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

TÍTULO: Desarrollo de componentes de interfaz de usuario para la representación de información mediante gráficas en el Polo PetroSoft

AUTORES

Maridalia Pérez Vázquez

Guillermo William Rodríguez Hidalgo

TUTOR

Ing. Maykel López Oliva

Ciudad de La Habana, 2010

Año del 51 aniversario de la Revolución

El valor de un hombre no se determina por lo que posee, ni aún por lo que hace, sino que está directamente expresado por lo que es él mismo.

Henri Frédéric Amiel

Dedicatoria

Le dedicamos este trabajo de diploma al Espíritu del Dios vivo.

Agradecimientos

Maridalia Pérez Vázquez:

Al Espíritu Santo por haber sido mi guía, maestro, consejero, Padre eterno, Dios fuerte, Príncipe de paz durante todo este tiempo.

A mis padres Joseíto y Ofelia por apoyarme siempre, y por darme la posibilidad de sentirme cuando estoy con ellos en sensación de hogar.

A mi novio Angel por su amor, cuidado y comprensión. Por ser el mejor regalo de Dios para mí.

A mis hermanas Cosito y Arelis, porque han sido mis segundas mamás, por estar pendientes a mis necesidades, por ser incondicionales. Verdaderamente si Dios me hubiera dado la oportunidad de escoger mis hermanas las escogería a ustedes dos porque son las mejores hermanas del mundo.

A mi hermana Marielena y a mis hermanos José Miguel y Ariel por apoyarme. Son los mejores que Dios me pudo haber dado.

A mis sobrinos por quererme tanto, y por compartir conmigo tantos momentos lindos de mi vida.

A mis amigas Cristina y Maribel por su confianza, por tener yo el privilegio de poder llamarme su amiga.

A mi compañero de tesis Guillermo por su paciencia conmigo, por saber trabajar en equipo a mi lado. Verdaderamente ha sido una bendición hacer la tesis unidos.

A mis hermanos de la UCI, son maravillosos. Por haber sido canales de bendición para mi vida durante todo este tiempo.

A Yadrián, mi hermano en Cristo, por habernos ayudado incondicionalmente. También a Roig por sus instrucciones.

A nuestro tutor Maykel López por haber contado conmigo para desarrollar este trabajo de diploma.

A la oponente Kizzy y al presidente del tribunal David por sus correcciones, porque nos fue de gran ayuda.

A todos los profesores y compañeros de la universidad que me han ayudado y brindado su apoyo.

Guillermo William Rodríguez Hidalgo:

Quiero agradecer al Espíritu Santo de Dios, porque obra de sus manos soy yo y por Él existo. Porque durante la realización del trabajo las veces que temí, me dio confianza, y cuando todo estaba oscuro, hizo que resplandeciera a mi alrededor. Por hacer especial cada día al despertar, dándole sentido y motivación a mi vida y renovar mis fuerzas para que prosiguiera adelante. Por ordenar mis pasos y mis pensamientos, y darme el entendimiento que necesité, pues toda ciencia y sabiduría viene de Él. Por amarme tanto, tanto, tanto. Y por moldearme de día en día con mucha paciencia y misericordia, para hacerme un mejor hombre que sirva a esta generación. Gracias mi Señor Jesús, por todo esto y por todo aquello que sabes que está en mi corazón.

Quiero agradecer además:

A mi madre, por su sobrado amor para conmigo. Por no detenerse sino valiente traspasar caminos largos unas veces y llenos de fango otras, para acalorar mi corazón con el fuego de amor de madre que había en el suyo. Por quitarse aún lo suyo, y hacerse débil para que yo estuviese fuerte y siguiera adelante en la vida y en mis estudios. A Dios le doy gracias por darme la mejor madre del mundo.

A mi papá por quien él mismo es para mí, por el privilegio de cada instante pasado a su lado. Por cada uno de sus consejos buenos. Y porque de él he aprendido a ser mejor y he tomado para ser un mejor padre en el futuro.

A mi padrastro Higinio, por hacerme como hijo suyo, y brindarme tanto su ayuda, aún sacrificándose por caminos difíciles.

A mis hermanos Yury, Dennis, Odennis, Yurenia y Beatriz, porque su vidas también inspiran la mía, y su cariño ha sido importante para mí.

A mis abuelos Armando (aunque ya no esté conmigo), Aldo, Arminda, Oneida y Julia además, por quererme tan especialmente. Por esperar lo mejor de mí.

A mi familia en general, por su confianza en mí lo cual me alentó, y porque el simple hecho de saber que me quieren de la manera que sé, ya alegra mi vida y me ayuda también a seguir adelante.

A mi compañera de tesis, por todo el tiempo de trabajo que juntos enfrentamos. Por ser tan responsable, trabajadora, optimista, y buena compañera de equipo. Por cada promesa de Dios que compartió conmigo, y cada consejo para ayudarme.

A mis amigos Ergly y Alexander por cada tiempo vivido juntos, cada aventura por toda Cuba, y por el privilegio de ver a Cristo a través de sus vidas, su carácter y manera de enfrentar cada circunstancia.

A mis hermanos en Jesucristo. Gracias por sus oraciones a Dios por mí, y por preocuparse por mí. Especialmente quiero agradecer a mi hermano Yadrián, por toda la ayuda brindada y por estar siempre presto ante las dudas referentes al desarrollo del producto.

Al tutor Maykel por en un inicio ayudarnos a identificar el problema científico sobre el cual podríamos desarrollar el trabajo de diploma, y por cada una de sus sugerencias a lo largo de todo el desarrollo de la tesis.

A la oponente Kizzy, por ser tan exquisita en corregir los errores del documento en cada uno de los cortes, para así finalizar con resultados correctos. Y al presidente David, por cada uno de los errores que nos señaló igualmente, pero para afinarnos a nosotros mismos como defensores del trabajo de diploma.

A la universidad, al país y a sus gobernantes por ser los medios por medio de los cuales Dios permitió que adquiriera los conocimientos para formarme como ingeniero en las ciencias informáticas.

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Dirección de Calidad de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

_____ _____

Firma del Autor Firma del Autor

Maridalia Pérez Vázquez **Guillermo William Rodríguez Hidalgo**

_____

Firma del Tutor

Maykel López Oliva

Resumen

Actualmente en la UCI y dentro de ella el Polo Productivo PetroSoft, genera un alto número de software relacionados con la industria del petróleo, pero teniendo en cuenta que en el mercado mundial muchos son los software que se realizan para esta área se hace compleja la tarea de buscar características que hagan que el producto de la UCI tenga un brillo en este mercado.

Una de estas características es la visualización de la información, basados en que las empresas petroleras generan cientos de miles de datos de muy distintas fuentes como la exploración, producción, procesamiento, transporte, refinación y medio ambiente, cada una con cantidades de procesos y formatos.

Este trabajo de diploma va dirigido a modelar, diseñar e implementar componentes de Interfaz de Usuario (IU) para graficar la información generada por las empresas petroleras. Las funcionalidades propuestas en este trabajo van a permitir al Polo PetroSoft contar con componentes estandarizados en la representación de información, homogeneizando las soluciones informáticas producidas por el polo, además de facilitar el trabajo a los desarrolladores de IU del mismo.

Palabras Claves

Visualización, Información, Componente, Interfaz, Metodología.

Índice

Introducción

Nuestro país ha experimentado un desarrollo acelerado en los últimos años en la esfera informática; así lo demuestra la creación de nuevos centros y planes de estudio como es el caso de la Universidad de las Ciencias Informáticas (UCI) y los Institutos Politécnicos de Informática (IPI), por solo mencionar algunos. Entre los principales objetivos que han propiciado esta iniciativa, se encuentran la informatización de la sociedad y la producción y exportación de software aportando así significativamente también al desarrollo de la economía.

El Polo de Soluciones Informáticas para la Industria Petrolera (PetroSoft), tiene a su cargo la creación de software para la industria del petróleo, pero teniendo en cuenta que determinar cuál es el que mejor se ajusta a las necesidades de esta industria, posee un elevado grado de dificultad, aún más, cuando existen en el mercado diversos software que cumplen funciones similares, se trabaja en aras de buscar características que hagan que el producto de la UCI tenga un realce en este mercado.

La industria petrolera realiza actividades típicamente integradas en forma vertical, lo que quiere decir que, todas las actividades económicas relacionadas con la explotación del petróleo, se suelen desarrollar bajo la dirección de una misma empresa; se supone luego, que el volumen de información que hay detrás de cada software desarrollado es bastante elevado; por lo que la Visualización de Información (VI) es una característica a tener en cuenta para mejorar los productos realizados en el Polo.

El término de Visualización de Información se relaciona con el diseño de representaciones visuales interactivas que permitan al usuario una rápida asimilación y comprensión de un volumen grande de información.

Por los motivos antes expuestos, se hace muy importante la actualización constante de las prácticas de ingeniería así como el estudio de los nuevos paradigmas que internacionalmente surgen y se aplican.

La creciente demanda de los sistemas computacionales actuales así como las exigencias en cuanto a factores como el aumento de la calidad y la reducción de los plazos de desarrollo de los mismos, son las causas principales de la búsqueda de la reutilización del software existente. El Desarrollo de Software Basado en Componentes (DSBC) trae aparejado diversos beneficios tales como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión; debido a que se

organiza el desarrollo a partir de elementos pre-elaborados que desarrollan alguna funcionalidad común y que pueden ser reutilizados una y otra vez.

Con el compromiso de desarrollar soluciones informáticas para la industria petrolera, hace poco más de un año y medio, por el Polo Productivo PetroSoft de la Facultad 9, emergen necesidades de homogeneizar procesos de iguales características para dichas soluciones. Con la puesta en marcha de más de diez productos informáticos contraídos con la Empresa Nacional Cuba Petróleo (CUPET) se distinguen similitudes en las soluciones informáticas independientemente del área a la que va dirigida. En la práctica, los proyectos de software son llevados a cabo por diferentes equipos multidisciplinarios, que en la mayoría de los casos poseen pobre comunicación entre ellos para desarrollar procesos de similares características. Esta situación provoca deficiencias en el proceso ingenieril del Polo, duplicando esfuerzo, degradando su calidad, poniendo en riesgo fechas de entrega pactadas con los clientes, divergencias de las soluciones con un mismo fin y por ende aumento en los costos del producto. Siguiendo esta vertiente, es perceptible esta situación en la capa de presentación de los productos, donde para similares objetivos de representación de información son aplicadas disímiles técnicas, menoscabando la identidad de las soluciones informáticas producidas por el Polo, siendo esta la **situación problemática** a la cual nos enfrentamos.

A partir de esto identificamos como **problema de la investigación**: ¿cómo facilitar la representación de información mediante gráficas para los desarrolladores de IU del Polo PetroSoft?

Debido a lo antes expuesto se hace urgente el desarrollo de componentes de IU que se utilicen para graficar la información que se genera en la Industria del Petróleo; específicamente en las áreas de la misma (Exploración, Perforación, Producción, Refinación, Comercialización), lo cual produce gran flujo de datos. De esta manera, el presente trabajo surge como una necesidad del Polo de poseer un producto propio y personalizado.

Para dar solución al problema fue trazado como **objetivo general**: desarrollar componentes básicos para la representación de información mediante gráficas en diferentes áreas de la Industria petrolera.

Según los análisis antes realizados se pretende dar entrada a la realización de componentes, que de acuerdo al concepto de reutilización, que plantea que los componentes se diseñan y desarrollan con el objetivo de poder ser reutilizados en otras aplicaciones, reduciendo el tiempo de desarrollo, mejorando la fiabilidad del producto final antes visto puedan estos ser usados en cualquier contexto distinto a aquel para el que fue diseñado. Componentes que tienen como objetivo final la representación de información mediante gráficas, que estandaricen y den una visión más clara de los diferentes indicadores a evaluar, así como el estado que presenten durante todo el proceso de producción de

petróleo. Teniendo en cuenta lo planteado con anterioridad, se definió como **objeto de estudio** la representación de información de la Industria petrolera mediante gráficas en las áreas de Exploración, Perforación, Producción, Refinación y Comercialización, estableciendo el **campo de acción** en la implementación de componentes de IU para el Grupo de Aplicación y Presentación del Polo PetroSoft. Como **idea a defender** se ha establecido que con la implementación de componentes de IU para la representación de información mediante gráficas se logrará facilitar la aplicación de los mismos en los productos del Polo PetroSoft, ganando en tiempo y homogeneización.

Para el cumplimiento del objetivo de este trabajo de diploma se ha propuesto desarrollar las siguientes **tareas de investigación**:

1. Identificar y caracterizar los diferentes tipos de gráficas que son utilizadas en las cinco áreas fundamentales de la Industria petrolera (Exploración, Perforación, Producción, Refinación y Comercialización).
2. Identificar componentes y librerías existentes, que brinden soluciones potenciales en la representación de información mediante gráficas.
3. Establecer los métodos, herramientas y procedimiento más factibles para el desarrollo de componentes de SW.
4. Desarrollar el producto de SW.

El principal resultado esperado de este trabajo, es la obtención de componentes de IU que generen gráficas para la representación de información de la industria petrolera en las áreas de Exploración, Perforación, Producción, Refinación y Comercialización. Con esto se logrará el aumento de la calidad de la producción del Polo PetroSoft, además de estandarizar los procesos de representación de información, y así lograr la identidad de las soluciones informáticas producidas por el Polo.

Para identificar y caracterizar los diferentes tipos de gráficas que son utilizadas en las cinco áreas fundamentales de la Industria petrolera se han empleado diferentes **métodos científicos de investigación**, tanto teóricos como empíricos.

Los métodos teóricos utilizados durante la investigación fueron el analítico - sintético, el análisis histórico - lógico y el inductivo - deductivo.

El método **analítico - sintético** con el objetivo de llegar a conocer, mediante el análisis de las diferentes teorías y documentos encontrados a lo largo de la presente investigación, los rasgos que caracterizan y distinguen cada una de las áreas de la Industria del petróleo, para la posterior selección de las gráficas de mayor impacto.

El **análisis histórico - lógico** para realizar un estudio analítico de la trayectoria histórica del objeto de estudio de la presente investigación y de cómo este ha influido históricamente en los procesos industriales de empresas petroleras; revisando de forma crítica cada uno de los documentos involucrados en la misma para poder profundizar en la importancia que tiene el mismo en la toma de decisiones y su utilidad en empresas petroleras de Cuba y el mundo.

Con el método **inductivo - deductivo** para arribar a conocimientos generalizadores de cada uno de los contenidos teóricos a profundizar en la presente investigación, a partir de los aspectos particulares y generales encontrados en los documentos involucrados en la presente indagación, analizándolos tanto de lo particular a lo general como viceversa.

La **modelación** es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. Se desarrollarán componentes de IU con el objetivo de representar la información de forma gráfica.

De los **métodos empíricos**, la **entrevista** con el objetivo de obtener información valiosa y familiarizarnos así con las peculiaridades y características de cada una de las cinco áreas que recorre el proceso de producción de petróleo, a partir de conversaciones planificadas entre los investigadores y personal competente ya sean especialistas pertenecientes a algún centro petrolero, o cualquier otro personal dentro de la universidad que conozca sobre el tema.

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

Capítulo 1. Fundamentación teórica: se expone el estado del arte del objeto de estudio de la presente investigación y se definen los elementos teóricos que lo sustentan. Se enuncian conceptos que posibilitan un mejor entendimiento de lo planteado en la situación problemática y el marco del problema en sentido general. Se enuncian y argumentan otras aplicaciones ya existentes que pueden dar solución de alguna manera al problema científico del presente trabajo.

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

Capítulo 3. Presentación de la solución propuesta: se expone de manera general cómo estarán modelados los componentes. Se exponen algunos de los principales artefactos generados en el flujo Modelación, como los requerimientos funcionales y no funcionales, el Diagrama de Casos de Uso del Sistema y la descripción textual de los mismos.

Capítulo 4. Construcción de la solución propuesta: se describe la construcción de los componentes que se propone. Se exponen algunos de los principales artefactos generados en los Flujos de Trabajo Modelación, Implementación, Prueba y Despliegue necesarios en el desarrollo del software que dará solución al problema científico de la presente investigación. Algunos de los artefactos encontrados son los diagramas de clases del diseño, los diagramas de componentes y de despliegue, las características generales de la implementación, el plan de pruebas y los casos de pruebas diseñados para aplicar al sistema desarrollado.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

Actualmente en la Industria del petróleo se hace uso de imágenes gráficas para la representación de información en las diferentes etapas que componen esta industria. En el presente capítulo se especifican algunos conceptos asociados a la investigación, además se definen los tipos de información, que ayudarán a la clasificación de los datos en las áreas más representativas de esta extensa industria. A su vez, para llevar a cabo la graficación de la información es necesario utilizar librerías gráficas de componentes, por lo que también se identifican y caracterizan las mismas con el objetivo de seleccionar en el próximo capítulo, la más útil. A partir de todos estos elementos se desarrolla la investigación, haciéndose más sencilla la comprensión de los temas que serán abordados en lo adelante.

1.2 Conceptos asociados al dominio del problema

A continuación se enuncian una serie de conceptos que harán más fácil el entendimiento del problema.

