Universidad de las Ciencias Informáticas

Facultad 5

Título: Arquitectura y Diseño del Componente de Búsqueda de Indicadores de Perforación. (SEnDI).

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

Autor: Julio García de León

Tutor: Ing. David Tavares Cuevas

La Habana, junio de 2011

"Si buscas resultados distintos, no hagas siempre lo mismo"

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2011.

Julio García de León

_____

Firma del Autor

DATOS DE CONTACTO

Nombre y Apellidos: Ing. David Tavares Cuevas.

e-mail: dtavares@uci.cu

Ingeniero en Ciencias Informáticas, graduado en la Universidad de Ciencias Informáticas (UCI). Profesor de la UCI, con varios años de experiencia en su desempeño laboral. Actualmente se desempeña como jefe del proyecto “Sistema para el Manejo Integral de la Perforación de Pozos de Petróleo (SIPP).”

Agradezco a toda mi familia por todo su apoyo, consagración y dedicación durante todos estos años, sin sus consejos, mis logros actuales no serían realidad. A mi madre hermosa por su comprensión y consejos en todos los momentos difíciles de mi vida. A mi padre por confiar siempre en mí y apoyar mis decisiones. A mi segunda madre Dirma por tener siempre tanta fe en las cosas que hago. Agradecer a mi hermano Guille por cada día servirme de ejemplo y mano derecha para seguir adelante, a mí hermana Delia Rosa, por ser uno de mis motivos más importantes para ser mejor persona y guía. Agradecer también a Asael por toda su preocupación y apoyo en todo momento. A mis abuelos que extraño y adoro por estar siempre orgullosos de mí. A mi primo Michel por ser mi hermano y por su preocupación y preguntarme tantas veces ¿Cuando terminas?

También agradecer a mi novia Dannelys y mi suegra Belquis por estar en estos últimos días tan pendientes de mis resultados, por confiar tanto en mí y por apoyarme siempre que lo necesité, a mi primo Vladi y su novia Yarielis, por su preocupación y por decir que estoy escapado, así como a mi tía Maribel y mi tío Papito por decir que soy un buen sobrino.

A mis amigos que son como hermanos, Eduardo, Yadiel, Arturo, Adriel, Adrian, Andy, Ernesto Gonzalez, Danae, de los cuales sobran las razones para estar agradecido toda la vida, de corazón, gracias a todos por existir. A mi laptop "Lala" por soportarme todo este tiempo.

Agradecer a mi tutor David Tavares por todas las largas noches de dedicación y empeño para lograr buenos resultados y a todos los miembros del tribunal por sus críticas y recomendaciones.

A las gemelas más lindas del mundo Katy y Jeni, asi como a mi hermanita Peniel, por estar siempre online por el chat para desearme toda la suerte del mundo y por recordarme siempre que puedo lograr grandes cosas. A Evelyn por toda la ayuda incondicional que siempre me brindo y por estar siempre pendiente de mis cosas, a Anay por darme siempre un motivo para querer las cosas lindas, a Yaimara por ser tan especial y por revisar el documento de tesis tantas veces como pudo.

A todos mis amigos de Matanzas, a los de Venezuela, a todos los que han compartido conmigo algún tiempo de estos intensos 5 años de universidad. A todo el piquete de la FEU que junto crecimos y nos hicimos mejores personas y mejores mueleros, en especial a Felipe y Yidian por sus buenos consejos y apoyo incondicional y en general a todas las personas que conozco.

A La Revolución y al Comandante por este genial proyecto que hizo posible mi formación como ingeniero informático y como profesional.

Dedico el resultado de este trabajo de diploma a toda mi familia por su apoyo constante, confianza y dedicación durante estos 5 años de universidad y en especial a mis dos madres, Zoraida y Dirma, mis dos padres, Julio y Asael, mis abuelos, Rosario, Negrito, Elpidia y Jose Ramon, este último mi inspiración para todas las cosas y a mis dos hermanos, los cuales adoro, Delia Rosa y Guille.

Dedico también este trabajo a todos mis amigos y hermanos como suelo llamarlos, por estar siempre pendientes de mis cosas y por su apoyo incondicional.

También dedico este trabajo a todas las personas lindas que durante mis 24 años he tenido la dicha de conocer y pretendo con este logro, estén cada día más orgullosos de conocer a alguien como yo, gracias por todo.

RESUMEN

En la actualidad existen enormes dificultades para la recuperación de información contenida en diversas fuentes de datos, es por esto que las herramientas para la realización de búsquedas dinámicas han experimentado un auge vertiginoso. El presente trabajo surge como una necesidad de la Dirección de Intervención y Perforación de Pozos (DIPP) y Centro de Investigaciones de Petróleo (CEINPET), para solucionar situaciones problemáticas existentes en la recuperación de información de indicadores de perforación. Con el fin de cumplimentar los objetivos propuestos se investigó acerca de los principales indicadores de perforación y se realizó un estudio de los principales sistemas existentes para la industria del petróleo. De esta forma se llegó a la conclusión de realizar la arquitectura y diseño del componente para buscar indicadores de perforación, con el objetivo de guiar y facilitar su futuro desarrollo e implementación.

Palabras clave: arquitectura, búsqueda, diseño, indicadores, perforación.

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Asistimos, cada día, al crecimiento exponencial de un volumen de información manipulada por los sistemas informáticos y a su carácter acumulativo, gracias al avance de las tecnologías de almacenamiento. En tanto la capacidad humana para la lectura se mantiene constante, los usuarios se enfrentan a la limitación de encontrar y leer únicamente lo que se ajusta a sus necesidades, ante un volumen informativo que supera con creces sus posibilidades de localización y entendimiento. En la actualidad existen enormes dificultades para la recuperación de información contenida en diversas fuentes de datos (Tec, 2009).

Los buscadores, además de influir decisivamente en la primera impresión de los usuarios, son herramientas cruciales, junto a la arquitectura de la información y a los sistemas de navegación para que los usuarios encuentren lo que buscan. Por ello, su utilización condiciona significativamente la usabilidad global del sistema, y debe prestársele especial atención según planteaba Nielsen, reconocido gurú de la usabilidad que ha comprobado en numerosas pruebas que más de la mitad de los usuarios, intentan encontrar la información que necesitan utilizando su buscador, aproximadamente un quinto intentan encontrarla siguiendo los enlaces que se les presentan y el resto muestran un comportamiento mixto (Nielsen, 2001).

Hoy en día las herramientas para la realización de búsquedas dinámicas han experimentado un auge extraordinario debido al avance de las Tecnologías de la Información y las Comunicaciones (TIC). La creciente informatización de casi la totalidad de las empresas, ha traído consigo la digitalización y almacenamiento de volúmenes considerables de informaciones, propiciado por el avance de las tecnologías de almacenamiento (Tec, 2009).

Existen diversos productos desarrollados para el sector del Petróleo, encaminados a resolver problemáticas dentro de esta esfera, principalmente en el proceso de recuperación de parámetros que ayudan en la toma de decisiones, generados en operaciones realizadas en los pozos en perforación. Actualmente no existe una solución que resuelva el problema planteado en este trabajo, están implementadas en sistemas operativos privativos y sus precios en el mercado son elevados.

La Oficina Central de Exploración Producción de la Unión Cuba Petróleo (CUPET) sufre estos problemas en la actualidad. Esta, se encuentra inmersa en nuevos proyectos de expansión de sus principales procesos productivos. Las expectativas que se plantean sobre las posibilidades reales del país de la exploración, perforación y producción de petróleo, en las aguas profundas del Golfo de México (ZEE), han propiciado un ambiente de trabajo favorable para la realización de convenios de trabajo entre la Universidad de las Ciencias Informáticas (UCI) y CUPET. Dada la necesidad de informatizar sus principales procesos, que elevaría grandemente la eficiencia y competitividad de

esta empresa en el marco nacional e internacional, surge el convenio de colaboración UCI-MINBAS.

En la UCI, se encuentra el Centro de Desarrollo de Informática Industrial (CEDIN); en él se encuentra el proyecto **Sistema para el Manejo Integral de la Perforación de Pozos de Petróleo (SIPP)**. Este proyecto está a punto de concluir su primer ciclo de desarrollo, y responde a las necesidades de la Unión Cuba Petróleo (CUPET). La Dirección de Intervención y Perforación de Pozos, previo al uso del sistema MaximusDrillPro (Producto del Proyecto SIPP), presentaba problemas para el acceso a la información para la toma de decisiones. Esto se debía a que no existía un orden para el almacenamiento de los datos, los cuales se guardaban en hojas de cálculo, utilizando Microsoft Excel 2003.

