado el día 16 de noviembre del 2009].

[14] http://www.adams-consulting.net/Neuralog_productos.htm [Consultado el día 16 de noviembre del 2009].

[15]Vega,Francisco. Diseño de la arquitectura de software para el proyecto de Petrofísica. Trabajo de titulación (Ingeniero en Ciencias Informáticas). Ciudad Habana, Universidad de las Ciencias Informáticas (UCI),2010.

[16] Hard-h2o.com. 2007. Diccionario Informático. [En línea] 2007. http://www.hard-h2o.com/diccionario-informatico.html.

[17]Abox.com [En línea] http://www.abox.com/productos.asp?pid=314 [Consultado el día 23 de febrero del 2010]

[18] http://msdn.microsoft.com/es-es/library/h1ek3akf(VS.80).aspx [Consultado el día 28 de febrero del 2010].

[19] http://blogs.finanzas.com/blogfiles/bat23/83770_ibe01072008.gif [Consultado el día 29 de febrero del 2010].

[20] http://200.23.8.83/manual_iris4/Integral%20Basico/grafica%20de%20pastel%20dimension.jpg [Consultado el día 29 de febrero del 2010].

[21] http://www.omnico.com.mx/GraficaLight1.JPG [Consultado el día 29 de febrero del 2010].

[22] 2005. Visitado en el Sitio HDS (Housing and Development Software). [Disponible en: http://www.hdsoftware.com/about_us.htm] [Consultado el día 10 de marzo del 2010].

[23] 2009. Visitado en el Sitio Schlumberger. [Disponible en: http://www.slb.com/content/services/software/geo/geoframe/petrophysics/index.asp?] [Consultado el 2 de marzo del 2010].

[24] Petrophysics, I., User Manual. 2007, Schlumberger, PGL-Senergy.

[25]Dr.Pere Graells,2000. Las TIC y sus aportaciones a la sociedad. [Disponible en: http://peremarques.pangea.org/tic.htm] [Consultado el día 10 de marzo del 2010].

[26] “Material_para_la_Conf_1.doc”. Visitado en el Sitio Teleformación.uci.cu. [Disponible en: http://eva.uci.cu/mod/resource/view.php?id=11402] . [Consultado el 26 de febrero del 2010].

[27] Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.2008.

Visitado en el Sitio Adonisnet's Weblog. [Disponible en: http://adonisnet.files.wordpress.com/2008/06/articulo-metodologia-de-sw-formato.doc. [Consultado el 24 de febrero del 2010].

[28] 2004. Visitado en el Sitio UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA FACULTAD DE INGENIERIA INGENIERO EN COMPUTACION (SERVIDOR YAQUI). [Disponible en: http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm]. [Consultado el 25 de febrero del 2010].

[29] Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 2004. El Proceso Unificado de Desarrollo de Software. s.l. : Addison Wesley, 2004. ISBN 84-7829-036-2.

[30] Rodríguez, Yudiel. INVENTARIO NACIONAL DE RECURSOS Y RESERVAS DE PETRÓLEO Y GAS. Trabajo de titulación (Ingeniero en Ciencias Informáticas). Ciudad Habana, Universidad de las Ciencias Informáticas (UCI), 2008.

[31] Angel Alvarez, Miguel and López, Daniel, Gutierrez,Manu. 2007. PHP 5: Programacion Orientada a Objeto. 2007.

[32]Robinson, S., Cornes, O., et al. Professional C#. Wrox Press Ltd. 2001. [Citado el: 10 de marzo de 2010.]

[33] URRIELLUnet. [En línea] [Citado el: 5 de Noviembre de 2008.] Disponible en: http://urriellu.net/es/articles-software/csharp-advantages.html

[34] 2007. Visitado en el Sitio Free Download Manager (Grupo Soluciones). [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/]. [Consultado el 25 de febrero del 2010].

[35] Visual Paradigm Design Group. 2009. Visual Paradigm. [Online] Noviembre 12, 2009. [Cited: 01 18, 2010.] http://www.visual-paradigm.com/.

[36] 2007. Visitado en el Sitio Universidad de Jaén (Departamento de Informática). [Disponible en: http://wwwdi.ujaen.es/asignaturas/isg/Software.html] . [Consultado el 25 de febrero del 2010].

