Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas.

Universidad de las Ciencias Informáticas
FACULTAD 9.

Graficador de Indicadores para el área de Perforación de la Industria Petrolera. (GIPP)

Autor: Noel Pérez Bello.

Tutor: Ing. Nilberto C. Chavez Marquez.

CO-Tutor: Ing. David Tavares Cuevas.

Ciudad de La Habana, Junio 2010.

“No se vive celebrando victorias, sino superando derrotas.”

Ernesto Che Guevara

A mis padres, porque hoy también se hacen realidad sus sueños.

Todo el esfuerzo y las horas de empeño para lograr realizar este trabajo quiero dedicarlos a las personas que han sido y son la razón de mi ser… a mi mamá Magalis y mi papá Noel, mi hermano Yusniel, mi tío Guille, mis abuelos Paula y Francisco, mis suegros Sonia y Miguel, mi segundo papá Lázaro, a Isabel, A Soili por el inmenso cariño que nos une… y a todos los familiares que de una u otra forma contribuyeron con mi formación.

A mis amigos por todos los momentos que he compartido con cada uno de ellos.

Gracias a todos.

"La función de un buen software es hacer que lo complejo aparente ser simple".

Grady Booch.

Antes que todo a la persona que ha sido mi faro guía durante todos estos largos años, sin ella no sería posible ser lo que soy, mis más grandes agradecimientos, para ese ser que sacrifica todo sin esperar nada a cambio, mi madre.

A mi padre, viejo gracias por enseñarme a ser un hombre de bien, este resultado también es tuyo.

A esa personita que amo con toda la vida: Soili, gracias por tanto sacrificio, por estar a mi lado todo este tiempo y por soportarme, este logro es de los dos.

A mi hermano Yusni, por ser siempre esa mano derecha junto a la mía y ser un ejemplo a seguir a lo largo de mis estudios.

A mis dos tesoros, mima y abuelo gracias por apoyarme.

A Lázaro por ser padre y amigo e Isabel por tanta "energía positiva".

A Sonia y Miguel, suegros gracias por su ayuda, sus consejos y por los que aún faltan.

A mi tutor Nilberto, gracias por enseñarme a ser un profesional.

A mi cotutor David, gracias por más que un tutor ser un amigo.

A mi amiga Anisleydis por la ayuda con la ortografía, gracias.

A mis amigos y compañeros del proyecto, por el apoyo de cada uno de vosotros.

A mis profes, por haberme enseñado a aprender. Muchos podrán ser los conocimientos olvidados después de los años pero la metodología del aprendizaje siempre quedará, y eso se lo debo a Uds.

A los miembros del tribunal por las críticas oportunas y los sabios consejos, al profesor Febe y Daniel por prestarme su ayuda, gracias.

A mis familiares a los que nunca defraudaré jamás.

A todas las personas que me han brindado su apoyo y han creído en mí durante todos estos largos años, que han influido de una forma u otra en mi crecimiento como persona y como profesional, de corazón un millón de gracias.

Al Comandante en Jefe, la Revolución y a la Universidad de las Ciencias Informáticas por darme la oportunidad de formarme como profesional.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 9 de la Universidad de las Ciencias Informáticas, así como a dicho centro a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2010.

_____	_____
Firma del Autor	**Firma del Tutor**
Noel Pérez Bello	Ing. Nilberto C. Chavez Marquez

Resumen:

En la actualidad existe un número de software que se utilizan para graficar datos, estos no pueden ser utilizados con los propósitos para los cuales se construyeron, pues se necesita de licencias, por lo que su utilización se ve limitada a las funcionalidades específicas para las cuales fueron creados.

El incremento que presenta el precio del petróleo junto a las cada vez más estrictas normas de seguridad medio ambiental, trae consigo la necesidad de desarrollar e implementar nuevas tecnologías en el campo de la perforación.

En la Universidad de Las Ciencias Informáticas (UCI) existe el Polo de Soluciones Informáticas para La Industria Petrolera (PetroSoft) en el cual se hace necesario contar con un Graficador para el área de Perforación de la Industria Petrolera, para apoyar el desarrollo de software relacionado con dicha área.

El presente trabajo surge como una necesidad, no solo del Polo sino también de la Dirección de Intervención y Perforación de Pozos (DIPP) y Centro de Investigaciones de Petróleo (CEINPET), para solucionar situaciones problemáticas existentes en las mismas. Debido a esta situación se analizó y se llegó a la conclusión de crear un componente para graficar datos del comportamiento de los indicadores en los procesos que se llevan a cabo en la perforación de pozos petroleros. Con el fin de cumplimentar los objetivos propuestos se investigó acerca de las distintas gráficas que se utilizan para la visualización de información referente al área de perforación, así como las herramientas (software) que se utilizan actualmente para el modelado de los mismos.

Palabras claves

Perforación, graficar, visualización, software, componente, indicadores, comportamiento.

Índice

Índice de Figuras

Índice de Tablas.

Introducción

La mayor parte de los documentos que se manejan en la Industria Petrolera contienen textos y gráficos. También se puede ver en libros, folletos y carteles publicitarios, que con el uso combinado de textos y gráficos se obtienen buenos resultados para comunicar ideas.

Cuando se maneja un gráfico en informática, interesa el resultado final y también la facilidad de manejo. Con los distintos tipos de gráficos que existen, se pueden analizar detalladamente fenómenos que se presentan a diario, pero el trabajo que demandan y la manera de manejar cada uno hace que sea importante conocer las características de los distintos gráficos que ofrecen grandes posibilidades para la representación de datos y pueden ser utilizados en múltiples situaciones, incluso para representar los resultados obtenidos por métodos de análisis más complicados.

Hoy en día el software para la visualización gráfica de procesos ha experimentado un auge extraordinario debido al avance de las Tecnologías de la Información y las Comunicaciones (TIC). La creciente informatización de casi la totalidad de las empresas, ha provocado efectividad, rapidez, menos costos en la representación de procesos con grandes flujos de información, para satisfacer las necesidades a la hora de optimizar servicios y productos. Ante esta notable demanda de soluciones informáticas existe un conjunto de Software vinculados a esta área, estos son programas que permiten manejar la información de modo sencillo y presentan funcionalidades para la visualización del flujo de datos.

La Oficina Central de Exploración Producción de la Unión Cuba Petróleo (CUPET) se encuentra inmersa en un proceso de expansión de sus principales procesos productivos, las expectativas que se plantean sobre las posibilidades reales del país de la exploración, perforación y producción de petróleo en las aguas profundas del Golfo de México (ZEE), han propiciado un ambiente de trabajo favorable para la realización de convenios de trabajo entre la UCI y CUPET, debido a la imperiosa necesidad de informatizar sus principales procesos de producción, que elevaría grandemente la eficiencia y competitividad de esta empresa en el marco internacional.

Actualmente en el Polo Productivo: Soluciones Informáticas para la Industria Petrolera (PetroSoft) no existe un componente, que adaptándose a la arquitectura a utilizar facilite la interacción entre los programadores y los procesos necesarios para graficar el comportamiento de los indicadores en el área de perforación petrolera. Se han encontrado problemas en los proyectos del Polo en los cuales, al no tener en cuenta este principio, han sido diseñadas y

programadas diferentes vías para cumplir el mismo objetivo, y en algunos casos, “reinventar la rueda”, lo que provocó invertir tiempo en actividades que no lo requerían, complicando el desarrollo y lejos de lograr un alto grado de motivación en los desarrolladores, son motivos para mantenerlos alejados, lo que provoca, en reiteradas ocasiones, retrasos en los cronogramas de entrega.

La Dirección de Intervención y Perforación de Pozos (DIPP) radicada en La Empresa de Producción y Extracción de Petróleo del CENTRO (EPEPC), controla todas las operaciones y demás actividades que se realizan en los pozos en perforación e intervención como las que se llevan a cabo en el Centro de Investigaciones de Petróleo (CEINPET). Actualmente en estos centros el proceso para visualizar el flujo de información en muchos de los casos se realiza manualmente y la visualización de la misma se hace de forma semiautomática, utilizándose diferentes herramientas ofimáticas como son:

Microsoft Excel, Microsoft Word, Microsoft Power Point, además de documentos en pdf, dando lugar a la intervención directa del hombre en el manejo y visualización de todo un gran cúmulo de información.

Esta situación ha provocado que la visualización de la información existente contenga errores y con ello inconsistencia e inseguridad, lo que dificulta la toma de decisiones, ocasionando que los tiempos de respuesta a las demandas de información sean elevados; lo que trae consigo, fallas en los partes y reportes emitidos, difícil acceso a la información y pérdida de la misma.

Para llevar a cabo los trabajos en dicha área, la empresa cuenta con un conjunto de software propietarios que se encargan de interpretar y graficar los registros de pozos. Dichos software restringen su uso por medio de licencias, lo que provoca, unido a la falta de acceso a las actualizaciones por ser costosas para el centro, que el software pueda solo ser usado en una computadora a la vez. Esto a su vez trae consigo que se acumule y se torne engorroso el trabajo, además de que parte de sus investigadores se vean excluidos de la utilidad que brindan dichos software; los mismos poseen redundancia pues coinciden en funcionalidades por lo que se hace necesario a estos centros contar con un software que agrupe las funcionalidades indispensables para contar con un producto de fácil manejo que permita visualizar los procesos con mayor claridad y calidad que se adapte a los gustos y necesidades de sus usuarios.

Para automatizar todo el flujo de información relacionada con dicho centro surge el proyecto Sistema de Información de Perforación de Pozos Petroleros (SIPP), el mismo necesita graficar

datos que permitan mejorar la visualización de toda la información referente al comportamiento de indicadores que se recogen de los procesos petroleros en la perforación de pozos.

Esta investigación surge como necesidad de dar solución a las situaciones antes expuestas; por lo que el **problema** a resolver queda planteado de la siguiente forma:

La escasez de un graficador que represente el comportamiento de los indicadores en el área de perforación de la Industria Petrolera, impide que se agilice el proceso de análisis de la información de los pozos petroleros en perforación.

Para dar solución al problema se ha trazado como **objetivo general** de la investigación: Desarrollar un Componente que Grafique el Comportamiento de los Indicadores para el área de perforación en pozos petroleros de la Unión Cuba Petróleo (CUPET).

Para darle cumplimiento a dicho objetivo se planteó como **objeto de estudio** el proceso de graficación del comportamiento de los indicadores existentes en el área de perforación en pozos petroleros de la Unión Cuba Petróleo (CUPET) y como **campo de acción** la automatización del proceso de graficación en el área de perforación de la Industria Petrolera.

Como **idea a defender** para la realización de la investigación se toma como base la premisa de que si se logran identificar los comportamientos de los principales indicadores en el área de perforación de la Industria Petrolera, se podrá diseñar e implementar un componente que grafique estos indicadores.

Para cumplir con los objetivos trazados y resolver el problema planteado, se proponen las siguientes **tareas de investigación**:

1. Describir un estudio del estado del arte de los componentes desarrollados con este fin a nivel nacional e internacional.
2. Identificar las metodologías más utilizadas en el desarrollo de componentes, así como métodos y herramientas a utilizar.
3. Identificar los principales indicadores utilizados en el área de perforación de la Industria Petrolera.
4. Modelar el componente para graficar indicadores.
5. Implementar el componente para graficar indicadores.
6. Validar la solución propuesta mediante la realización pruebas de software.

El **resultado que se espera** de la investigación es, un componente para graficar indicadores del área de perforación para la Industria Petrolera.

Para la realización de las tareas de investigación se han empleado los **métodos de investigación** que se describen a continuación.

Métodos Teóricos: Permiten conocer las relaciones que fluyen alrededor del objeto de estudio.

Entre estos se empleó:

- ✓ **Analítico-Sintético**, con el objetivo de analizar y aumentar los conocimientos entorno al objeto de estudio a partir de consultar la bibliografía científica correspondiente, para después, haciendo uso de la síntesis, lograr resumir y exponer los resultados obtenidos del análisis.
- ✓ **Histórico-Lógico**, con el objetivo de identificar y verificar la existencia de sistemas que se encarguen de los procesos antes mencionados y entender cómo se realizan dichos procesos, comprender y dominar todo el entorno que rodea a dicha empresa y que interviene en el manejo de información sobre todo de forma gráfica.
- ✓ **Modelación**, para la caracterización de las funcionalidades que tendrá el componente cuando se están definiendo los requisitos funcionales.

Métodos Empíricos: Permiten la observación y el análisis inicial de la información.

Entre estos se empleó:

- ✓ **Observación**, con el objetivo de obtener un registro visual de toda la información referente a los procesos de perforación y el funcionamiento del centro en una situación real.

- ✓ **Entrevista**, al personal que se encarga del análisis en los sistemas de perforación para recopilar información confiable y real de las necesidades del centro, y que se tornan indispensables para la solución del problema de investigación; a los líderes de proyecto para obtener información sobre las herramientas de graficación que se utilizan en el Polo PetroSoft.

El contenido del presente trabajo de diploma está estructurado de la siguiente manera:

Capítulo 1. "Fundamentación Teórica": en este capítulo se analizan los principales aspectos relacionados con la graficación de los procesos de perforación en la Industria Petrolera. Se enuncian conceptos que posibilitan un mejor entendimiento a lo planteado en la situación problemática y el problema en general. Se hace referencia al comportamiento de los indicadores en el área de perforación y a los sistemas automatizados existentes vinculados al campo de acción, y se plantean los objetivos generales y específicos.

Capítulo 2. "Tendencias y tecnologías actuales a considerar**":** se describen las metodologías, lenguajes de programación y tecnologías a considerar para su posterior utilización en el desarrollo del componente, analizando sus ventajas y desventajas, características, estableciendo comparaciones y seleccionando propuestas con el fin de dar cumplimiento al objetivo general de la presente investigación con la mayor eficiencia y calidad posible.

Capítulo 3. "Presentación de la solución propuesta": se describen los procesos del negocio relacionados con el objeto de estudio. Se presentan los requisitos funcionales y no funcionales que debe tener el software y se muestra la concepción del sistema a partir del diagrama de casos de uso y las descripciones de los mismos.

Capítulo 4. "Construcción de la solución propuesta": se desarrolla el diseño de la propuesta partiendo de la explicación de los patrones de diseño que se utilizan para modelar el software, se propone la arquitectura para guiar la realización de los principales aspectos del sistema, se describe la fase de implementación y se presenta el diagrama de componentes, además de describir las pruebas exploratorias, funcionales y del sistema realizadas.

Capítulo 1. Fundamentación Teórica.

1.1 Introducción.

En este capítulo se abordarán aspectos fundamentales sobre la representación gráfica, constatando su utilidad en el proceso de análisis estadístico y la presentación de datos. Se describen los distintos tipos de gráficos que se pueden utilizar para reflejar el comportamiento de indicadores y su correspondencia con las etapas del proceso de Perforación Petrolera. Se aborda de forma detallada el objeto de estudio y la situación problemática y se analizan los principales software generadores de reportes gráficos, así como conceptos relacionados con los procesos en la industria petrolera.

1.2 Conceptos asociados al dominio del problema.

Para mejor entendimiento y desenvolvimiento de las áreas temáticas que se abordarán en el presente y posteriores capítulos, se mostrarán una serie de conceptos identificados durante la investigación realizada.

1.2.1 Componente.

Varios autores han definido los componentes de software, de acuerdo a sus necesidades específicas para diferentes contextos. Esto hace que se manejen una gran cantidad de definiciones.

Un componente es: (12)

- Según el SEI[1], una implementación opaca de funcionalidad, sujeta a composición por terceros y que cumple con un modelo de componentes.
- Para Michael Sparling[2], un paquete de servicios de software, implementados independientemente y con neutralidad de lenguaje, distribuidos en un contenedor

[1] SEI, Instituto de Ingeniería de Software de la Universidad Carnegie Mellon en Estados Unidos para desarrollar modelos de evaluación y mejora en el desarrollo de software.

[2] Michael Sparling es director de Tecnología en Castek, Canadá, una consultora en Ingeniería de Software.

reemplazable que los encapsula y que son accesibles por vía de una o más interfaces públicas.

- Según Clemens Szyperski[3], una unidad binaria de producción, adquisición y despliegue independiente, que interactúa con otras para formar un sistema que funciona.