1.2.1 Visualización de la información (VI)

Siendo la VI una disciplina bastante novedosa, en la medida que van surgiendo productos que requieren de su utilización, aumentan el número de investigadores interesados en este campo y se publican nuevas definiciones. A continuación se muestra un ejemplo:

- Representar visualmente espacios y estructuras de información que faciliten una rápida asimilación y comprensión, y la posibilidad de identificar y extraer patrones a partir de una gran cantidad de información, incrementando el valor. (Zhu and Chen 2005)

1.2.2 Tipologías de Visualización de Información

La visualización aparece en varias formas y es usada en varios campos. Los diferentes tipos de visualización en ocasiones son identificados por el tipo de datos que muestran, qué representación visual utilizan o en qué áreas de aplicación son usados. Entre los diferentes tipos de visualización se encuentran:

- Pictogramas

- Mapas
- Cartogramas
- Histogramas
- Gráficos
    - Gráficos de barra
    - Gráficos de líneas
    - Gráficos circulares
    - Gráficos de áreas
    - Gráficos de superficie
    - Gráficos de anillos
    - Gráficos de dispersión
    - Gráficos de burbujas
    - Gráficos radiales
- Redes neuronales artificiales (RNA)

Pictogramas

También llamada gráfica de imágenes o pictografía. Es un diagrama que utiliza imágenes o símbolos para mostrar datos para una rápida comprensión. La mayor frecuencia se identifica por la mayor acumulación de símbolos y es más funcional y natural que cualquier otro sistema. (eumednet)

Figura 1: Pictogramas del sistema señalético de las Olimpiadas de Pekín 2008 (estonova.com)

Mapas

Los mapas se utilizan para representar una región de la Tierra en un plano. En ellos podemos representar diferentes características, como el clima, la flora, la población, etc. (labrandero 2007)

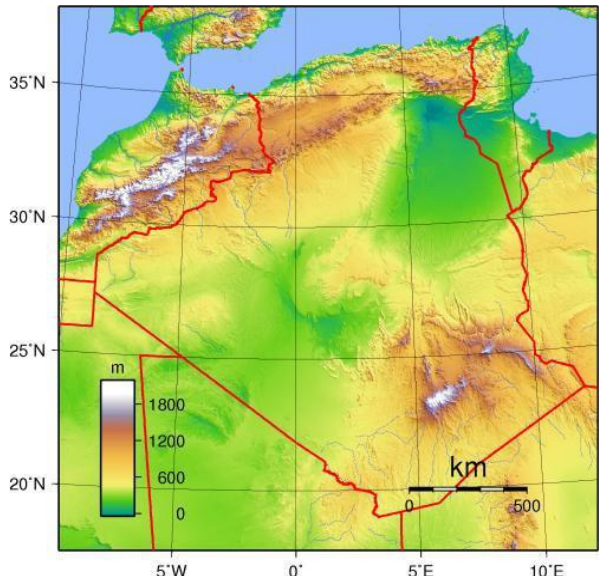


Figura 2: Mapa topográfico de Argelia (vmapas.com)

Cartogramas

Un cartograma es un tipo de gráfico, semejante a un mapa, en el que los límites geográficos y el área que contienen están distorsionados en función del valor de una cierta variable georreferenciada (Georreferenciada: una variable que se puede asociar a coordenadas geográficas) que se quiere representar. (Dürsteler 2008)

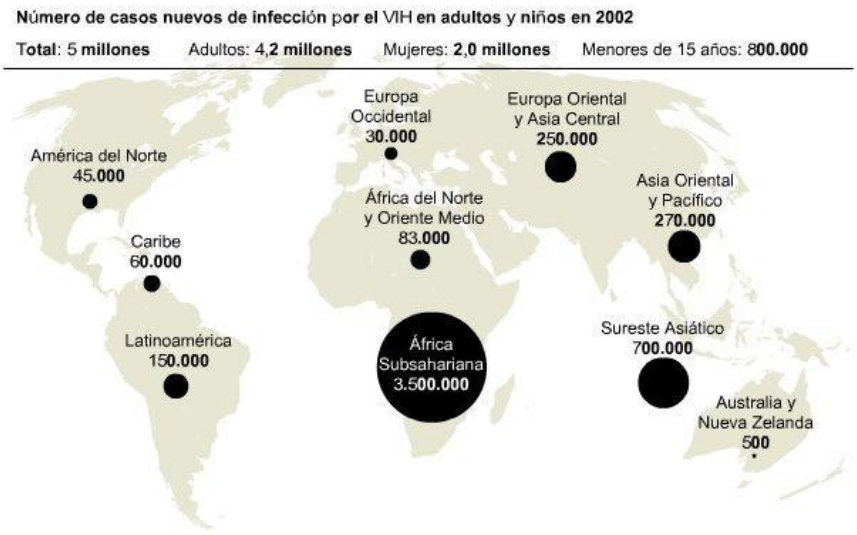


Figura 3: Cartograma con puntos (www.ua.es 2006)

Histogramas

Es un resumen gráfico de los valores producidos por las variaciones de una determinada característica, representando la frecuencia con que se presentan distintas categorías dentro de dicho conjunto.

Es una gráfica de la distribución de un conjunto de medidas. Un histograma es un tipo especial de gráfica de barras que despliega la variabilidad dentro de un proceso. Toma datos variables y despliega su distribución. Los patrones inusuales o sospechosos pueden indicar que un proceso necesita investigación para determinar su grado de estabilidad. (Calidad 2000)

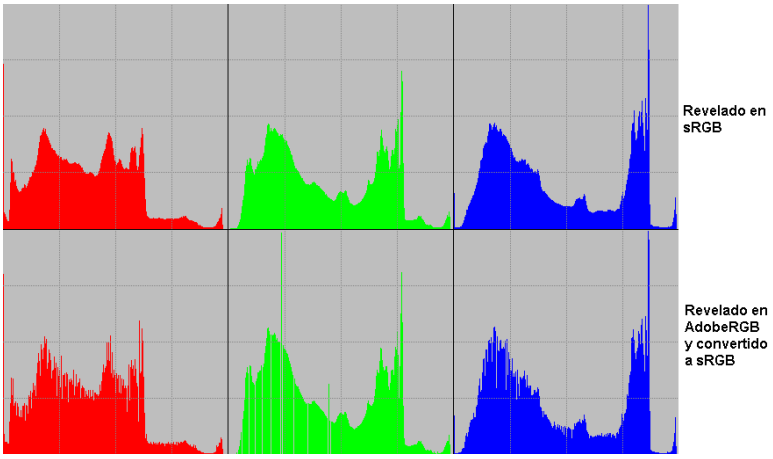


Figura 4: Histograma (img236.imageshack.us)

Gráficos de Barras

Representan valores usando trazos verticales o horizontales, aislados o no unos de otros, según la variable a graficar, si es discreta o continua. Este tipo de gráfico es útil para mostrar cambios de datos en un período de tiempo o para ilustrar comparaciones entre elementos. (Colegio 2007)

Se pueden trazar datos que se organizan en columnas o filas de una hoja de cálculo en un gráfico de barras. Los gráficos de barras muestran comparaciones entre elementos individuales. (Office)

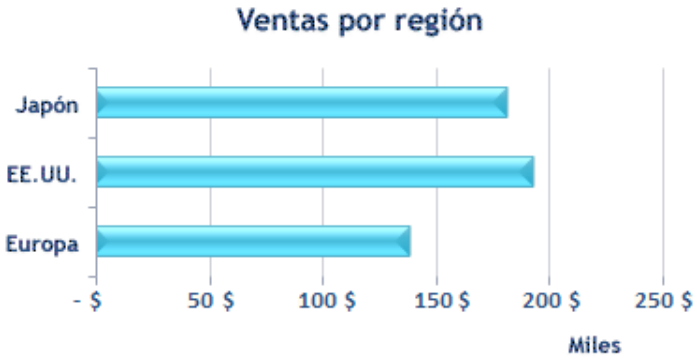


Figura 5: Gráfico de barras horizontales en 3D (Microsoft)

Gráficos de Líneas

Los gráficos de líneas muestran una serie como un conjunto de puntos conectados mediante una sola línea. Los gráficos de líneas se usan para representar grandes cantidades de datos que tienen lugar durante un período continuado de tiempo. (Microsoft 2009)

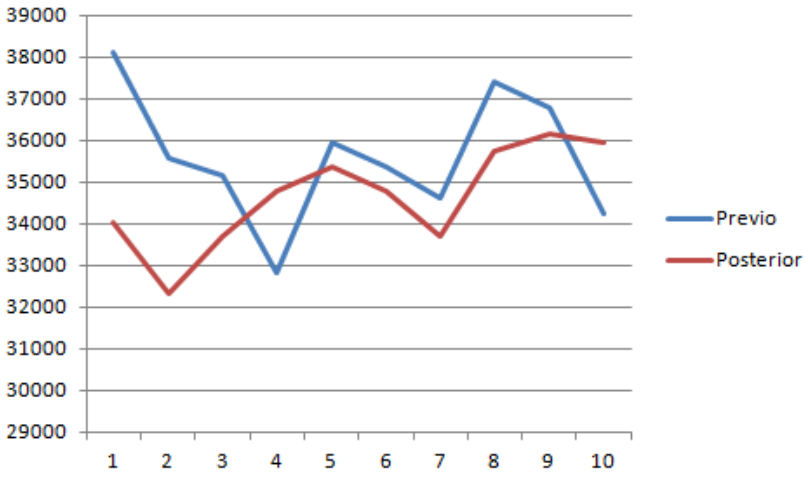


Figura 6: Gráfico de Línea (Ferri)

Gráficos Circulares

Estos gráficos, también conocidos como gráficos de Tartas (pastel) o Sectores, muestran los datos como un círculo dividido en secciones de colores o diseños, se divide un círculo en tantas porciones como clases tenga la variable, de modo que a cada clase le corresponde un arco de círculo proporcional a su frecuencia absoluta o relativa. (Díaz and Fernández 2001)

Figura 7: Gráfico Circular en 3D (Microsoft)

Gráficos de áreas

Un gráfico de área muestra sus datos como áreas llenas de colores o diseños y destacan la magnitud del cambio en el tiempo utilizándose para llamar la atención hacia el valor total en una tendencia. Por

ejemplo, se pueden trazar los datos que representan el beneficio en el tiempo en un gráfico de área para destacar el beneficio total. (Microsoft 2009)

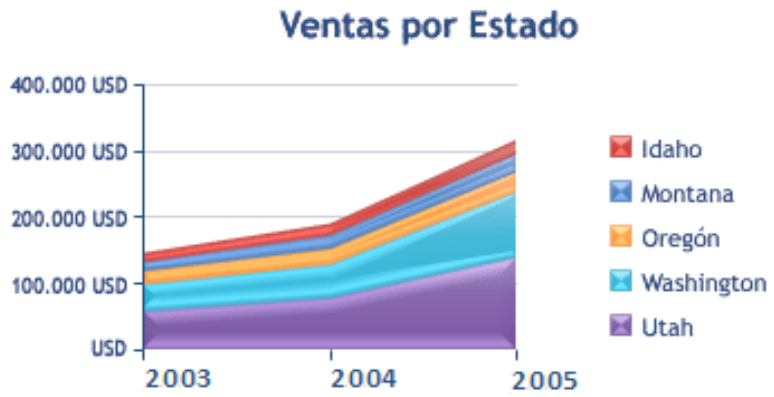


Figura 8: Gráfico de área (Microsoft)

Gráficos de superficie

Un gráfico de superficie es útil cuando busca combinaciones óptimas entre dos conjuntos de datos. Como en un mapa topográfico, los colores y las tramas indican áreas que están en el mismo rango de valores. (Scribd 2008)

Puede utilizar un gráfico de superficie cuando ambas categorías y series de datos sean valores numéricos o para ver la evolución de un dato sujeto a tres variables. (Martin 2002)

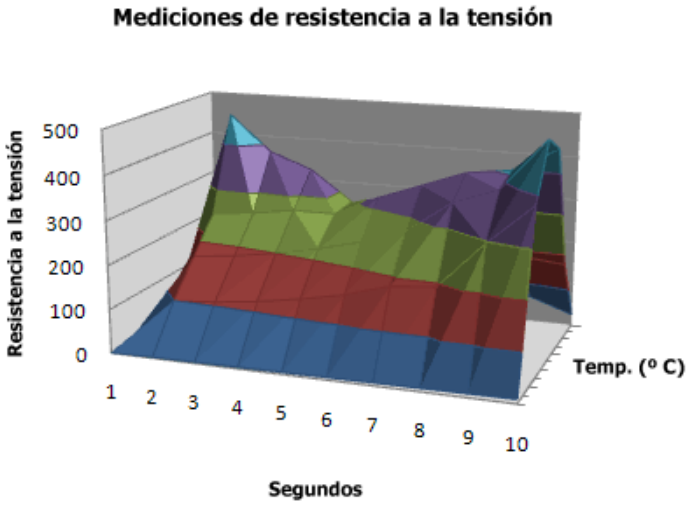


Figura 9: Gráfico de Superficie (Microsoft)

Gráficos de anillos

Los gráficos de anillos muestran los datos de los valores en formato de porcentajes de un total. Las categorías se representan mediante sectores individuales. Los gráficos de anillos suelen utilizarse para mostrar porcentajes. Son idénticos en cuanto a funciones a los gráficos circulares. (Microsoft 2008)

Un gráfico de anillos muestra grupos de categorías, grupos de series y series de valores como sectores. El tamaño de los sectores viene determinado por el valor de la serie como un porcentaje del total de todos los valores. (Microsoft 2008)

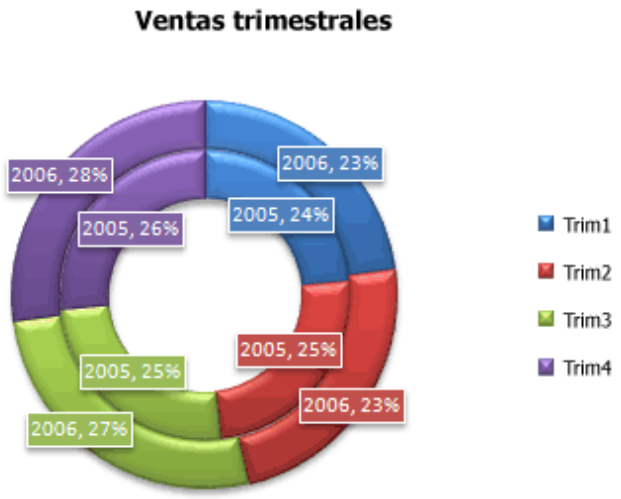


Figura 10: Gráfico de Anillo (Microsoft)

Gráficos de dispersión o tipo XY

Los gráficos de dispersión muestran la relación entre los valores numéricos de varias series de datos o trazan dos grupos de números como una serie de coordenadas XY. (Microsoft 2008)

Un gráfico de dispersión tiene dos ejes de valores y muestra un conjunto de datos numéricos en el eje horizontal (eje X) y otro en el eje vertical (eje Y). Combina estos valores en puntos de datos únicos y los muestra en intervalos irregulares o agrupaciones. (Microsoft 2008)

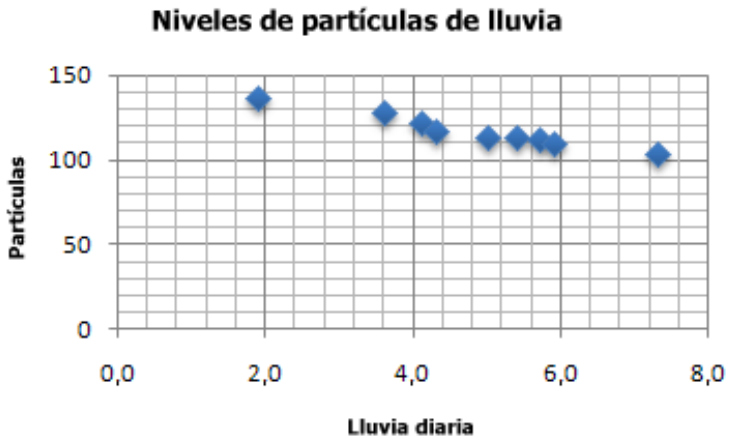


Figura 11: Gráfico de dispersión con solo marcadores (Microsoft)

Gráficos de burbujas

El gráfico de burbujas presenta los datos como una serie de burbujas, donde el tamaño de las burbujas está en proporción a la cantidad de datos. (Crystal Decisions 2003)

Es una variación de un gráfico de dispersión en el que los puntos de datos (puntos de datos: valores individuales trazados en un gráfico y representados con barras, columnas, líneas, sectores, puntos y otras formas denominadas marcadores de datos. Los marcadores de datos del mismo color constituyen una serie de datos), se reemplazan por burbujas, y el tamaño de las burbujas representa una dimensión adicional de los datos. (Microsoft 2007)

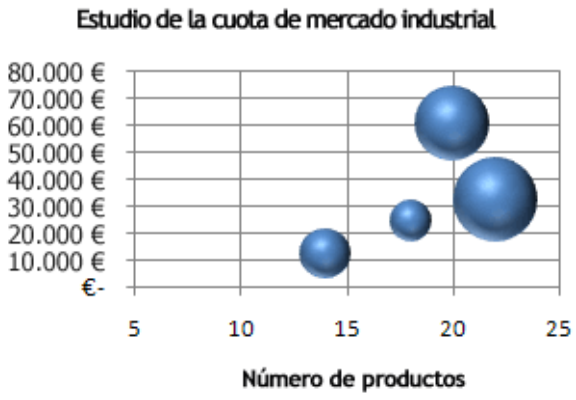


Figura 12: Gráfico de burbujas (Microsoft)

Gráficos radiales

Un gráfico radial, también conocido como gráfico de araña o de estrella a causa de su aspecto, representa los valores de cada categoría a lo largo de un eje independiente que se inicia en el centro del gráfico y finaliza en el anillo exterior.

Los gráficos radiales comparan los valores agregados de varias series de datos.