Luego de la implantación del sistema antes mencionado, se solucionaron estos problemas parcialmente, ya que esta primera versión de la solución aún no presenta funcionalidades que permitan el acceso dinámico a los datos. Los reportes generados por el sistema, no pueden ser personalizados, ni configurados dinámicamente según la necesidad de información de los especialistas. Además si se desea vincular información de varios indicadores para la toma de decisiones, se tendrían que generar varios reportes. Entre los componentes conceptualizados por el equipo de análisis de SIPP se encuentra el desarrollo de un componente que permita la búsqueda de indicadores, el cual forma parte del Subsistema Presentación, tiene como objetivo realizar búsquedas que el usuario determine de manera dinámica mediante criterios de búsquedas. Esto permitiría un acceso pleno y flexible a la información, lo cual contribuiría directamente al proceso en la toma de decisiones.

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

¿Cómo garantizar el acceso dinámico a la información resultado del proceso de perforación, contenida en la base de datos del SIPP?

Para dar solución al problema se ha trazado como **objetivo general** de la investigación: Realizar la arquitectura y diseño del Componente de Búsqueda de Indicadores de Perforación para el área de perforación en pozos petroleros. Para darle cumplimiento a dicho objetivo se planteó como **objeto de estudio** el proceso de recuperación de información de indicadores en el área de perforación en pozos petroleros enmarcados en el **campo de acción** del proceso de recuperación de información en el **Sistema para el Manejo Integral de la Perforación de Pozos de Petróleo y Gas (SIPP).** Como **idea a defender** para la realización de la investigación, se propone que: con la realización de la propuesta de arquitectura y diseño del Componente de Búsqueda de Indicadores de

Perforación, se podrá implementar un componente que permita el acceso dinámico a la información contenida en la base de datos del SIPP.

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

1. Descripción del estado del arte de los componentes desarrollados con este fin a nivel nacional e internacional, con el objetivo de identificar ventajas y desventajas de estos.
2. Identificación de las metodologías más utilizadas en el desarrollo de componentes, así como métodos y herramientas a utilizar para el correcto desarrollo de la investigación.
3. Identificación de los principales indicadores utilizados en el área de perforación de la Industria Petrolera para obtener el conocimiento genérico y abstracto para su manipulación.
4. Modelación del componente para la búsqueda de indicadores.
5. Evaluación de la propuesta de solución para garantizar la calidad.

El resultado que se espera de la investigación radica en, la arquitectura y diseño del componente para buscar indicadores de perforación de pozos de petróleo, que dotará a los especialistas del CUPET de una herramienta para realizar búsqueda de indicadores y tomar mejores decisiones con la vinculación de estos.

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

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

Entre estos se empleó:

- Analítico-Sintético, con el objetivo de analizar y aumentar los conocimientos respecto al proceso de recuperación de información de los indicadores en el área de perforación en pozos petroleros de la empresa CUPET, a partir de consultar la bibliografía científica correspondiente, para después, haciendo uso de la síntesis, lograr resumir y exponer los resultados obtenidos del análisis.
- Histórico-Lógico, con el objetivo de identificar y verificar la existencia de sistemas que se encarguen del proceso de recuperación de información y entender cómo se realizan dichos procesos, comprender y dominar todo el entorno que rodea al Sistema para el Manejo Integral de la Perforación de Pozos de Petróleo y Gas, así como los demás factores que intervienen en esos procesos de recuperación.
- Modelación, para la caracterización de las funcionalidades que tendrá el componente de búsqueda de indicadores cuando se estén definiendo los requisitos funcionales.

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

Entre estos se empleó:

- Observación, con el objetivo de obtener un registro visual de toda la información referente a los procesos de perforación, el funcionamiento del centro en una situación real y del Sistema para el Manejo Integral de la Perforación de Pozos de Petróleo y Gas.
- Entrevista, al personal que se encarga del análisis en los sistemas de perforación para recopilar información confiable y real de las necesidades del centro, y que se tornan indispensables para la solución del problema de investigación; a los líderes de proyecto para obtener información sobre las herramientas de búsqueda más utilizadas en el Centro CEDIN.

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

Capítulo 1. "Fundamentación Teórica": En este capítulo se analizan los principales aspectos relacionados con la recuperación de información de los procesos de perforación en la Industria Petrolera. Se enuncian conceptos que posibilitan un mejor entendimiento a lo planteado en la situación problemática y problema en general. Se hace referencia al comportamiento de los indicadores en el área de perforación, se presenta la descripción del estado de arte con las soluciones que existen alrededor del objeto de estudio, se plantean los objetivos generales y específicos.

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

Capítulo 3. "Presentación de la solución propuesta": Se describen los procesos del negocio relacionados con el objeto de estudio. Se presentan los requisitos funcionales y no funcionales que debe tener el software, se muestra la concepción del sistema a partir del diagrama de casos de uso y las descripciones de los mismos. Se desarrolla el diseño de la propuesta de solución partiendo de la explicación de los patrones de diseño que se utilizan para modelar el software, se propone la arquitectura para guiar la realización de los principales aspectos del sistema.

Capítulo 4. "Validación de la propuesta de solución": En este capítulo se describen las técnicas, métodos y atributos de calidad que pueden ser usados para evaluar la arquitectura y diseño de la propuesta de solución. Para su validación se selecciona además uno de estos métodos para realizar la evaluación y dejar descritos los resultados.

CAPÍTULO 1. Fundamentación Teórica.

1.1. Introducción.

En este capítulo se abordarán aspectos fundamentales sobre la recuperación de información de los procesos de perforación en la Industria Petrolera, constatando su utilidad en el proceso de análisis estadístico y toma de decisiones. Se hace referencia al comportamiento de los indicadores en el área de perforación, así como la descripción de los más importantes. Se presenta la descripción del estado de arte con las soluciones que existen alrededor del objeto de estudio. Se describen los distintos tipos de buscadores que existen para recuperar informaciones de sitios web dinámicos. Se aborda de forma detallada el objeto de estudio y la situación problemática y se analizan los principales conceptos relacionados con los procesos en la industria petrolera.

1.2. Conceptos asociados al dominio del problema.

Para mejor entendimiento y desenvolvimiento de las áreas temáticas que se abordarán en el presente y posteriores capítulos, se mostrarán un grupo de conceptos identificados durante la investigación realizada, así como la descripción de cada uno facilitando así la comprensión de los temas abordados.

Componente.

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

Un componente es:

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

[1] SEI, Instituto de Ingeniería de Software de la Universidad Carnegie Mellon en Estados Unidos para desarrollar modelos de evaluación y mejora en el desarrollo de software.
[2] Michael Sparling es director de Tecnología en Castek, Canadá, una consultora en Ingeniería de Software.

- Según Clemens Szyperski[3], una unidad binaria de producción, adquisición y despliegue independiente, que interactúa con otras para formar un sistema que funciona.
- Bertrand Meyer[4] establece siete criterios que ayudan a definir lo que es un componente en el ambiente de un sistema y las cualidades que le dan valor en el marco de un proyecto de CBD (Component Based Development).

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

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

Buscar.

Se conoce con el término de buscar a aquella actividad que se caracteriza principalmente por un propósito de descubrimiento. El objeto de esa búsqueda es dar con alguna región en particular que se presume existe, descubrir algún tipo de recurso que servirá para por ejemplo enriquecer las arcas de una nación como puede ser el petróleo, gas, carbón, algún otro tipo de mineral o estar orientada a la búsqueda de determinada información en pos de satisfacer alguna necesidad (Florencia, 2008).

La acción de buscar es tan antigua y se viene poniendo en práctica desde que la humanidad se propuso dar sus primeros pasos, los seres humanos buscan cosas, información, que contribuyan a la supervivencia, porque la búsqueda es uno de los tres pilares que conforman el proceso de la

[3] Clemens Szyperski es Profesor Agregado en la Facultad de Tecnología de Información de la Universidad Queensland en Australia y es autor de un libro fundamental en el tema de componentes.
[4] Bertrand Meyer es el autor del lenguaje de programación orientado a objetos Eiffel y fundador de ISE Inc.

investigación científica, que junto a los otros dos, descripción y explicación, completan el círculo (Florencia, 2008).