[37] http://msdn.microsoft.com/es-es/library/zw4w595w.aspx [Consultado el 20 de febrero del 2010].

[38] Vitter,D; Templeman J. Visual Studio .NET, 2002. [Citado el: 10 de diciembre del 2010].

[39]. Powers, R; Snell, M. Microsoft Visual Studio 2008 UNLEASHED, 2008. [Citado el: 12 de Enero de 2010].

[40] "Material_para_la_Conf_1.doc". Visitado en el Sitio Teleformación.uci.cu.[Disponible en: http://teleformacion.uci.cu/mod/resource/view.php?id=11402]. [Consultado el 26 de febrero del 2010].

[41] Danis Rego Castillo, Alfonso Chaveco Laurencio. "DESARROLLO DE UNA INTRANET PARA EL GRUPO EMPRESARIAL DE TRANSPORTE POR OMNIBUS". Universidad de las Ciencias Informáticas (UCI). Ciudad de La Habana, Junio 26 de 2007. 91.

[42]Zulma Cataldi, Fernando J. Lage Facultad Regional Buenos Aires. Sistemas Tutores Inteligentes: Procedimientos, métodos, técnicas y herramientas para su creación. [Disponible en: http://www.virtualeduca.info/ponencias/558/Entorno%20STI%20Frame%2003-09-09%20eje%205.doc].

[43]2008. solvingsoft.com . Visitado en el sitio[http://www.solvingsoft.com/wordpress] [Consultado el día 10 de marzo del 2010].

[44]http://upload.wikimedia.org/wikipedia/commons/4/4d/Rup_espanol.gif

[45] Ingeniería de Software 1. Conferencia #4. Fase de Inicio. Flujo de trabajo de requerimientos. Modelo de Diseño. Curso 2008-2009. 27.

[46] Filiberto López, Luis Javier. " Implementación de un componente de software para la visualización tridimensional de Neuroimágenes". Universidad de las Ciencias Informáticas (UCI). Ciudad de La Habana, Junio de 2009. 30.

[47]. UCI, Ingeniería de Software II. Material de apoyo.Conferencia de Diseño. 2008. [Disponible en:] http://teleformacion.uci.cu/mod/resource/view.php?id=21363.

[48] Perry, D.E. and A.L. Wolf, Foundations for the Study of Software Architecture. Software Engineering Notes, 1992. Vol. 17 no. 4.

[49]Reynoso, C. and N. Kiccillof (2004) Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.

[50]Ing. Yoan Arlet Carrascoso Puebla e Ing. Enrique Chaviano Gómez Propuesta de arquitectura orientada a servicios para el módulo de inventario del ERP cubano. Disponible en: http://www.gestiopolis.com/administracion-estrategia/erp-arquitectura-orientada-a-servicios.htm

[51]Diseñando aplicaciones distribuidas. Disponible en: http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml. [Consultado el día 15 de marzo del 2010].

[52] Luz María Hernández Rodríguez. UML y Patrones. Introducción al análisis y diseño orientado a objetos. México. Cámara Nacional de la Industria Editorial Mexicana. 1999.499.

[53] Charla 1: Introducción a los patrones de diseño. Algunos patrones básicos. 18.

[54] Ingeniería de Software 2. Conferencia #1. Continuación del FT Análisis y Diseño. Modelo de Diseño. Curso 2008-2009. 4.

[55] Edgar Cabrera Báez. Reynier Pérez González. Universidad de las Ciencias Informáticas. Ciudad de la Habana. Julio del 2008. 119.

[56] Ingeniería de Software 1. Conferencia #7. Fase de Elaboración. Flujo de trabajo de Análisis y Diseño. Curso 2008-2009. 18.

[57] Ingienería de software 2. Conferencia #4. Fase de Elaboración. Flujo de trabajo Pruebas. [Disponible en http://eva.uci.cu/mod/resource/view.php?id=2241&subdir=/Conferencias_IS2_05-06]. Curso 2005-2006.

[58] Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.2008. Visitado en el Sitio Adonisnet's Weblog. [Disponible en: http://adonisnet.files.wordpress.com/2008/06/articulo-metodologia-de-sw-formato.doc. [Consultado el 25 de febrero del 2010].

[59] 12 de noviembre del 2007. Visitado en el Sitio blogspot.es. [Disponible en: http://eproano334.blogspot.es/]. [Consultado el 24 de abril del 2010].