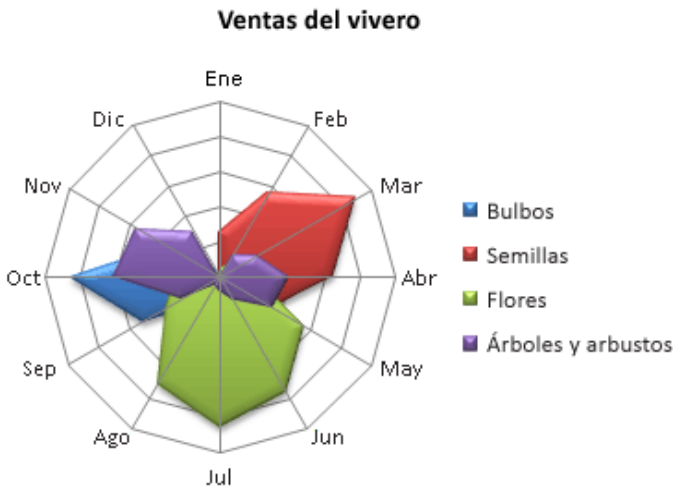


Figura 13: Gráfico radial relleno (Microsoft)

Redes Neuronales Artificiales (RNA)

En las últimas décadas las Redes Neuronales Artificiales (RNA) han recibido un interés particular como una tecnología para minería de datos, puesto que ofrece los medios para modelar de manera efectiva y eficiente problemas grandes y complejos. Los modelos de RNA son dirigidos a partir de los datos, es decir, son capaces de encontrar relaciones (patrones) de forma inductiva por medio de los algoritmos de aprendizaje basado en los datos existentes, más que requerir la ayuda de un modelador para especificar la forma funcional y sus interacciones. (Salas 2007)

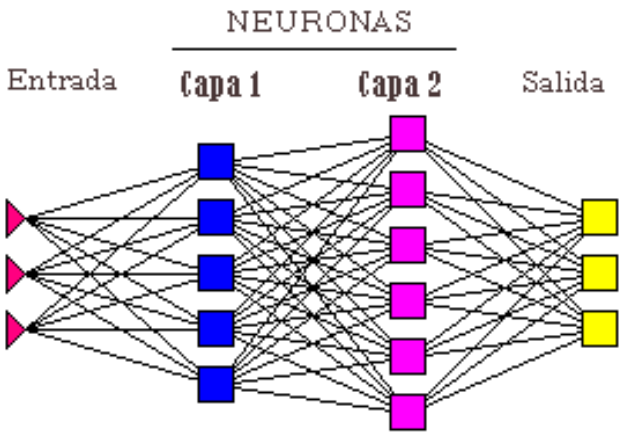


Figura 14: Esquema de una RNA de dos capas de neuronas intermedias

1.3 Visualizaciones más utilizadas en las áreas de la industria de petróleo

En investigaciones llevadas a cabo por especialistas e ingenieros en esta rama, se han analizado un conjunto de software, que fueron seleccionados en correspondencia con cada una de las áreas de la Industria del petróleo a través de un muestreo no probabilístico accidental, con el fin de analizar las visualizaciones de información que los mismos utilizan. Basándose en ello se definen las visualizaciones más utilizadas en cada una de las áreas de esta industria por separado.

1.3.1 Exploración

- Las visualizaciones más abundantes utilizadas en la actualidad en el área, según los estudios realizados (Gráficos de Superficie en 3D, Gráficos de Área en 3D, Histogramas, Gráficos de línea, Gráfico de dispersión, Mapas y Cartogramas), donde cada una de ellas tiene un papel fundamental y logran cumplimentar sus objetivos.
- Los Gráficos de Área en 2D: La evolución del tiempo determina en gran medida los factores geológicos, de ahí la propuesta de utilizar estas visualizaciones, ya que suelen usarse para mostrar la magnitud del cambio de una variable en el tiempo y donde finalmente pueden verse sus partes como un todo. Un ejemplo de los datos que pudieran representarse en ellos pueden ser: cantidad de yacimientos descubiertos, cantidad de yacimientos por descubrir, porciento de sustancias inorgánicas encontradas, entre otros de este tipo de característica.

- Los Gráficos de Barra: Estos gráficos pudieran tener un uso similar al de los gráficos de áreas en 2D, teniendo como característica fundamental, que suelen ser fáciles de comprender por el usuario. Son muy usados para representar información estadística, donde puede observarse la evolución de una determinada característica en el tiempo permitiendo hacer comparaciones. Además en ellos se pueden mostrar varias series de datos siendo representadas con variedades de colores. (Cairó 2009)

1.3.2 Perforación

En el área de Perforación quedan propuestas las siguientes visualizaciones para los software desarrollados en el Polo PetroSoft relacionados con la Perforación Petrolera:

- Las visualizaciones más abundantes utilizadas en la actualidad en el área según los estudios realizados (Gráficos de área, Gráficos de líneas, Histogramas, Gráficos de Superficie en 3D, Gráficos de Áreas en 3D y Mapas), pudiéndose constatar la efectividad de cada una de ellas, además de lograr la mejor comprensión de la información por parte del usuario como objetivo fundamental.
- Los Cartogramas: Su semejanza con los mapas le confieren a los cartogramas gran importancia, es por ello que se proponen dentro de las visualizaciones a utilizar en esta área. Tienen la facilidad de poder mostrarle a los usuarios la localización de una región determinada y mostrar datos sobre dicha base geográfica. Pueden ser ideales para mostrar determinados aspectos que ayudarían a la selección de los equipos de perforación, como son: rango de profundidades del pozo, tamaño de los agujeros y revestidores, dimensiones de la subestructura: altura y espacio libre inferior, entre otros.
- Los Gráficos de Burbujas: Teniendo en cuenta que en el área se hace un intenso trabajo con estimaciones, análisis de costos y datos financieros, se propone el uso de estas visualizaciones. Son ideales en la representación de datos estadísticos y pueden resultar fáciles de comprender por el usuario. Su mayor utilidad pudiera estar en la planificación de los pozos, donde se analizan factores como: análisis de presión de poros, selección de profundidad de asentamiento de la tubería, planeamiento del completamiento del pozo, estimación de costos, entre otros.

- Los Gráficos de Barras: Estos gráficos pudieran utilizarse indistintamente con los gráficos de burbujas, los cuales tienen además la posibilidad de poder mostrar distintas series de datos estableciéndose fácil comparaciones entre ellos, haciendo uso de diferentes colores. (Cairó 2009)

1.3.3 Producción

Se proponen el uso de las siguientes visualizaciones para los Software desarrollados en el Polo PetroSoft relacionados con el área:

- Las visualizaciones más abundantes utilizadas en la actualidad en el área según los estudios realizados por investigaciones (Gráficos de barras, Gráficos circulares, Gráficos de líneas, Gráficos de dispersión, Gráficos de áreas en 2D y Cartogramas), ya que las mismas logran facilitar la comprensión de la información por parte de los usuarios, siendo éste el motivo que conlleva a su uso.
- Los Gráficos de Líneas en combinación con los Histogramas: La combinación de los Histogramas con los Gráficos de Líneas resulta una visualización capaz de mostrar perfectas comparaciones entre diferentes series de datos, donde además se puede observar la variabilidad de las características en el tiempo y son fáciles de comprender por los usuarios. Uno de los ejemplos que pudieran mencionarse es la representación de la producción de acuerdo a las reservas consideradas.
- Los Histogramas: Estos gráficos son un tipo especial de gráficos de barras, lo que hace que su uso sea bastante favorable para representar información estadística, siendo muy fácil de entender. (Cairó 2009)

1.3.4 Refinación

En la presente área se proponen el uso de las siguientes visualizaciones para los Software desarrollados en el Polo PetroSoft relacionados con la Refinación Petrolera:

- Las visualizaciones más abundantes utilizadas en la actualidad en el área según el estudio realizado (Histogramas, Gráficos de líneas, Gráficos de barras comparativas, Gráficos circulares y Gráficos de burbujas), las cuales logran transmitirle la información a los usuarios de una forma muy amena.

- Los Gráficos de Dispersión: Estos gráficos resultan muy exitosos cuando se desea mostrar similitudes entre cantidades de datos y realizar comparaciones entre ellos. Quedan propuestos para su uso teniendo en cuenta que pudieran utilizarse en las comparaciones que se establecen en el área, referentes a la cantidad de determinadas sustancias que tiene el crudo en su estado natural.
- Los Cartogramas: Teniendo en cuenta la importancia que se le ha dado a dicha visualización en las áreas anteriores, se considera que la misma resulta ser una herramienta poderosa para la comprensión de determinados datos en la empresa, teniendo en cuenta que brinda la posibilidad de la localización de una región además de una característica de la misma que se desee representar. En la Refinación un cartograma puede ser muy útil para representar datos como: localizaciones de las refinerías en funcionamiento, localizaciones de las refinerías que no están en funcionamiento, principales productos que se obtienen en determinadas refinerías, entre otros. (Cairó 2009)

1.3.5 Comercialización

En la presente área se propone:

- ✓ Mantener el uso de las visualizaciones comunes que se utilizan actualmente (Gráficos de barra, Gráficos circulares, Gráficos de burbujas, Gráficos de dispersión y Cartogramas), teniendo en cuenta que las mismas son ideales para representar grandes volúmenes de datos estadísticos y financieros; partiendo del principal objetivo de la comercialización que es precisamente la rentabilidad de la empresa. (Cairó 2009)

1.4 Conceptos asociados al Componente de software

Uno de los enfoques en los que actualmente se trabaja constituye lo que se conoce como Desarrollo de Software Basado en Componentes (DSBC), que trata de sentar las bases para el diseño y desarrollo de aplicaciones distribuidas basadas en componentes de software reutilizables. Dicha disciplina cuenta actualmente con un creciente interés, tanto desde el punto de vista académico como desde el industrial, en donde la demanda de estos temas es cada día mayor. El desarrollo de software basado en componentes permite reutilizar piezas de código preelaborado que permiten realizar diversas tareas, conllevando a diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión.

1.4.1 Componente de software

- En esencia, un componente es una pieza de código preelaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Son los "ingredientes de las aplicaciones" que se juntan y combinan para llevar a cabo una tarea. (Terreros 2010)
- Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio. (Educativa 2009)

Como conclusión de estas definiciones un componente de software equivale a partes de software que se pueden combinar con otros componentes para generar un conjunto aún mayor tales como otro componente, subsistema o sistema. Un componente define una o más interfaces desde donde se puede tener acceso a los servicios que este ofrece a los demás componentes. Un componente puede presentarse en forma de código fuente o código objeto; puede estar escrito en un lenguaje funcional, procedural o en un lenguaje orientado a objetos; y puede ser tan simple como un botón GUI (Graphical User Interface) o tan complejo como un subsistema.

Por otro lado, existe también una confusión habitual entre el concepto de componente con el de clase, objeto, módulo, entre otros. Con el fin de distinguirlas entre sí, se recogen las definiciones más aceptadas de otras entidades software, así como sus principales diferencias.

Clase: Las clases no hacen referencia explícita a sus dependencias y requisitos. Las clases suelen construirse mediante herencia de implementación, y por tanto no suelen permitir una instanciación e instalación por separado de la clase base y de sus clases hijas. Esto puede causar problemas si se realiza composición tardía (pudiendo aparecer, por tanto, el problema de la clase base frágil).

Módulo: Un módulo es un conjunto de clases, junto con opcionalmente otros elementos no orientados a objetos, como pueden ser procedimientos y funciones.

Paquetes/Librerías: Un paquete es un conjunto de clases, usualmente agrupadas conceptualmente. Los paquetes no suelen ser ejecutables, y pueden ser consideradas como la versión orientada a objetos de las librerías tradicionales.

Subsistema: Un subsistema es una agrupación de elementos de modelado que representa una unidad de comportamiento en un sistema físico. De acuerdo a la especificación de UML 1.3, un subsistema es una subclase de Classifier y de Package, y por tanto no tiene un comportamiento propio, sino que organiza elementos ejecutables que lo poseen (por ejemplo, componentes).

Recurso: Un recurso es una colección no modificable de elementos con tipo.

Framework: Los marcos de trabajo suelen estar compuestos de componentes, de los cuales unos están fijados por el propio marco, y otros son los que proporciona el usuario para especializar el marco de trabajo. (Fuentes, Troya et al.)

Sin embargo más allá de su definición existen algunas características claves para que un elemento pueda ser catalogado como componente:

– *Identificable:* Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.

– *Auto contenido:* Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.

– *Puede ser remplazado por otro componente:* Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.

– *Con acceso solamente a través de su interfaz:* Debe asegurar que estas no cambiarán a lo largo de su implementación.

– *Sus servicios no varían:* Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.

– *Bien Documentado:* Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.

– *Es genérico:* Sus servicios debe servir para varias aplicaciones.

– *Reutilizado dinámicamente:* Puede ser cargado en tiempo de ejecución en una aplicación.

– *Independiente de la plataforma:* Hardware, Software, Sistema Operativo. (Rojas and García 2004)

1.4.2 Beneficios de los componentes

El primer beneficio es que con componentes los desarrolladores minimizan el tiempo de desarrollo al reducir trabajo ya hecho. La fiabilidad del sistema es mayor.

Se mejora la calidad, ya que un componente de software desarrollado para ser reutilizado es más propenso a ser libre de errores, lo que asegura la calidad y fiabilidad del mismo. (Pressman 2002)

Se mejora la productividad ya que las aplicaciones creadas a partir de componentes reusables requieren menor tiempo para el análisis, diseño y codificación consiguiendo la funcionalidad requerida por el usuario, pero con menos esfuerzo. (Stepanek 2005)

1.5 Interfaz de Usuario (IU)

Con el objetivo de entender el tipo de componente que en este trabajo de diploma de propone desarrollar, se expresa la definición de IU.

La Real Academia Española define el término interfaz (del inglés interface, superficie de contacto) como una conexión física y funcional entre dos aparatos o sistemas independientes. Generalizando esta definición, dados dos sistemas cualesquiera que se deben comunicar entre ellos la interfaz será el mecanismo, entorno o herramienta que hace posible dicha comunicación.

Cuando uno de los sistemas que se comunican es un ser humano se pasa al concepto de interfaz de usuario. Por un lado se tiene un sistema físico o informático y por otro a una persona que desea interaccionar con él, darle instrucciones concretas, siendo la IU la herramienta que entiende a ambos y es capaz de traducir los mensajes que se intercambian. (Moreno 2005)

Los componentes que se desarrollarán son de IU porque los mismos se crearán bajo el concepto de que a través de ellos el desarrollador pueda crear interfaces gráficas, con las cuales el usuario pueda interaccionar con el sistema, modificando y actualizando las gráficas que visualizan datos referentes a las cinco principales áreas de la Industria del petróleo, ya antes mencionadas.

1.6 Análisis de librerías para gráficos

Para que se puedan generar las gráficas pertinentes es necesario hacer uso de una librería que grafique las más utilizadas en la industria petrolera. A continuación se realiza un estudio exhaustivo de librerías que brindan gran potencialidad, con el objetivo de seleccionar la más efectiva para llevar a cabo el desarrollo de los componentes.

Se puede decir que empresas como SunMicrosystems se han encargado de mejorar y dar continuidad al desarrollo de bibliotecas con funcionalidades gráficas enteramente codificadas en Java. Estas han facilitado el trabajo de los desarrolladores a la hora de comenzar proyectos desde cero, donde haya que crear interfaces gráficas. Entre el grupo de librerías que se utilizan en Java para crear Interfaces Gráficas de usuarios podemos identificar las siguientes:

1.6.1 JFreeChart

JFreeChart es una librería para gráficos escrita 100% en Java, que facilita la muestra gráficos de calidad profesional en aplicaciones ya sean web o de escritorio. Entre las características principales de esta biblioteca se tienen las siguientes:

- ✓ Se pone a disposición la documentación API (Interfaz de Programación de Aplicaciones) de la librería. Esta es consistente y bien documentada, con soporte para un amplio rango de tipos de gráficas.
- ✓ Entre la gama de gráficos que genera JFreeChart se incluyen: gráficas de pastel (2D y 3D), gráficas de barras (regular y apiladas, con un efecto 3D opcional), gráficas de líneas, de burbujas, de dispersión y de área, combinación de gráficas, diagramas de Pareto, diagramas de Gantt, tablas y gráficos de símbolos métricos, gráficos de mapa de obleas, y series de tiempo.
- ✓ Posee un diseño flexible fácilmente extensible, y la posibilidad de ser usado tanto en tecnologías de servidor (aplicaciones Web) y de cliente (Swing, por ejemplo).
- ✓ Soporte para varios tipos de salida, incluyendo componentes Swing, archivos de imagen como PNG y JPEG, y formatos gráficos de vectores (incluyendo PDF, EPS y SVG).
- ✓ JFreeChart es Open Source, más específicamente, software libre. Se distribuye bajo los términos de la GNU Lesser General Public Licence (LGPL), que permite su uso en aplicaciones propietarias.
- ✓ El código y la especificación de la misma es gratuita, sin embargo, la guía de desarrollador de JFreeChart, así como el acceso a algunos ejemplos asociados, se pueden adquirir pagando. (Gallego 2009)
- ✓ Como requerimientos JFreeChart requiere la plataforma Java 2 (JDK versión 1.3 o posterior). (JFree.org 2009)

Es importante destacar que la librería JFreeChart tiene dependencia de otro proyecto de código abierto, el proyecto JCommon.

JCommon, es una colección de clases de utilidad, usadas tanto por JFreeChart como por JFreeReport, y en general para cualquier tipo de aplicación Java. Esta librería incorpora una gran cantidad de clases, que pueden ser muy útiles para el desarrollo de aplicaciones.