Motor de Búsqueda.

Un motor de búsqueda es un sistema informático que busca archivos almacenados en servidores de datos. Un ejemplo son los buscadores de Internet (algunos buscan sólo en la Web pero otros buscan además en noticias, servicios como Gopher, FTP, etc.) cuando se pide información sobre algún tema. Las búsquedas se hacen con palabras clave o con árboles jerárquicos por temas; el resultado de la búsqueda es un listado de direcciones Web en los que se mencionan temas relacionados con las palabras clave buscadas (Motores, 2009).

Se pueden clasificar en dos tipos:

- Índices temáticos: Son sistemas de búsqueda por temas o categorías jerarquizados (aunque también suelen incluir sistemas de búsqueda por palabras clave). Se trata de bases de datos de direcciones Web elaboradas "manualmente", es decir, hay personas que se encargan de asignar cada página web a una categoría o tema determinado.

- Motores de búsqueda: Son sistemas de búsqueda por palabras clave. Son bases de datos que incorporan automáticamente páginas web mediante "robots" de búsqueda en la red.

Recuperación de Información.

La recuperación de información, llamada en inglés information retrieval (IR), es la ciencia de la búsqueda de información en documentos, búsqueda dentro de los mismos, búsqueda de metadatos que describan documentos, o también la búsqueda en bases de datos relacionales, ya sea a través de internet, intranet, para textos, imágenes, sonido o datos de otras características, de manera pertinente y relevante. (Salvador Oliván, 2008).

Con lo anteriormente expuesto, se puede determinar que el término buscar, es la acción de descubrir algo para satisfacer alguna necesidad o completar un trabajo de estudio. Estas búsquedas son realizadas mediante software denominados motores de búsquedas que son los encargados de buscar informaciones en un determinado medio o sistema de almacenamiento. La realización de estas acciones, utilizando estos sistemas es lo conocido como proceso de recuperación de información, constituye un factor clave en el desarrollo de sistemas informáticos que manipulan grandes volúmenes de información.

Indicador.

Elemento de estadística que permite conocer el avance de un programa o actividad. Puede consistir en porcentajes, etapas, número de operaciones, etcétera, implica la comparación entre lo programado y lo alcanzado, proporcionando la desviación en la ejecución del programa o actividad en el período determinado (ife.org, 2005).

Elemento que indica cierta condición, capacidad o medida numérica que, al registrarse, recopilarse y analizarse, facilita que los conceptos más complejos sean más susceptibles de medición y permite a los administradores y evaluadores comparar los resultados reales del programa con los resultados que se esperan (ife.org, 2005).

Petróleo.

El petróleo es una mezcla heterogénea de compuestos orgánicos, principalmente hidrocarburos insolubles en agua, que se extrae de lechos geológicos continentales o marítimos. También es conocido como petróleo crudo o simplemente crudo. Es un líquido aceitoso, inflamable, cuyo color varia de incoloro a negro (Paneque, 2010).

Pozo de Petróleo y Gas.

Entidad donde se realiza la perforación, la cual agrupa un gran número de entidades y trabajadores, como son las compañías de servicio de lodo, direccionales, contratistas, logísticas, entre otras. (Paneque, 2010)

Industria del Petróleo.

La industria petrolera incluye procesos globales de exploración, extracción, refino, transporte (frecuentemente a través de buques petroleros y oleoductos) y mercadotecnia de productos del petróleo. (Paneque, 2010)

CUPET.

Unión de Empresas de Cuba encargadas de toda la actividad petrolera desde la exploración hasta la refinación, así como de satisfacer las necesidades del mercado nacional de hidrocarburos, a partir del incremento de la extracción de crudos nacionales. (Rodríguez, 2010)

Arquitectura de Software.

Son muchas las definiciones que se han dado sobre el concepto de Arquitectura de Software, según Pressman la arquitectura no es el software operacional, más bien, es la representación que

capacita al ingeniero del software para: analizar la efectividad del diseño para la consecución de los requisitos fijados; considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil; y reducir los riesgos asociados a la construcción del software (Pressman, 2002).

La definición de este concepto planteada por Bass, Clements y Kazman plantea que la Arquitectura de Software de un sistema de programa o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos (Bass, 1998).

En el sitio oficial del Instituto de Ingeniería de Software (SEI por siglas en inglés) se encuentra publicada una definición en la que se expone que la Arquitectura de Software sirve para modelar el sistema y el proyecto en desarrollo, definiendo las asignaciones de trabajo que deben llevarse a cabo por los equipos de diseño e implementación. La arquitectura es el soporte principal de las cualidades del sistema, tales como el rendimiento, modificabilidad y la seguridad, ninguno de los cuales se puede lograr sin una visión arquitectónica unificadora; Es un artefacto para el análisis temprano de un proyecto, permite asegurarse de que el enfoque de diseño de un sistema tiene un rendimiento aceptable. En resumen, la arquitectura es el pegamento conceptual que sostiene todas las fases del proyecto junto a todas las partes interesadas. (SEI, 2011)

El concepto dado por el SEI es uno de los más completos y al unirlo con los demás conceptos analizados se puede concluir que la arquitectura es el pegamento conceptual que sostiene todas las fases del proyecto junto a todas las partes interesadas, en ella se incluyen los principales componentes, cómo se comportan y cómo interactúan entre ellos. La Arquitectura de Software aporta grandes beneficios en cuanto al ahorro de tiempo, la organización del trabajo, el entendimiento entre las partes que intervienen en el desarrollo y posterior mantenimiento del software, además brinda una visión de cómo quedará estructurado el sistema y el modo en que sus componentes interactúan entre sí.

1.3. Buscador para el acceso dinámico a la información.

En la Universidad de las Ciencias Informáticas, se encuentra el Centro de Desarrollo de Informática Industrial (CEDIN); en él se encuentra el proyecto **Sistema para el Manejo Integral de la Perforación de Pozos de Petróleo (SIPP)**. Dicho proyecto está a punto de concluir su primer ciclo de desarrollo, y responde a las necesidades de la Unión Cuba Petróleo (CUPET).

En muchas ocasiones, a las unidades técnicas se les solicita realizar los reportes sobre la situación existente en los pozos en perforación; para ello es necesario la utilización de la información perteneciente al área de perforación, el trabajo suele demorar varios días y hasta semanas debido

a que los datos deben ser extraídos de varias hojas de cálculo de Microsoft Excel para lograr conformar el reporte solicitado y en ocasiones suele ser considerable el volumen de información a procesar. Esta demora provoca que el proceso sea lento y difícil. La Dirección de Intervención y Perforación de Pozos (DIPP), previo al uso de la solución MaximusDrillPro (Producto del Proyecto SIPP), presentaba este tipo de problemas para el acceso a la información para la toma de decisiones. Esto se debía a que no existía un orden para el almacenamiento de los datos, los cuales se guardaban en hojas de cálculo, utilizando Microsoft Excel 2003, un software del paquete de oficina de la empresa Microsoft.

Luego de la implantación de la solución antes mencionada, se solucionaron los problemas parcialmente, ya que esta primera versión de la solución aun no presenta funcionalidades que permitan el acceso dinámico a los datos. Entre los componentes conceptualizados por el equipo de análisis de SIPP se encuentra el desarrollo de un componente que permita la búsqueda de indicadores. El Componente Búsqueda de Indicadores, el cual forma parte del Subsistema Presentación, tiene como objetivo realizar búsquedas que el usuario determine de manera dinámica. Esto permitiría un acceso pleno y flexible a la información, lo cual contribuiría directamente al proceso en la toma de decisiones.

En la Unión Cuba Petróleo se cuenta con un conjunto de software propietarios, entre ellos WellWizard, que se encargan de interpretar los registros de pozos. Estos software poseen licencias privativas (copyrigth), lo que unido al difícil acceso de las actualizaciones, provoca que parte de los trabajadores que laboran con el mismo se vean excluidos de las principales funcionalidades que brindan. Ninguno de estos software mencionados anteriormente no permiten la recuperación dinámica de información con motores de búsquedas ni otras herramientas; se plantea la necesidad de modelar la arquitectura y diseño de un componente que permita recuperar la información del comportamiento de indicadores para el área de la perforación de la Industria Petrolera, lo que posibilitará, el ahorro de tiempo y esfuerzo para realizar dicho proceso.