Bertrand Meyer[4] establece siete criterios que ayudan a definir lo que es un componente en el ambiente de un sistema y las cualidades que le dan valor en el marco de un proyecto de CBD (Component Based Development). (12)

1. Puede ser usado por otros elementos de software (clientes).
2. Puede ser usado por clientes sin la intervención del desarrollador.
3. Incluye una especificación de todas sus dependencias (plataformas de hardware y software, versiones, otros componentes).
4. Incluye una especificación precisa de todas las funcionalidades que ofrece.
5. Puede ser usado basándose solamente en esa especificación.
6. Se puede componer con otros componentes.
7. Puede ser integrado en un sistema en forma rápida y sin dificultad.

De todos los conceptos antes citados, se concluye que los componentes son unidades de composición de aplicaciones de software, que poseen un conjunto de interfaces y requisitos , y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio, aclarando que no todos los artefactos que se generan se pueden considerar componentes aunque se logre reutilizarlos en más de una aplicación; pues para construir un componente se debe desarrollar un modelo que permita generarlos en función de su posible reutilización en diferentes aplicaciones.

1.2.3 Gráfica.

Una gráfica es la representación de datos, generalmente numéricos, mediante líneas, superficies o símbolos, para ver la relación que esos datos guardan entre sí. También puede ser un conjunto de puntos, que se plasman en coordenadas cartesianas, y sirven para analizar el

[3] Clemens Szyperski es Profesor Agregado en la Facultad de Tecnología de Información de la Universidad Queensland en Australia y es autor de un libro fundamental en el tema de componentes.

[4] Bertrand Meyer es el autor del lenguaje de programación orientado a objetos Eiffel y fundador de ISE Inc.

comportamiento de un proceso, o un conjunto de elementos o signos que permitan la interpretación de un fenómeno. (2)

1.2.4 Graficador.

Un Graficador es un registrador que presenta en forma de gráfica la información de salida, indicando las tendencias. Los dispositivos gráficos se utilizan comúnmente en la actualidad, junto con las computadoras, para tener un registro permanente del material que se está procesando en una computadora. (10)

1.2.5 Indicador.

Elemento de estadística que permite conocer el avance de un programa o actividad. Puede consistir en porcentajes, etapas, número de operaciones, etcétera, implica la comparación entre lo programado y lo alcanzado, proporcionando la desviación en la ejecución del programa o actividad en el período determinado. (4)

Elemento que indica cierta condición, capacidad o medida numérica que, al registrarse, recopilarse y analizarse, facilita que los conceptos más complejos sean más susceptibles de medición y permite a los administradores y evaluadores comparar los resultados reales del programa con los resultados que se esperan.(9)

1.3 Objeto de Estudio.

En la actualidad es usual encontrar un gráfico hecho a mano. Generalmente se emplean sistemas graficadores de microcomputadoras. Esto no invalida la necesidad de conocer las reglas y convenciones establecidas con respecto a la confección de los mismos. Dada la libertad que brindan algunos de esos sistemas, en más de una oportunidad se han visto gráficos confeccionados por estos medios que presentan errores, entre otras cosas, por seleccionar un tipo de gráfico no adecuado para la información que se desea representar.(13)

Se puede entonces decir que el proceso de graficación no es más que la secuencia de acciones necesarias que se ejecutan para la confección de una gráfica a través de un graficador.

1.3.1 Descripción General.

El proceso de graficación en el área de perforación es de vital importancia para la interpretación cuantitativa y cualitativa de los datos en la Industria Petrolera. Se basa fundamentalmente en el almacenamiento de información referente al costo de perforación, metrajes de perforación diario, distribución del tiempo de perforación por etapas y la perforación direccional del pozo (proyección vertical u horizontal); estos indicadores son almacenados en documentos para arribar a pronósticos certeros de la situación actual existente en los pozos petroleros.

A partir del análisis de la información gráfica es posible arribar a conclusiones que favorecen o no las decisiones que se tomarán posteriormente en la perforación o inversión de un pozo petrolero; si es factible perforar un nuevo pozo, hasta qué profundidad se ha perforado y cuántos días han demorado en alcanzar esa profundidad. Algunas empresas encargadas de realizar software para la industria del petróleo utilizan esta información para hacer representaciones gráficas con gran calidad y precisión como la compañía Schlumberger que emplea el Software WellWizard utilizado en los Centros Petroleros Cubanos.

Este proceso tiene además gran importancia para los Geólogos, Químicos, Direccionales, Supervisores del pozo, investigadores y trabajadores interesados en los temas referentes a las perforaciones petroleras ya que una gran cantidad de decisiones son obtenidas a partir del análisis gráfico.

Realizar un gráfico no es una cosa elemental y no hay recetas para su construcción, hacer un buen gráfico consiste en la comprensión de lo complejo, no en la complicación de lo simple y esto no se consigue sin un cierto esfuerzo y la clara comprensión de qué es lo que se quiere hacer y a quién va dirigido.

El proceso de graficación se divide en tres partes (**Anexo 1**): (15)

1. **Para qué**. La decisión del propósito para el que se realiza la representación gráfica. Este determina el tipo de datos que hay que recoger y sobre los que hay que preguntarse de qué tipo han de ser, (cuantitativos, secuenciales, categorías) y lo más importante: ¿son relevantes para lo que se pretende?
2. **Cómo**. De qué manera se representará. Un aspecto fundamental en este apartado es que los gráficos interesan porque ***revelan diferencias***.

Por ello ***refinarlos, reordenarlos*** y representar, no los propios datos, sino los derivados de su tratamiento estadístico suelen revelar aspectos que de otra forma resultan confusos.

Una vez refinado hay que escoger qué metáfora visual es más efectiva. A veces, para pocos datos, una tabla o incluso una frase, puede ser más clara que un gráfico. En ocasiones cambiar la paleta de colores o el tipo de gráfico puede clarificar enormemente el panorama.

3. **¿Funciona?** Por más bonito y elegante que haya quedado, si no cumple con el objetivo perseguido en el primer apartado, habrá fracasado. La clave está en ***revisar y experimentar con lo realizado hasta encontrar una mejora***.

Variar los ***colores*** reduciendo la saturación de lo menos importante y aumentándola para los datos más relevantes, modificar la tipografía, el tamaño de las letras, eliminar todo lo que sea posible (retículas que no aportan nada, rótulos superfluos, datos redundantes) sin perder la información relevante que proporcionan en ocasiones resultados sorprendentemente mejorados.

1.3.1.1 Representación de información Mediante Gráficos Estadísticos.

"Estadística es un grupo de técnicas o metodologías que se desarrollaron para la recopilación, presentación y análisis de los datos y para el uso de tales datos." (14)

En estadística se denomina gráficos a aquellas imágenes que, combinando la utilización de sombreado, colores, puntos, líneas, símbolos, números y un sistema de referencia (coordenadas), permiten presentar información cuantitativa. La utilidad de los gráficos es doble, ya que pueden servir no sólo como sustituto a las tablas, sino que también constituyen por sí mismos una poderosa herramienta para el análisis de los datos, siendo en ocasiones el medio más efectivo no sólo para describir y resumir la información, sino también para analizarla. (1)

La presentación de datos estadísticos por medio de gráficos es considerada una tarea importante en el proceso de comunicación de información. Usualmente cuando alguien recibe en sus manos un documento con gráficos, la primera mirada se dirige a estos. La exposición de datos mediante gráficos es algo que se realiza a diario y en forma casi natural por personas de las más disímiles profesiones. En comparación con otras formas de presentación de datos, los gráficos permiten, de una mirada, comprender el comportamiento de una variable, aún de variables muy complejas, por lo tanto ahorran tiempo al analista de información. Los gráficos estadísticos permiten usar las habilidades visuales para procesar información.

1.3.1.2 Algunos tipos de gráficos en perforación.

La perforación por su parte toma como base, los datos que fueron extraídos de la exploración además de datos tomados durante el proceso de perforación del pozo, es por ello que al hacer el análisis del área, se demuestra que cuando se representan datos en un tipo de gráfico determinado se puede llegar a conclusiones distintas.

En correspondencia con ello están los software realizados para la perforación, en los cuales abundan el uso de visualizaciones utilizando gráficos como:

- Gráficos de Barras: También se le llama gráfico de columnas, es útil para presentar o comparar varios conjuntos de datos, en el área de perforación se utilizan para mostrar cambios de datos en un período de tiempo o para ilustrar comparaciones entre elementos.
- Gráficos de Pastel: Un gráfico de pastel muestra los datos como un círculo dividido en secciones de colores o diseños, en la industria petrolera se utiliza mucho para representar un grupo de datos (por ejemplo el porcentaje de la distribución del tiempo de perforación de un pozo).

- Gráficos de Área: Cuando se hace necesario mostrar la variación de una o varias variables en correspondencia con el tiempo, se acude en los software de perforación, a dichos gráficos. Es común ver su uso con abundantes colores, donde a cada uno de ellos le corresponde una variable.
- Gráficos de Líneas: Son usados cuando se está en presencia de una variable en función del tiempo, siendo más comprensible la variación que un gráfico de área.
- Gráficos de Superficie y Gráficos de Áreas en 3D: Estos gráficos son muy utilizados en la presente área ya que son capaces de representar determinadas características geológicas que favorecen cuando se va a perforar. Un ejemplo lo constituyen las características de las capas del suelo donde se realizará la perforación.

1.3.1.3 Comportamiento de Indicadores en el Área de Perforación.

Para poder conocer el comportamiento de un indicador en el área de perforación deben utilizarse técnicas y herramientas para medir los parámetros necesarios a evaluar, que permitan conocer la interacción entre los mismos. En la perforación petrolera un indicador no es más que números

o índices que tienen un valor indicativo, datos que son obtenidos, pues sirven de instrumento de medida.

Unas veces son números que contienen un elevado grado de información, por ejemplo, un número relativo de los costos de perforaciones diarias, el metraje diario de un pozo petrolero, el porcentaje de pozos perforados con respecto al plan. Otras veces son números con carácter de una medida, por ejemplo la rentabilidad de un proyecto (si es rentable o no perforar un pozo petrolero); la profundidad (DEPTH) medida en pies por hora; la inclinación de un pozo (ángulo con respecto a la vertical) en grados; la profundidad vertical verdadera (TVD) en metros.

1.3.1.4 Perforación Direccional.

Perforar direccionalmente es dirigir un pozo de manera controlada a lo largo de una trayectoria planeada para alcanzar un objetivo geológico. (3)
La planificación de un pozo direccional se realiza como un paso previo a la perforación del mismo. Es responsabilidad del Planificador Direccional, quien generalmente trabaja en conjunto con el Ingeniero de Perforación, Geólogos y Supervisores del Pozo.

Cuando se planifican pozos direccionales se utilizan distintos métodos como el Tangencial, Ángulo Promedio, Radio de Curvatura y Curvatura Mínima este último es el más preciso y el estándar usado en la industria.

Métodos de cálculo de trayectoria.

Los métodos de cálculo de trayectoria utilizan grupos de datos llamados estaciones de survey, los que proporcionan inclinación azimut y profundidad medida. Estas mediciones son proporcionadas por los sensores de las herramientas de medición mientras se perfora (MWD, por sus siglas en inglés) y la profundidad es proporcionada por los sensores del bloque o por la cuenta de tubería. Existen varios modelos de survey basados en diferentes suposiciones de la forma del pozo entre estaciones de survey. Excepto del método de la tangente, las demás proporciones de mediciones son virtualmente idénticas. El método más usual es el de mínima curvatura. Este método asume que el pozo es una curva constante entre las estaciones de survey y que es tangente al ángulo medido en cada estación. (3)
En la figura 1, el ∆Norte es el cambio en la coordenada Norte en pies, el ∆Este es el cambio en la coordenada Este, ∆Vert es el cambio en la coordenada de profundidad vertical verdadera

(TVD, por sus siglas en inglés), ∆MD es el cambio en la profundidad medida, θ es el ángulo de inclinación, Φ es el cambio en azimut, RF es el factor de relación y DL es el "dog leg" pata de perro o cambio total de ángulo en el intervalo. (3)

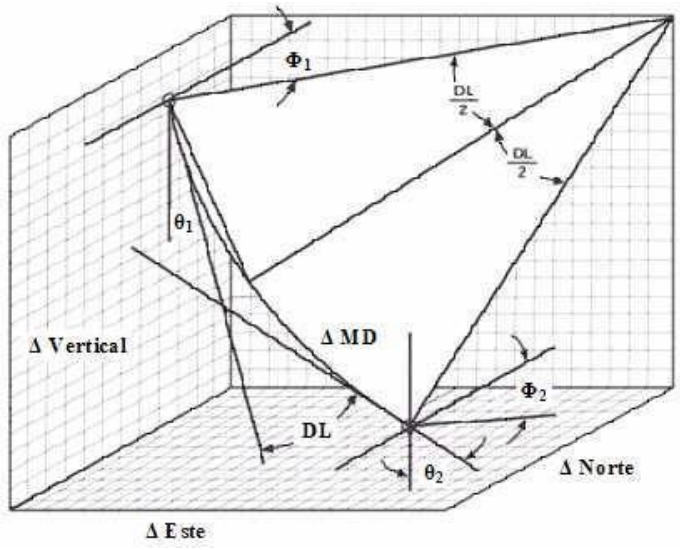


Figura 1. Método de Mínima Curvatura.

1.3.2 Descripción actual del dominio del problema.

Actualmente el proyecto vinculado al área de perforación SIPP necesita contar con un componente que automatice el proceso de graficación del comportamiento de los indicadores de dicha área, por lo que se hace necesaria la descripción del entorno donde coexiste el negocio y las organizaciones que la rodean.

El entorno del negocio estudiado se encuentra dividido en tres áreas fundamentales: (11)

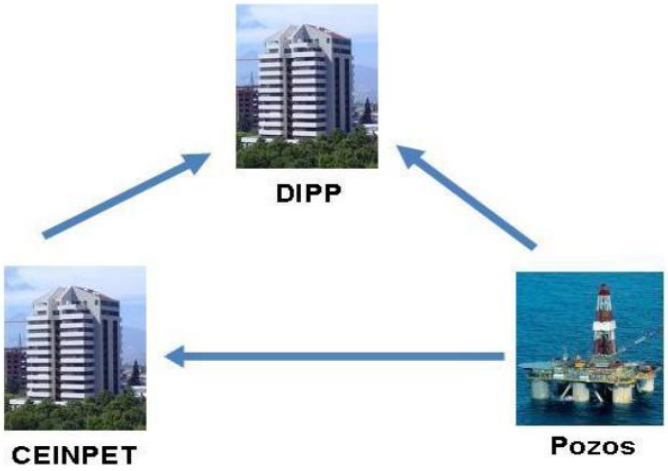


Figura 2. Distintas áreas del negocio

DIPP: Se encuentra ubicada en el municipio Varadero perteneciente a la provincia de Matanzas, aquí se confeccionan una serie de informes, partes y reportes a mano con el software Microsoft Office Excel con las informaciones que se reciben diariamente a las 8:00 de la mañana por correo electrónico de los pozos petroleros en perforación y de CEINPET, en caso de no existir la conexión la misma es gestionada por teléfono. Estos documentos generados contienen información de la perforación diaria, donde se incluyen informaciones de la cantidad de combustible, barrenas, herramientas, camisas y productos químicos utilizados en los pozos, así como las operaciones y actividades que se realizarán en el día, además de la cantidad de dinero invertido. Los mismos son manipulados por las secretarias de la DIPP, aprobada por el Jefe de Despacho, y guardada en formato duro y digital para tomar decisiones en momentos posteriores. (11)

CEINPET: Perteneciente a la Unión de CUPET, la misma está localizada en Boyeros, provincia Ciudad Habana. Aquí diariamente la información es recibida en forma de reporte a las 7:00 de la mañana por correo electrónico al igual que en la DIPP en caso de error con la conexión se realiza por teléfono, luego de ser recibida y gestionada a mano utilizando el software excel (Programa perteneciente al paquete de Office creado por la Microsoft) para realizar el parte diario de geología y el paint para dibujar poco a poco la columna litológica que contiene todos los datos de litología de los pozos en perforación, es decir los tipos de rocas, formaciones existentes en los distintos intervalos en los cuales se han perforado, edad de dichas formaciones, para luego ser guardados en formato duro y digital, llevando de esta forma un historial con todos los documentos realizados en una carpeta. (11)

Pozos Petroleros en Perforación: Están ubicados por toda la parte norte de nuestro país principalmente en las provincias de La Habana y Matanzas. En los mismos laboran una serie de personas que contribuyen a la gestión de la información generada en cada uno de los pozos las cuales se describen a continuación. (11)

Geólogo: Realiza diariamente un estudio consecuente de las características de las rocas, del terreno, analizando todo tipo de formación, obtiene información del WellWizard, elabora el Reporte Diario de Geología, lo guarda en formato duro y digital y se lo entrega al Supervisor de Pozo, además de enviárselo al geólogo que trabaja en la oficina del CEINPET. (11)

Supervisor de Pozo: Recibe diariamente información en formato duro y digital del químico, geólogo, direccional y analiza minuciosamente el software WellWizard, examina los parámetros recibidos, confeccionando con estos datos una serie de reportes, los cuales los guarda en formato duro y digital y envía diariamente a las 12:00 de la noche a la DIPP y a CEINPET. (11)

1.3.3 Situación Problemática.

En la Universidad de las Ciencias Informáticas, en la Facultad 9, se encuentra el Polo Productivo: Soluciones Informáticas para la Industria del Petróleo (PetroSoft); en él se encuentra el proyecto Sistema de Manejo Integral de Perforación de Pozos (SIPP), este proyecto está actualmente en fase de construcción y tiene la necesidad de poseer una función dentro del sistema que le permita graficar una serie de datos representativos del estado de los pozos en perforación, además el Polo PetroSoft, no cuenta con una base de componentes para la construcción de sistemas, es decir no se mantiene una línea única de trabajo para implementar estas características comunes, con la calidad, flexibilidad y rendimiento que aplicaciones de este tipo demandan, lo que conlleva a la existencia de gran dependencia entre los procesos generales y específicos del negocio.