1.6.2 RChart

RChart es un componente de Java muy conocido que ha estado disponible desde el 2000 y ha sido portado también a .NET y lenguajes de programación PHP. Su última versión es RChart 2.1. RChart es muy popular debido a su sencillez, utilidad y flexibilidad. Flexible porque se puede utilizar en muchos escenarios, como applets, servlets y JSP, en los controles de Java (Swing y SWT) o dispositivos móviles (J2ME con MIDP / LWUIT y Android). En su paquete de descarga se incluyen librería y applets

de java para crear gráficos estadísticos e incluirlos en aplicaciones Java y páginas Web de una manera sencilla y cómoda. (Java4less 2000)

Entre las características principales de esta biblioteca se tienen las siguientes:

- ✓ Con RChart se pueden crear distintos tipos de gráficos así como combinar estos. Entre ellos encontramos: gráficas de líneas 2D y 3D, de área, gráficas circulares 2D y 3D, de burbuja, Gauge Circular, gráficas de barras 2D y 3D, de barras apiladas, gráficas de candlestick, de punto, de radar, de eventos, B-Splines (aproximación), mínimos cuadrados, y la funcionalidad de Servlets incluido en su código fuente.
- ✓ Puede configurar casi todas las características, tipo de letra, color, estilo de línea, imágenes y estructura.
- ✓ Algunas opciones avanzadas son: exportar a GIF, JPG, PNG, leer el gráfico de una URL, Integración con JavaScript, líneas con anchura y estilo, posición de la leyenda, datos "real time" (datos actualizables), fechas automáticas en ejes, zoom y scroll, escalas automáticas, entre otras.
- ✓ Puede llamar a funciones JavaScript (o abrir una página HTML) cuando el usuario hace click sobre un punto o barra.
- ✓ Compatible con navegadores antiguos. Además es compatible con JDK 1.1 y 1.2. y existe una licencia especial que incluye el código fuente de la misma.
- ✓ Además tiene precios a partir de unos 35 Euros (unos 40 dólares), y se puede adquirir el código fuente ya que el acuerdo de licencia no es restrictivo.
- ✓ Entre las limitaciones de la versión shareware es que posee el texto de aviso junto a la leyenda, así como que tiene limitación en el número de elementos que se dibujan.
- ✓ Licencia: Shareware - Versión de Prueba.
- ✓ *RChart Visual Builder* es una herramienta que acelera el desarrollo de sus gráficas.(Java4less 2000)

1.6.3 TeeChart

TeeChart para Java es una extensa biblioteca gráfica para desarrolladores de Java de licencia comercial y disponible para cualquier plataforma por lo menos en sus últimas versiones. Es extremadamente portátil y puede utilizarse en todos los entornos de programación Java estándar. Entre sus principales características se enuncian las siguientes: (Steema 2010)

- ✓ TeeChart ofrece literalmente docenas de tipos de gráficos. Utiliza los formatos preparados o mezclados y combina los tipos de datos de la serie. La mayoría de los gráficos se pueden exhibir en 2D y 3D. Utiliza el Editor Gráfico para agregar y configurar múltiples ejes o añadir en tiempo de ejecución. Tiene prácticamente una cantidad ilimitada de ejes que pueden añadirse y ser configurados de forma independiente.
- ✓ Proporciona completo, rápido y fácil uso de gráficos en tecnologías .NET, Java, ActiveX / COM (Component Object Model), PHP y Delphi VCL (Visual Component Library) para negocios, aplicaciones financieras y científicas.
- ✓ TeeChart para Java soporta los principales entornos de desarrollo Java, incluyendo CodeGear JBuilder, IBM Eclipse, NetBeans Sun, IntelliJ IDEA y Oracle JDeveloper.
- ✓ Posee herramientas de código libre para ofrecer anotación, de bandas de color con transparencia, líneas personalizadas, numeración de las páginas, el cursor del travesaño, consejos mouseOver, fondo de la pantalla de la imagen, rotación de gráfico, arrastre el punto de información y punto más cercano.
- ✓ Por otro lado TeeChart puede exportar como un archivo, o copiar en el portapapeles en cualquiera de los formatos descritos en el resumen de características. En formato nativo TeeChart puede usarse para crear diseños con el Editor Gráfico del valor de plantilla de base de datos o almacenamiento de archivos y la importación en tiempo de ejecución.
- ✓ El archivo de ayuda y documentación incluye una guía de referencia, una guía de usuario, tutoriales detallados con explicaciones, una ayuda de usuario y un ejemplo de proyecto integral con el código completo.

Las últimas versiones de TeeChart, entre las cuales la más reciente es la anunciada el 4 de Febrero del 2010, cuando se anunció una versión de TeeChart para Java 2.0. Estas últimas actualizaciones están disponibles con gastos de envío en línea con un precio para Europa de $10 dólares, mientras que para el resto del mundo era de $ 18 dólares.

La Licencia de desarrollador es de $ 439 o su equivalente € 321.42. A la vez que la concesión de licencias para educación, entre los que figuran académicos y organizaciones no lucrativas (organismos no gubernamentales) pueden solicitar un descuento del 50% en una compra directa. Las licencias educativas no deben ser utilizadas con fines comerciales y se limitan a uso exclusivamente académico.

1.6.4 JetChart

JetChart es una biblioteca de clases gráficas que abarca una amplia gama de funcionalidades destinadas a la visualización de datos y análisis, en forma de diferentes tipos de gráficos en Java. Existen varios tipos de gráficos de Java que son compatibles, como gráficos de líneas, gráficos de barras, gráficos circulares, gráficos de dispersión, gráficos financieros, gráficos de series de tiempo, entre otros.

Características como el zoom, la interacción del usuario por medio de hacer clic o arrastrando los puntos de datos, información sobre herramientas, escalas múltiples, gráficos de apilamiento, y otros, están disponibles con sólo unas pocas líneas de código o configuración de los parámetros de los subprogramas de gráfico.

JetChart es totalmente compatible con JDK 1.1.x y las versiones más recientes, y apoya la translucidez de color y el antialiasing usando un adaptador de Java 2, por lo que es posible desarrollar gráficos de Java con buena visibilidad y no irregulares líneas.

La librería gráfica de Java puede ser fácilmente integrada en las aplicaciones y applets de éxitos, y también soporta GIF, JPEG, PNG y SVG como codificación de las imágenes de gráfico desde aplicaciones que no sean visibles. (Oiticica 2009)

Especificaciones de la Biblioteca

Fecha de salida: 17/9/2003

Sistema Operativo: Windows 95, Windows Me, Windows 98, Windows NT, Windows XP, Windows 2000.

Requisitos adicionales: JVM 1.1.7.

Modelo de Licencia: Gratis o Libre para probar (Free to try).

Precio: $160.00

1.7 Conclusiones

Con el desarrollo de este capítulo se ha logrado definir de forma clara todos los aspectos teóricos conceptuales, necesarios para la realización de la investigación propuesta. También se han analizado los conceptos fundamentales referentes al desarrollo basado en componentes de software, así como los beneficios que estos presentan.

En la descripción hecha de las diferentes librerías para gráficos, se considera que cada una de ellas ofrece características potenciales, pero solo una es libre. Esto se ha hecho con el objetivo de seleccionar la más conveniente para generar las gráficas de los componentes a desarrollar.

A partir del resultado del estudio plasmado en este capítulo, se puede proceder a seleccionar las tecnologías necesarias para el desarrollo de los componentes de software para graficar.

Capítulo 2: Tendencias y tecnologías actuales

2.1 Introducción

En la actualidad, a nivel mundial existe gran variedad tecnológica que brinda excelentes posibilidades para que grandes y pequeñas empresas puedan dar soluciones efectivas a sus necesidades. En este capítulo se hace un análisis de las tendencias y tecnologías actuales a considerar para ser usadas en el desarrollo de los componentes que dará solución al problema científico de la presente investigación. Con el fin de adoptar las mejores opciones para cumplir el objetivo general de este trabajo de diploma son analizadas las metodologías de desarrollo de software más representativas, el lenguaje de programación a utilizar y otros aspectos referentes a la tecnologías actuales que son de gran importancia en esta toma de decisiones. Luego del análisis de estos elementos, se hace indispensable establecer cuáles de estos serán empleados. Se hace una exposición y fundamentación de la tecnología escogida para modelar, implementar y garantizar el correcto funcionamiento de los componentes.

2.2 Metodologías de desarrollo de software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, entre otros.). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, entre otros. (Computación. 2004)
La comparación y/o clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. A grandes rasgos, si tomamos como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, podemos clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. Por otra parte, considerando su filosofía de desarrollo, las metodologías se clasifican en Ágiles y Tradicionales (Pesadas). A continuación se explica brevemente cada una de estas categorías de metodologías.

2.2.1 Metodologías ágiles

Un proceso es ágil cuando el desarrollo de software es incremental, es decir, entregas pequeñas de software, con ciclos rápidos; cooperativo, lo que implica que el cliente y los desarrolladores trabajan

juntos constantemente con una cercana comunicación; sencillo, cuando el método en sí mismo es fácil de aprender y modificar, bien documentado; y adaptable, lo que permite realizar cambios de último momento. (Computación. 2004)

En resumen metodologías, denominadas Metodologías Ágiles, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

2.2.1.1 Proceso Unificado Ágil (AUP)

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés, es una versión simplificada del Proceso Unificado de Rational (RUP). Describe en una forma simple, fácil de entender y brinda un enfoque de desarrollo de software utilizando técnicas ágiles y conceptos del RUP. El proceso unificado (Unified Process o UP) es un marco de desarrollo de software iterativo e incremental. (Flores and Cordero)

La Fig. 35 muestra el ciclo de vida de esta metodología. AUP abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente. El modelado agrupa los tres primeros flujos de RUP (Modelamiento del negocio, Requerimientos y Análisis y Diseño). Dispone de cuatro fases igual que RUP: Creación, Elaboración, Construcción y Transición.

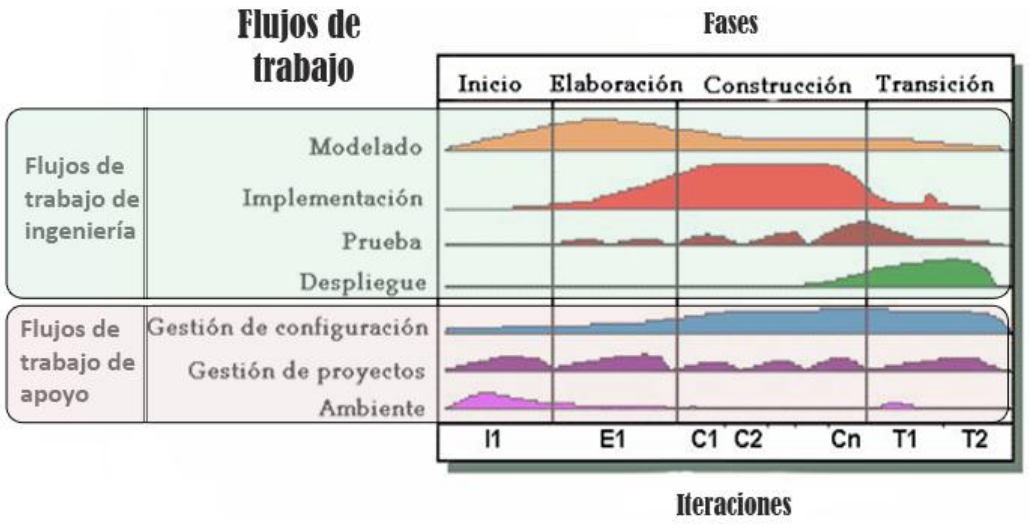


Figura 15: Esquema que muestra los flujos de trabajo y las fases de AUP

El ciclo de vida de AUP, se puede ver en sus cuatro fases, y su naturaleza iterativa se expresa en sus siete disciplinas que son:

Modelado: Comprender el negocio de la organización, comprender el dominio del problema abordado por el proyecto, e identificar una solución al mismo que sea viable.

Implementación: Transformar el modelo realizado en código ejecutable y realizar pruebas de nivel básico, en particular pruebas unitarias.

Prueba: Realizar una evaluación objetiva para asegurar la calidad. Esto incluye buscar defectos, validar que el sistema funcione como debería, y verificar que se cumplen los requerimientos.

Despliegue: Planificar la liberación del sistema.

Gestión de configuración: Administrar el acceso a los artefactos del proyecto.

Gestión de proyectos: Dirigir las actividades que forman parte del proyecto.

Ambiente: Facilitar todo el entorno que permita el normal desarrollo del proyecto.

AUP divide el ciclo de desarrollo en 4 fases:

Inicio: Identificación del alcance y dimensión del proyecto, propuesta de la arquitectura y de presupuesto del cliente.

Elaboración: Confirmación de la idoneidad de la arquitectura.

Construcción: Desarrollo incremental del sistema, siguiendo las prioridades funcionales de los implicados.

Transición: Validación e implantación del sistema.

AUP es una metodología de desarrollo ágil, la cual se basa en los siguientes principios:

- ✓ El personal sabe lo que está haciendo. La gente no va a leer detallado el proceso de documentación, pero algunos quieren una orientación de alto nivel y/o formación de vez en cuando.
- ✓ Simplicidad. Todo se describe concisamente utilizando un puñado de páginas, no miles de ellos.
- ✓ Agilidad. El ajuste a los valores y principios de la Alianza Ágil.
- ✓ Centrarse en actividades de alto valor. La atención se centra en las actividades esenciales para el de desarrollo, no todas las actividades que suceden forman parte del proyecto.
- ✓ Herramienta de la independencia. Usted puede usar cualquier conjunto de herramientas que usted deseé con el ágil UP. Lo aconsejable es utilizar las herramientas más adecuadas para el

trabajo, que a menudo son herramientas simples o incluso herramientas de código abierto. (Flores and Cordero)

AUP define actividades que el equipo de desarrolladores debe realizar para construir, validar y entregar un software que satisfaga las necesidades de los involucrados.

2.2.1.2 Extreme Programming (XP)

Es la más destacada de los procesos ágiles de desarrollo de software formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Las características fundamentales del método son:

- ✓ Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- ✓ Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- ✓ Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- ✓ Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- ✓ Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- ✓ Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- ✓ Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda

corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- ✓ Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que en más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores. (Figueroa, Solís et al.)

Ventajas

- ✓ Apropiado para entornos volátiles.
- ✓ Estar preparados para el cambio, significa reducir su coste.
- ✓ Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio.
- ✓ Permitirá definir en cada iteración cuales son los objetivos de la siguiente
- ✓ Permite tener realimentación de los usuarios muy útil.
- ✓ La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas

- o Delimitar el alcance del proyecto con el cliente.

Para mitigar esta desventaja se plantea definir un alcance a alto nivel basado en la experiencia.

2.2.2 Metodologías tradicionales (no ágiles)

Las metodologías no ágiles son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo; llamadas también metodologías tradicionales o clásicas, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema. (Computación. 2004)

Entre las principales metodologías tradicionales tenemos los ya tan conocidos RUP y MSF entre otros, que centran su atención en llevar una documentación exhaustiva de todo el proyecto y centran su atención en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

2.2.2.1 El Proceso Unificado de Desarrollo de Software (RUP)

El Proceso Unificado de Desarrollo de Software (Rational Unified Process) conocido como RUP (siglas en inglés), es una metodología de software que permite el desarrollo de aplicaciones a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto.

El proceso de desarrollo constituye un marco metodológico que define en términos de metas estratégicas, objetivos, actividades y artefactos (documentación) requeridos en cada fase de desarrollo. Esto permite enfocar esfuerzo de los recursos humanos en términos de habilidades, competencias y capacidades a asumir roles específicos con responsabilidades bien definidas.

El RUP es una de las metodologías robustas o pesadas, que presenta entre sus características ser un proceso de desarrollo orientado a objetos, utiliza el Lenguaje Unificado de Modelado (UML) como lenguaje de representación visual. Este proceso unificado define "Quién", "Cómo", "Cuándo" y "Qué" debe hacerse en el proyecto. Tiene tres características fundamentales: es iterativo e incremental, centrado en la arquitectura y dirigido por casos de usos.

Dirigido por casos de uso: "Los casos de uso representan los requisitos de software capturados durante el flujo de trabajo de requisitos, la planificación del proyecto se hace en términos de casos de uso, los desarrolladores crean realizaciones de casos de uso en términos de clases y subsistemas, los componentes se incorporan en los incrementos y cada uno realiza un conjunto de casos de uso, y por último se verifica que el sistema implementa los casos de uso correctos para el usuario. En otras palabras los casos de uso guían la arquitectura del sistema, enlazan todas las actividades del desarrollo y dirigen el proceso de desarrollo". (Jacobson, Booch et al. 2004)

Centrado en la arquitectura: "La arquitectura representa la forma del futuro sistema en términos de vistas arquitectónicas, sobre la cual el equipo de desarrollo y usuarios deben estar de acuerdo, ya que estas describen los elementos del modelo más importantes para su desarrollo, la arquitectura va madurando en las interacciones comenzando con los casos de uso relevantes desde el punto de vista arquitectónico". (Jacobson, Booch et al. 2004)

Iterativo e incremental: "El Proceso Unificado propone que cada fase se desarrolle en iteraciones, ya que el incremento en la complejidad de los sistemas actuales hace que sea factible dividir el trabajo en partes más pequeñas o mini-proyectos. Cada mini-proyecto es una iteración que resulta un

incremento. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros". (Jacobson, Booch et al. 2004)

El período de vida del software esta particionado en ciclos, cada ciclo consta de cuatro fases: concepción o inicio, elaboración, construcción y transición y cada vez que termina un ciclo se produce una versión del sistema.

Es ideal para proyectos cuyos requisitos no son variables y para grandes equipos de desarrollo. Sin embargo puede adaptarse a diferentes condiciones. En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo, los 6 primeros son flujos de ingeniería y los tres últimos de apoyo.