1.4. Proceso de recuperación de indicadores en el área de perforación en pozos petroleros.

El proceso donde se accede a una información previamente almacenada, mediante herramientas informáticas que permiten establecer ecuaciones de búsqueda específicas, donde dicha información ha debido ser estructura previamente a su almacenamiento, es conocido como proceso de recuperación. Este se lleva a cabo mediante consultas a la base de datos donde se almacena la información estructurada, mediante un lenguaje de interrogación adecuado (Salvador Oliván, 2008). Es necesario tener en cuenta los elementos clave que permiten hacer la búsqueda,

determinando un mayor grado de pertinencia y precisión, como son: los índices y palabras clave, que juntos conforman un indicador en el área de perforación de pozos. Se puede entonces decir que el proceso de recuperación de indicadores no es más que la secuencia de acciones necesarias que se ejecutan para obtener una información a través de un buscador de indicadores.

1.4.1. Descripción General.

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

A partir del análisis de los resultados obtenidos por la búsqueda, es posible arribar a conclusiones que favorecen a agilizar las decisiones que se tomarán posteriormente en la perforación o inversión de un pozo de petróleo; si es factible perforar un nuevo pozo, hasta qué profundidad se ha perforado y cuántos días han demorado en alcanzar esa profundidad. Algunas empresas utilizan esta información para hacer representaciones gráficas y generar reportes con gran calidad y precisión. Un ejemplo de ello, es la compañía DATALOG que emplea el Software WellWizard utilizado en la mayoría de los pozos en perforación de CUPET. A pesar de ello, todavía no es posible acceder a esta información de manera dinámica y mediante criterios de búsquedas definidos por el usuario.

Realizar un motor de búsqueda orientado a elementos del negocio no es de fácil construcción y hacer un buen componente de búsqueda consiste en la comprensión de lo complejo, no en la complicación de lo simple y esto no se consigue sin un cierto esfuerzo y la clara comprensión de qué es lo que se quiere obtener y de qué forma se va a visualizar la información resultante.

Etapas del proceso de búsqueda y recuperación de la Información: (La biblioteca, 2001)

1. **Identificación de la necesidad de información.** ¿Qué información se necesita?, es importante identificar la información que se necesita en función de las tareas a resolver, luego concretar la necesidad de información, determinar sus características, si son exhaustiva o cobertura (temática, geográfica y cronológica), nivel de compresión, tener en cuenta idioma y por último el formato.
2. **Identificación de las fuentes de información.** ¿Qué fuentes de información utilizar?, identificar el tipo de fuentes apropiadas en función de la necesidad de información.

3. **Localización y recuperación de la información.** ¿Cómo y dónde localizar la información?, hay que determinar una o varias estrategias, diseñar los itinerarios de búsqueda, buscar fuentes de información en diferentes entornos.
4. **Selección y valoración de la información.** ¿Qué se encontró de lo que se buscaba?, analizar y valorar los resultados de la búsqueda. Obtener la información más útil o relevante en función de la necesidad de información y el nivel requerido.
5. **Evaluación del proceso.** ¿Se ha realizado todo el proceso de búsqueda correctamente?

Este proceso tiene además gran importancia para la industria petrolera y específicamente para los geólogos, químicos, direccionales, supervisores del pozo, investigadores y trabajadores interesados en los temas referentes a las perforaciones petroleras ya que una gran cantidad de decisiones son tomadas con la información obtenida a partir del análisis de indicadores.

Comportamiento de Indicadores en el Área de Perforación.

Para poder conocer el comportamiento de un indicador en el área de perforación deben utilizarse técnicas y herramientas para medir los parámetros necesarios a evaluar, que permitan conocer la interacción entre ellos. En la perforación petrolera un indicador no es más que números o índices que tienen un valor indicativo, datos que son obtenidos, pues sirven de instrumento de medida.

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

Perforación Direccional.

Perforar direccionalmente es dirigir un pozo de manera controlada a lo largo de una trayectoria planeada para alcanzar un objetivo geológico. La planificación de un pozo direccional se realiza como un paso previo a la perforación del mismo. Es responsabilidad del Planificador Direccional, quien generalmente trabaja en conjunto con el Ingeniero de Perforación, Geólogos y Supervisores del Pozo. Cuando se planifican pozos direccionales se utilizan distintos métodos como el Tangencial, Ángulo Promedio, Radio de Curvatura y Curvatura Mínima este último es el más preciso y el estándar usado en la industria.

1.4.2 Descripción actual del dominio del problema.

Actualmente el proyecto vinculado al área de perforación SIPP necesita contar con un componente que automatice el proceso de recuperación de la información de los indicadores de dicha área, por lo que se hace necesaria la descripción del entorno donde coexiste el negocio y las organizaciones que la rodean. El entorno del negocio estudiado se encuentra dividido en tres áreas fundamentales:

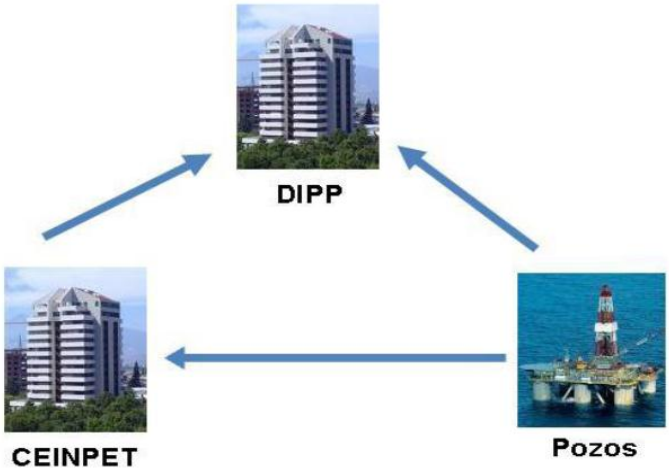

Figura 1. Distintas áreas del negocio.

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

CEINPET: Perteneciente a la Unión de CUPET, la misma está localizada en El Cerro, provincia La Habana. Aquí diariamente la información es recibida en forma de reporte a las 7:00am por correo electrónico al igual que en la DIPP en caso de error con la conexión se realiza por teléfono, luego de ser recibida y gestionada a mano utilizando el software *Microsoft Excel* para realizar el parte diario de geología y el *Microsoft Paint* para dibujar poco a poco la columna litológica que contiene todos los datos de litología de los pozos en perforación, es decir los tipos de rocas, formaciones existentes en los distintos intervalos en los cuales se han perforado, edad de dichas formaciones,

para luego ser guardados en formato duro y digital, llevando de esta forma un historial con todos los documentos realizados en una carpeta.

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

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

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

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

Existen diversos productos desarrollados para el sector del Petróleo, encaminados a resolver problemáticas dentro de esta esfera, principalmente en el proceso de recuperación de parámetros que ayudan en la toma de decisiones, generados en operaciones realizadas en los pozos en perforación. Actualmente no existe una solución que resuelva el problema planteado en este trabajo, están implementadas en sistemas operativos privativos y sus precios en el mercado son elevados. Próximamente se realiza un análisis de las principales características que poseen, identificando debilidades y fortalezas que servirán de base a la propuesta a desarrollar.

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

La perforación es la segunda etapa en la búsqueda de petróleo. Para realizarla, se perfora un pozo mediante la utilización de una sonda. La profundidad de un pozo es variable, dependiendo de la región y de la profundidad a la cual se encuentra la estructura geológica o formación seleccionada con posibilidades de contener petróleo. Una vez comprobada la existencia del petróleo, otros pozos se perforarán para evaluarse la extensión del yacimiento. Esta información es la que va a determinar si es comercialmente viable o no producir el petróleo descubierto. Durante estas etapas se generan volúmenes considerables de datos y reportes, debido a esto se han elaborado distintas aplicaciones para ayudar al mejor cumplimiento de las labores de los especialistas en sus distintas áreas.

WellWizard.

Sistema Integrado de Datos (WellWizard) es un sistema versátil que utiliza cualquier servicio de flujo de datos de pozo. Todos los datos geológicos y de perforación están integrados, en tiempo real, a través de la interfaz gráfica WellWizard. WellWizard IDS es configurable para integrar datos de múltiples fuentes, incluyendo la perforación direccional, de medición durante la perforación, el registro durante la perforación, de ensayo y producción.