[60] http://i.technet.microsoft.com/Cc526621.CMSU_graphics_Ops_CIR_MSF_Milestones_and_Phases(en-us,TechNet.10).gif.

Bibliografía Consultada

[1] Ing. Juan Gabardini, Ing. Lucas Campo. Balanceo de Metodologías Orientadas al Plan y Ágiles. Herramientas para la Selección y Adaptación. Buenos Aires. Argentina. 2004. 7.

[2] Martha D. Delgado Dapena. Definición del modelo del negocio y del dominio utilizando Razonamiento Basado en Casos. Centro de Estudios de Ingeniería de Sistemas. Cuba. 11.

Glosario de Términos

Petrofísica: La petrofísica se ocupa del estudio de las propiedades físicas de las rocas como reservorios de hidrocarburos a través de análisis de laboratorio en núcleos y de registros de pozos, es decir, es el estudio de las propiedades físicas y químicas que describen la incidencia y el comportamiento de las rocas, los sólidos y los fluidos.

Petróleo: Es una mezcla heterogénea de compuestos orgánicos, principalmente hidrocarburos insolubles en agua. También es conocido como petróleo crudo o simplemente crudo.

Casos de Uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.

Objeto: Un objeto es la representación de un concepto para un programa, y contiene toda la información necesaria para abstraer dicho concepto: los datos que describen su estado y las operaciones que pueden modificar dicho estado.

Clase: Una clase es un contenedor de uno o más datos (variables o propiedad miembro) junto a las operaciones de manipulación de dichos datos (funciones/métodos). Las clases pueden definirse como estructuras, uniones o clases pudiendo existir diferencias entre cada una de las definiciones según el lenguaje. Además las clases son agrupaciones de objetos que describen su comportamiento.

Herencia: En la programación orientada a objetos, la herencia es un mecanismo que permite derivar una clase de otra, de manera que extienda su funcionalidad.

Polimorfismo: En programación orientada a objetos el polimorfismo se refiere a la posibilidad de definir clases diferentes que tienen métodos o atributos denominados de forma idéntica, pero que se comportan de manera distinta.

Interfaz: Representa el uso de un tipo para describir el comportamiento visible externamente de cualquier elemento del modelo.

Anexos

Anexo 1 Gestionar Pista

Caso de Uso del sistema:	Gestionar Pista
Actores:	Petrofísico
Propósito	Realizar cambios en las propiedades de las pistas para realizar el estudio de la representación del registro de pozo.
Resumen:	El Caso de Uso se inicia cuando el Petrofísico solicita eliminar, adicionar, guardar una pista.
Precondiciones:	Debe existir una pista seleccionada.
Referencias	RF2.1,RF2.2,RF2.3
Prioridad	Crítico

Flujo normal de los eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Petrofísico solicita gestionar la pista.	2. El sistema le muestra las posibles formas para gestionar las pistas (adicionar, eliminar, guardar).
3. El Petrofísico selecciona la forma deseada.	4. El sistema Gestiona la selección: Si selecciona la opción Adicionar Pista ir a la **Sección Adicionar Pista**. Si selecciona la opción Eliminar Pista ir a la **Sección Eliminar Pista**. Si selecciona la opción Guardar Pista ir a la **Sección Guardar Pista**.

Prototipo de Interfaz "* Principal*"

Figura 16. Prototipo de interfaz "Principal".

Sección Adicionar Pista	
	1. El sistema muestra criterios de edición para la pista creada. El sistema muestra además las opciones Agregar curvas, Eliminar curvas, Adicionar pista, Cancelar.
2. El Petrofísico llena los campos requeridos para editar la representación de la pista y selecciona la opción deseada.	3. El sistema Gestiona la Selección Si selecciona la opción Adicionar pista el sistema procede a visualizar la representación editada. Si selecciona la opción Cancelar se cierra la ventana sin guardar los cambios efectuados. Si seleccciona la opción Agregar curvas el sistema procede a agregar una nueva curva a la pista. Si selecciona la opción Eliminar curvas el sistema procede a eliminar una curva de la pista.