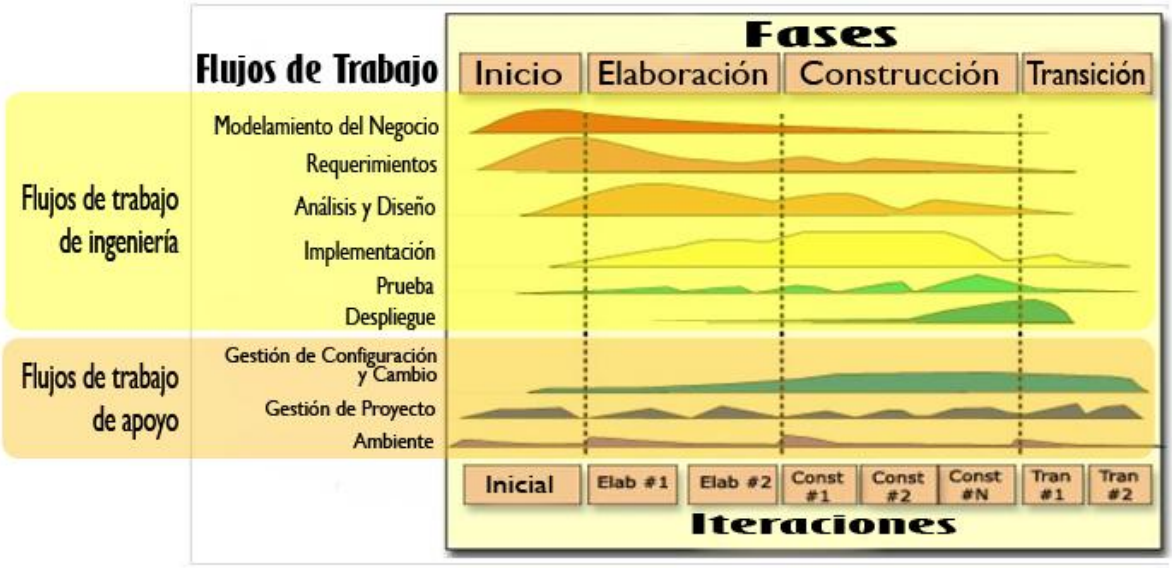


Figura 16: RUP: Fases, flujos de trabajo e iteraciones

Cada flujo de trabajo cumple con algunas actividades específicas. En el funcionan trabajadores específicos y producen y consumen artefactos también definidos.

Cada fase representa un estado del proyecto, y produce un hito que sirve de entrada a la próxima fase. Todos los flujos se aplican en todas las fases, si bien algunos tienen más carga de trabajo que otros en algunas fases específicas.

2.2.2.2 Microsoft Solution Framework (MSF)

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

Todo proyecto es separado en *cinco principales fases*:

- Visión y Alcances.
- Planificación.
- Desarrollo.
- Estabilización.
- Implantación.

Visión y Alcances: La fase de visión y alcances trata uno de los requisitos más fundamentales para el éxito del proyecto, la unificación del equipo detrás de una visión común. El equipo debe tener una visión clara de lo que quisiera lograr para el cliente y ser capaz de indicarlo en términos que motivarán a todo el equipo y al cliente. Se definen los líderes y responsables del proyecto, adicionalmente se identifican las metas y objetivos a alcanzar; estas últimas se deben respetar durante la ejecución del proyecto en su totalidad, y se realiza la evaluación inicial de riesgos del proyecto.

Planificación: Es en esta fase cuando la mayor parte de la planeación para el proyecto es terminada. El equipo prepara las especificaciones funcionales, realiza el proceso de diseño de la solución, y prepara los planes de trabajo, estimaciones de costos y cronogramas de los diferentes entregables del proyecto.

Desarrollo: Durante esta fase el equipo realiza la mayor parte de la construcción de los componentes (tanto documentación como código), sin embargo, se puede realizar algún trabajo de desarrollo durante la etapa de estabilización en respuesta a los resultados de las pruebas. La infraestructura también es desarrollada durante esta fase.

Estabilización: En esta fase se conducen pruebas sobre la solución, las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento.

Implantación: Durante esta fase el equipo implanta la tecnología base y los componentes relacionados, estabiliza la instalación, traspasa el proyecto al personal soporte y operaciones, y obtiene la aprobación final del cliente. (Figueroa, Solís et al.)

Modelo de roles

El modelo de equipos de MSF (MSF team model) fue desarrollado para compensar algunas de las desventajas impuestas por las estructuras jerárquicas de los equipos en los proyectos tradicionales.

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

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

La comunicación se pone en el centro del círculo para mostrar que está integrada en la estructura y fluye en todas direcciones. El modelo apoya la comunicación efectiva y es esencial para el funcionamiento del mismo. (Figueroa, Solís et al.)

2.2.3 Metodologías Ágiles versus Metodologías Tradicionales

La Tabla 1 recoge esquemáticamente las principales diferencias de las metodologías ágiles con respecto a las tradicionales (“no ágiles”). Estas diferencias que afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización.

La siguiente tabla recoge una comparación entre las Metodologías Ágiles y Metodologías Tradicionales con el objetivo de demostrar que para el desarrollo de la solución propuesta, resulta necesario escoger una metodología ágil, debido a todas las ventajas que las mismas presentan.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1: Diferencias entre Metodologías Ágiles y Tradicionales

Se ofrece a continuación una comparativa entre cada uno de las etapas más comunes del desarrollo de software y los enfoques de metodologías revisados.

Modelos Tradicionales	Etapa	Modelos Ágiles
Planificación predictiva y "aislada"	Análisis de requerimientos Planificación	Planificación adaptativa: Entregas frecuentes + colaboración del cliente
Diseño flexible y Extensible + modelos +	Diseño	Diseño Simple: Documentación Mínima +

Documentación exhaustiva		Focalizado en la comunicación
Desarrollo individual con Roles y responsabilidades estrictas	Codificación	Transferencia de conocimiento: Programación en pares + conocimiento colectivo
Actividades de control: Orientado a los hitos + Gestión miniproyectos	Pruebas Puesta en Producción	Liderazgo-Colaboración: empoderamiento +auto-organización

Tabla 2: Diferencia por etapas y enfoques metodológicos

2.2.4 El Proceso Unificado Ágil (AUP) como base en el desarrollo de la solución propuesta

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto, recursos técnicos y humanos, tiempo de desarrollo y tipo de sistema. Estas son algunas de las cuestiones a valorar a la hora de escoger una metodología.

Teniendo en cuenta las particularidades de cada una de las metodologías de desarrollo, se ha seleccionado *Agile Unified Process (AUP)*, la cual es una metodología ágil. AUP proporciona un ambiente de desarrollo de software iterativo e incremental. A menudo es calificado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un proyecto de software, pero dado que es un marco de procesos, puede ser adaptado al proyecto que se esté desarrollando. AUP propone en una versión simplificada, los mismos roles y artefactos de RUP, que es la metodología con la cual se ha familiarizado la universidad; sólo se utilizan los artefactos que son imprescindibles y realmente necesarios para la realización del producto. Además AUP adopta muchas de las técnicas de XP y otros procesos ágiles que aún mantienen la formalidad de RUP.

De igual forma AUP se preocupa especialmente de la gestión de riesgos, es decir, propone para aquellos elementos con alto riesgo prioridad en el proceso de desarrollo y que sean abordados en etapas tempranas del mismo.

2.3 El Lenguaje Unificado de Modelado (UML) versión 2.0 como soporte de la modelación de la solución propuesta

El Lenguaje Unificado de Modelado (Unified Modeling Language), conocido como UML por sus siglas en inglés, es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. Este contiene diagramas que permiten la modelación de los diferentes componentes del sistema. Es utilizado en varias metodologías aunque presenta una estrecha relación con la que será utilizada en este trabajo de diploma: el Proceso Unificado Ágil (AUP), ya que este hace uso de todos los diagramas propuestos por este lenguaje.

Con el surgimiento del UML llegaron a su fin las llamadas "guerras de métodos" de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás y surgió precisamente por la necesidad de crear un estándar único que debía ser, ante todo, universal. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos, de igual forma, los clientes, desarrolladores y otras personas involucradas o interesadas, pueden comprender el funcionamiento y la estructura de un sistema determinado.

UML agrupa los diagramas en tres tipos diferentes.

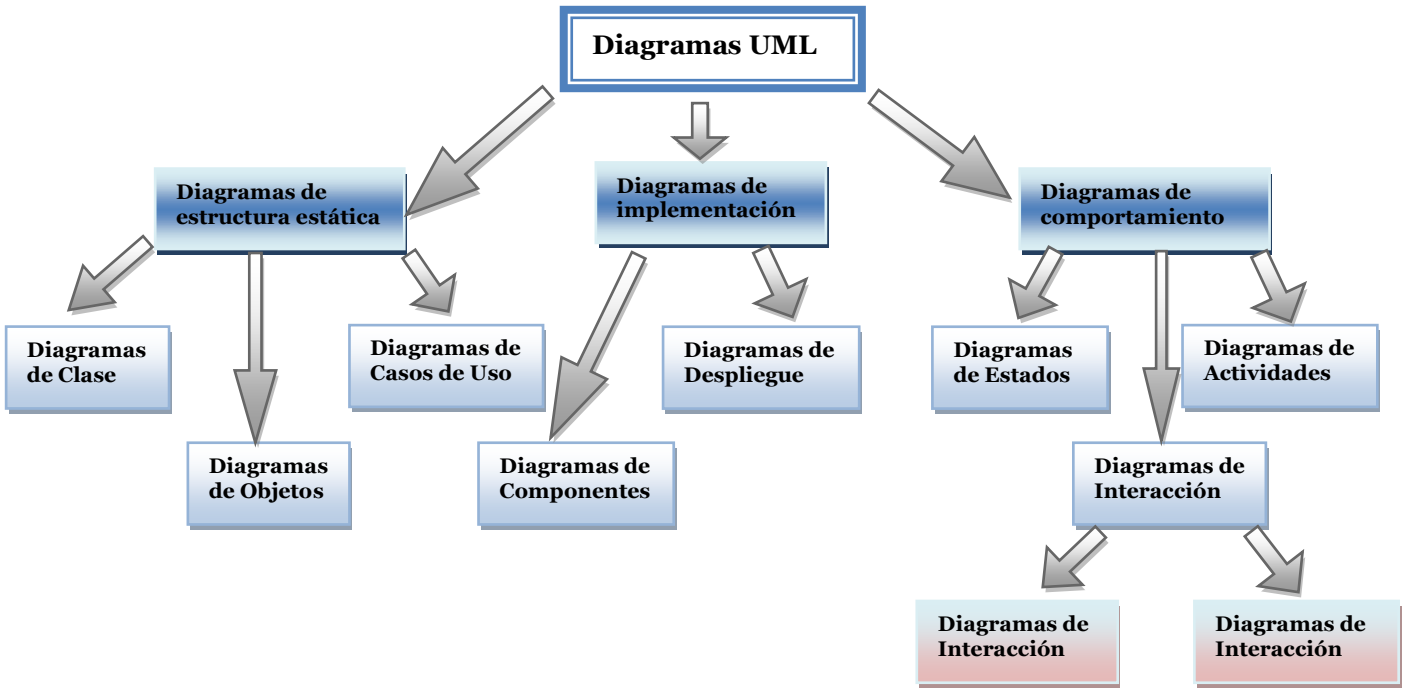


Figura 17: Diagramas UML

Algunas de las características que presenta este lenguaje son:

- ✓ Incluye “estereotipos” como mecanismo de extensibilidad.
- ✓ Aporta un lenguaje para expresar restricciones mediante fórmulas bien formadas como OCL (Object Constraint Language).
- ✓ Puede describir cualquier tipo de sistemas en términos de diagramas orientado a objetos.

Es ideal para el modelado de sistemas orientados a objetos ya que incluye la representación de la abstracción, herencia, polimorfismo, encapsulamiento o encapsulación, envío de mensajes, asociaciones y agregación. Permite además detectar con mayor facilidad las dependencias y dificultades implícitas del sistema, con él se pueden modelar tanto sistemas de software y de hardware como organizaciones del mundo real.

2.4 Herramientas CASE (Computer-Aided Software Engineering)

Las herramientas CASE son un conjunto de programas y ayudas que brindan asistencia técnica a analistas, ingenieros de software y desarrolladores para el análisis de requisitos, modelado visual y documentación durante parte o todo el ciclo de vida de un proyecto de software. (Informática 1999) La selección de esta herramienta está estrechamente relacionada con la metodología de desarrollo de software y el lenguaje de modelado a utilizar.

2.4.1 Rational Rose Enterprise Edition

Rational Rose Enterprise Edition es una herramientas CASE y está considerada como una de las más poderosas para el moldeamiento visual y también una de las más usadas y populares en el campo de la informática. Permite crear y refinar, logrando así un modelo completo que representa el dominio del problema y el sistema de software. Permite especificar, analizar y diseñar el sistema antes de codificarlo.

Brinda la posibilidad de generar código en distintos lenguajes de programación a partir de un diseño en UML y proporciona mecanismos para realizar la denominada ingeniería inversa, o sea, la realización de los diagramas una vez conocido el código. Soporta de forma completa la especificación del UML. Es realmente una de las herramientas más técnicas y de fácil uso pues viene acompañada de un sistema de ayuda bastante amigable, al igual que los estereotipos y diagramas que a partir de él se generan.

Presenta dos aspectos que limitan su uso y estos son, en primer lugar, que no es soportada en el Sistema Operativo GNU/Linux, y en segundo, que es una herramienta propietaria, lo que significa que es necesario pagar por su licencia para poder utilizarla.

2.4.2 Visual Paradigm for UML Enterprise Edition

"El paradigma visual para el lenguaje unificado de modelado UML es una potente plataforma, la cual está diseñada para una amplia gama de usuarios, incluidos los analistas de sistemas, ingenieros de software, etc. Esta herramienta facilita la interoperabilidad con otras herramientas de modelado de UML además de permitir la transición de análisis para el diseño." (Stella 2005)

El Visual Paradigm for UML Enterprise Edition es una herramienta CASE poderosa y fácil de usar. Permite representar todo tipo de diagramas UML para las distintas fases como la captura de requisitos, análisis, diseño e implementación. Presenta, al igual que otras herramientas de modelado visual, una serie de ventajas tales como la generación del código fuente en java, C#, C entre otros a partir de diagramas de clases y además aplicar ingeniería inversa en los lenguajes Java, C++, CORBA IDL, PHP, XML, Ada y Python.

Unas de las características que presenta es:

- ✓ Navegación intuitiva entre el modelo visual y el código.
- ✓ Modela todos los diagramas de UML.
- ✓ Validación de modelos en tiempo real.
- ✓ Presenta recursos centrado en la interfaz para mejorar la usabilidad
- ✓ Diagrama de diseño automático.
- ✓ Permite exportar diagramas como imagen en el formato JPG, PNG y SVG.
- ✓ Presenta sub-diagramas de apoyo para todos los modelos UML.
- ✓ Diseño centrado en caso de uso y enfocado al negocio que le permite generar un software con mayor calidad.
- ✓ Importa Racional Rose Project.

Una de las ventajas distintivas que presenta esta herramienta CASE es que apoya la sincronización del código java, lo que le permita la generación de código en este lenguaje de programación a partir de modelos y viceversa. Esta herramienta CASE es una de las que soporta el análisis textual, una técnica que se utiliza para la captura de requisitos. Una característica fundamental que presenta esta herramienta es la disponibilidad de múltiples plataformas: es soportada tanto en el Sistema Operativo

Windows como en el GNU/Linux. Importante es señalar que existen algunas de sus versiones que son gratuitas.

2.4.3 Comparación entre Visual Paradigm for UML Enterprise Edition y Rational Rose Enterprise Edition

Durante el análisis de las características de estas dos herramientas CASE tenidas en cuenta para modelar la solución propuesta, ha sido posible notar que ambas son bastante poderosas y usadas por los desarrolladores de software de todo el orbe. Ambas poseen una fortaleza técnica indiscutible y sus características son altamente reconocidas en todo el mundo. Establecer una breve comparación entre ambas resultaría ilustrativo para tomar una correcta decisión de cuál es la apropiada para utilizar en este trabajo de diploma.

Criterios de comparación	Visual Paradigm for UML Enterprise Edition	Rational Rose Enterprise Edition
UML	Modela todos sus diagramas.	Modela todos sus diagramas.
Ingeniería inversa	Sí.	Sí.
Sistemas Operativos que lo soportan	GNU/Linux y Windows.	Windows.
Licencia	Tiene algunas versiones gratuitas.	Es una herramienta propietaria.

Tabla 3: Comparación entre Visual Paradigm y Rational Rose

2.4.4 Visual Paradigm for UML 6.4 Enterprise Edition como herramienta para el modelado de la solución propuesta

Visual Paradigm for UML 6.4 Enterprise Edition es sin dudas la elección más apropiada para modelar la solución propuesta. Para tomar esta decisión se han tenido en cuenta un conjunto de factores de irrefutable relevancia para el presente equipo de desarrollo. Esta es una herramienta que permite el modelado orientado a objetos, y es, a la vez, orientada al Lenguaje Unificado de Modelado (UML) que

fue el escogido en este trabajo de diploma como soporte para la modelación. El visual Paradigm fue una de las herramientas establecidas por el polo productivo PetroSoft como tecnología a utilizar, en el desarrollo del presente trabajo de diploma. Además es la más usada en la universidad, y esto trae consigo la comodidad de contar con la ayuda de especialistas en el tema.

El Rational Rose Enterprise Edition es una herramienta muy usada pero es software propietario y no es multiplataforma, al contrario del Visual Paradigm que es una herramienta multiplataforma y algunas de sus versiones son gratuitas.

2.5 Lenguaje de Programación Java

Java es un lenguaje de programación orientado a objetos que debido a su plataforma J2EE se ha hecho muy popular en Internet. Una de las principales características que presenta este lenguaje es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware, permite resolver problemas de alta complejidad y soportar las características de encapsulación, herencia, polimorfismo y enlace dinámico.