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

No posee ninguna herramienta para la recuperación dinámica de información de indicadores de perforación.

WellFlo 3.

WellFlo es una suite de programas que modelan y optimizan pozos de crudo, gas y redes de producción para ingenieros petroleros que diseñen, pronostiquen desempeños, diagnostiquen problemas de pozo u optimicen producciones desde instalaciones existentes. La versión 3 de WellFlo brinda un nuevo nivel de sofisticación, velocidad y precisión. Al mismo tiempo, el programa permanece suficientemente simple de usar, asegurando un aprendizaje rápido y un alto nivel de productividad un factor importante en la altamente exigente industria del petróleo y gas de hoy.

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

Modelado de Pozos y realización de Pozos Horizontales. (Modelado de Pozos 2009)

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

No posee ninguna herramienta para la recuperación dinámica de información de indicadores de perforación.

InfoProd.

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

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

Algunas características:

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

No posee ninguna herramienta para la recuperación dinámica de información de indicadores de perforación.

Strater.

Strater es una potente e innovadora herramienta para el registro de pozos y el trazado de perforaciones. Con esta herramienta los geólogos ya no tendrán que invertir más tiempo y dinero para crear registros profesionales de pozos. Strater es una herramienta potente, sencilla de utilizar y con un precio muy asequible. Ofrece una flexibilidad insuperable para el diseño y formateo de registros. Su avanzada interfaz de usuario permite que el diseño y la visualización de sus datos sean más fáciles que nunca. Proporciona varias funcionalidades para simplificar la tarea de importar datos, crear el diseño de la perforación exacta que el usuario requiere y obtener la salida en el formato necesario, ya sea impreso o exportado a formato electrónico para incluir en un informe o presentación.

No posee ninguna herramienta para la recuperación dinámica de información de indicadores de perforación.

Resultados del análisis de las soluciones existentes.

Después de analizar las principales herramientas utilizadas en el proceso de gestión de los datos de la perforación de pozos, en el ámbito nacional e internacional, se pudo determinar que estas no poseen ninguna herramienta que permita de manera dinámica obtener información de los indicadores de perforación, ya que las mismas brindan esta información mediante detallados reportes y gráficas de distintos tipos no configurables dinámicamente por el usuario.

1.6. Patrones de Diseño.

Los patrones son soluciones comunes a problemas de diseño de software orientado a objetos y que además poseen ciertas características de efectividad para resolver ese problema. Son reusables ya que pueden ser aplicados en otros diseños o problemas. Son una disciplina problema solución que está en constante evolución entre los diseñadores y desarrolladores que trabajan con lenguaje orientado a objetos.

Un patrón es:

- La abstracción de una forma concreta que puede repetirse en contextos específicos.
- Una información que captura la estructura esencial y la perspicacia de una familia de soluciones probadas con éxito para un problema repetitivo que surge en un cierto contexto y sistema.
- Una unidad de información nombrada, instructiva e intuitiva que captura la esencia de una familia exitosa de soluciones probadas a un problema recurrente dentro de un cierto contexto.

Los Patrones de diseño se clasifican en:

- Patrones de creación: Inicialización y configuración de objetos.
- Patrones estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases u objetos se agrupan, para formar estructuras más grandes.
- Patrones de comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

Aunque cuando se habla sobre los patrones salen a relucir los patrones de diseño como los más importantes en el mundo de la programación orientada a objeto, estos no son los únicos que existen ya que actualmente hay una inmensa cantidad de patrones que sería imposible para un programador el tratar de conocerlos todos, a continuación se muestra una cantidad modesta de estos:

- **Patrones de Arquitectura:** Soluciones probadas para estructurar los componentes, como son Modelo-Vista-Controlador, arquitectura en tres capas, pares a pares (peer to peer), arquitectura orientada a servicios, por citar ejemplos.
- **Patrones Web:** Soluciones probadas para la creación de sitios web, como son la maquetación en tres columnas, efectos de rollover, estructuras de blog.
- **Patrones de Diseño:** Las soluciones probadas para el diseño de software.
- **Patrones de Programación:** Soluciones específicas para algoritmos y estructuras de control, como son algoritmos de ordenación, procedimientos de recursión e iteración, etc.
- **Patrones de Refactorización:** Soluciones para simplificar el código, como la variable explicativa, extracción de métodos, sobrecarga de constructores.

1.7. Conclusiones Parciales.

En este capítulo fueron abordados diferentes temas con fundamento teórico, entre los más importantes se encuentran los diferentes conceptos sobre las áreas temáticas que se abordarán en la investigación, descripción detallada del objeto de estudio, descripción del problema a resolver, así como algunos de los principales software existentes en la industria del petróleo y que se utilizan en nuestro país. Se concluye que el proceso de recuperación de datos a través de indicadores es un proceso complejo el cual requiere identificar correctamente los indicadores a analizar. De los sistemas analizados solamente el WellView de Pelotón, contiene una funcionalidad para buscar datos, pero solo lo hace para texto dentro de los reportes con que trabaja.

CAPÍTULO 2. Herramientas y tecnologías actuales a considerar.

2.1. Introducción.

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

2.2. Metodologías de Desarrollo de Software.

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

Grady Rumbaugh[5] dio la siguiente definición:

"Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo". (Jacobson, 2000)

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte se tienen aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán, las cuales son conocidas como "metodologías complejas o formales". Otra propuesta es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software, siendo esta la filosofía de las llamadas "metodologías ágiles", las cuales dan mayor valor al individuo, a la colaboración con el cliente y al

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

desarrollo incremental del software con iteraciones muy cortas. (Hamar, 2004) Se analizarán a continuación varias de las metodologías más conocidas, abordando algunas de sus características.

RUP (Rational Unified Process)

RUP es una de las metodologías pesadas más conocidas y utilizadas en el desarrollo de software. Esta metodología divide el desarrollo en 4 fases que definen su ciclo de vida:

- Inicio: El objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
- Elaboración: Etapa en la que se determina la arquitectura óptima del proyecto.
- Construcción: Se obtiene la capacidad operacional inicial.
- Transmisión: Obtener el producto acabado y definido.

RUP determina 6 flujos de trabajos que se puede observar en la figura 2 y serán descritos a continuación:

Ingeniería o modelado del negocio: Analizar y entender las necesidades del negocio para el cual se está desarrollando el software.
Requisitos: Proveer una base para estimar los costos y tiempo de desarrollo del sistema.
Análisis y diseño: Trasladar los requisitos analizados anteriormente a un sistema automatizado y desarrollar una arquitectura para el sistema.
Implementación: Crear software que se ajuste a la arquitectura diseñada y que tenga el comportamiento deseado.
Pruebas: Asegurarse de que el comportamiento requerido es correcto y que todo lo solicitado está presente.
Despliegue: Producir distribuciones del producto y distribuirlo a los usuarios.

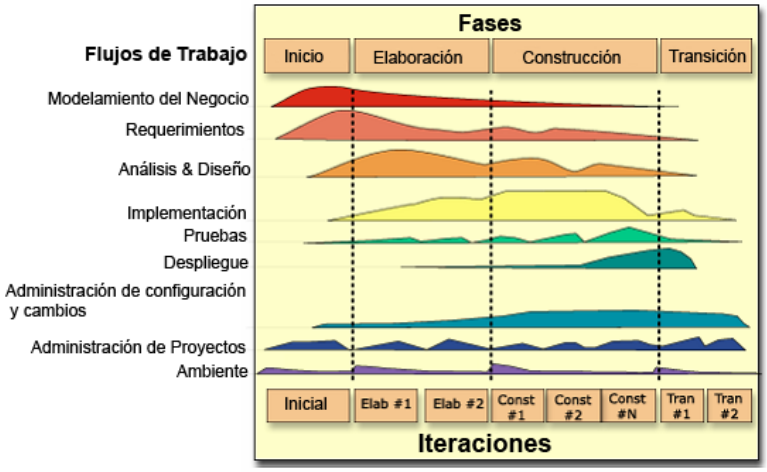

Figura 2. Fases e iteraciones de RUP.

Entre sus características se encuentra que utiliza el Lenguaje Unificado de Modelado (UML), para la confección de todos los esquemas de los sistemas; es una metodología dirigida por casos de uso, centrada en la arquitectura, iterativo e incremental, eliminando así los errores cometidos en las iteraciones previas; es usada comúnmente para proyectos de larga duración y alta complejidad; los elementos que componen son: las *actividades*, los *trabajadores* y los *artefactos;* en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo de software; se representa mediante un número diferente de vistas arquitectónicas llamadas el "modelo de vista 4+1" que se muestra en la figura 3 (Kruchten, et al., 2000).