Prototipo de Interfaz "Adicionar Pista"

Figura 17. Prototipo de Interfaz "Adicionar Pista"

Sección Eliminar Pista	
	1. El sistema muestra un listado con todas las pistas. El sistema muestra además los botones de Aceptar, Cancelar.
2. El Petrofísico selecciona la pista deseada y elige la opción a ejecutar.	3. El sistema Gestiona la Selección: Si selecciona la opción Aceptar se continúa con el paso 4 de la Sección Eliminar Pista. Si selecciona la opción Cancelar se cierra la ventana sin eliminarse la pista.
	4. El sistema elimina la pista seleccionada.
	5. El sistema actualiza la numeración de las pistas siguientes a la eliminada.
	6. El sistema muestra un mensaje confirmando que la pista ha sido eliminada.

Prototipo de Interfaz "Eliminar Pista"

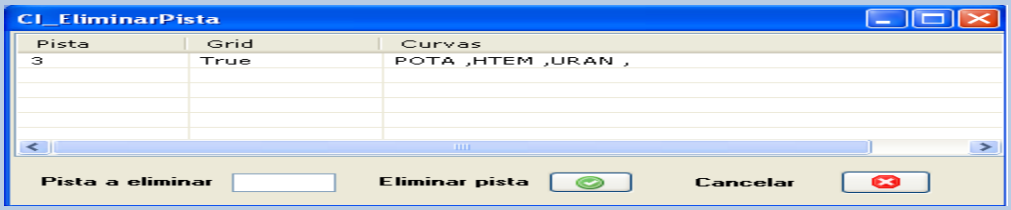

Figura 18. Prototipo de Interfaz "Eliminar Pista"

Prototipo de Interfaz "Advertencia para Eliminar Pista!"

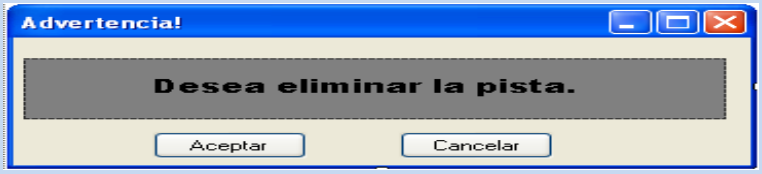

Figura 19. Prototipo de Interfaz Advertencia para Eliminar Pista.

Prototipo de Interfaz "Confirmación para Eliminar Pista!"

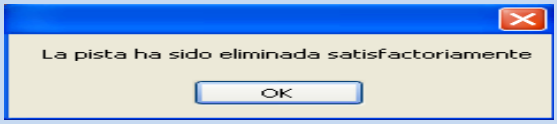

Figura 20. Prototipo de Interfaz Confirmación para Eliminar Pista.

Sección Guardar Pista	
	1. El sistema realiza una copia de la pista seleccionada.

Prototipo de Interfaz "Guardar Pista".

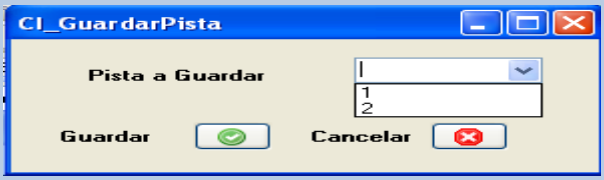

Figura 21. Prototipo de Interfaz "Guardar Pista".

Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
Poscondiciones	Se logra gestionar la pista

Tabla 6. Descripción del caso de uso del sistema Gestionar Pista.

Anexo 2 Encuesta Realizada a los trabajadores del CEINPET (Petrofísico).

1. ¿Con qué objetivo necesitan que les elaboremos un software que permirta visualizar la información contenida en los registros de pozos? Argumente.
2. ¿Qué problemas presenta actualmente el CEINPET, que motiven a la realización de un software que permirta visualizar la información contenida en los registros de pozos? Argumente.
3. ¿No cumple(n) todas las expectativas el(los) software que están usando?
4. ¿Es muy costoso pagar las licencias del (de los) software que están usando cuando se vencen?
5. ¿Para realizar el proceso de Visualización de Registros, necesita de un pedido del CEINPET, de CUPET, o de alguna persona en específico?
6. ¿Qué funcionalidades necesita que tenga el sistema para visualizar curvas de registro de pozo? Enumerarlas.
7. ¿Cómo se realiza el trabajo de Visualizar Registros?
8. ¿Cuál es el orden de las acciones para la Visualización de Registros?
9. ¿Resulta disponible la información necesaria para el proyecto?