Dentro de las tareas que se llevan a cabo en CEINPET y en la DIP se encuentra la gestión de la información de los pozos petroleros y su representación gráfica, mucha de la información se halla de manera dispersa y desorganizada pues la manipulación de la información se realiza utilizando herramientas ofimáticas como el Excel.

En muchas ocasiones, a las unidades técnicas de producción se les solicita realizar los reportes sobre la situación existente en los pozos petroleros para ello es necesario la utilización de la

información perteneciente al área de perforación y a partir de ésta, realizar gráficos que representen el comportamiento de los distintos indicadores se torna engorroso. Este trabajo suele demorar varios días y hasta semanas debido a que los datos deben ser introducidos en el Excel para lograr las gráficas correspondientes al reporte solicitado y en ocasiones suele ser considerable el volumen de información a procesar. Esta demora provoca que el servicio prestado por la DIPP y CEINPET a los pozos petroleros no sea tan rápido como se desea y la atención a sus clientes, demore mucho tiempo.

En CEINPET se cuenta con un conjunto de software propietarios, entre ellos WellWizard, que se encargan de interpretar y graficar los registros de pozos, estos software poseen licencias, lo que unido al difícil acceso de las actualizaciones, por su costo, provoca que parte de los trabajadores que laboran con el mismo se vean excluidos de las principales funcionalidades que brindan. Por tal razón se plantea la necesidad de desarrollar un componente que permita graficar el comportamiento de indicadores para el área de la perforación de la Industria Petrolera, lo que posibilitará, el ahorro de tiempo y el esfuerzo para automatizar dicho proceso.

1.4 Análisis de las soluciones existentes vinculadas al campo de acción.

En la actualidad existen diversos productos desarrollados para el sector del Petróleo, encaminados a resolver problemáticas dentro de esta esfera, principalmente en el proceso de visualización de parámetros que ayudan en la toma de decisiones, generados en operaciones realizadas en los pozos en perforación.

Las aplicaciones realizadas hoy en el mundo están muy desarrolladas y se relacionan con las distintas fases para la búsqueda del crudo, partiendo de la exploración que es la primera fase y se caracteriza por la búsqueda de las llamadas trampas de petróleo, en aquellas regiones donde se cree que las condiciones geográficas han sido favorables para su formación, a través de métodos geológicos y geofísicos en esta fase se estudian los terrenos minuciosamente utilizando distintos métodos y apoyándose de la visualización gráfica se interpretan las distintas propiedades que tiene el suelo donde se puede o no formar un yacimiento, además de generar con estudios más profundos una proyección futura para la extracción de este importante producto.

La **perforación** es la segunda etapa en la búsqueda de petróleo. Para realizarla, se perfora un pozo - el pionero - mediante la utilización de una sonda. La profundidad de un pozo es variable,

dependiendo de la región y de la profundidad a la cual se encuentra la estructura geológica o formación seleccionada con posibilidades de contener petróleo. Una vez comprobada la existencia del petróleo, otros pozos se perforarán para evaluarse la extensión del yacimiento. Esta información es la que va a determinar si es comercialmente viable o no producir el petróleo descubierto. (5)

Durante estas etapas se generan volúmenes considerables de informaciones y reportes, debido a esto se han elaborado distintas aplicaciones para ayudar al mejor cumplimiento de las labores de los especialistas en sus distintas aéreas.

Actualmente CEINPET, centro para el cual se propone la solución posee un sistema adquirido de la firma Schlumberger (WellWizard) que se encarga de interpretar y graficar los registros de los pozos, este software restringe su uso mediante una licencia, lo que provoca unido a la falta de acceso a las actualizaciones que el centro tenga que invertir mucho capital para su mantenimiento.

1.4.1 WellWizard.

Sistema Integrado de Datos (WellWizard) es un sistema versátil que utiliza cualquier servicio de flujo de datos de pozo. Todos los datos geológicos y de perforación están integrados, en tiempo real, a través de la interfaz gráfica WelWizard.

WelWizard IDS es configurable para integrar datos de múltiples fuentes, incluyendo la perforación direccional, de medición durante la perforación, el registro durante la perforación y de ensayo y de producción.

Posee un Sistema de Archivo que puede crear reportes de barras, mixtas y registros geológicos que se imprimen en papel continuo o guardan en archivos PDF. Todos los archivos son automáticamente reflejados en un servidor remoto que proporciona rápido acceso a todos los televidentes con WITSML capacidades de exportación e importación. Los usuarios también tienen la habilidad de fijar los parámetros de alarmas y el cambio de unidades para completar la personalización.

1.4.2 WellFlo 3.

WellFlo es una Suite de Programas que Modelan y Optimizan Pozos de Crudo y Gas y Redes de Producción para ingenieros petroleros que diseñen completaciones, pronostiquen desempeños, diagnostiquen problemas de pozo u optimicen producciones desde instalaciones existentes. La versión 3 de WellFlo brinda un nuevo nivel de sofisticación, velocidad y precisión. Al mismo tiempo, el programa permanece suficientemente simple de usar, asegurando un aprendizaje rápido y un alto nivel de productividad un factor importante en la altamente exigente industria del petróleo y gas de hoy. (6)

A continuación un resumen de las principales características de modelación de pozos que posee el software:

Modelado de Pozos y realización de Pozos Horizontales. (7)

- Modelado Preciso de pozos Horizontales.
- Opciones de influjo de estado semi-permanente o permanente.
- Modelado de pérdida de presión a lo largo de la sección horizontal.
- Múltiples puntos de influjo.
- Modelado de múltiples huecos de drenaje.

Salida Gráfica Disponible para Cálculos de Caída de Presión (contra profundidad vertical medida o real). (7)

- Perfil a lo largo del pozo para Presión, Temperatura; tasas de flujo de Fase; gradientes de presión total, gravitacional, de fricción y aceleración; retenciones de fase “No slip” e insitu; regímenes de flujo; velocidad de descarga de Turner; velocidades de fase n-situ y superficiales.
- La data medida de presión y temperatura puede ser graficada.

1.4.3 InfoProd.

InfoProd es un sistema integrado para la producción de petróleo y gas, que posee una base de datos y Sistema de Análisis. Los datos se pueden representar gráficamente en diferentes formatos de diagrama. Estos datos pueden ser exportados a hojas de cálculo y procesadores de texto. Los usuarios también pueden definir informes personalizados.

También los informes pueden ser generados en diversas formas gráficas (coordenadas cartesianas, gráficos circulares, gráficos de barra, mapas de burbuja, etc.).También proporciona funciones de ajuste y la extrapolación que permite un análisis completo de la declinación.

Algunas características:

- Gestión integrada de flujo.
- Exportación de datos a otras aplicaciones.
- Reporte diario de las generaciones.
- Proceso Mensual.
- Estadísticas de producción.

1.4.4 Strater.

Strater es una potente e innovadora herramienta para el registro de pozos y el trazado de perforaciones. Con esta herramienta los geólogos ya no tendrán que invertir más tiempo y dinero para crear registros profesionales de pozos. Strater es una herramienta potente, sencilla de utilizar y con un precio muy asequible. (8) (**Anexo 2**)

Con Strater se puede visualizar gráficamente: (8)

- Profundidad o elevación.
- Notas, comentarios y otros datos de texto.
- Petrología, % 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 (Unidades Térmicas Británicas) y datos de contenido de cenizas.
- Virtualmente cualquier tipo de dato de profundidad o intervalo.

Strater ofrece una flexibilidad insuperable para el diseño y formateo de registros. Su avanzada interfaz de usuario permite que el diseño y la visualización de sus datos sean más fáciles que

nunca, Strater incluye 13 tipos de registros 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 se pueden citar: (8)

- Visualizar registros basados en profundidad o elevación.
- Visualizar profundidad y/o líneas de rejilla variable.
- 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.
- Crear registros continuos sencillos o de múltiples páginas.
- Formatear diferentes tipos de registro para obtener presentaciones lo más informativas posible.

Strater proporciona varias funcionalidades para simplificar la tarea de importar datos, crear el diseño de la perforación exacta que el usuario requiere y obtener la salida en el formato necesario, ya sea impreso o exportado a formato electrónico para incluir en un informe o presentación. (8)

1.5. Conclusiones Parciales.

En este capítulo se trataron diferentes temas, como los tipos de gráficos que existen en el área de perforación, diferentes conceptos sobre las áreas temáticas que se abordarán en la investigación, así como algunos de los principales software existentes en la industria del petróleo y que se utilizan en nuestro país, realizándose un estudio del proceso de graficación para área de perforación de manera general. Después de haber analizado los conceptos y definiciones abordadas fue posible percatarse de las ventajas que proporciona el uso de los componentes reutilizables para agilizar la implementación y garantizar un mejor mantenimiento en una aplicación, para ello en el próximo capítulo se definirán las tecnologías y herramientas a utilizar para su desarrollo.

Capítulo 2. Tendencias y Tecnologías Actuales a Considerar.

2.1 Introducción.

Actualmente existen disímiles herramientas y tecnologías para dar soluciones a problemas complejos que se presentan a diario y que favorece un mayor desempeño de grandes y medianas empresas. En el siguiente capítulo se argumentará sobre las tecnologías y herramientas utilizadas en el desarrollo de software. Se analizarán las metodologías de desarrollo de software más representativas, así como métodos adoptados para construir un componente y se hace una exposición de la tecnología escogida para modelar, implementar y garantizar el correcto funcionamiento de las funcionalidades requeridas por el software.

2.2 Metodologías de Desarrollo de Software.

Las metodologías de desarrollo de software, constituyen uno de los temas más polémicos en el mundo del desarrollo de aplicaciones. Una metodología de desarrollo de software permite producir organizada y económicamente software de alta calidad, además su uso persigue, como objetivo principal, lograr el éxito rotundo de los proyectos de producción de software, para lo cual imponen un proceso disciplinado con el fin de hacerlo más predecible y eficiente.

Grady Rumbaugh[5] dió la siguiente definición:

"Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo". (21)

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte se tienen aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán, las cuales son conocidas

[5] Científico de la computación y metodologista escribió varios libros sobre UML y RUP junto a Ivar Jacobson y Grady Booch.

como “metodologías pesadas o formales”. Otra propuesta es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software, siendo esta la filosofía de las llamadas “metodologías ágiles”, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. (19) Se analizarán a continuación varias de las metodologías más conocidas, abordando algunas de sus características.

2.2.1 Proceso Ágil Unificado (AUP).

El Proceso Ágil Unificado (Agil Unified Process AUP), es una versión simplificada del Proceso Unificado de Desarrollo de Software (RUP), que describe de forma simple y fácil de comprender el uso de técnicas ágiles para el desarrollo de aplicaciones que permanezcan dentro de los conceptos de RUP. No es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos, utiliza el Lenguaje Unificado de Modelado (UML) como representación visual.

AUP al igual que RUP se caracteriza por ser ***dirigido por casos de uso*** donde los casos de uso definen lo que el usuario desea a partir de la captura de requisitos y la modelación del negocio. Es ***centrado en la arquitectura***, característica que brinda una visión completa del sistema, se describen los procesos del negocio que son más importantes, para comprenderlo, desarrollarlo y producirlo de una forma eficaz. ***Iterativo e Incremental*** donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo.

La figura 3 representa el ciclo de vida de AUP, con respecto a RUP las disciplinas han cambiado. Primero, la disciplina de Modelado abarca las disciplinas de Modelado del Negocio, Requerimientos y de Análisis y Diseño. Segundo, la disciplina de la Administración de la Configuración y Cambios ahora es la disciplina de la Administración de la Configuración.

La naturaleza serial en AUP es capturada en cuatro fases: (22)

1. **Iniciación:** El objetivo es identificar el alcance inicial del proyecto, una arquitectura potencial de su sistema, y obtener la financiación inicial del proyecto y la aceptación del involucrado.
2. **Elaboración**: El objetivo es mejorar la arquitectura del sistema.

3. **Construcción**: El objetivo es construir software funcional en una base regular e incremental, la cual cumpla con las necesidades de prioridad más alta de los involucrados de su proyecto.
4. **Transición**: El objetivo es validar y desplegar su sistema en su ambiente de producción.

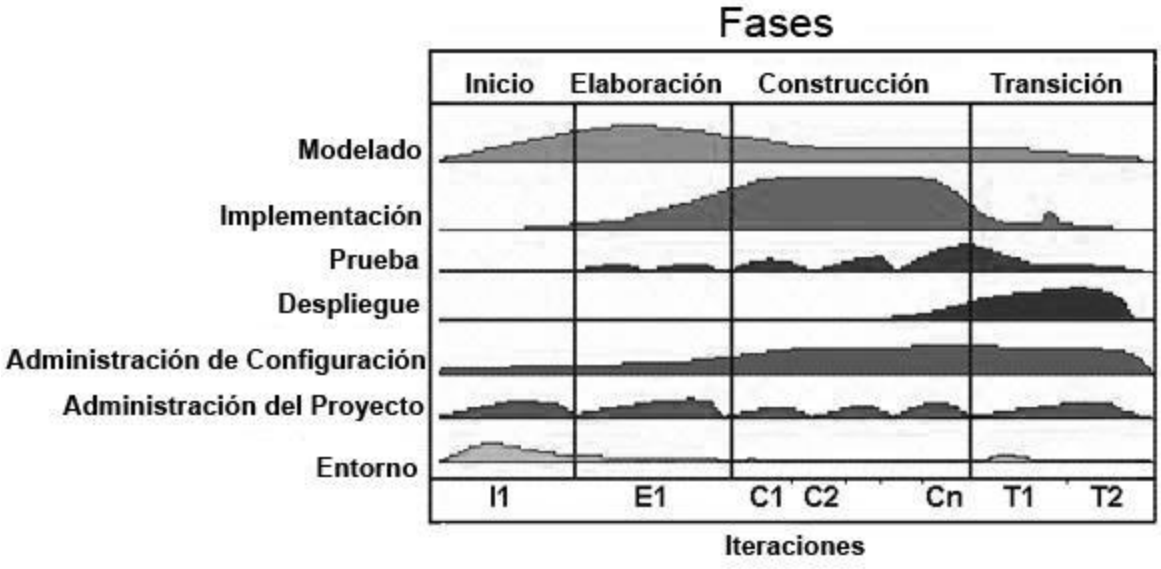


Figura 3. Fases y flujos de trabajo definidos por AUP.