Figura 3. Modelo de vista 4+1.

Proceso Ágil Unificado (AUP).

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

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

La figura 4 representa el ciclo de vida de AUP, con respecto a RUP las disciplinas han cambiado. Primero, la disciplina de Modelado abarca las disciplinas de Modelado del Negocio, Requerimientos

y de Análisis y Diseño. Segundo, la disciplina de la Administración de la Configuración y Cambios ahora es la disciplina de la Administración de la Configuración.

La naturaleza serial en AUP es capturada en cuatro fases: (UP, 2005)

1. **Iniciación:** El objetivo es identificar el alcance inicial del proyecto, una arquitectura potencial de su sistema, y obtener la financiación inicial del proyecto y la aceptación del involucrado.
2. **Elaboración**: El objetivo es mejorar la arquitectura del sistema.
3. **Construcción**: El objetivo es construir software funcional en una base regular e incremental, la cual cumpla con las necesidades de prioridad más alta de los involucrados de su proyecto.
4. **Transición**: El objetivo es validar y desplegar su sistema en su ambiente de producción.

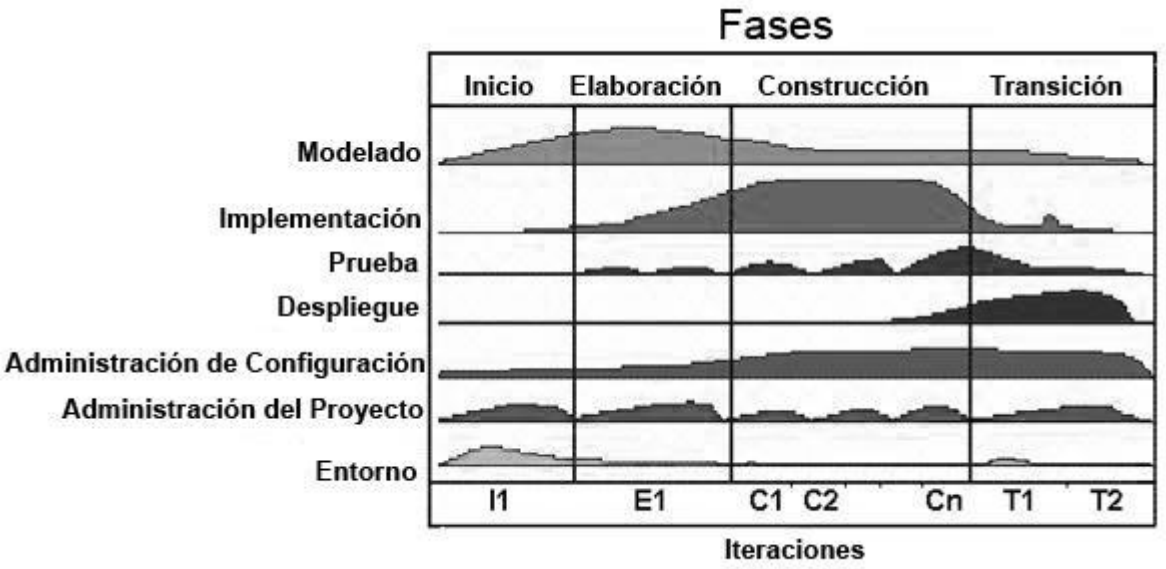

Figura 4. Fases y flujos de trabajo definidos por AUP.

Las disciplinas son ejecutadas en una manera iterativa, definiendo las actividades, las cuales los miembros del equipo ejecutan para construir, validar y liberar software funcional que cumpla con las necesidades de sus involucrados. Las disciplinas son: (UP, 2005)

- **Modelado:** El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se aborda en el proyecto, e identificar las soluciones viables para manejar el dominio del problema.
- **Implementación**: El objetivo de esta disciplina es transformar su modelo(s) en código ejecutable y llevar a cabo un nivel básico de las pruebas, en particular, la unidad de prueba.

- **Pruebas**: El objetivo de esta disciplina es ejecutar una objetiva evaluación para asegurar la calidad. Esto incluye la detección de defectos, validaciones de que el sistema funciona como fue diseñado, y verificar que se cumplan los requerimientos.
- **Despliegue**: El objetivo de ésta disciplina es planificar la entrega del proyecto de desarrollo y ejecutar el plan, para dejar disponible el sistema al usuario final.
- **Administración de la Configuración**: La meta de esta disciplina es manejar el acceso a sus productos de trabajo de proyecto. Esta no sólo incluye el rastreo de versiones del trabajo del producto en el tiempo, sino también el control y administración de los cambios en estos productos.
- **Administración del Proyecto**: El objetivo de esta disciplina es dirigir las actividades a lo largo del proyecto. Esto incluye la administración del riesgo, dirección del personal (asignación de tareas, rastreo del progreso, etc.), y coordinación con personas y sistemas fuera del alcance del proyecto para asegurar su liberación a tiempo y dentro del presupuesto.
- **Entorno**: El objetivo de esta disciplina es soportar el resto del esfuerzo asegurando que el proceso apropiado, las guías (normas y directrices), y herramientas (hardware y software) estén disponibles para cuando el equipo las necesite.

Programación Extrema (XP).

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

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. (Letelier Penadés, 2006)

Características de XP, la metodología se basa en: (Mendoza Sánchez, 2004)

- **Pruebas Unitarias**: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.

- **Refabricación**: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares**: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

De las metodologías de desarrollo de software que anteriormente se describen, se decidió optar por una metodología de desarrollo ágil, ya que estas son las recomendadas para la construcción de componentes como la propuesta de solución. A continuación se realiza una comparación entre estas.

Análisis Comparativo entre metodologías ágiles.

AUP constituye una metodología formal, ampliamente configurable, con facilidad para la construcción de componentes reutilizables, que puede ser aplicada a casi cualquier proyecto de desarrollo de software, permitiendo adaptar sus artefactos, roles, y modelos a las especificidades del proyecto que esté siendo desarrollado. XP es una metodología ágil, que requiere interacción con el cliente en todo momento, que es utilizada principalmente en proyectos donde se automaticen procesos con requisitos inestables, y que tiene como meta entregar el producto al cliente lo más rápido posible.

Tabla 1. Comparación entre metodologías.

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

Metodología Seleccionada.

Para cumplir los objetivos de esta investigación, se hace necesario utilizar una metodología altamente personalizable, que proporcione facilidades para su configuración, y que no requiera una constante interacción con el cliente. Además, que proporcione artefactos formales que posibiliten la correcta documentación del componente que se desarrollará para su posterior utilización. Además

de ser la única de las metodologías estudiadas que garantiza la elaboración de las fases de un producto de software orientado a objetos y brindar nuevos artefactos permitiendo modelar el sistema de forma correcta; se enfoca en las actividades de alto nivel, no en cada actividad posible por la que podría pasar un proyecto como lo hace RUP, y al estar basado en principios de desarrollo ágil. Debe garantizar una rápida reacción ante los cambios ocasionados en los requisitos del cliente y centrará los esfuerzos de desarrollo del producto en las tareas de mayor valor, dedicando especial atención a la excelencia técnica y al buen diseño. Por lo antes expuesto, la metodología seleccionada para guiar el proceso de desarrollo de la solución propuesta en esta investigación es AUP, la cual se utilizará, según las necesidades que se planteen para el desarrollo del producto.

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

Desde los inicios de la ingeniería de software el hombre necesitó modelar mediante dibujos sus conceptos. Con ello se facilita el análisis y comprensión de sus ideas. Con el desarrollo de esta industria y los procesos de desarrollo de software en equipo surge la necesidad de crear lenguajes visuales que estandarizarán el modelado. Este debía cumplir con una premisa: la modelación de un sistema debe ser comprendida por desarrolladores de cualquier parte del mundo. Para dar solución a lo mismo es creado el Lenguaje Unificado de Modelado por Grady Booch, Jim Rumbaugh e Ivar Jacobson.

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Esta herramienta de modelado ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (Tecnologia, 2001)

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema, en UML 2.0 se definen una serie de diagramas adicionales a los establecidos en las versiones anteriores de UML, el conjunto de diagramas se encuentra organizado en torno a dos categorías: diagramas estructurales y diagramas dinámicos o de comportamiento (Figura 5).