Una de sus principales características es que es distribuido, pues proporciona una colección de clases para su uso en aplicaciones de red, que permiten establecer y aceptar conexiones con servidores o clientes remotos.

Otras características de Java son:

Robustez: Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros) eliminando la necesidad de liberación explicita de memoria.

Indiferencia en la arquitectura: soporta aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.

Alto rendimiento: Java soporta sincronización de múltiples hilos de ejecución (multithreanding) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. “Este lenguaje de alto rendimiento soporta la concurrencia a través de threads lo cual significa que se puede dividir una aplicación en varios flujos de control independientes, cada uno de los cuales lleva a cabo funciones de manera concurrente”. (Navarra)

Dinámica: El lenguaje java y sus sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases solo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda procedente de fuentes muy variadas, incluso de la red.

Seguridad: no contempla la posibilidad de manipular la memoria mediante el uso de punteros ni la capacidad de transformación de números en direcciones de memoria, evitando así todo acceso ilegal a la memoria. El compilador de Java hace posible lo anterior mediante una verificación sistemática de conversiones.

2.5.1 Java como lenguaje de programación para la implementación de la solución propuesta

Java, por ser un lenguaje prácticamente sin características dependientes del implementador y con una extensa librería utilitaria cuya interfaz de programación está fuertemente estandarizada, puede desarrollar programas en casi todas las plataformas.

El lenguaje de programación Java presenta una desventaja con respecto al C y al C++, esta radica precisamente en la velocidad de ejecución. Java es más lento debido a que utiliza muchos de los componentes auxiliares como librerías, base de datos y dispositivos gráficos acelerados, entre otros; además, no se compila directamente en el lenguaje de máquina del CPU en uso, sino que utiliza un programa llamado Máquina Virtual de Java (JVM) que consume grandes recursos de memoria.

A pesar de esta desventaja, se ha escogido Java como lenguaje de programación a emplear para la implementación del software que dará solución al problema científico de la presente investigación y para incorporarse a los estándares establecidos por el polo productivo PetroSoft basados principalmente por su seguridad y portabilidad.

2.6 NetBeans IDE 6.8 como Entorno de Desarrollo Integrado (IDE) a utilizar en el desarrollo de la solución propuesta

NetBeans no es más que un Entorno de Desarrollo Integrado (IDE) basado en código abierto y en una plataforma de aplicación de escritorio genérica. Una de las características que presenta esta herramienta es que puede cargar módulos dinámicamente y además permite montar una aplicación de módulos, lo cual posibilita beneficiarse de trabajos realizados por otros. Cuenta también con las bibliotecas visuales API que se encargan de la visualización de datos y la creación de fichas de un texto de entrada. NetBeans IDE 6.8 permite que el lenguaje y las mejoras en la plataforma sean de fácil acceso para los desarrolladores y ayuda a acelerar el desarrollo de aplicaciones. Es el primer IDE que brinda soporte completo para Java EE 6 y Sun GlassFish Enterprise Server v3 y ofrece PHP mejorado, soporte para JavaFX y C/C ++. El NetBeans 6.8 IDE también proporciona un mejor soporte para JSF 2.0/Facelets, Java Persistence 2.0, EJB 3.1 incluyendo el uso de EJB en aplicaciones Web y servicios Web RESTful, así como también mejoras en la plataforma NetBeans.

2.7 JFreeChart 1.0.13 como librería a utilizar para generar las gráficas en el desarrollo de la solución propuesta

Para el desarrollo de los componentes de interfaz de usuario, lo cual es el propósito de este trabajo de diploma, es necesario usar una librería para gráficos. Se escoge JFreeChart entre las existentes, en su versión 1.0.13 (la más actual), que además de brindar funcionalidades de gran calidad y ser escrita en Java, se escoge por ser libre.

2.8 Conclusiones

Durante el presente capítulo se han enunciado y caracterizado los principales componentes tecnológicos a tener en cuenta en el desarrollo de la solución propuesta en este trabajo de diploma. Se han comparado las distintas posibilidades y se ha escogido, entre otras, la metodología de desarrollo de software, el lenguaje de programación, y el entorno de desarrollo integrado, además de la librería gráfica apropiada para conseguir de manera óptima el objetivo general de la presente investigación. Se ha asumido como principal premisa la de lograr la libertad tecnológica del producto, teniendo siempre en cuenta el soporte sobre diferentes sistemas operativos y el cumplimiento con los principios de la comunidad de software libre.

Capítulo 3: Presentación de la solución propuesta

3.1 Introducción

En éste capítulo se exponen los artefactos generados correspondientes al flujo de trabajo de modelado. Debido a la poca estructuración de los procesos de negocio y para poder comprender el contexto en el cual se desarrollan los componentes se determinó desarrollar un Modelo de Dominio, donde se expone un marco conceptual y las relaciones entre estas definiciones. Por otra parte, se enumeran los requerimientos funcionales y no funcionales, agrupándose los primeros en Casos de Uso, con el fin de estructurar el Diagrama de Casos de Uso del Sistema.

3.2 Modelo de Dominio

El Modelo de Dominio o Modelo Conceptual, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del problema. Un Modelo del Dominio es una representación de las clases conceptuales del mundo real, no de componentes software. El modelo desarrollado no se trata de un conjunto de diagramas que describen clases de software u objetos de software con responsabilidades, sino que puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio. Aprovechando las bondades de los diagramas UML para representar conceptos, el Modelo de Dominio se presenta en forma de diagrama de clases donde figuran los principales conceptos y roles del sistema en cuestión.

3.2.1 Conceptos Fundamentales

Para una mejor comprensión del Diagrama del Modelo de Dominio estructurado en el siguiente epígrafe, a continuación se proporciona un marco conceptual con las definiciones identificadas. Estas son:

- **Desarrollador:** Persona que usará los componentes para el desarrollo de aplicaciones.
- **Componente:** Clase conceptual de la cual heredan los demás componentes.
- **Gráfica de Área:** Componente de interfaz de usuario para representar gráficas de área.
- **Gráfica de Línea:** Componente de interfaz de usuario para representar gráficas de línea.

- **Gráfica de Dispersión:** Componente de interfaz de usuario para representar gráficas de dispersión.
- **Gráfica de Barras:** Componente de interfaz de usuario para representar gráficas de barras.
- **Gráfica de Burbujas:** Componente de interfaz de usuario para representar gráficas de burbujas.
- **Gráfica Circular:** Componente de interfaz de usuario para representar gráficas circulares.
- **Histograma:** Componente de interfaz de usuario para representar histogramas.
- **IDE NetBeans:** Plataforma sobre la cual van a ser integrados los componentes.
- **Polo PetroSoft:** Polo productivo integrado por los desarrolladores que usarán los componentes.

3.2.2 Diagrama del Modelo de Dominio

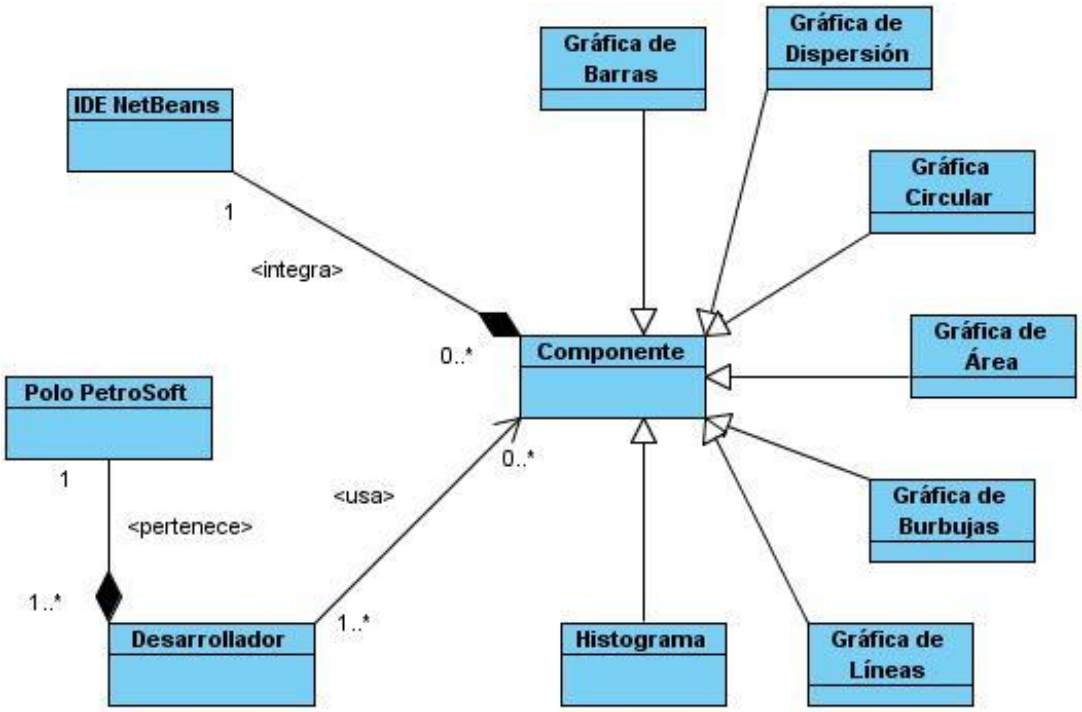


Figura 18: Diagrama del Modelo de Dominio

3.3 Propuesta de los componentes

La solución que se plantea es la de desarrollar componentes de interfaz de usuario para graficar, los cuales podrán ser integrados al IDE NetBeans. El propósito de estos componentes es que puedan ser utilizados por desarrolladores del Polo PetroSoft (aunque no se limita su uso a otros desarrolladores), para desarrollar proyectos en lenguaje Java, en el IDE NetBeans, en los que se requiera visualizar gráficas. Los componentes que se desarrollarán podrán generar diferentes tipos de gráficas, haciendo uso de la librería para gráficos JFreeChart. El desarrollador pudiera usar directamente la librería, pero le será más cómodo graficar usando los componentes. Esto le permitirá disminuir la cantidad de líneas de código así como el tiempo de trabajo que hasta el momento emplean para realizar gráficas dinámicas en sus proyectos, ya que los mismos darán la posibilidad de crear gráficas muy fácilmente, con tan solo arrastrar hasta el área de trabajo los componentes deseados. También se brinda la posibilidad de modificar las gráficas, alterando las propiedades de los componentes con nuevos valores definidos, como igualmente se trabaja con el resto de los componentes agregados en el JDK.
Para hacer uso de los componentes, los desarrolladores deberán seguir los pasos que se muestran en el Manual de Usuario 1, ya que se requiere que sean incluidos los componentes a la paleta de componentes del NetBeans.

3.4 Requerimientos

Como parte del modelado se definieron los requerimientos, los cuales son una "condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo" (Jacobson, Booch et al. 2004). Los requerimientos se clasifican en funcionales y no funcionales. Los requerimientos funcionales "son capacidades o condiciones que el sistema debe cumplir, se mantienen invariables sin importar con que propiedades o cualidades se relacionen por lo que no alteran la funcionalidad del producto" (Jacobson, Booch et al. 2004). Y los requerimientos no funcionales "son las propiedades o cualidades que el sistema debe tener" (Jacobson, Booch et al. 2004). Los requerimientos no funcionales son aquellos requisitos que hacen que el sistema sea usable, rápido, confiable y agradable para los usuarios.

3.4.1 Requerimientos Funcionales

(Para acceder a la descripción de los requerimientos ver el Artefacto Especificación de Requisitos)
A través de los requerimientos funcionales, se puede expresar una especificación más detallada de las responsabilidades de los componentes. Con ellos, se pretende determinar de manera clara y concisa lo que deben hacer los componentes a desarrollar siguiendo un enfoque funcional.

A partir del modelo de dominio representado anteriormente, se alcanzó la visión necesaria del objeto a automatizar y quedaron determinadas las funcionalidades que desarrollan los componentes. Se definieron 11 requerimientos funcionales que quedaron expresados como funcionalidades de los mismos. Estos se recogen en la siguiente tabla:

RF1	Visualizar gráfica de barras.
RF2	Visualizar gráfica circular.
RF3	Visualizar gráfica de área.
RF4	Visualizar gráfica de dispersión.
RF5	Visualizar gráfica de burbujas.
RF6	Visualizar gráfica de línea.
RF7	Visualizar histograma.
RF8	Mostrar propiedades de los componentes.
RF9	Mostrar eventos de los componentes.
RF10	Modificar propiedades de los componentes.
RF11	Modificar eventos de los componentes.

Tabla 4: Requerimientos funcionales

3.4.2 Requerimientos No Funcionales

En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto debido a que forman una parte significativa de la especificación. Normalmente están vinculados a requisitos funcionales, es decir una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. (Para acceder a la descripción de los requerimientos deberá remitirse al artefacto Especificación de Requisitos).

Usabilidad

- Los componentes podrán ser usados por aquellos desarrolladores que tengan conocimiento básico de programación java usando el NetBeans.

Rendimiento

- Tipos de respuestas para el procesamiento de las gráficas, menores a los 5 milisegundos.

Apariencia o interfaz externa

- Los componentes deberán poseer una interfaz sencilla, amigable, lo más atractiva y clara posible para el usuario, además su funcionamiento debe ser lo más fácil posible de comprender.

Requerimientos de Hardware

- Procesador Pentium 3 (o superior).
- Memoria RAM mínima de 256 MB.
- 1Gb de espacio libre en el disco como mínimo.

Requerimientos de Software

- Sistema Operativo tanto Windows (win9.x o versión superior) como Linux (cualquiera de sus distribuciones).
- NetBeans IDE en cualquiera de sus versiones, así como cualquier otro IDE capaz de importar componentes con extensión .jar, y la JVM (Máquina Virtual de Java).
- Adicionar al IDE de desarrollo los archivos .jar JFreeChart y JCommon de la librería JFreeChart.

3.5 Definición de los casos de uso del sistema

3.5.1 Descripción de los actores

Actor	Descripción
Desarrollador	Responsable de usar los componentes de acuerdo a sus necesidades.

3.5.2 Diagramas de casos de uso del sistema

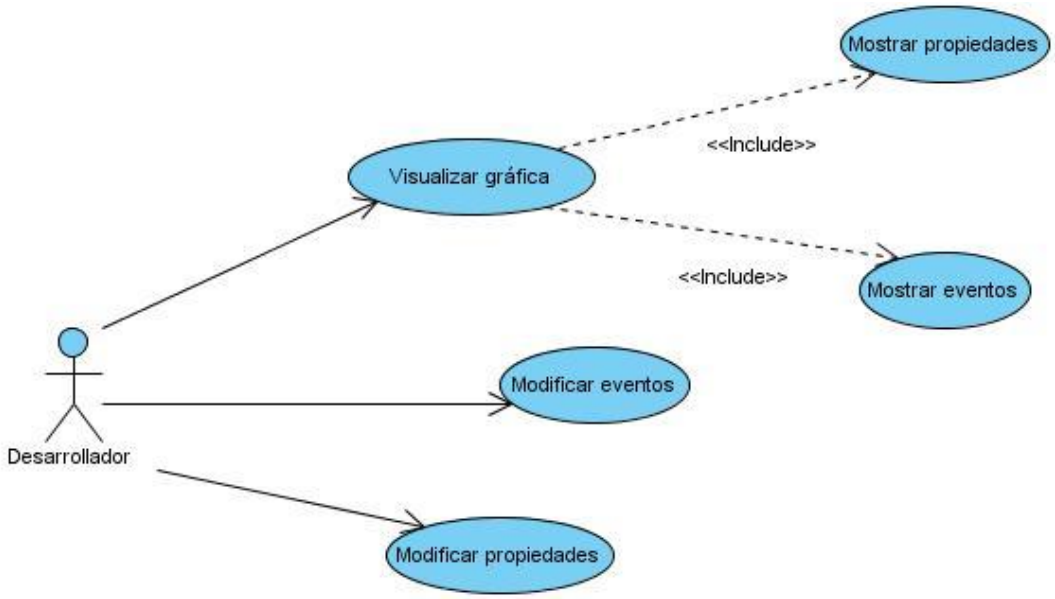


Figura 19: Diagrama de Casos de Uso del Sistema

3.5.3 Listado de los casos de uso del sistema

CU-1	Visualizar gráfica
Actor	Desarrollador
Descripción	El Desarrollador selecciona el componente en cuestión, este mostrará la gráfica correspondiente a su selección.
Referencia	RF1-RF7

CU-2	Mostrar propiedades
Actor	-
Descripción	Después de seleccionar el componente deseado, se muestran las propiedades del mismo.
Referencia	RF8

CU-3	Mostrar eventos
Actor	-
Descripción	Después de seleccionar el componente deseado, se muestra los eventos del mismo.
Referencia	RF9

CU-4	Modificar propiedades
Actor	Desarrollador
Descripción	El desarrollador modifica las propiedades de la gráfica de acuerdo a sus necesidades.
Referencia	RF10

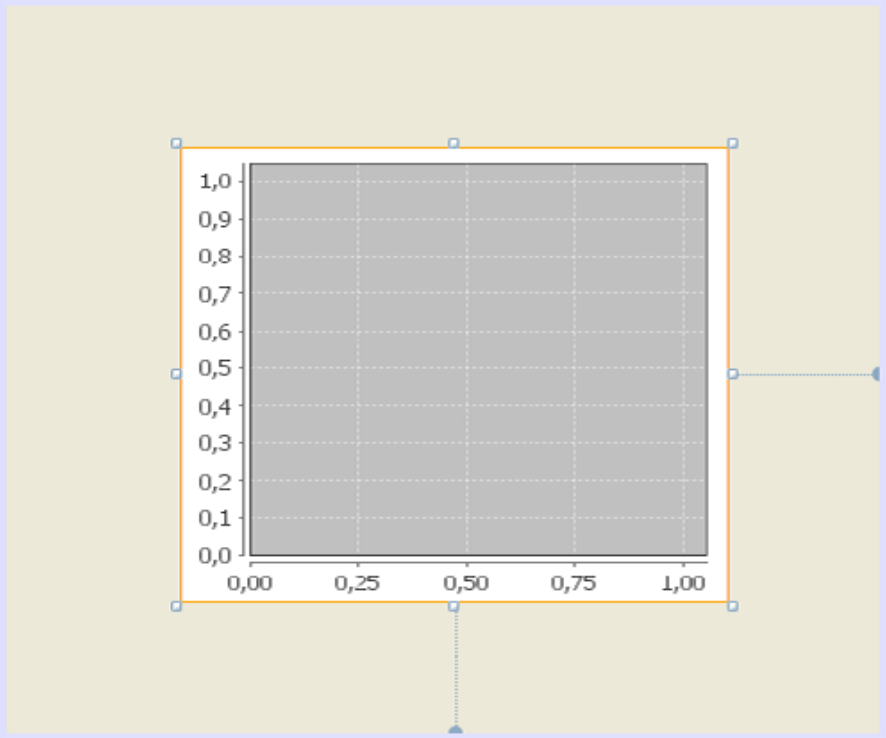
CU-5	Modificar eventos
Actor	Desarrollador
Descripción	El desarrollador modifica los eventos de la gráfica de acuerdo a sus necesidades.