Las disciplinas son ejecutadas en una manera iterativa, definiendo las actividades, las cuales los miembros del equipo ejecutan para construir, validar y liberar software funcional que cumpla con las necesidades de sus involucrados. Las disciplinas son: (22)

- **Modelado:** El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se aborda en el proyecto, e identificar las soluciones viables para manejar el dominio del problema.
- **Implementación**: El objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y llevar a cabo un nivel básico de las pruebas, en particular, la unidad de prueba.
- **Pruebas**: El objetivo de esta disciplina es ejecutar una objetiva evaluación para asegurar la calidad. Esto incluye la detección de defectos, validaciones de que el sistema funciona como fue diseñado, y verificar que se cumplan los requerimientos.
- **Despliegue**: El objetivo de ésta disciplina es planificar la entrega del proyecto de desarrollo y ejecutar el plan, para dejar disponible el sistema al usuario final.

- **Administración de la Configuración**: La meta de esta disciplina es manejar el acceso a sus productos de trabajo de proyecto. Esta no sólo incluye el rastreo de versiones del trabajo del producto en el tiempo, sino que también el control y administración de los cambios en estos productos.
- **Administración del Proyecto**: El objetivo de esta disciplina es dirigir las actividades a lo largo del proyecto. Esto incluye la administración del riesgo, dirección del personal (asignación de tareas, rastreo del progreso, etc.), y coordinación con personas y sistemas fuera del alcance del proyecto para asegurar su liberación a tiempo y dentro del presupuesto.
- **Entorno**: El objetivo de esta disciplina es soportar el resto del esfuerzo asegurando que el proceso apropiado, las guías (normas y directrices), y herramientas (hardware y software) estén disponibles para cuando el equipo las necesite.

2.2.2 Programación Extrema (XP).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales y considerada una de las más exitosas dentro del desarrollo del software, promoviendo el trabajo en equipo preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. (23)

Características de XP, la metodología se basa en: (24)

- **Pruebas Unitarias**: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- **Refabricación**: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares**: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un

proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

2.2.3 Desarrollo de Componentes de Software Reutilizables (CSR).

En la actualidad, no existe una metodología patentizada y validada para el desarrollo de componentes de software reutilizables. Sin embargo, diversas empresas y universidades han personalizado el Proceso Unificado de Desarrollo según sus necesidades, logrando un modelo a seguir válido, y que garantiza el cumplimiento de los objetivos trazados. Entre ellas se puede mencionar el Método WATCH-Component, marco metodológico orientado al desarrollo de aplicaciones empresariales caracterizadas por estar basadas en la reutilización de componentes, emplear tecnología web y ser de pequeña a mediana escala. Consta de tres modelos: el modelo de producto, que captura las propiedades de los CSR; el modelo del proceso, que describe las actividades necesarias para realizar la producción; y el modelo del grupo de desarrollo, que describe la estructura y los roles del personal que participa en el proyecto de desarrollo. (19)
El Método WATCH-Component: (20)

- Es un método desarrollado expresamente para producir componentes de software reutilizable.
- Consta de tres modelos:

  - Modelo del producto: Captura las propiedades de los CSR.
  - Modelo del proceso: Describe las actividades necesarias para producir CSR.
  - Modelo del grupo de desarrollo: Describe los actores y roles del grupo de desarrollo de CSR.

2.2.4 Análisis Comparativo.

AUP constituye una metodología formal, ampliamente configurable, con facilidad para la construcción de componentes reutilizables, que puede ser aplicada a casi cualquier proyecto de desarrollo de software, permitiendo adaptar sus artefactos, roles, y modelos a las especificidades del proyecto que esté siendo desarrollado. Extreme Programming es una metodología ágil, que requiere interacción con el cliente en todo momento, que es utilizada principalmente en

proyectos donde se automaticen procesos con requisitos inestables, y que tiene como meta entregar el producto al cliente lo más rápido posible.

Metodología	Dirigido por casos de uso	Desarrollo iterativo incremental	Construcción de componentes reutilizables	Participación activa del cliente
XP	No	No	No	Si
AUP	Si	Si	Si	No

Tabla 1. Comparación entre metodologías.

2.2.5 Metodología Seleccionada.

Para cumplir los objetivos de esta investigación, se hace necesario utilizar una metodología altamente personalizable, que proporcione facilidades para su configuración, y que no requiera una constante interacción con el cliente. Además, que proporcione artefactos formales que posibiliten la correcta documentación del componente que se desarrollará para su posterior utilización, además de ser la única de las metodologías estudiadas que garantiza la elaboración de las fases de un producto de software orientado a objetos y brindar nuevos artefactos permitiendo modelar el sistema de forma correcta, se enfoca en las actividades de alto nivel, no en cada actividad posible por la que podría pasar un proyecto como lo hace RUP, y al estar basado en principios de desarrollo ágil, garantizará una rápida reacción ante los cambios ocasionados en los requisitos del cliente y centrará los esfuerzos de desarrollo del producto en las tareas de mayor valor, dedicando especial atención a la excelencia técnica y al buen diseño. Por lo antes expuesto, la metodología seleccionada para guiar el proceso de desarrollo de la solución propuesta en esta investigación es AUP, la cual se utilizará, según las necesidades que se planteen para el desarrollo del producto.

2.3 Lenguaje Unificado de Modelado (Unified Modeling Language, UML).

Desde los inicios de la ingeniería de software el hombre necesitó modelar mediante dibujos sus conceptos. Con ello se facilita el análisis y comprensión de sus ideas. Con el desarrollo de esta industria y los procesos de desarrollo de software en equipo surge la necesidad de crear lenguajes visuales que estandarizaran el modelado. Este debía cumplir con una premisa: la

modelación de un sistema debe ser comprendida por desarrolladores de cualquier parte del mundo. Para dar solución a lo mismo es creado el Lenguaje Unificado de Modelado por Grady Booch, Jim Rumbaugh e Ivar Jacobson.

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Esta herramienta de modelado ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.(25)

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema, en UML 2.0 se definen una serie de diagramas adicionales a los establecidos en las versiones anteriores de UML, el conjunto de diagramas se encuentra organizado en torno a dos categorías: diagramas estructurales y diagramas dinámicos o de comportamiento (**Anexo 3**).

UML 2.0 es la mayor revisión que se le ha hecho a UML desde la versión 1.0. El modelo conceptual en el 2.0 ha sido reestructurado completamente y nuevos diagramas han sido incorporados. Los diagramas tradicionales también han sido mejorados, por ejemplo, los diagramas de secuencia y de despliegue; en el primero es posible hacer referencias a otros diagramas de secuencia, además de incluir flujos alternos, opcionales, lazos, entre otros y en el segundo, sobre los diferentes nodos de la infraestructura de red se colocan, a modo de artefactos (elementos físicos simples), los elementos componentes del software. (26)

En UML 2.0 los diagramas aparecen dentro de un marco (frame) que posee una etiqueta para indicar el tipo de diagrama, lo que permite al usuario una referencia rápida al diagrama invocado.

Esta versión proporciona a los analistas, arquitectos y desarrolladores; herramientas cada vez más potentes que les permite aprovechar mejor los modelos y como consecuencia generar una mayor cantidad de código reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

2.4 JavaScript como lenguaje de Programación.

En la actualidad existen múltiples lenguajes de programación, para aplicaciones de escritorio, aplicaciones web, inteligencia artificial, entre otros muchos tipos de aplicaciones que se pudieran mencionar.

JavaScript es un lenguaje de programación orientado a objetos, implementado generalmente como un componente integrado en el navegador web, que entre otras cosas permite el desarrollo de interfaces de usuario enriquecidas y sitios web dinámicos, además de estas utilidades posee gran cantidad de librerías que permiten y facilitan el rápido desarrollo de aplicaciones, ya que es libre y de código abierto.

Ventajas de JavaScript: (31)

- **Fácil de aprender, rápido y potente.**

  - Lenguaje sencillo de aprender con utilización de funciones para crear grandes aplicaciones web, trabaja con muchas propiedades de los exploradores web sin necesidad de cargar ninguna máquina virtual para ejecutar el código, utiliza una arquitectura orientada a objetos parecida a la de Java o C++.

- **Usabilidad.**

  - JavaScript es uno de los lenguajes más utilizados en la Web la mayoría de los navegadores web lo reconocen por lo que se encuentra ampliamente difundido en las grandes redes.

- **Reducción de la carga del servidor.**

  - Adoptado JavaScript, se puede hacer cargo de gran parte de las funciones del cliente de las cuales se encargaba el servidor, con JavaScript es posible validar lo**s** elementos antes de que el usuario se los envíe al servidor de esta forma se reduce la cantidad de transacciones que se efectúan a través de http.

2.5 Notación JSON.

JSON, Notación de objetos en JavaScript en inglés, *JavaScript Object Notation* es un formato sencillo para el intercambio de información. El formato JSON permite representar estructuras de datos (arrays) y objetos (arrays asociativos) en forma de texto. La notación de objetos mediante JSON es una de las características principales de JavaScript y es un mecanismo definido en los fundamentos básicos del lenguaje. (38)

JSON es una forma de escribir las propiedades de los objetos en JavaScript para que éstos puedan ser enviados entre servidor y cliente y viceversa; remplaza a XML o al envío de texto plano, este formato de datos es más liviano que XML por que la comunicación puede ser rápida. JSON permite definir arrays y objetos de una manera concisa. Esto supone una gran ventaja respecto a la notación tradicional de los objetos y los arrays. Los arrays normales se definen de forma abreviada como una lista de valores separados por comas y encerrados entre corchetes, por lo que se puede concluir que JSON es un formato para intercambiar datos de una forma fácil y precisa.

2.6 Librerías de Desarrollo.

En las aplicaciones web o en cualquier software, se necesita mostrar datos con mucha frecuencia; ya que se recopila demasiada información, mostrarla con números o palabras no es efectivo. Se necesita usar diagramas gráficos fáciles de entender; obviamente la creación de gráficas no es un tema trivial sino que requiere de una dosis de dedicación y esfuerzo, sin embargo para graficar existen disímiles librerías gráficas, estas tienen la capacidad de que permiten generar imágenes para facilitar los fines antes descritos en gráficas de barra, líneas, área y pastel; a continuación se describen varias librerías.

2.6.1 JpGraph.

JpGraph es una librería PHP que permite crear gráficos matemáticos y estadísticos de manera sencilla y con absoluto control. Esta librería soporta una amplia variedad de tipos de gráficos para todas las necesidades. (16)

Esta librería dispone de una extensa documentación y tutoriales para aprender a manejarla. En la documentación se encuentran numerosos ejemplos de su uso, desde los que se puede partir para solucionar nuestras necesidades, utiliza la librería gráfica GD que está escrita en lenguaje C y permite crear y manipular gráficos fácilmente exportándolos en distintos formatos (GIF, JPG y PNG). JpGraph es configurable con distintos tipos de colores, leyendas, tipologías e imágenes de fondo a continuación se describen estas características detalladamente.

Características: (16)

- Soporte para GD1 y GD2, la librería auto detecta qué versión del GD.
- Soporte para incluir texto a las imágenes y soporte para tipos de letra.

- Soporte para niveles de transparencia.
- Soporte para gráficos complejos de Gantt.
- Manejo de las escalas para los ejes del gráfico.
- Soporta formatos PNG, GIF y JPG.
- Soporte para gráficas de barras horizontales.
- Soporte para gráficas de tipo científico.
- Soporte para generación de la escala automática dependiendo de los datos.
- Soporta varios tipos de relleno para las gráficas.
- Documentación muy completa y con una referencia completa de las funciones.

2.6.2 FusionCharts Free.

FusionCharts Free es un recurso para graficar todo tipo de datos usando componentes Flash que está pesando para su uso en páginas web. Con este se pueden crear todo tipo de gráficas animadas, corre en cualquier navegador y soporta los lenguajes de programación*:* PHP, Python, Ruby on Rails, ASP, ASP.NET, JSP, ColdFusion, HTML, incluidas presentaciones PowerPoint. (17)

FusionCharts Free no sólo ofrece una amplia variedad de gráficos, un amplio soporte de lenguajes de programación, sino que también se descarga en un paquete que incluye una ayuda muy completa (en inglés) con abundantes ejemplos y demostraciones. (17)

Características de FusionCharts Free: (18)

- Facilidad de uso.
- Se ejecuta en una variedad de plataformas.
- Reduce la carga en sus servidores.
- Un gran número de tipos de gráficos posible.
- Potente AJAX / JavaScript integración.

2.6.3 Jqplot

Jqplot es un sistema para creación de representaciones gráficas utilizando JQuery, JQuery es una biblioteca o framework de JavaScript que proporciona constantemente nuevas soluciones de

alta calidad de forma gratuita, ya que es software libre y de código abierto, posee GNU[6] (General Public License). JQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. Una de las ventajas es que es independiente del navegador, ocupa muy poco espacio y simplifica las habilidades para trabajas con JavaScript.

Jqplot es un plugin de JQuery que permite crear todo tipo gráficas con datos y con una apariencia muy acabada, probado en varios navegadores como Internet Explorer, Safari y Opera. Es una librería que ofrece muchas ventajas pues todo se realiza del lado del cliente lo que proporciona más rapidez en el proceso de graficación y posee una gran cantidad de ejemplos y documentación para facilitar el trabajo.

Se trata de un desarrollo de código abierto que ofrece:

- Crear varios tipos de gráficos entre los que se encuentran los de barras, tortas y líneas.
- Agregar fechas personalizadas a los ejes cartesianos.
- Agregar sombras y puntos de arrastres en los gráficos.

2.6.4 Librería Escogida.

JpGraph es una de las librerías más utilizadas con php, posee una fácil creación de gráficos estadísticos generando gráficas en formatos de imágenes GIF, PNG y JPG , se puede utilizar a través de licencia LGPL permitiendo utilizar la herramienta en cualquier software propietario; mientras que FusionChartsFree es una herramienta con múltiples ventajas para graficar, pero al crear imágenes del tipo Flash el tiempo de respuesta es mucho mayor que JpGraph; Jqplot es una librería escrita en JavaScript del framework Jquery no hace falta ni php, ni Flash, ni nada más que algo de html y otro poco de JavaScript, las imágenes generadas son de gran calidad y con muy buena presentación con respecto a las demás librerías, buenos colores, no están muy cargadas y lo mejor es muy simple de usar.

En el momento de decidir incluir gráficos en una aplicación tenemos varias opciones, escoger la adecuada a veces no es fácil, pero elegir una opción que proporcione gran cantidad de

[6] Licencia Pública General que busca eliminar las restricciones de uso, copia, modificación y distribución de software.

funcionalidades, que tenga una interfaz amigable entre otros aspectos que la hacen atractivas y usable es una buena propuesta por lo que la librería seleccionada para graficar es Jqplot pues se adapta a las necesidades que se plantean para el desarrollo del producto de forma eficiente.

2.7 Entorno de Desarrollo NetBeans IDE 6.8.

Un entorno de desarrollo integrado o, en inglés, ***Integrated Development Environment*** (‘*IDE*’), es un programa compuesto por un conjunto de herramientas para un programador. (29). Puede dedicarse exclusivamente para un lenguaje de programación o bien para varios. Teniendo en cuenta el entorno de desarrollo y el lenguaje de programación a usar *PHP*, las opciones más atractivas para la elección de un IDE son *Eclipse y NetBeans*, que gozan de reconocimiento a nivel mundial pues son entornos muy potentes para la programación con PHP.

Eclipse posee una serie de plugins con los cuales brinda una mayor funcionalidad al entorno de desarrollo. Este IDE es de gran utilidad para el desarrollo de aplicaciones web, lo cual se hace más sencillo y cómodo con la utilización de los antes mencionados plugins.

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar, y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación, como C / C + +, e incluso lenguajes dinámicos como PHP, JavaScript, Groovy, y Ruby. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito y de código abierto, desarrollado por una comunidad extensa de desarrolladores.

NetBeans IDE dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco sus funcionalidades son ampliables mediante la instalación de packs. (30) El IDE se ejecuta en muchas plataformas incluyendo Windows, Linux, Mac OS X y Solaris, además se puede configurar e instalar fácilmente para satisfacer las necesidades de los desarrolladores.

Características de NetBeans IDE 6.8: (33)

- Soporte PHP Ampliado: Expande el soporte de los lenguajes dinámicos con apoyo para PHP 5.3 y el esquema de Symfony acelera el desarrollo de aplicaciones web PHP.

- JavaFX ™: Código de finalización mejorado, sugerencias y navegación para JavaFX en el editor NetBeans.