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

de la infraestructura de red se colocan, a modo de artefactos (elementos físicos simples), los elementos componentes del software. Esta versión proporciona a los analistas, arquitectos y desarrolladores; herramientas cada vez más potentes que les permite aprovechar mejor los modelos y como consecuencia generar una mayor cantidad de código reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

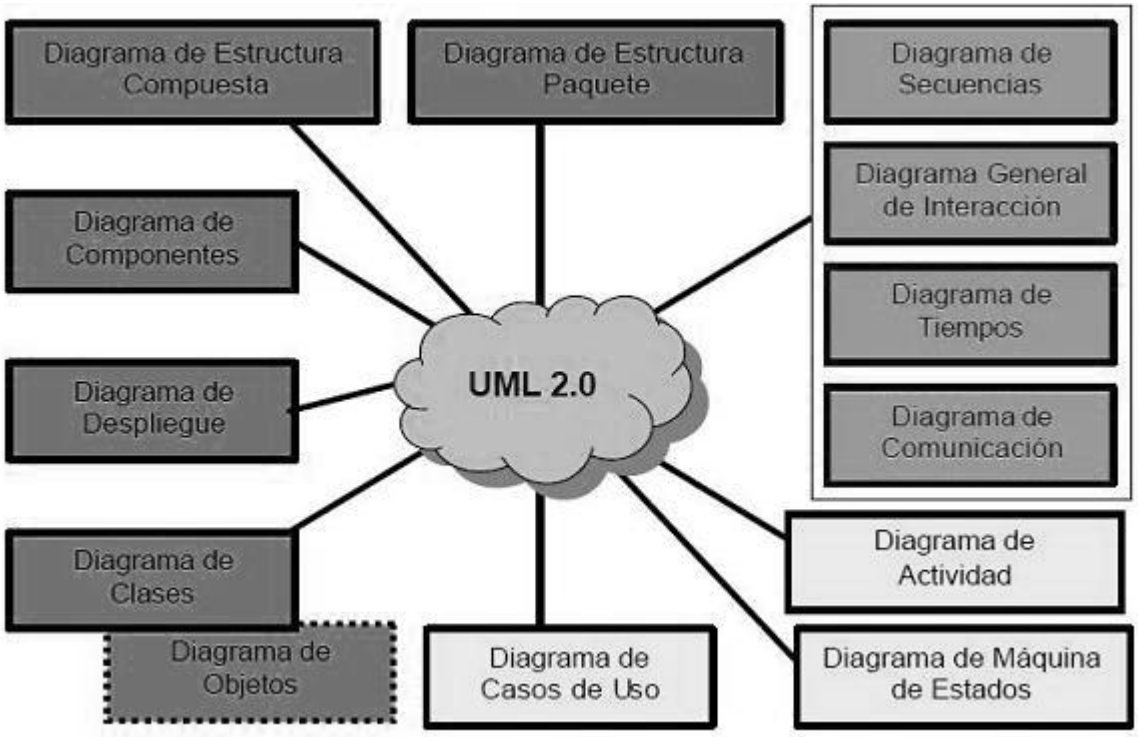

Figura 5. Diagramas UML 2.0.

2.4. Lenguajes de Programación.

Un lenguaje de programación es independiente de la computadora en la que se utiliza. Cada lenguaje se representa en forma simbólica y está diseñado para describir el conjunto de acciones consecutivas que un programa debe ejecutar. Es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. (Wilson, 1993)

Durante el desarrollo del componente serán utilizados los dos lenguajes que a continuación se proponen, uno para toda la lógica de negocio del lado del servidor y el otro para la interfaz de usuario y capa de presentación del lado del cliente.

2.4.1. PHP

Es el acrónimo de Hipertext Preprocesor. Es un lenguaje script interpretado del lado del servidor lo cual permite acceder a los recursos existentes en el servidor, como puede ser una base de datos. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. (Hinostroza, 2007).

Al ser un lenguaje que se interpreta en el servidor no es necesario que su navegador lo soporte, sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP. (Mancilla, 2001). Algunas de las características de este lenguaje son la compatibilidad con las bases de datos más comunes como MysSQL, PostgreSQL, Oracle, MS SQL Server, SybasemSQL, Informix, entre otras; está soportado por una gran comunidad de desarrolladores lo que hace que cuente con la ayuda de una cantidad de programadores considerable, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente; presenta un soporte multiplataforma tanto de diversos Sistemas Operativos, como servidores HTTP y de bases de datos. (Hinostroza, 2007).

PHP es un lenguaje de programación de código abierto con una enorme variedad de funciones que permiten desarrollar múltiples funcionalidades como enviar un e-mail, subir un archivo, crear una imagen en tiempo de ejecución, interactuar con diversos protocolos de comunicación, interactuar con documentos XML, autenticación, creación dinámica de documentos PDF, entre otras. (Hinostroza, 2007).

2.4.2. JavaScript.

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

Ventajas de JavaScript:

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

Lenguaje sencillo de aprender con utilización de funciones para crear grandes aplicaciones web, trabaja con muchas propiedades de los exploradores web sin necesidad de cargar ninguna

máquina virtual para ejecutar el código, utiliza una arquitectura orientada a objetos parecida a la de Java o C++.

- **Usabilidad.**

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

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

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

2.5. Ajax.

Ajax, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. (Castellano, 2004.)

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML. (Castellano, 2004.)

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Ajax es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.

- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano, JSON y hasta EBML.

2.6. Entorno de Desarrollo NetBeans IDE 6.8.

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

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

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

NetBeans IDE dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco sus funcionalidades son ampliables mediante la instalación de packs. (Softonic, 2009) El IDE se ejecuta en muchas plataformas incluyendo Windows, Linux, Mac OS X y Solaris, además se puede configurar e instalar fácilmente para satisfacer las necesidades de los desarrolladores.

Características de NetBeans IDE 6.8: (Línea, 2011)

- Soporte PHP Ampliado: Expande el soporte de los lenguajes dinámicos con apoyo para PHP 5.3 y el esquema de Symfony acelera el desarrollo de aplicaciones web PHP.
- JavaFX ™: Código de finalización mejorado, sugerencias y navegación para JavaFX en el editor NetBeans.

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

2.7. Herramientas Case.

Las herramientas CASE[6] son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida del software.

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

2.7.1. Rational Rose.

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

Incluye también estas funciones: (IBM, 2011)

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

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

2.7.2. Visual Paradigm.

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (VP, 2011)

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

Algunas de sus características principales son: (VP, 2011)

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

2.7.3. Herramienta Seleccionada.

La herramienta seleccionada para modelar la solución es el Visual Paradigm para UML en su versión 6.4. La decisión se basa, fundamentalmente, en el hecho de que la herramienta posee una plataforma denominada SDE capaz de integrarse con el entorno de desarrollo seleccionado, permitiendo realizar el proceso de ingeniería en ambos sentidos e incluye muchas funcionalidades para el lenguaje PHP, además de presentar una interfaz de usuario de fácil uso para realizar los diagramas y artefactos que se generan durante el desarrollo del software.

2.8. Servidor Web Apache 2.2.

El servidor Apache es desarrollado por la Apache Software Fundation. Es considerado la plataforma de servidores Web de código abierto más potente y utilizada del mundo, pudiendo ejecutarse en sistemas operativos como Linux, Macintosh y Windows. Las capacidades de este servidor pueden ser ampliadas incorporándole nuevos módulos, pues su diseño modular es altamente configurable. Por su sencillez, robustez, flexibilidad, estabilidad y eficiencia ha logra ser el servidor Web más utilizado en el mundo, muy superior a los demás existentes, esto es demostrado por empresa Netcraf pues afirma que un 61.13 % de las aplicaciones publicadas en Internet en abril del 2011 usan Apache como su servidor Web. Se señala como desventaja el que no posee interface gráfica que facilite su configuración (Apache Foundation, 2011).

Se escoge Apache como servidor web pues es la solución usada para la mayoría de los sitios Web. La versión 2.2 es una profunda revisión del servidor Apache, las principales revisiones de código se han llevado a cabo para crear una arquitectura realmente escalable, en la actualidad es considerada la plataforma Web más utilizada del mundo, pues aumentan cada día el número de usuarios que aceptan este código fuente abierto en su infraestructura, es muy usado en nuestra universidad con amplia disposición de documentos e información, popular (fácil conseguir ayuda/soporte en Internet y otros sitios) y con amplia aceptación en toda la red. En el Anexo 1, se realiza una comparación entre los servidores web más importantes.