3.5.4 Descripción Textual de Casos de Uso del sistema

3.5.4.1 Descripción del Caso de Uso Visualizar gráfica

Para mayor detalle de todos los casos de uso ver Artefacto Modelo de Casos de Uso del Sistema.

Caso de Uso:	**Visualizar gráfica**
Actores:	Desarrollador
Resumen:	Permite al actor que al cargar el componente seleccionado se visualice la gráfica mostrando sus propiedades, eventos y ayuda.
Precondiciones:	-
Referencias	RF1 - RF9
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El Desarrollador selecciona el componente en la paleta del IDE.	2- Se visualiza la interfaz del componente en el área de trabajo del IDE, además de sus propiedades y eventos correspondientes.
Prototipo de Interfaz 	
Poscondiciones	El componente debe visualizar su gráfica respectiva.

3.6 Conclusiones

En este capítulo se establecieron las características principales de los componentes, así como el proceso para su funcionamiento. Se muestra el modelo del dominio, conjuntamente con la especificación de los requisitos funcionales y no funcionales. Además se elaboró el modelo de casos de uso del sistema, que a través del diagrama confeccionado se resume en el caso de uso “Visualizar gráfica” todos los tipos de gráficas que se quieren visualizar en la solución propuesta.

Capítulo 4: Construcción de la solución propuesta

4.1 Introducción

Luego de un completo entendimiento de las funcionalidades y procesos del sistema, como parte del flujo de trabajo de modelado de la metodología de desarrollo AUP, se puede realizar el diseño y como parte del flujo de trabajo de implementación, la construcción de los componentes. En este capítulo se abordan brevemente los patrones de diseño. Además se realiza la descripción de la construcción de los componentes utilizando diagrama de clases de diseño, un artefacto necesario porque ofrece una idea de cómo fue la concepción de la arquitectura de los componentes a desarrollar. Se presenta el modelo de implementación que resultó del diseño así como algunos resultados obtenidos en las pruebas de los componentes desarrollados.

4.2 Patrones de diseño GRASP (patrones generales de software para asignar responsabilidades)

Un patrón es una descripción de un problema y la solución a la que se le da un nombre, y que además se puede aplicar a nuevos contextos.

Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas no se puede extraer software de forma independiente y reutilizarlo. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades**.**

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). 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.

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.

Creador: Ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases. 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 y posibilidades. (Gutierrez 2006)

4.3 Propuesta de Arquitectura de los componentes

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: Reproducción en papel de un dibujo técnico, un plano cartográfico o un diseño de ingeniería) comentados.

No hay estándares en cuanto a la forma y lenguaje a utilizar en estos blueprints. De todas formas, existe 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. (Casanovas 2004)

Los componentes de IU para la representación de información mediante gráficas en la Industria del petróleo heredan las funcionalidades del componente JPanel perteneciente al paquete de componentes Swing.

La arquitectura que aplica Swing para sus componentes no se basa estrictamente en el tradicional patrón MVC (Modelo-Vista-Controlador), sino en un modelo de arquitectura separable. El mismo consiste en la unión de la vista y el controlador como una sola entidad, ya que entre ellos existe una estrecha conexión, manteniendo así intacto el modelo. Esto se aplica debido a que era muy difícil escribir un controlador genérico que no conocía detalles sobre la vista. Por lo tanto, se derrumbaron estas dos entidades en una única interfaz de usuario objeto (IU Objeto), como se muestra en la Fig. 20. (Fowler 2010)

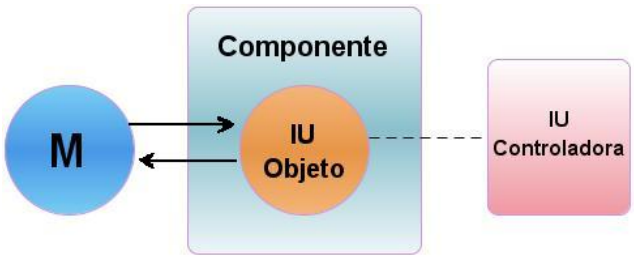


Figura 20: Modelo de Arquitectura de los componentes.(Fowler 2010)

Debido a esto se propone la arquitectura para los componentes a desarrollar, como se muestra en la Fig. 20. Este diagrama representa el casi patrón de diseño MVC (propuesto por Swing), ya que el mismo es heredado por defecto para cada uno de los componentes. En la implementación se hace una restructuración para cada uno de ellos. Los mismos están compuestos por una sola clase que hereda de la clase JPanel, siendo la interfaz la vista del componente. Esta clase actúa como controladora de sus funciones y como modelo además, debido a que contiene parte de la información persistente, representada a través de los atributos.

La arquitectura propuesta anteriormente facilitará la comunicación entre todas las partes interesadas en el desarrollo de los componentes. Constituye un modelo comprensible de cómo está estructurado cada uno de ellos y de cómo trabajan junto a otros componentes. La misma servirá para la toma de decisiones tempranas de diseño con un profundo impacto en la construcción de la solución propuesta.

4.4 Diagrama de Clases del Diseño

Como parte del modelado de los componentes a continuación se muestra el Diagrama de Clases del Diseño. Este diagrama posee una vista general de la organización de los componentes, y estos relacionados con las clases que usan. Debido a la particularidad de cada uno de ellos, es decir, su sencillez a la hora de ser diseñado, se muestra de una forma simple las relaciones entre las clases.

En el siguiente diagrama se puede observar el paquete *JFreeComponents* el cual contiene las clases que corresponden a cada uno de los componentes que heredan de la clase JPanel perteneciente al paquete de componentes Swing. *JFreeComponents* a su vez está asociado a otros paquetes *JFreeChart* y *JExcelAPI* que contiene algunas de las clases de estas librerías que los componentes usan.

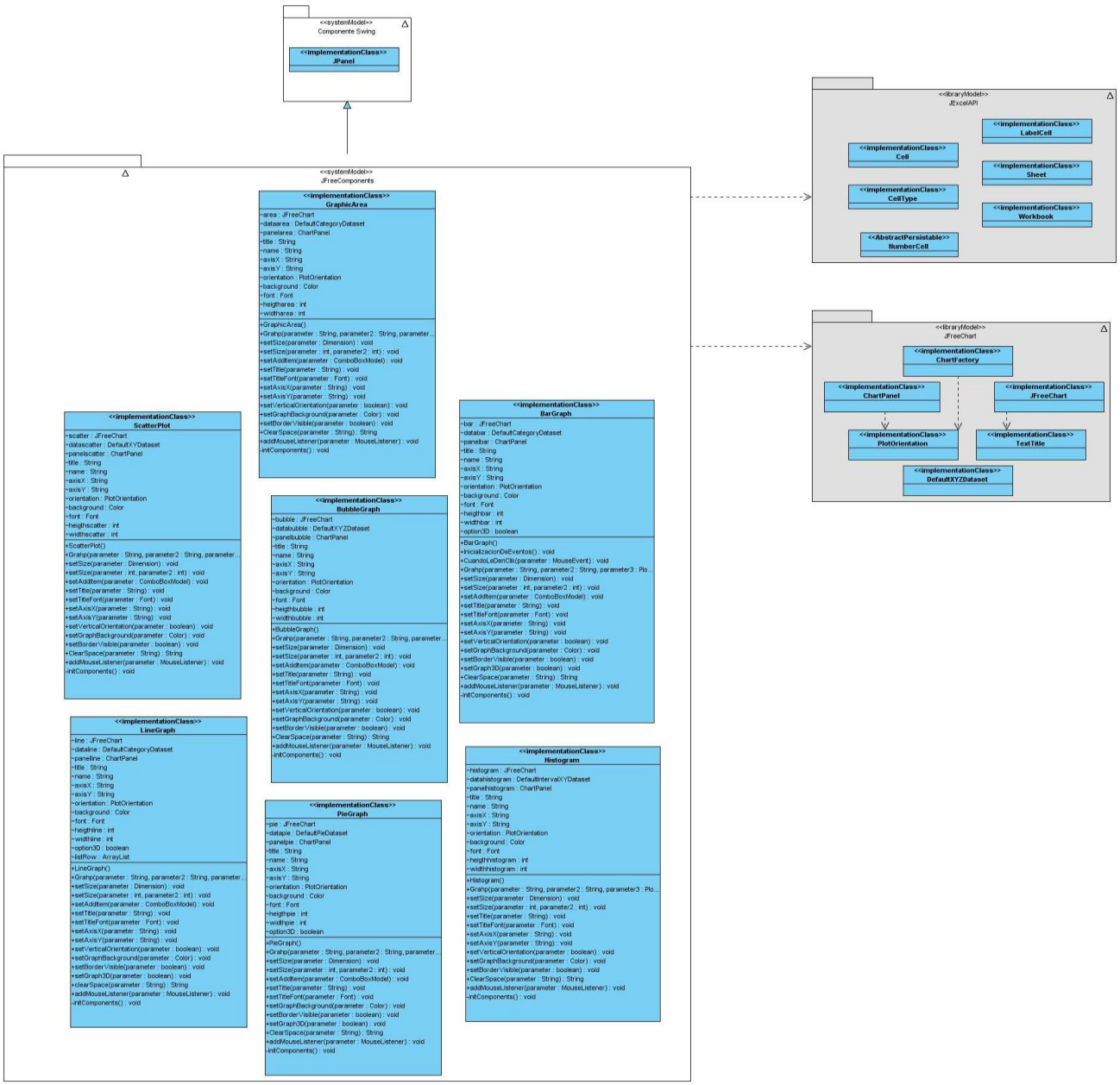


Figura 21: Diagrama de clases del diseño (vista general)

A continuación se muestra el diagrama de clases del diseño de forma particular para el componente Gráfica de Barras, con el objetivo de mostrar las relaciones que existen entre cada uno de los componentes a desarrollar con las clases de la librería. Para cada uno de ellos existen las mismas relaciones.

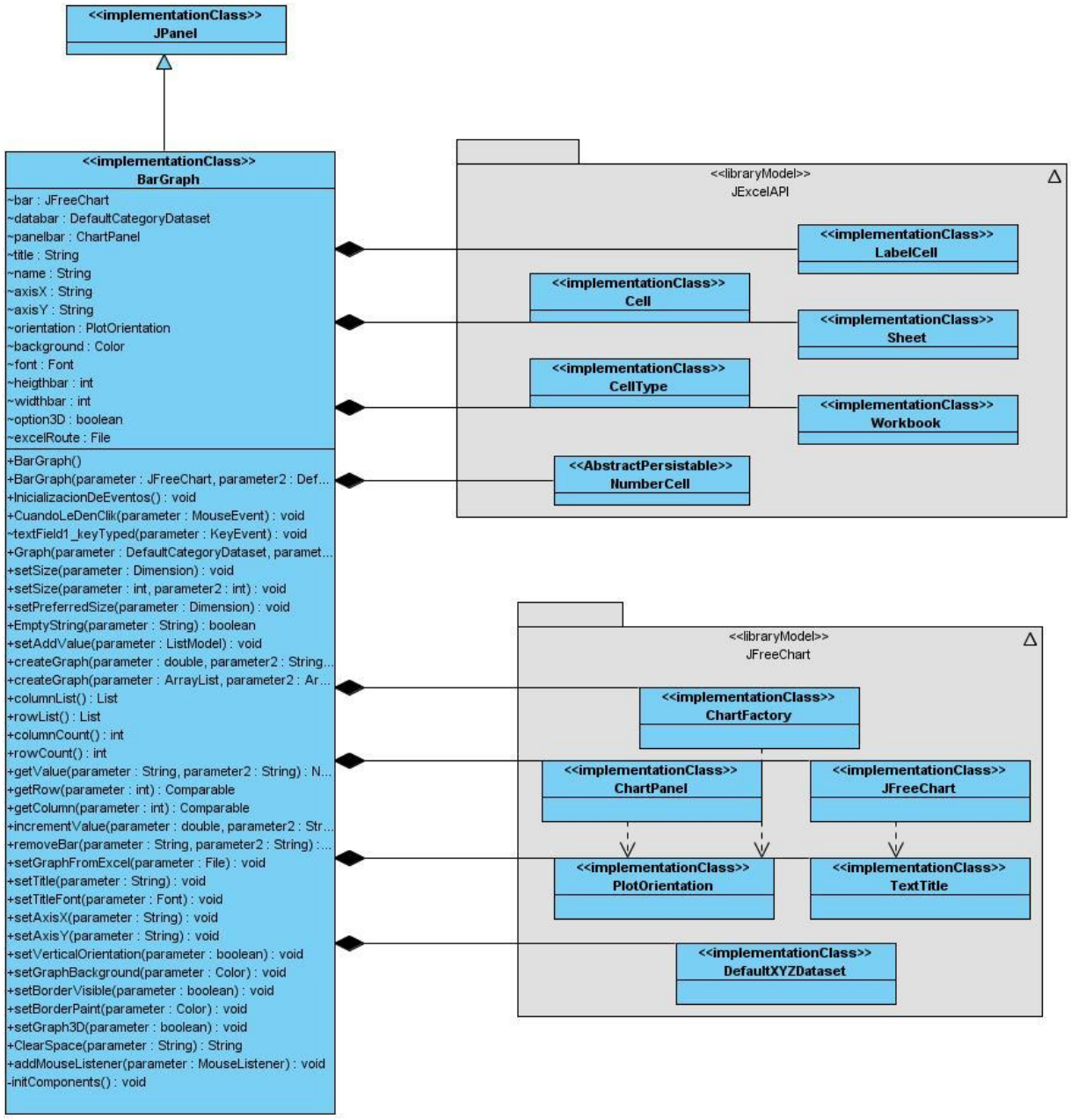


Figura 22: Diagrama de clases del diseño del componente Gráfica de Barra

4.5 Modelo de Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos

encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (MeRinde 2010)

Uno de los artefactos que se generan en este modelo es el Diagrama de Componentes, el cual 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.

En el Diagrama que aparece a continuación se muestra la relación de dependencia que existe entre los paquetes *JFreeComponents* y *Librerías*. El primero contiene todos los componentes a desarrollar (en este caso solo se muestran dos componentes, para que el diagrama no se haga engorroso), los cuales se agrupan en el mismo paquete debido a sus similares características. El segundo agrupa las librerías JExcelAPI y JFreeChart, la cual depende a su vez de la librería JCommon.

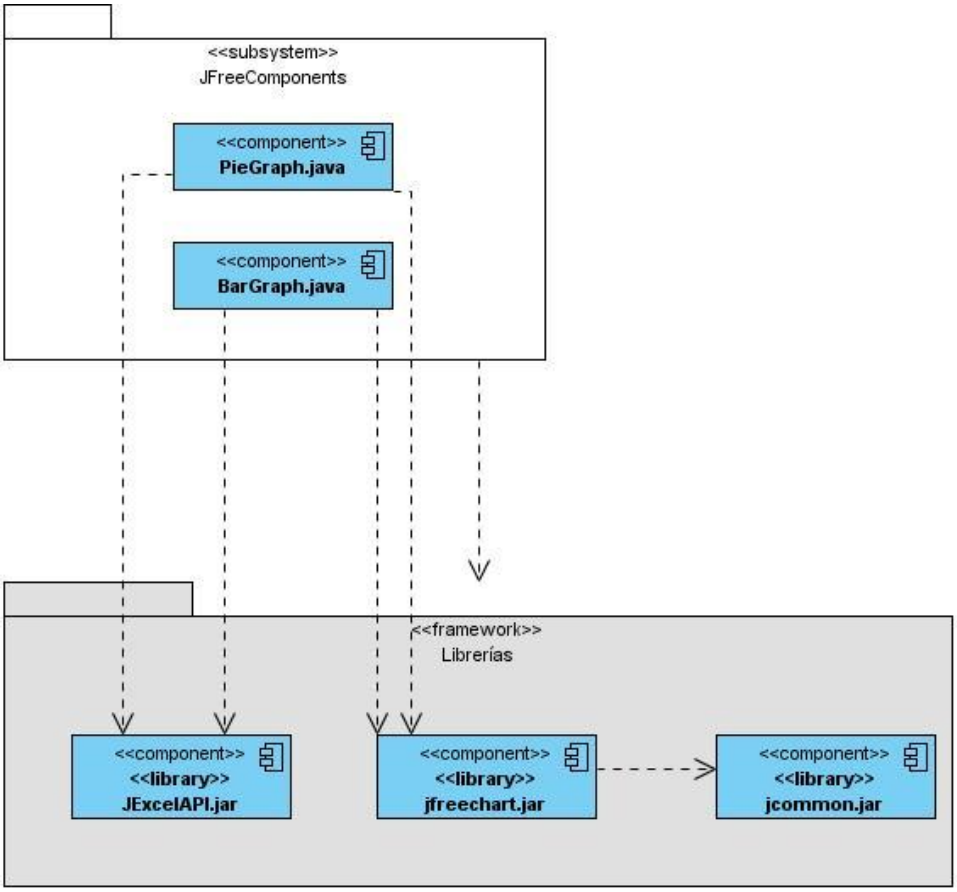


Figura 23: Diagrama de Componentes

4.5 Pruebas

La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.Las mismas son un elemento crítico para la garantía de calidad del software y representan una visión final de las especificaciones, del diseño y de la codificación. (Pressman 2002)

Se puede ir realizando pruebas desde la fase de inicio del software hasta la fase de construcción, siendo esta última fase donde tiene mayor volumen el flujo de trabajo de prueba. Uno de los tipos de prueba que existen son las pruebas de caja negra.