Es importante destacar que el IDE NetBeans 6.8 se integra con el framework Symfony, este soporte incluye características como: (34)

- El completado de código ofrece las variables de Symfony adecuadas en los archivos de sus *vistas*.
- Los proyectos de Symfony existentes son reconocidos y también se pueden crear nuevos.
- Atajos de teclado asignables para acciones específicas de Symfony.
- Fácil navegación entre *vistas* y *acciones*, de ida y vuelta.
- Posibilidad de ejecutar comandos de Symfony.

Aunque ambos IDE son similares en cuanto a las funcionalidades y ventajas que ofrecen para el programador, se selecciona NetBeans 6.8 pues es superior que eclipse en cuanto al entorno gráfico además tiene la facilidad de integrarse fácilmente con el framework Symfony propuesto en la solución.

2.8 Herramientas Case.

Las herramientas CASE[7] son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida del software.

Constituyen un soporte automatizado para el desarrollo y mantenimiento del software. Se pueden ver como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales. Existen múltiples herramientas de este tipo especializadas con diferentes propósitos: para la fase de diseño, para generar código, algunas tienen una visión de desarrollo orientada a procesos sin la capacidad de modelamiento, mientras otras proveen el modelamiento sin incluir los procesos de Análisis o Diseño. Entre las herramientas CASE más conocidas y utilizadas a escala internacional, se pueden mencionar ERWIN, EasyCASE, OracleDesigner, PowerDesigner, Rational Rose y Visual Paradigm.

[7] Case: **C**omputer **A**ided **S**oftware **E**ngineering, Ingienería de Software Asistida por Computadoras.

2.8.1 Rational Rose.

IBM Rational Rose Enterprise es uno de los productos más completos de la familia Rational Rose. Todos los productos de Rational Rose dan soporte a Unified Modeling Language (UML), pero no son compatibles con las mismas tecnologías de implementación. Rational Rose Enterprise es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Al igual que todos los productos de Rational Rose, ofrece un lenguaje de modelado común que agiliza la creación del software. (27)

Incluye también estas funciones: (27)

- Soporte a modelos de análisis, ANSI C++, Rose J y Visual C++ según el documento "Design Patterns: Elements of Reusable Object-Oriented Software".
- Los componentes del modelo se pueden controlar independientemente, lo que permite una gestión y un uso de modelos más granular.
- Nuevo: Soporte para compilación y descompilación de las construcciones más habituales de Java 1.5.
- Generación de código en lenguaje Ada, ANSI C++, C++, CORBA, Java y Visual Basic, con funciones configurables de sincronización entre los modelos y el código.
- Soporte para Enterprise Java Beans 2.0.
- Funciones de análisis de calidad de código.
- Complemento de modelado Web que incluye funciones de visualización, modelado y herramientas para desarrollar aplicaciones Web.
- Modelado en UML para diseñar bases de datos, que integra los requisitos de datos y aplicaciones mediante diseños lógicos y analíticos.
- Creación de definiciones de tipo de documento DTD en XML.
- Integración con otras herramientas de desarrollo de IBM Rational.
- Integración con cualquier sistema de control de versiones compatible con SCC, como IBM Rational ClearCase.
- Posibilidad de publicar en las Web modelos e informes para mejorar la comunicación entre los miembros del equipo.

2.8.2 Visual Paradigm.

Visual Paradigm para UML 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. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (28)

Además soporta las últimas versiones de UML, se integra con diferentes entornos de desarrollo como Eclipse y NetBeans y proporciona código y compatibilidad con diferentes lenguajes como C++, PHP, Java, Python y otros

Algunas de sus características principales son: (28)

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversión.
- Interoperabilidad con modelos UML2 (meta modelos UML 2.x para plataforma Eclipse) a través de XMI.
- Ingeniería de ida y vuelta.
- Ingeniería inversa -Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código-Modelo a código, diagrama a código.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML Importación y exportación de ficheros XMI.
- Integración con Visio -Dibujo de diagramas UML con plantillas de MS Visio Editor de figuras.
- Importación y exportación de ficheros XMI.

2.8.3 Herramienta Seleccionada.

La herramienta seleccionada para modelar la solución es el Visual Paradigm para UML en su versión 6.4. La decisión se basa, fundamentalmente, en el hecho de que la herramienta posee

una plataforma denominada SDE capaz de integrarse con el entorno de desarrollo seleccionado, permitiendo realizar el proceso de ingeniería en ambos sentidos e incluye muchas funcionalidades para el lenguaje PHP, además de presentar una interfaz de usuario de fácil uso para realizar los diagramas y artefactos que se generan durante el desarrollo del software.

2.9 Servidor Web Apache 2.2.

Un servidor Web es un programa que permite acceder a páginas Web contenidas en un ordenador. El Servidor Apache: es un sistema de código abierto para plataformas Windows, Unix, Macintosh y otras que implementa el protocolo HTTP (HiperText Transfer Protocol) que está basado en el envío de mensajes y establece el conjunto de normas mediante las cuales se envían las peticiones de acceso a una web y la respuesta de esa web.

Este servidor posee varias características que lo favorecen como por ejemplo:

- Es una tecnología gratuita de código fuente abierta, lo cual le atribuye transparencia a este.
- Corre en varios Sistemas Operativos, lo que provoca que sea una herramienta prácticamente universal.
- Es un servidor altamente configurable de diseño modular. Permite aumentar fácilmente su capacidad e instalar cualquier módulo para cumplir una función específica. Otra cosa importante es que cualquiera que posea experiencia en la programación de C o Perl puede escribir un módulo para realizar una acción determinada.
- Permite personalizar la respuesta ante posibles errores. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log facilitando de este modo el control de las acciones realizadas en el servidor.

El Servidor HTTP Apache Versión 2.2 presenta las siguientes mejoras sobre la versión 2.0: (35)

- Módulos de cacheo mejorados (mod_cache, mod_disk_cache, mod_mem_cache).
- Una nueva estructura para el soporte de autenticación y autorización que remplaza los módulos de autenticación proporcionados en las versiones anteriores.
- Soporte para balanceo de carga proxy (mod_proxy_balancer)

- Soporte para manejo de archivos grandes (más grandes de 2 GB) en plataformas de 32 bits.

- Se han realizado los siguientes cambios a la configuración httpd predeterminada:

    - Los módulos mod_cern_meta y mod_asis ya no se cargan por defecto.
    - Ahora el módulo mod_ext_filter se carga por defecto.

Se escoge Apache como servidor web pues es la solución usada para la mayoría de los sitios Web. La versión 2.2 es una profunda revisión del servidor Apache, las principales revisiones de código se han llevado a cabo para crear una arquitectura realmente escalable, hoy en día es considerada la plataforma Web más utilizada del mundo, pues aumentan cada día el número de usuarios que aceptan este código fuente abierto en su infraestructura, es muy usado en nuestra universidad con amplia disposición de documentos e información, popular (fácil conseguir ayuda/soporte en Internet y otros sitios) y con amplia aceptación en toda la red.

2.10 Herramientas seleccionadas para desarrollar la solución.

A modo de resumen de todo lo planteado en los epígrafes anteriores de este capítulo, se puede concluir que la solución propuesta se desarrollará en el lenguaje de programación JavaScript usando como formato para intercambio de información JSON y utilizando la librería gráfica Jqplot, sobre el entorno de desarrollo integrado (IDE) NetBeans 6.8 y el Servidor Web Apache 2.2 . Como herramienta de modelado se utilizará el Visual Paradigm for UML 6.4 con lenguaje de modelado UML 2.0, y para guiar todo el proceso de desarrollo se utilizará AUP como metodología.

2.11 Conclusiones Parciales.

En este capítulo se realizó un análisis de las tecnologías a utilizar en el desarrollo de la propuesta de solución. Se fundamentó la elección del lenguaje de programación, las herramientas, el entorno de desarrollo, la metodología de desarrollo de software, así como el uso de otras tecnologías. Finalmente se planteó la propuesta que incluye dichos aspectos. Se ha asumido como característica principal la de lograr una mayor libertad tecnológica, teniendo en cuenta los principios de la comunidad de software libre y la portabilidad del producto. A partir de este capítulo, se abordará el desarrollo de la propuesta de solución.

Capítulo 3 Presentación de la solución propuesta.

3.1 Introducción.

En el presente capítulo se describe la propuesta de solución. Para ello se muestran los procesos del negocio mediante el modelo de dominio, se brinda una concepción general de los conceptos asociados al mismo, se particularizan las funcionalidades y las características del componente a desarrollar detallando los requisitos funcionales y no funcionales y se muestra el diagrama de casos de uso del sistema describiendo cada caso de uso para su mejor comprensión.

3.2 Entorno donde trabajará el sistema.

Como punto de partida, es necesario aclarar que los procesos involucrados en esta investigación, no representan procesos “reales” de negocio, existentes en las organizaciones estudiadas, sino procesos genéricos que se manifiestan de manera directa o indirecta en su trabajo cotidiano. Constituyen una generalización de las principales actividades desarrolladas por estas instituciones, obtenida como resultado del estudio de los correspondientes negocios. Por estas razones, se decidió realizar un modelo conceptual de dominio para proporcionar una mayor comprensión del área de acción de la solución propuesta.

Un Modelo de Dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (21)

Un modelo de Dominio se representa cuando no se tienen procesos bien definidos; ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. Para la construcción de un modelo de dominio se representan los conceptos de importancia en el área de interés, así como de las relaciones entre estos. El resultado puede expresarse en un **modelo conceptual de dominio**.

3.2.1 Conceptos principales del entorno.

Dato: Unidad de información, que en este contexto se refiere a la representación numérica del sistema de coordenadas en el plano (x, y).

Directivo: Persona con alto o medio grado de autoridad formal que desarrolla funciones o responsabilidades de dirección de personas, en este caso se refiere a jefes y administradores relacionados con entidades petroleras.

Especialista: Persona experta en una materia determinada, profesional que domina una especialidad.

Geólogo: Especialista encargado de realizar diariamente un estudio consecuente de las características de las rocas, del terreno, analizando todo tipo de formación, información que utiliza para elaborar el Reporte Diario de Geología.

Gráfica: Una gráfica es una representación de datos, generalmente numéricos, que guardan relación entre sí, en forma de líneas, superficies o símbolos; que sirven para analizar el comportamiento de un proceso, conjunto de elementos o signos que facilitan la interpretación de un fenómeno.

Indicador: Un indicador expresa el avance de un programa o actividad, la diferencia entre lo programado y lo alcanzado, es una magnitud utilizada para medir y comparar datos efectivamente obtenidos en la ejecución de un proyecto, programa o actividad, que expresa el comportamiento de determinados valores en el transcurso del tiempo o bien indica un análisis cualitativo o cuantitativo, o ambos de una situación determinada.

Secretaria del Despacho: Es la encargada(o) de gestionar la información relacionada con la DIPP.

Supervisor del Pozo: Analiza, examina y supervisa diariamente los parámetros recibidos del pozo en perforación, confeccionando con estos datos una serie de reportes, los cuales se utilizan posteriormente en la DIPP y a CEINPET.

3.2.2 Eventos principales del Entorno.

Graficación: La graficación es un evento mediante el cual los datos deseados van a ser traducidos a un tipo de gráfica, la cual mostrará de forma más elocuente la relación que existe

entre los mismos. Estas gráficas se pueden representar o no en un sistema de coordenadas en el plano en dependencia del tipo que sean.

3.2.3 Modelo Conceptual de Dominio.

Se pueden representar los principales conceptos presentes en los procesos antes descritos, así como las principales relaciones entre ellos, como se muestra en la *figura 5*.

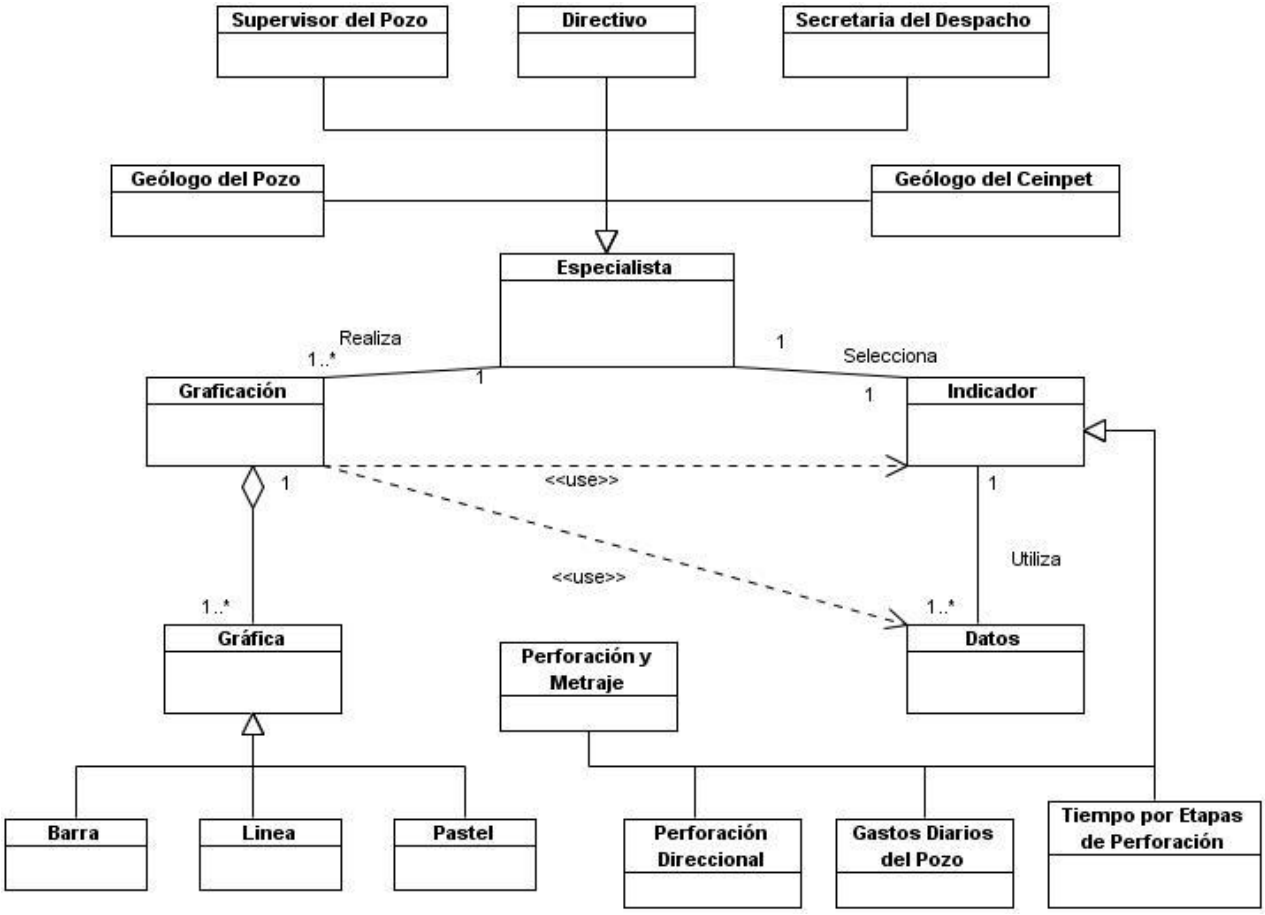


Figura 4. Diagrama de clases del modelo del dominio.

3.2.4 Glosario de Términos del Dominio.

Sistema de coordenadas en el plano.

Sistema de referencia conformado por dos rectas perpendiculares que se cortan en el origen y dividen el plano en cuatro cuadrantes, cada punto en el plano puede nombrarse mediante dos números (x, y), donde a (x) se le denomina abscisa y (y) ordenada. Las coordenadas varían de positivo a negativo según su cuadrante.

Perforación

Resultado final de construir un pozo utilizando un taladro y la colocación de una tubería de revestimiento desde la superficie hasta una formación que contenga hidrocarburos, con todos los equipos de producción necesarios.

3.3 Especificación de los Requisitos del software.

A partir de este punto se modela la solución propuesta. Para ello se identifican sus requisitos, tanto funcionales como no funcionales, y se modelan las funcionalidades en términos de casos de uso del sistema.

3.3.1 Requisitos Funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir.

El sistema debe permitir:

RF1 Generar gráficas de indicadores de perforación.

Este requisito se encarga de visualizar la información de los diferentes indicadores ubicados en los reportes de perforación petrolera.

1.1 Graficar el metraje diario del pozo.

Este requisito tiene la función de visualizar el metraje de perforación planificado contra el metraje real.

1.2 Graficar la perforación diaria del pozo.

Este requisito se encarga de visualizar la profundidad del instrumento planificada por días contra la profundidad real.

1.3 Graficar la perforación vertical del pozo.

Este requisito se encarga de visualizar el comportamiento de la profundidad vertical del pozo planificada contra la esperada, teniendo en cuenta el ángulo de inclinación y el ángulo con respecto a la horizontal.

1.4 Graficar la perforación horizontal del pozo.

Este requisito se encarga de visualizar el comportamiento de la profundidad horizontal del pozo planificada contra la esperada, teniendo en cuenta la dirección Norte/Sur y Este/Oeste.

1.5 Graficar el comportamiento de los gastos diarios del pozo.

Este requisito se encarga de visualizar el comportamiento de los gastos, en cuanto a gastos en CUC, gastos en lodo de perforación y gastos en moneda total (CUC + Moneda Nacional); además del porcentaje de los mismos.

1.6 Graficar el resumen del tiempo por etapas de perforación.

Este requisito se encarga de visualizar el comportamiento de cada actividad en la perforación en una etapa determinada, así como el por ciento de horas dedicado a cada etapa.

3.3.2 Requisitos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, puesto que las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable es el sistema, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Existen varios tipos de requisitos no funcionales. A continuación se presentan los correspondientes a la solución propuesta.

RNF 1 Usabilidad.

- La solución debe ser de fácil comprensión, configuración y utilización por los desarrolladores en la confección de sistemas petroleros desarrollados en plataformas Web.

RNF2 Diseño e implementación.

- Lenguaje de Programación JavaScript.
- Se debe generar la documentación de todas las clases, métodos y recursos creados.
- La solución debe ser extensible de acuerdo a las necesidades de uso.

- La solución debe permitir una configuración flexible para satisfacer las necesidades de diferentes negocios.
- Se utiliza Visual Paradigm para el análisis y diseño de la aplicación.

RNF3 Hardware.

- Requisitos mínimos: Procesador Pentium III a 1.8 GHz, Memoria: 256 Mb RAM, Disco Duro: 80 Gb.
- Requisitos recomendados: Procesador: Pentium IV 2.4 GHz, Memoria: 512 Mb RAM, Disco Duro: 80 Gb.

RNF4 Software.

- Por parte del cliente se requiere un navegador capaz de interpretar JavaScript.

RNF5 Soporte.

- El componente debe estar documentado para garantizar un buen desempeño de los desarrolladores a la hora de interactuar con él, además debe brindar garantía de funcionamiento, adaptación y configuración. Adicionalmente debe proveer el código fuente para que se potencie su extensión y su evolución posterior.

RNF6 Portabilidad.

- La solución debe ser independiente de la plataforma en que se utilice, propiedad que hereda del lenguaje utilizado JavaScript lenguaje interpretado que puede ejecutarse en diferentes plataformas como Windows y Linux.

RNF7 Confiabilidad.

- La solución debe brindar garantía de un tratamiento adecuado de las excepciones.

3.4 Descripción del Sistema Propuesto.

3.4.1 Descripción de los actores.

Los actores del negocio son aquellas personas o sistemas que obtienen un resultado de valor a partir de uno o varios procesos del negocio. Los procesos que constituyen objeto de estudio en el

presente trabajo no se corresponden con procesos existentes en un negocio determinado, sino que son un conjunto de actividades repetitivas que son fuertes candidatas a la automatización en todo proceso de informatización de un sistema petrolero. En la siguiente tabla se describe el actor de la solución propuesta.

Actor	Justificación
Aplicación Cliente	El actor utiliza el graficador implementado para poder representar el comportamiento de los diferentes indicadores de perforación utilizados.

Tabla 2. Actores del Sistema.

3.4.2 Caso de Uso del Sistema.

Los casos de uso son requisitos funcionales que describen de una manera detallada el comportamiento del sistema con los distintos actores que interactúan con él. En ellos se describe una secuencia determinada de eventos que realiza un actor en interacción con la aplicación. A continuación se muestra el diagrama de casos de uso del sistema a desarrollar.

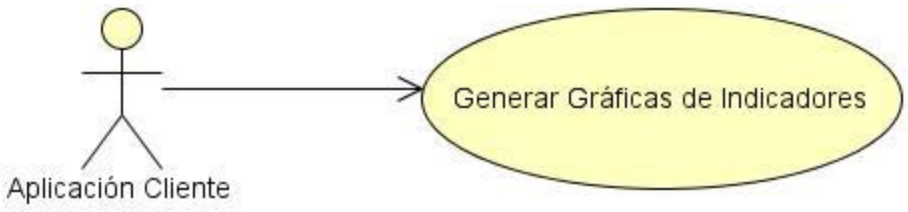


Figura 5. Diagrama de casos de uso del sistema.

3.4.3 Descripción textual de los Casos de Uso del Sistema.

El objetivo principal de detallar cada caso de uso es describir su flujo de sucesos en detalle, incluyendo cómo comienza, termina e interactúan con los actores. (36)

3.4.3.1 Caso de Uso Generar Gráficas de Indicadores.

Caso de uso	**Generar Gráficas de Indicadores.**
Actores	Aplicación Cliente.
Propósito	Generar gráficas de indicadores de perforación.
Resumen	El caso de uso se inicia cuando la aplicación cliente requiere del servicio para

	generar la gráfica de un indicador en específico.
Precondiciones	Es necesario tener los datos que se van a utilizar para generar las gráficas.
Referencias	RF1, RF1.1,RF1.2, RF1.3, RF1.4, RF1.5, RF1.6
Prioridad	Crítico.

Flujo Normal de los eventos.	
Acción del actor.	**Respuesta del sistema.**
1. La aplicación cliente invoca la función de generar gráficas de indicadores. 2. Entra los datos (datos a utilizar, indicador a graficar).	3. Verifica que los datos a utilizar y el indicador a graficar sean correctos. 4. El sistema Gestiona la selección del indicador a graficar : a) Metraje Diario ir a la sección" **Graficar Metraje diario**". b) Perforación Diaria ir a la sección" **Graficar Perforación diaria".** c) Perforación Vertical ir a la sección" **Graficar Perforación vertical".** d) Perforación Horizontal ir a la sección" **Graficar Perforación horizontal".** e) Gastos Diarios ir a la sección" **Graficar Gastos diarios".** f) Tiempo por Etapas ir a la sección" **Graficar Tiempo por etapas".** 5. Se genera la gráfica del indicador correspondiente. 6. Termina el caso de uso.

Sección Graficar Metraje diario.	
Acción del actor.	**Respuesta del sistema.**
1. Selecciona el indicador metraje diario.	2. El sistema gestiona el indicador metraje diario.

Sección Graficar Perforación diaria	
Acción del actor.	**Respuesta del sistema.**
1. Selecciona el indicador perforación diaria.	2. El sistema gestiona el indicador perforación diaria.

Sección Graficar Perforación vertical

Acción del actor.	Respuesta del sistema.
1. Selecciona el indicador perforación vertical.	2. El sistema gestiona el indicador perforación vertical.
Sección Graficar Perforación horizontal	
Acción del actor.	**Respuesta del sistema.**
1. Selecciona el indicador perforación horizontal.	2. El sistema gestiona el indicador perforación horizontal.
Sección Graficar Gastos diarios	
Acción del actor.	**Respuesta del sistema.**
1. Selecciona el indicador gastos diarios.	2. El sistema gestiona el indicador gastos diarios.
Sección Graficar Tiempo por etapas	
Acción del actor.	**Respuesta del sistema.**
1. Selecciona el indicador tiempo por etapas.	2. El sistema gestiona el indicador tiempo por etapas, con las etapas a graficar.
Flujo alterno de los eventos.	
Acción del actor.	**Respuesta del sistema.**
Viene del paso 3 del flujo normal	1. Si los datos o el indicador a graficar son incorrectos se lanza una excepción: **"Error: Verifique que los datos y el indicador sean correctos"**.
Poscondiciones.	Se obtiene la gráfica correspondiente.

Tabla 2. Generar Gráficas de Indicadores.

3.5 Conclusiones Parciales.

En el presente capítulo se realizó una descripción de la propuesta de solución a través de la modelación del dominio y el planteamiento de los requisitos funcionales y no funcionales, describiendo el entorno donde trabajará el sistema. Con el desarrollo de la solución propuesta se creó una base sólida para guiar los esfuerzos del sistema basándose en el caso de uso identificado, ganando claridad en la concepción del sistema a construir y sentando las bases para el diseño y la implementación del componente que se abordará en el próximo capítulo.

Capítulo 4 Construcción de la Solución Propuesta.

4.1 Introducción.

En el actual capítulo se exponen los principales elementos que sustentan la construcción de la solución propuesta. Se mostrarán varios artefactos que tienen como objetivo mostrar las principales actividades de los flujos de trabajo de diseño, implementación y prueba, se dará un acercamiento más detallado al sistema mostrando la arquitectura propuesta vinculando las clases y librerías que se utilizan. También se abordarán brevemente los patrones de diseño y se presentará el modelo de implementación mediante el diagrama de despliegue que resultó del diseño así como algunos resultados obtenidos en las pruebas al sistema desarrollado.

4.2 Patrones de Diseño Utilizados.

Un patrón de diseño es un conjunto de reglas que describen cómo afrontar tareas y solucionar problemas que surgen durante el desarrollo de software (36), es la solución recurrente para un problema típico y en un contexto determinado. La solución se refiere a la respuesta al problema, por lo que ayuda a resolver las dificultades de diseño en problemas similares. Los patrones comunican soluciones de diseño a los desarrolladores y arquitectos que los leen y los utilizan.

Lo importante de los patrones no es expresar nuevas ideas de diseño. Es justamente lo contrario: los patrones pretenden codificar conocimiento, estilos y principios existentes y que se han probado que son válidos. A continuación se muestran algunos de los patrones más relevantes utilizados en el diseño de la propuesta de solución.

4.2.1 Factory Simple

Patrón **Fábrica Simple** en inglés ***Simple Factory*** es un patrón de creación que se encarga de crear instancias de objetos de manera que ya no se tendrán que instanciar directamente proporcionando a los programas una mayor flexibilidad para decidir qué objetos usar.

Su objetivo es devolver una instancia de múltiples tipos de objetos, normalmente todos estos objetos provienen de una misma clase padre mientras que se diferencian entre ellos por algún aspecto de comportamiento. (36)

El objeto ***Factory*** será el encargado de decidir según los parámetros asignados, el tipo de objeto que devolverá; en este caso la utilización del patrón se pone de manifiento en la clase GraphFactory que devolverá un tipo de gráfica determinado según el parámetro que se le pase.

4.2.2 Patrones de Asignación de Responsabilidades.

Para la realización del diseño de la aplicación informática a implementar, se utilizaron los Patrones Generales de Software para Asignación de Responsabilidades (GRASP), en inglés (Responsability Assignment Software Patterns). Se considera que más que patrones son una serie de "Buenas Prácticas" de aplicación recomendables en el diseño de software. Entre los patrones GRASP se encuentran los siguientes:

Experto: el GRASP de Experto en información es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo, lo cual permite que se conserve el encapsulamiento, soportando un bajo acoplamiento y una alta cohesión, la aplicación del patrón se manifiesta en todas las clases pues cada una es la experta en realizar una responsabilidad según información que posee.

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. La nueva instancia deberá ser creada por la clase que: tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase. Este patrón brinda soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases, en el diseño del componente este patrón se evidencia en clases como GeneradorGraph y GraphFactory encargadas de generar y crear gráficas.

Alta cohesión y bajo acoplamiento: se pueden separar, aunque están íntimamente ligados, de hecho si se aumenta mucho la cohesión del sistema software, se tiene un alto acoplamiento entre las clases, y por el contrario si reduce mucho el acoplamiento, se verá mermada la cohesión.

Alta cohesión: este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan bajo acoplamiento, soporta mayor capacidad de reutilización.

Este patrón se pone de manifiesto en la mayoría de las clases porque cada una es capaz de realizar sus responsabilidades sin la utilización de las demás.

Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases, este patrón se evidencia en la mayoría de las clases pues no dependen de ninguna otra clase para realizar sus responsabilidades .

Controlador: Este patrón funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos y la que los envía a las distintas clases según el método llamado, la utilización de este patrón se evidencia en la clase Graficador que es la encargada de manejar el sistema de forma global, o sea maneja los eventos del sistema generado por un actor externo.

4.3 Propuesta de Arquitectura del Componente.

La arquitectura de software son las estructuras del sistema, la cual consta de los elementos del software, las propiedades de estos elementos que son externamente visibles y las relaciones entre ellos. (39)

La Arquitectura de Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los elementos definidos a pasos posteriores del diseño.

El objetivo principal de la Arquitectura de Software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en la realización de un sistema. Para conseguirlo, la arquitectura de software construye abstracciones, materializándolas en forma de diagramas (blueprints[8]) comentados.

No hay estándares en cuanto a la forma y lenguaje a utilizar estos blueprints. De todas formas, existe un consenso en cuanto a la necesidad de organizar dichas abstracciones en vistas tal y como se hace al diseñar un edificio. La cantidad y tipos de vistas difieren en función de cada tendencia arquitectónica.

[8] Reproducción en papel de un dibujo técnico, un plano cartográfico o un diseño de ingeniería.

A continuación se describe en un diagrama la arquitectura propuesta para la construcción del componente Graficador de Indicadores de Perforación Petrolera (GIPP).

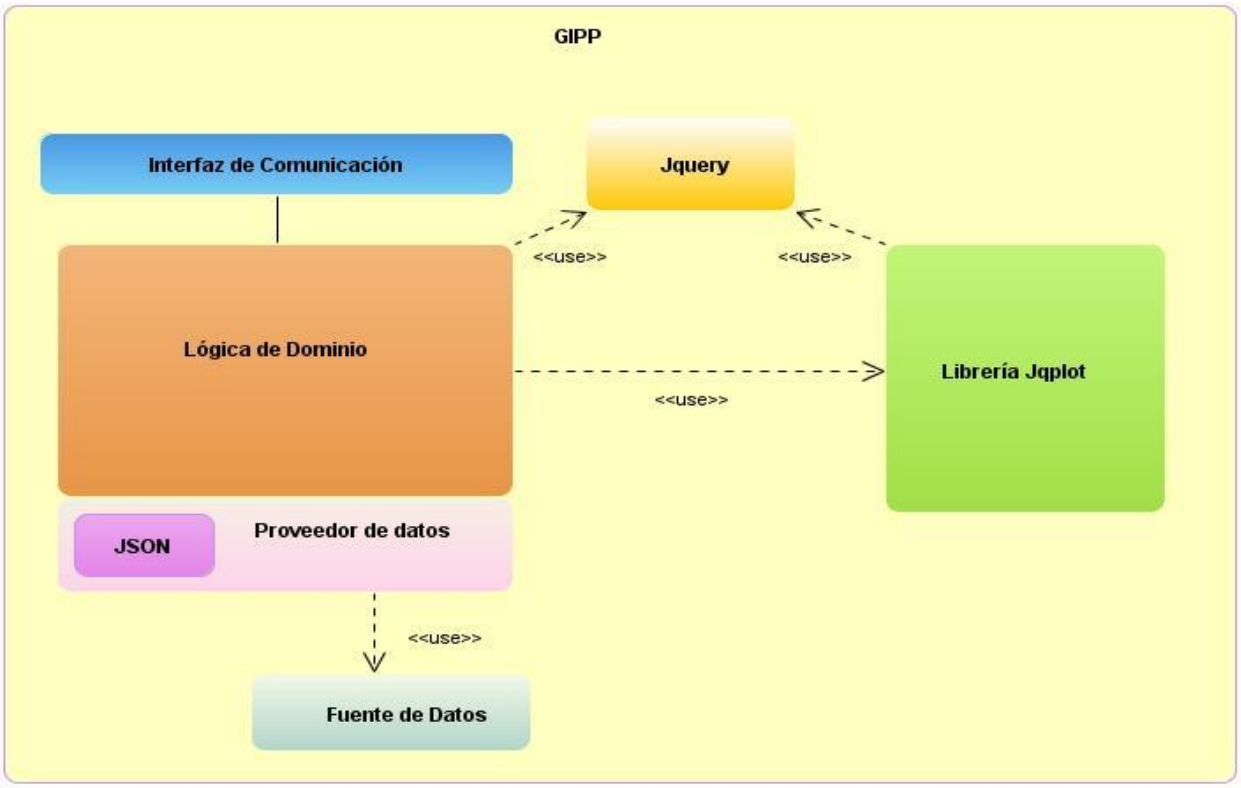


Figura 6. Modelo de Arquitectura del componente GIPP.