4.5.2 Prueba de Caja Negra

Las pruebas de caja negra, también denominadas *Pruebas de Comportamiento*, se centran en los requisitos funcionales del software.

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y de terminación.

Estas pruebas se refieren a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software. (Pressman 2002)

4.5.3 Pruebas de los componentes

Se entiende por caso de prueba, según la Ingeniería del software, al conjunto de condiciones o variables bajo las cuáles el analista determinará si el requisito de una aplicación es parcial o completamente satisfactorio. A continuación se muestra un caso de prueba realizado al caso de uso Modificar propiedades en el Componente Gráfica de Barras.

Entrada	Resultados	Condiciones
El desarrollador inserta todos los datos correctamente en cada una de las propiedades.	El componente modifica los valores de las propiedades, y actualiza la gráfica.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.
El desarrollador inserta los datos de forma incorrecta de las propiedades seleccionadas.	El componente informa el error a través de un mensaje de diálogo.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.
En el caso que el desarrollador selecciona la propiedad **"addValue"** para adicionar nuevas barras.	El componente activa un ListModel para que se introduzcan los datos necesarios para adicionar la nueva barra.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.
El desarrollador selecciona la opción "Aceptar" habiendo pasado valores nulos o de tipo incorrecto.	El componente muestra un mensaje de alerta informando el formato en que se deben pasar los valores para que sea adicionada correctamente las nuevas barras.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.
En el caso que el desarrollador selecciona las propiedades **"axisX"** o **"axisY"**, si le adiciona como valor una cadena que solo contenga el caracter espacio.	El componente muestra un mensaje de alerta informando que la cadena no solo debe contener espacios.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.
En el caso que el desarrollador selecciona las propiedad **"foregroundAlpha"** y le pasa un valor no numérico o fuera del rango 0-100.	El componente muestra un mensaje de alerta informando que el valor debe ser numérico o en ese rango.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.
En el caso que el desarrollador	El componente activa una	El componente debe estar

selecciona la propiedad **"addFromExcel"** para generar un gráfica.	ventana de archivos para seleccionar el archivo .xls.	seleccionado para que se puedan modificar cada una de sus propiedades.
En caso que el archivo seleccionado no es de extensión .xls.	El componente muestra un mensaje de alerta informando que el archivo debe de ser de tipo .xls.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.
En caso de que este archivo .xls posee en sus celdas algún valor que no se corresponda con el formato de la gráfica.	El componente muestra un mensaje de alerta informando el error.	El componente debe estar seleccionado para que se puedan modificar cada una de sus propiedades.

Tabla 5: Caso de prueba del Caso de Uso Modificar Propiedades

Durante la ejecución de las pruebas se diseñaron los casos de prueba, se utilizó una PC y dos probadores, no se identificaron no conformidades. Las pruebas se hicieron en un ambiente real del producto lo que garantizará la calidad del mismo. Las mismas se aplicaron utilizando la técnica de caja negra lo que hace posible el cumplimiento de los requisitos funcionales del producto.

4.6 Conclusiones

En este capítulo se mostraron los resultados obtenidos en la etapa de diseño, de implementación y prueba. Se describe la fase de implementación y prueba, construyéndose el diagrama de implementación, describiéndose los casos de prueba para demostrar que se cumplieron con los objetivos de forma satisfactoria. Como resultado final se obtiene la primera versión de los componentes y su documentación, para dar cumplimiento a los requerimientos funcionales planteados por el cliente.

Conclusiones Generales

Mediante el presente trabajo Luego de dar por vencidos los objetivos presentados y las tareas trazadas para el desarrollo de los componentes de IU, se llegó a las siguientes conclusiones:

- ✓ Se realizó un estudio sobre la representación de información en las cinco áreas principales de la Industria del petróleo.
- ✓ Se estudiaron de forma comparativa el lenguaje de programación, las herramientas, técnicas y tecnologías a utilizar en la implementación de los componentes.
- ✓ Se realizaron satisfactoriamente las actividades necesarias de los flujos de trabajo: Modelado, Implementación y Prueba, para dar cumplimiento a los requerimientos planteados por el cliente.
- ✓ Se diseñaron e implementaron las funcionalidades de los componentes de IU para la representación de información mediante gráficas en el Polo PetroSoft.
- ✓ Con la implantación de este producto, se logrará facilitar el trabajo a los desarrolladores en la capa de presentación de las soluciones del Polo.

Recomendaciones

Se recomienda que:

- ✓ Los componentes sean usados por grupos de desarrolladores de proyectos que requieran representar información a través de gráficas.
- ✓ Continuar el estudio del desarrollo de componentes, para que puedan ser creados otros con el propósito de facilitar al desarrollador su trabajo.

Bibliografía Citada

Cairó, I. D. (2009). Propuesta de visualización de información para los software de PetroSoft. Ciudad de la Habana, Universidad de las Ciencias Informáticas**:** 93.

Calidad, S. L. p. l. (2000). "Histograma." Retrieved 4 Noviembre, 2009, from http://www.ongconcalidad.org/histogr.pdf.

Casanovas, J. (2004, 9 Septiembre 2004). "Usabilidad y arquitectura del software." Retrieved Mayo, 2010, from http://www.desarrolloweb.com/articulos/1622.php.

Colegio. (2007). "Tipos de gráficos estadísticos." Retrieved 5 Diciembre, 2009, from http://www.corsaje.edu.co/descargas%20tecnologia2007/.

Computación., D. d. S. I. y. (2004, Marzo 2010). "Proceso de Desarrollo de Software." from https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Forms/DispForm.aspx?ID=6&Source=https%3A%2F%2Fpid.dsic.upv.es%2FC1%2FMaterial%2Fdefault.aspx&RootFolder=%2FC1%2FMaterial%2FDocumentos%20Disponibles.

Crystal Decisions, I. (2003). "Tipos de gráficos." Retrieved 10 Noviembre, 2009, from http://www.willydev.net/crystaldesde0/html/crconcharttypes.htm.

Díaz, P. and P. Fernández. (2001). "Representación gráfica en el Análisis de Datos." Retrieved 3 Diciembre, 2009, from http://www.fisterra.com/mbe/investiga/graficos/graficos.asp.

Dürsteler, J. C. (2008) "Cartogramas." InfoVis.net **Volume**, DOI: 197

Educativa, T. (2009). "Desarrollo de un sistema evaluador." Retrieved 20 Enero 2010, from http://www.tecnologiaeducativa.net/web2/2009/08/4-desarrollo-de-un-sistema-evaluador-basado-en-componentes/.

estonova.com Pictograma basado en la señalética. picpek.jpg, estonova.com. **500px × 484px**.

eumednet. "Pictogramas." Retrieved 2 Diciembre, 2009, from http://www.eumed.net/libros/2007a/239/3e.htm.

Ferri, F. clavesleids. **469 x 280 - 16 KB - png**.

Figueroa, R. G., C. J. Solís, et al. "Metodologías Tradicionales vs. Metodologías Ágiles." Retrieved 9 Marzo, 2010, from http://www.google.com.cu/#hl=es&q=Roberth+G.+Figueroa&meta=&aq=&aqi=&aql=&oq=Roberth+G.+Figueroa&gs_rfai=&fp=bf8281a9b87c25dc.

Flores, L. E. and J. L. Cordero. "Proceso Unificado Ágil (AUP)." Retrieved 2 Marzo, 2010, from http://www.mex.tl/images/18149/METODOLOGIAS%20AGILES.pdf.

Fowler, A. (2010, 2010). "A Swing Architecture Overview." Retrieved Junio, 2010, from http://java.sun.com/products/jfc/tsc/articles/architecture/.

Fuentes, L., J. M. Troya, et al. "Desarrollo de Software Basado en Componentes." Retrieved 1 Febrero, 2010, from http://www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf.

Gallego, F. M. (2009, 13 Septiembre 2009). "JFreeChart." Retrieved 18 Enero, 2010, from http://gavab.escet.urjc.es/wiki/Carmar/JFreeChart.

Gutierrez, J. A. S. (2006, 17 Agosto 2006). "Patrones Grasp (Craig Larman) Parte I." Retrieved Abril, 2010, from http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

img236.imageshack.us Histograma. todosvi6.gif. **880 x 513 - 20 KB - gif**.

Informática, S.-J. d. (1999). "Herramientas CASE." Retrieved Abril 2010, from http://jhoel-jp.tripod.com/paginas/herramientascase.pdf.

Jacobson, I., G. Booch, et al. (2004). El Proceso Unificado de Desarrollo de Software. La Habana, Editorial Félix Varela.

Java4less. (2000). "Gráficas para la plataforma Java[TM] " Retrieved Febrero, 2010, from http://www.java4less.com/graficos_s.htm#Features.

JFree.org. (2009). "Requirements." Retrieved 3 Febrero, 2010, from http://www.jfree.org/jfreechart/.

labrandero. (2007). "Mapas y Gráficos." Retrieved 12 Diciembre, 2009, from http://lacomunidad.elpais.com/labrandero/2007/8/25/mapas-y-graficos.

Martin, C. (2002). "La representacion de datos de forma grafica." Retrieved 2 Noviembre, 2009, from http://www.desarrolloweb.com/articulos/875.php.

MeRinde. (2010, 2010). "Modelo de Implementación " Retrieved Mayo, 2010, from http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=495&Itemid=291.

Microsoft Gráfico circular en 3D. ZA100902833082. **371px × 201px**.

Microsoft Gráfico de anillos. ZA102147833082. **390px × 294px**.

Microsoft Gráfico de área. ZA100902863082. **364px × 196px**.

Microsoft Gráfico de barras en 3D. ZA100902843082. **386px × 217px**.

Microsoft Gráfico de burbujas. ZA102147893082. **322px × 331px**.

Microsoft Gráfico de superficie. ZA102146813082. **438px × 342px**.

Microsoft Gráfico de tipo XY (Dispersión). ZA102146193082. **391px × 246px**.

Microsoft Gráfico radial relleno. ZA102153973082. **390px × 294px**.

Microsoft. (2007). "Presentar los datos en un gráfico de burbuja." Retrieved 10 Noviembre, 2009, from http://office.microsoft.com/es-es/outlook/HA012337493082.aspx.

Microsoft. (2008). "Gráficos de anillos." Retrieved 19 Noviembre, 2009, from http://msdn.microsoft.com/es-es/library/ms159178%28SQL.90%29.aspx.

Microsoft. (2008). "Gráficos XY (de dispersión)." Retrieved 10 Noviembre, 2009, from http://msdn.microsoft.com/es-es/library/ms155790%28v=SQL.90%29.aspx.

Microsoft. (2009). "Gráficos de áreas." Retrieved 3 Noviembre, 2009, from http://msdn.microsoft.com/es-es/library/ms159211.aspx.

Microsoft. (2009, Julio 2009). "Gráficos de líneas." Retrieved 8 Abril 2010, 2010, from http://msdn.microsoft.com/es-es/library/ms159640.aspx.

Moreno, L. (2005) "Qué son las interfaces, las interfaces web y las interfaces gráficas." desarrolloweb.com **Volume**, DOI:

Navarra, C. d. T. I. d. l. U. d. "Introducción a Java." Retrieved 15 Enero, 2010, from http://www.unav.es/SI/manuales/Java/indice.html.

Office, M. "Tipos de gráficos disponibles." Retrieved 8 Abril, 2010, from http://office.microsoft.com/es-es/help/HA012337373082.aspx.

Oiticica, R. A. (2009). "The JetChart Library." Retrieved Febrero, 2010, from http://www.jinsight.com/jetchart/index.html.

Pressman, R. S. (2002). Ingeniería del software, un enfoque práctico.

Rojas, M. A. and J. C. M. García. (2004, 30 Septiembre 2004). "Introducción y Principios básicos del Desarrollo de Software basado en Componentes." Retrieved Enero, 2010, from http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf.

Salas, R. (2007). "Redes Neuronales Artificiales." Retrieved 10 Octubre, 2009, from http://www.inf.utfsm.cl/~rsalas/Pagina_Investigacion/docs/Apuntes/Redes%20Neuronales%20Artificiales.pdf.

Scribd. (2008). "Gráfico de superficie." Retrieved 6 Enero, 2010, from http://webcache.googleusercontent.com/search?q=cache:1EkHRQKmk5UJ:www.scribd.com/doc/6894116/Grafico-de-superficie+gr%C3%A1fico+de+superficies&cd=6&hl=es&ct=clnk&gl=cu.

Steema. (2010). "Main features." Retrieved Marzo, 2010, from http://www.steema.com/teechart/java#.

Stella. (2005, 8 Febrero 2005). "Visual Paradigm for UML Enterprise Edition Released " Retrieved 20 Enero, 2010, from http://www.javalobby.org/forums/thread.jspa?forumID=17&threadID=17168.

Stepanek, G. (2005). Projects Secrets : Why software projetcts fail, apress.

Terreros, J. C. (2010). "Desarrollo de Software basado en Componentes " Retrieved 14 Diciembre, 2009, from http://msdn.microsoft.com/es-es/library/bb972268.aspx.

vmapas.com Mapa topográfico. Mapa_Topografico_Algeria.jpg, vmapas.com. **1574 x 1507 - 2655 KB**

www.ua.es (2006). Cartograma con puntos. Cartograma_con_puntos-Num_casos_2002.jpg. **616 x 369 - 39 KB - jpg**.

Zhu, B. and H. Chen (2005). Information Visualization, Annual Review of Information Science and Technology.

Glosario

Antialiasing: Cualquier técnica que reduce la apariencia accidentada de los bordes en imágenes digitales tanto planas como en tres dimensiones.

API: Una API (acrónimo en inglés de *Application Programming Interface)* o Interface de Programación de Aplicaciones (en español) es un conjunto de funciones que facilitan el intercambio de mensajes o datos entre dos aplicaciones.

Applets: Es otra manera de incluir código a ejecutar en los clientes que visualizan una página web. Se trata de pequeños programas hechos en Java, que se transfieren con las páginas web y que el navegador ejecuta en el espacio de la página.

CORBA IDL (*Interface Definition Language*): Es un estándar que se usa para definir interfaces a los objetos en la red.

GIF (acrónimo en inglés de *Graphics Interchange Format*): Es un formato de compresión de imágenes limitado a 256 colores. Los archivos tipo gif utilizan un algoritmo de compresión de datos que está patentado.

GUI (acrónimo en inglés de *Graphical User Interface*): Una Interfaz Gráfica de Usuario es un conjunto de formas y métodos que posibilitan la interacción de un sistema con los usuarios.

IDE: Un entorno de desarrollo integrado (acrónimo en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación.

JDK (acrónimo en inglés de *Java Development Kit*): El Java Development Kit, JDK por sus siglas en inglés, es un grupo de herramientas para el desarrollo de software provisto por Sun Microsystems Inc. Incluye las herramientas necesarias para escribir, testear, y depurar aplicaciones y applets de Java.

JFreeReport: Biblioteca de la clase de Java para generar informes.

JSP (acrónimo en inglés de *Java Server Pages*)**:** Es una tecnología similar a los Servlets que ofrece una conveniente forma de agregar contenido dinámico a un archivo HTML por utilizar código escrito en

Java dentro del archivo utilizando etiquetas especiales que son procesadas por el servidor Web antes de enviarlas al cliente.

Oblea (en inglés, *wafer*): En microelectrónica, un wafer es una capa delgada de material semiconductor, generalmente de silicio.

Open Source (en español *Código Abierto*): es el término con el que se conoce al software distribuido y desarrollado libremente.

PNG (acrónimo en inglés de *Portable Network Graphics*): Es un formato de compresión de imágenes aprobado por el World Wide Web Consortium (W3C) como sustituto del formato .gif. Los archivos tipo .gif utilizan un algoritmo de compresión de datos que está patentado, mientras que el formato .png no está patentado y no necesita licencia para su utilización.

Señalética: Trasmitir una señal, reproducen el contenido de un mensaje sin referirse a su forma lingüística.

Shareware: Software con autorización de redistribuir copias, pero debe pagarse cargo por licencia de uso continuado.

Servlets: Los Servlets son módulos escritos en Java que se utilizan en un servidor, que puede ser o no ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java. Los servlets son para los servidores lo que los applets para los navegadores, aunque los servlets no tienen una interfaz gráfica.

SVG (acrónimo en inglés de *Scalable Vector Graphics*): es una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados.

Swing: Conjunto de componentes proporcionados por las APIs de Java para trabajar en un entorno gráfico de ventanas en aplicaciones cliente. Se empezó a utilizar a partir de las versiones 1.1 como una extensión aparte de la librería estándar Java, y más adelante se incluyó en la versión 1.2 como parte de las librerías estándar Java.

Visualización: Formación de la imagen mental de un concepto abstracto.