En la implementación del componente Graficador de Indicadores de Perforación Petrolera (GIPP) se hace uso de dos contenedores principales, Lógica de Dominio y Librería Jqplot, ambos contenedores se comunican y utilizan la librería Jquery para facilitar el trabajo con JavaScript, lenguaje en que será implementado el componente. Los datos se proveerán desde cualquier fuente de datos que garantice la entrega de los mismos en el formato JSON[9]. La interfaz facilitará la interacción con el componente ya que será la puerta de entrada para interactuar con él.

La arquitectura propuesta anteriormente facilitará la comunicación entre todas las partes interesadas en el desarrollo del componente GIPP, constituye un modelo comprensible de cómo está estructurado el sistema y de cómo trabajan junto sus componentes, servirá para la toma de decisiones tempranas de diseño con un profundo impacto en la construcción del sistema.

[9] Notación de Objetos en JavaScript, es un formato sencillo para el intercambio de información.

4.3 Modelo de Diseño.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de la implementación tienen impacto en el sistema a considerar.(21) Lo principal de esta etapa es la elaboración de los diagramas de clases del diseño, para la confección del mismo se elaboran los diagramas de interacción para obtener una visión general de las clases, interfaces, métodos y atributos que intervienen.

4.3.1 Diagramas de Interacción del Diseño (Colaboración).

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 (37).

Los diagramas de colaboración destacan la organización de los objetos que participan en una interacción, y la secuencia de acciones que se realizan en ella, para una mejor comprensión del diseño se realizó un diagrama para cada sección del caso de uso Generar Gráficas de Indicadores, se pueden apreciar a partir del **Anexo 4**.

4.3.2 Diagrama de Clases del Diseño.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. (37)

Incluye la siguiente información: (37)

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Navegabilidad.
- Dependencias.

A diferencia del Modelo Conceptual, un Diagrama de Clases de Diseño muestra definiciones de entidades de software, más que conceptos del mundo real. El diagrama de clases del diseño es el diagrama más importante de la fase de diseño de un software, este brinda una visión general para comenzar con la implementación del producto. Para la creación del diagrama de clases del diseño primeramente se identifican las entidades software necesarias para desarrollar el sistema que se pretende realizar, se definen las relaciones existentes entre ellas y los atributos y

métodos según la necesidad del sistema. Para la definición de forma más sencilla de los métodos en cada clase es muy factible el apoyo en los diagramas de interacción (colaboración o secuencia).

Es importante destacar que en el sistema a desarrollar al igual que en la mayoría de los software el diagrama de clases del diseño se actualiza, pues aparecen nuevos elementos, a medida que avanza el desarrollo del sistema hasta obtener la versión final que se corresponde con el diseño definitivo del software.

El diagrama de clases del diseño resultante correspondiente al software de graficación, se puede observar en el **anexo 10**.

4.4 Generalidades de la Implementación.

4.4.1 Estilo de Código.

Durante su vida útil, un programa será manejado por muchas manos y observado para agregarle nuevas funcionalidades o con el objetivo de comprender el código, las convenciones de código pueden ayudar a reducir la fragilidad de estos. Adoptar un estándar aporta consistencia, pues las diferentes porciones de código (que pueden ser provenientes de diversos lenguajes) guardan un patrón que se puede reconocer, mejora la legibilidad del código. Permite que otras personas puedan colaborar más eficazmente pues la “redacción” del código no les resulta incomoda y en consecuencia, el mantenimiento del software es menos pesado como resultado de los aportes anteriormente mencionados. (Ver **Anexo 11**)

4.5 Modelo de Implementación.

Un modelo de implementación describe al igual que los elementos del diseño, cómo son las clases, cómo se organizan los componentes de acuerdo al lenguaje de programación utilizado y al entorno de implementación, y muestra las dependencias de los componentes entre sí.

4.5.1 Vista de Gestión del Modelo.

Los paquetes reflejan la arquitectura de alto nivel de un sistema: su descomposición en subsistemas y sus dependencias. Están organizados de manera funcional, siguiendo un cierto principio racional, tal como funcionalidad común, implementación estrechamente relacionada, y un punto de vista común. Una dependencia entre paquetes resume las dependencias entre los contenidos del paquete. (40)

A continuación se muestra el diagrama de paquetes del componente GIPP:

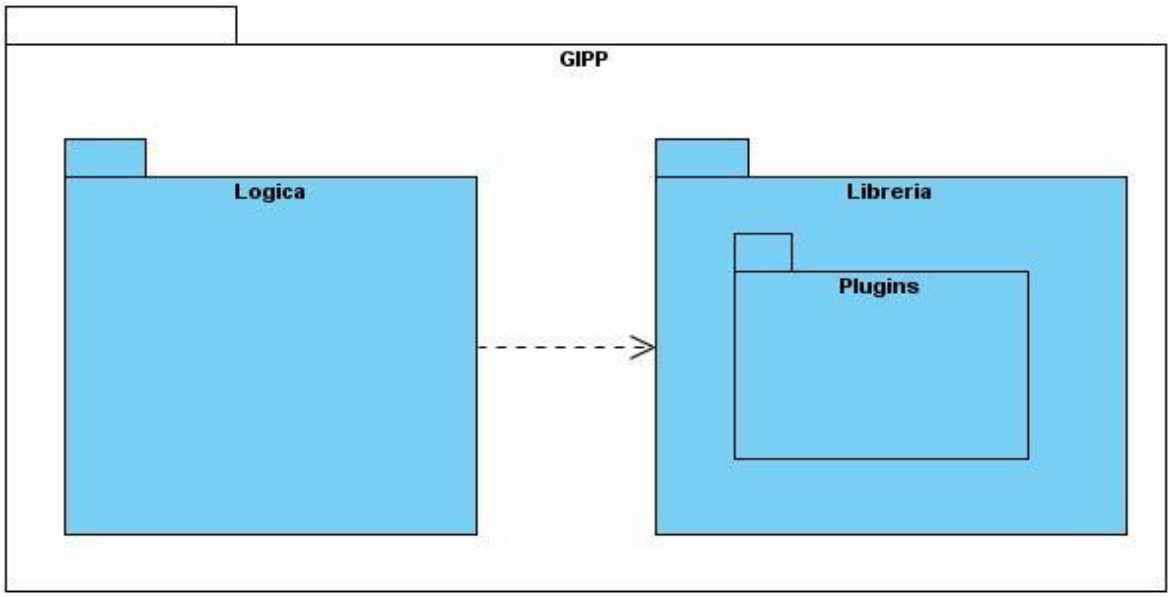


Figura 7. Vista de Gestión del Modelo del componente GIPP.

4.5.2 Diagrama de Componentes.

El Diagrama de Componentes muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. Se utiliza para modelar la vista estática de un sistema y muestra la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. (Ver **Anexo 12**)

4.6 Modelo de Despliegue.

El diagrama de despliegue es un modelo de objetos que describe la topología de la arquitectura física del sistema por medio de la distribución de funcionalidades entre nodos interconectados. Estos nodos y sus enlaces son elementos de hardware sobre los que puede ejecutarse el software, donde cada nodo representa un dispositivo de procesamiento y cada enlace representa los mecanismos de comunicación que se establecen entre dichos nodos, además sobre los diferentes nodos se colocan, a modo de artefactos (elementos físicos simples), los elementos componentes del software.

El diagrama de despliegue del componente GIPP (**figura 12**) representa dos nodos interconectados PC Cliente y Servidor Web, en la PC Cliente se hará uso de la aplicación que mediante el protocolo HTTP se comunicará con el componente GIPP que estará ubicado en el

Servidor Web y que proporcionará una interfaz (**IGraph**) para garantizar el punto de acceso al componente.

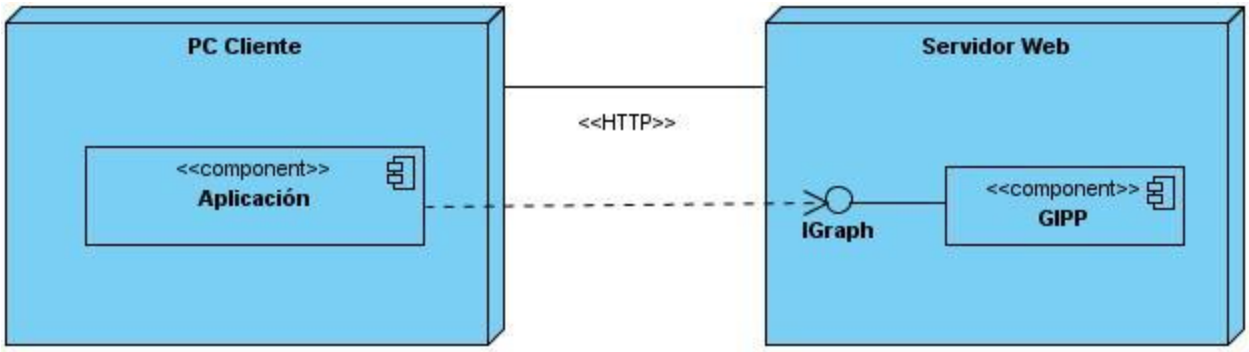


Figura 8. Diagrama de Despliegue.

4.7 Tratamiento de errores.

En la implementación del componente se tuvo en cuenta el tratamiento de errores a través de mensajes de fácil comprensión y lo más descriptivos posible, manteniendo el flujo de información de posibles riesgos o errores cometidos, mostrando además mensajes condicionales.

El tratamiento de errores se aprecia en diferentes clases que tratan los posibles errores a la hora de inicializar los valores de las diferentes variables e instancias, asegurando así, que cada una tome el valor que le corresponde. En caso contrario se captura el error ocurrido.

El tratamiento de excepciones o errores es un detalle considerado en cada acción ejecutada. Se evitan de esta manera operaciones innecesarias y aumenta el tiempo útil disponible por el usuario. Los mensajes de error mostrados son de fácil comprensión para el usuario, pueden observarse cuando:

- Se envían datos a las gráficas que poseen errores.
- No se especifica el indicador correcto.
- En caso de que no se pueda continuar ejecutando la opción deseada porque una secuencia no se pueda realizar, se emite un mensaje de alerta al usuario.

4.8 Pruebas del sistema propuesto.

El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad del software, la prueba es un elemento crítico para ello.

En ese proceso se ejecuta el sistema a probar bajo condiciones específicas y se le aplica un conjunto de estímulos diseñados ingenierilmente, con el objetivo de detectar insatisfacción de los requerimientos planteados (41); todas las actividades y sus resultados son observados y registrados, donde se realiza una evaluación del sistema o componente, para encontrar el número máximo de errores con una cantidad mínima de esfuerzo.

Pruebas de Caja Negra.

Las pruebas de caja negra se concentran en los requisitos funcionales del software; estas técnicas de prueba se concentran en el dominio de la información del software, derivando casos de prueba mediante partición de los dominios de entrada y salida de un programa en forma tal que proporcione cobertura completa. (40)

Objetivo El objetivo de realizarle este tipo de prueba al componente, es para detectar el incorrecto o incompleto funcionamiento de este, comprobar que la entrada se acepta de forma adecuada y que se produce un resultado correcto.

Alcance El proceso de pruebas de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar la interacción con el usuario.

Descripción Estas pruebas permiten evaluar todos los requisitos funcionales del programa.

Casos de Prueba: Para verificar que se cumplieran los requerimientos funcionales establecidos anteriormente, se le realizó la prueba a cada sección contenida en el caso de uso **Generar Gráficas de Indicadores** conformando una interfaz donde se muestra el resultado esperado, esta se realizó con el objetivo de evaluar la interacción del usuario final (Expertos que interactuaran con las gráficas). **(**Ver a partir del **anexo 13)**

Caso de Uso	Generar Gráficas de Indicadores
Caso de Prueba	Graficar Metraje Diario.
Entrada	El usuario selecciona el indicador deseado, y presiona el botón visualizar. **Indicador :** metraje
Resultado Esperado	Gráfica de Metraje Diario
Resultado de la prueba	Al seleccionar el indicador metraje se obtuvo satisfactoriamente el Cronograma de Metraje Diario.
Condiciones	Debe especificarse los datos con los que trabajará el componente, ancho y altura en pixeles.

Tabla 3. Caso de Prueba Graficar Metraje Diario.

Pruebas de Caja Blanca

Las pruebas de caja blanca están dirigidas a las funciones internas del software, se comprueban los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

Objetivos

El objetivo de realizar este tipo de prueba al componente es que se garantice que se ejerciten por lo menos una vez todos los caminos independientes de cada método, todos los bucles en sus límites operacionales, así como las estructuras internas de datos para asegurar su validez.

Alcance.

El proceso de pruebas de caja blanca se va a concentrar principalmente en validar que cada método funcione apropiadamente.

Descripción.

La prueba de Caja Blanca es considerada como uno de los tipos de pruebas más importantes que se le aplican a los programas informáticos, logrando como resultado la disminución de los errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

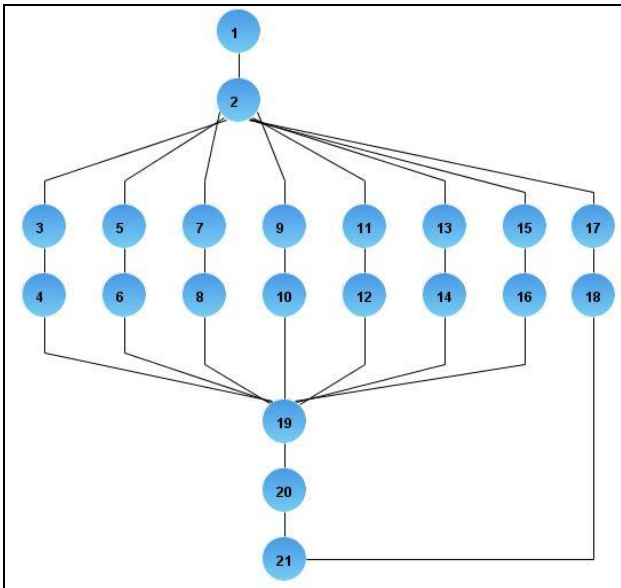
Métrica de la complejidad ciclomática.

Se realizó el cálculo de la complejidad ciclomática a todos los bloques del código dentro del componente, lo cual ayudó a reflejar una medida de la complejidad del código escrito, teniendo en cuenta el número de destinos posibles. Además permitió conocer el esfuerzo a realizar en cada una de las pruebas, el cual es exactamente el valor de la complejidad ciclomática en cada elemento. A partir de esta medida, se diseñaron pruebas que forzaron el recorrido de estos caminos, lo cual garantiza que se ejecute al menos una vez cada sentencia del programa y que cada condición se ejecute en sus variantes verdaderas y falsas.

Ejemplo 1: Método para construir una gráfica según un indicador.

Fragmento de Código

```
ejecutarGraph: function(indicador) {

 var opciones ; 1
   switch (indicador) {  2
      case 'metraje':  3
         opciones =   this._constGraphMetraje();  4
         break; 4

      case 'perforacion':   5
         opciones =  this._constGraphPerforacion();   6
         break;  6

      case 'pvertical':  7
         opciones = this._constGraphPVertical();  8
         break;  8

      case 'phorizontal':   9
         opciones =  this._constGraphPHorizontal();  10
         break;  10

      case 'costodiario':  11
         opciones = this._constGraphCostoDiario();  12
         break;   12

      case 'costoxciento':   13
         opciones = this._constGraphPCostoxCiento();   14
         break;  14

      case 'tiempoxetapa':   15
         opciones = this._constGraphTiempoxEtapa();  16
         break;   16

      default:  17
         throw new Error('Indicador Incorrecto'); 18

   } 19

   return opciones; 20

}  21
```



Complejidad ciclomática V (G)

V(G) = N. de aristas - N. de nodos + 2
V(G) = 27 - 21 + 2
V(G) = 8

Caminos independientes:

1-2-3-4-19-20-21
1-2-5-6-19-20-21
1-2-7-8-19-20-21
1-2-9-10-19-20-21
1-2-11-12-19-20-21
1-2-13-14-19-20-21
1-2-15-16-19-20-21
1-2-17-18-21

Tabla 4 Grafo de flujo para el método que realiza la construcción de una gráfica.

Ejemplo 2: Método para generar una gráfica.

```
generarGraphIndicador: function(indicador,datos,ancho,altura) {

if (ancho == null ||datos == null || altura == null || datos == null) 1
        throw new Error('Parámetros Incorrectos'); 2

  $('.Graph').html(''); 3

  $('.Graph').attr({

     id: indicador

  }); 3

  $('#'+ indicador).attr({

     style: 'width:'+ ancho +'px'+ ';'+'height:'+ altura +'px'

  }); 3

  var generar  = new GeneradorGraph( datos); 3

  var opciones = generar.ejecutarGraph(indicador); 3

  var objdatos = new Datos( datos ); 3
```

```
    var datnumericos ;  4

    if( indicador != 'tiempoxetapa') 4

        datnumericos = objdatos.numericos();  5

    else

        datnumericos = objdatos.datoPastel();  6

$.jqplot(indicador,datnumericos,opciones );  7

} 8
```

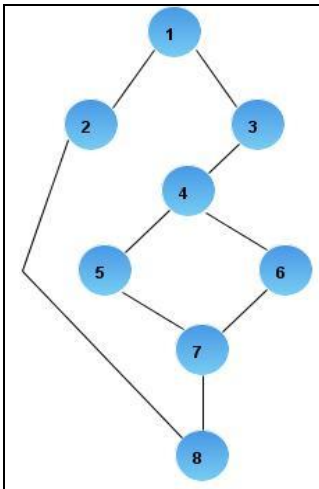
	**Complejidad ciclomática V (G)** V(G) = N. de aristas - N. de nodos + 2 V(G) = 9 - 8 + 2 V(G) = 3 Caminos independientes: 1-2-8 1-3-4-5-7-8 1-3-4-6-7-8

Tabla 5 Grafo de flujo para el método que genera la gráfica.

Casos de pruebas por método analizado.

Se procedió en cada método a calcular la complejidad ciclomática que arrojo para el primer método 8 y 5 respectivamente quiere decir que existen a lo sumo 8 y 5 caminos lógicos para recorrer el algoritmo.

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los **casos de pruebas** para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.

Los ***casos de prueba*** son un conjunto de condiciones o variables bajo las cuales se determina si el requisito de una aplicación es parcial o completamente satisfactorio.

Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:

Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento.

Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Los casos de pruebas realizados para los métodos **ejecutarGraph** (**Anexo 19**) y **generarGraphIndicador** (**Anexo 20**) fueron satisfactorios, pues los resultados obtenidos fueron iguales a los resultados esperados.

4.9 Conclusiones Parciales.

A modo de conclusión se puede decir que en dicho capítulo se representaron aspectos como el diagrama de clases del diseño y los diagramas de colaboración, se describieron los patrones de diseño utilizados y la arquitectura propuesta; se representa el diagrama de despliegue que muestra como están distribuidos los dispositivos de software entre los diferentes nodos, además de exponer el diagrama de componentes.

Se realizaron dos casos de prueba de caja blanca analizando los métodos más importantes en la implementación del componente, estas pruebas guían la calidad del sistema, y determinan si están creadas las condiciones para continuar avanzando en el seguimiento del producto. Además de proporcionar una idea completa del tratamiento de errores y los estándares a utilizar como estrategias de codificación; todos los elementos antes mencionados son de gran importancia para futuras versiones del graficador.

Conclusiones Generales.

La presente investigación recoge todo el proceso de desarrollo del componente, en la cual se realizó un estudio del marco teórico conceptual, tecnologías, tendencias y procesos de desarrollo de software actuales; siendo la base a partir de la cual, permitió obtener una herramienta que arroja resultados positivos para la comprensión de diferentes procesos en la Industria Petrolera, logrando un sistema con flexibilidad de configuración y portabilidad, desarrollado con tecnologías libres que se puede integrar a cualquier aplicación web tomando como área de interés la perforación petrolera.

La aplicación de dicha herramienta es de gran importancia, pues permite disponer de una solución capaz de desarrollar gráficas de indicadores, disminuyendo los costos en la producción de sistemas petroleros, y garantizando la rápida entrega de sistemas y el incremento de la calidad, proporcionando al especialista en pozos petroleros un servicio que sustituirá el trabajo con herramientas ofimáticas para analizar los procesos vinculados a la representación de indicadores de perforación petrolera, ahorrando recursos monetarios al país.

Se puede concluir que la presente investigación aporta una solución factible a la situación problemática que lo originó y que su explotación significará una mejora considerable en la calidad y eficiencia del desarrollo de sistemas petroleros, cumpliéndose de esta forma los objetivos de la investigación.

Recomendaciones.

Teniendo como base los resultados de esta investigación y la experiencia adquirida durante el desarrollo de la misma, se proponen las siguientes recomendaciones:

1. Se recomienda la profundización en el estudio de los procesos de graficación en el área de perforación petrolera, a fin de agilizar la comprensión de las técnicas para graficar en esta área y permitir una refinación de los requisitos del componente desarrollado.

2. Ampliar las funcionalidades del sistema, basándose en otros tipos de gráfica que existen, las cuales responden a detalles específicos de los distintos procesos de la perforación petrolera según sus características.

3. Incluir la solución propuesta en aplicaciones que se desarrollan sobre plataforma web para valorar su eficiencia y efectividad, así como su impacto en la visualización de los datos arrojados en los procesos de perforación petrolera.

Bibliografía Referenciada.

1. **Molinero, Luis.** [En línea] [Citado el: 31 de 10 de 2009.] http://www.seh-lha.org/graficos.htm..

2. Scribd. *Scribd.* [En línea] [Citado el: 31 de 10 de 2009.] http://www.scribd.com/doc/11854611/Graficacion.

3. **Crespo Cevallos, Hugo Andrés.** *Análisis del sistema GEO-Pilot para Perforación dirigida en Pozos Petroleros.* Quito : s.n., Marzo, 2008.

4. [En línea] [Citado el: 03 de 11 de 2009.] http://www.ife.org.mx/documentos/Reforma_Electoral/link_glosario.htm.

5. Petrobras. *Petrobras.* [En línea] [Citado el: 02 de 11 de 2009.] http://www2.petrobras.com.br/EspacoConhecer/esp/SobrePetroleo/ExploracaoProducao_perfuracao.asp..

6. Modelaje de Pozos. *Modelaje de Pozos.* [En línea] [Citado el: 01 de 11 de 2009.] http://modelaje-de-pozos.blogspot.com/search/label/WellFlo.

7. Modelaje de pozos. *Modelaje de pozos.* [En línea] [Citado el: 01 de 11 de 2009.] http://modelaje-de-pozos.blogspot.com/search/label/Software.

8. AddLink. *AddLink.* [En línea] [Citado el: 01 de 11 de 2009.] http://www.addlink.es/productos.asp?pid=430.

9. [En línea] [Citado el: 04 de 11 de 2009.] http://erc.msh.org/readroom/espanol/vocab.htm.

10. **Aguiar Guerra, Randi y Refeca González, Luis Manuel.** *Graficador de datos en el plano para simulador de procesos.* Ciudad de La Habana : s.n., julio, 2008.

11. **Álvarez González, Maikel Hugo.** *Diseño de la base de datos del Sistema de Información de Perforación de Pozos (SIPP).* Ciudad de La Habana : s.n., junio, 2009.

12. **Gayoso, Manuel.** *Desarrollo basado en componentes para sistemas de gestión empresarial.* Montevideo : s.n., Diciembre, 2002.

13. **Arenas Gutiérrez, René.** CECAM. *CECAM.* [En línea] [Citado el: 05 de 11 de 2009.] http://www.cecam.sld.cu/pages/rcim/revista_4/articulos_html/rene.htm#5.

14. **Waserman, Neter.** [En línea] [Citado el: 31 de 10 de 2009.] http://ntics-ntics-estadistica.blogspot.com/2007/08/definiciones.html.

15. **Dürsteler, Juan.** InfoVis.net. *InfoVis.net.* [En línea] 21 de 01 de 2002. [Citado el: 13 de 11 de 2009.] http://www.infovis.net/printMag.php?num=73&lang=1.

16. Unijimpe. *unijimpe.* [En línea] [Citado el: 21 de 11 de 2009.] http://blog.unijimpe.net/jpgraph-graficos-con-php/

17. Internet y Tecnología. [En línea] [Citado el: 22 de 11 de 2009.] http://pixelco.us/blog/10-poderosos-recursos-para-graficar-datos/.

18. Downloads Portal. Downloads Portal. [En línea] [Citado el: 22 de 11 de 2009.] http://spanish.downloads-portal.com/programacion/librerias-y-componentes/fusioncharts-free_application-52558.html

19. **Hamar, Vanessa.** *Aspectos metodológicos del desarrollo y reutilización de componentes de software.* 2004.

20. **Montilva, Jonás.** *Aspectos metodológicos del desarrollo de componentes de software reutilizable.* 2004.

21. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. 84-7829-036-2.

22. Agile UP. [En línea] [Citado el: 11 de 03 de 2010.] http://cgi.una.ac.cr/AUP/html/overview.html.

23. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* **Letelier Penadés, Patricio.** España : s.n., 2006.

24. **Mendoza Sanchez., María A.** *Metodologías De Desarrollo De Software.* Peru : s.n., 2004.

25. Tecnologías. *Tecnologías.* [En línea] [Citado el: 25 de 11 de 2009.] http://www.geopalm.cl/tecnologia/uml.html .

26. **ANACHE, I. y JOEL, M.** *OMG UML 2.0 Marcando un hito en el desarrollo de software.* Habana : s.n., 2005.

27. IBM. *IBM.* [En línea] [Citado el: 26 de 11 de 2009.] http://www-142.ibm.com/software/products/es/es/enterprise.

28. [En línea] [Citado el: 30 de 11 de 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

29. [En línea] [Citado el: 01 de 12 de 2009.] http://stedwin.wordpress.com/2009/10/20/informacion-util-y-de-interes/.

30. Softonic. *Softonic.* [En línea] [Citado el: 10 de 12 de 2009.] http://netbeans-ide.softonic.com/.

31. **Brandendaugh, Jerry.** *Programación de aplicaciones Javascript.* Madrid : 84-415-1070-9, 2000.

32. **Potencier, Fabien y Zaninotto, François.** *Symfony la guía definitiva.* 2008.

33. *Estamos en Línea.* [En línea] [Citado el: 20 de 02 de 2010.] http://www.estamosenlinea.com.ve/2009/12/17/sun-microsystems-presenta-netbeans-ide-6-8/.

34. *VivaPhp.* [En línea] [Citado el: 19 de 02 de 2010.] http://vivaphp.com.ar/frameworks/symfony-para-netbeans.

35. *REDHAT.* [En línea] [Citado el: 20 de 02 de 2010.] https://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/es-ES/Deployment_Guide/s2-httpd-v2-features.html.

36. **Mariñán, Pérez Martín.** *Patrones de Diseño.*

37. **Larman, Craig.** *UML y Patrones,Introducción al análisis y diseño orientado a objetos.* Mexico : s.n., 1999. 970-17-0261-1.

38. **Eguíluz Pérez, Javier.** LibrosWeb. *LibrosWeb.* [En línea] [Citado el: 01 de 03 de 2010.] http://www.librosweb.es/ajax.

39. **Software Engineering Institute, Carnegie Mellon.** [Online] 2007. [Cited: 04 05, 2010.] http://www.sei.cmu.edu/architecture/start/moderndefs.cfm.

40. **S. Pressman, Roger.** *Ingeniaría de Software. Un enfoque práctico.* Madrid : Quinta Edición, 2001.

41. e-Quallity. [En línea] [Citado el: 02 de 05 de 2010.] http://www.e-quallity.net/definiciones.php.

Bibliografía Consultada.

1. **Ciudad Ricardo, Febe Angel.** *¿CÓMO CONFECCIONAR UN ESTADO DEL ARTE?* Ciudad de La Habana : s.n., 2008.

2. **Hernández León, Rolando Alfredo y Coello González, Sayda.** *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de La Habana : EDUNIV, 2002. 959-16-0343-6.

3. **Gómez Franco, José Lino y Roa, Tatiana.** repsol. *repsol.* [En línea] 199. http://www.canariasdicenoarepsol.org/monitoreo/Curso%20industria%20petrolera.pdf.

4. infoil. *infoil.* [En línea] http://www.infoil.com.ar/productsInfoProd.html.

5. **Puentes, Eredio.** *Introducción a la Industria Petrolera Cubana.* Ciudad de La Habana : s.n., 2009.

6. **Barberii, Efraín.** *El Pozo Ilustrado.* Caracas : FONCIED, septiembre, 1998.

7. *Planificación de Pozos Direccionales.* [Presentación en power point] s.l. : Schlumberger, Febrero, 2008.

8. Weatherford. *Weatherford.* [En línea] http://www.inter-log.com/pros_integrateddata.html.

9. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. 84-7829-036-2.

10. **Ciudad Ricardo, Ángel Febe y Yosnel, Herrera Martínez.** *DoMet COMO PROPUESTA PARA LA MODELACIÓN DE ENTORNOS ORGANIZACIONALES COMPLEJOS Y DIFUSOS.* Ciudad de La Habana : s.n., 2006.

11. Jqplot. *Jqplot.* [En línea] http://www.jqplot.com/.

Glosario de Términos.

CAD: Estilo para el tratamiento de datos gráficos.

CEINPET: Centro de Investigaciones de Petróleo.

CUPET: Unión Cubana del Petróleo (Cuba-Petróleo), empresa nacional dedicada a las labores relacionadas con los hidrocarburos y sus derivados.

DISWIN: Proporciona capacidades de envío completas a todas las empresas de transporte por camión clasificadas. DISWIN es el software para maximizar la productividad y hacer ahorrar tiempo y el dinero.

DST: Genera informes relacionados con la Dirección de Servicios Tecnológicos.

DIPP: Dirección de Intervención y Perforación de Pozos.

EPEPC: Empresa de Perforación y Extracción de Petróleo de Centro.

Estadística: Por estadística se entiende una batería de recursos científicos por los cuales se puede recolectar, organizar, resumir, presentar y analizar datos numéricos de un conjunto de observaciones.

Gráficos: Gráficos, en informática, es el nombre dado a cualquier imagen generada por un ordenador.

HTTP: Protocolo de Transferencia de Hipertexto en inglés (Hypertext Transfer Protocol), es un protocolo de comunicación que permite la interacción entre los servidores y el navegador, para intercambiar archivos (texto, gráfica, imágenes, sonido, video y otros archivos de multimedia).

JavaFX ™: Familia de productos y tecnologías de Sun Microsystems para la creación de Aplicaciones de Internet Enriquecidas (RIAs), aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas.

SIPP: Sistema de Información de Perforación de Pozos Petroleros.

MWD: Measurement while drilling. Midiendo mientras se perfora.

PetroSoft: Polo Informático para el desarrollo de soluciones informáticas para la industria del petróleo.

Qt: Es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores.

GTK+: GTK+ o The GIMP Toolkit es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME, XFCE y ROX aunque también se puede usar en el escritorio de Windows, MacOS y otros.

RDQ: Índices RQD (*Rock Quality Designation*) se define como el porcentaje de recuperación de testigos de más de 10 cm de longitud (en su eje) sin tener en cuenta las roturas frescas del proceso de perforación respecto de la longitud total del sondeo. Además saber que para determinar el RQD en el campo o zona de estudio de una operación minera, existen tres procedimientos de cálculo.

Schlumberger: Compañía de servicios de yacimiento petrolífero que entrega una variedad de servicios y soluciones a la industria de petróleo internacional.

T V D: Total vertical deep. Profundidad Vertical Total.
WITSML: Well-Site Information Transfer Standard Markup Language. Lenguaje para marcar normas para transferir información del sitio del pozo.

XMI: XMI es el nombre que recibe el estándar para el intercambio de metamodelos usando XML.