2.10. Conclusiones Parciales.

En este capítulo se realizó un análisis de las herramientas y tecnologías a utilizar en el desarrollo de la propuesta de solución. Se fundamentó la elección de los lenguajes de programación, las herramientas, el entorno de desarrollo, la metodología de desarrollo de software, así como el uso de otras tecnologías, concluyendo que: las selección de herramientas y tecnologías multiplataforma, permite obtener un componente con alto grado de portabilidad; el hecho de que estas herramientas y tecnologías procedan del software libre, contribuyen a que el costo de desarrollo y mantenimiento del componente sea mínimo. Se puede concluir que la solución propuesta se desarrollará en los lenguajes de programación JavaScript para el lado del cliente y PHP para la lógica de negocio del lado del servidor, usando como formato para intercambio de información entre lenguajes Ajax, sobre el entorno de desarrollo integrado (IDE) NetBeans 6.8 y el Servidor Web Apache 2.2. Como herramienta de modelado se utilizará el Visual Paradigm para UML 6.4 con lenguaje de modelado UML 2.0, y para guiar todo el proceso de desarrollo se utilizará AUP como metodología.

CAPÍTULO 3. Presentación de la solución propuesta.

3.1. Introducción.

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

3.2. Entorno donde trabajará el sistema.

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

Un Modelo de Dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (Jacobson, 2000)

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

3.2.1. Conceptos principales del entorno.

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

Búsqueda: Se conoce con el término de buscar a aquella actividad que se caracteriza principalmente por un propósito de descubrimiento. El objeto de esa búsqueda puede ser dar con

alguna región en particular que se presume existe, descubrir algún tipo de recurso que servirá para, por ejemplo enriquecer las arcas de una nación como ser el petróleo, gas, carbón o algún otro tipo de mineral o estar orientada a la búsqueda de determinada información en orden a satisfacer alguna necesidad o completar un trabajo de estudio.

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

3.2.2. Eventos principales del Entorno.

Búsqueda: La búsqueda es un evento mediante el cual podemos obtener los datos deseados de manera ágil y específica. En este caso, las búsquedas permiten la toma de decisiones según los criterios de búsquedas.

3.2.3. Modelo Conceptual de Dominio.

Se pueden representar los principales conceptos presentes en los procesos antes descritos, así como las principales relaciones entre ellos, como se muestra en la *figura 6*.

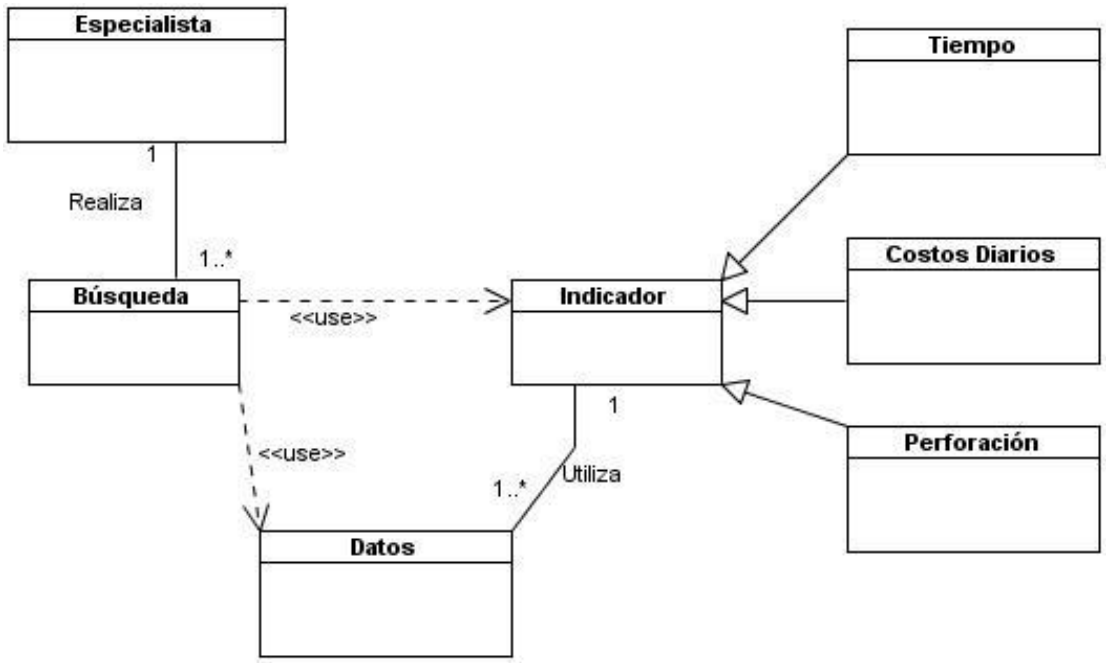

Figura 6. Modelo Conceptual de Dominio.

3.3. Especificación de los Requisitos del software.

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

3.3.1. Requisitos Funcionales.

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

El sistema debe permitir:

RF1 Realizar búsquedas de indicadores de perforación.

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

1.1 Buscar indicadores de la perforación diaria del pozo.

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

1.2 Buscar el comportamiento de los costos diarios del pozo.

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

1.3 Buscar el resumen del tiempo por etapas de perforación.

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

3.3.2. Requisitos No Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, puesto que las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable es el sistema, pueden marcar la diferencia entre un producto bien aceptado y uno con

poca aceptación. Existen varios tipos de requisitos no funcionales. A continuación se presentan los correspondientes a la solución propuesta.

RNF 1 Usabilidad.

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

RNF2 Diseño e implementación.

- Lenguaje de Programación JavaScript y PHP.
- Se debe generar la documentación de todas las clases, métodos y recursos creados.
- La solución debe ser extensible de acuerdo a las necesidades de uso.
- La solución debe permitir una configuración flexible para satisfacer las necesidades de diferentes negocios.
- Se utiliza Visual Paradigm para el análisis y diseño de la aplicación.

RNF3 Hardware.

- Requisitos mínimos: Un computador con Procesador Pentium III a 1.8 GHz, Memoria: 128 Mb RAM, Disco Duro: 40 Gb.
- Requisitos recomendados: Un computador con Procesador: Pentium IV 2.4 GHz, Memoria: 512 Mb RAM, Disco Duro: 80 Gb.

RNF4 Software.

- Por parte del cliente se requiere un navegador capaz de interpretar JavaScript, HTML y CSS.

RNF5 Soporte.

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

RNF6 Portabilidad.

- La solución debe ser independiente de la plataforma en que se utilice, propiedad que hereda de los lenguajes utilizados, JavaScript y PHP que pueden interpretarse en diferentes plataformas como Windows y Linux.

RNF7 Confiabilidad.

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

3.4. Descripción de la arquitectura del sistema.

La estructura de la arquitectura será descrita a través del modelo 4+1 vista que propone AUP, este modelo describe la Arquitectura de Software usando cinco vistas concurrentes, donde cada vista se refiere a un conjunto de intereses de los diferentes interesados (stakeholders) del sistema permitiendo así una visión más detallada de la arquitectura del sistema. Este modelo consta con las siguientes vistas (UP, 2005):

Figura 7. Modelo 4+1 vista.

- La **vista lógica** representa los elementos de diseño más importantes para la arquitectura del sistema. Este describe las clases más importantes, su organización en paquetes y subsistemas, y estos a su vez en capas.
- La **vista de procesos** suministra una base para la comprensión de la organización de los procesos de un sistema, ilustrados en el mapeo de las clases y subsistemas en procesos e hilos. Solo suele usarse cuando el sistema presenta procesos concurrentes o hilos.
- La **vista de implementación** describe la organización estática del software en su ambiente de desarrollo.
- La **vista despliegue** suministra una base para la comprensión de la distribución física de un sistema a través de nodos. Suele utilizarse cuando el sistema está distribuido.

- La **vista de casos de uso** representa la selección y descripción de los casos de usos arquitectónicamente significativos brindando una noción general del sistema.

3.4.1. Vista de Casos de Uso

En esta vista se representa los actores y los casos de usos arquitectónicamente significativos para el sistema; estos serían los casos de uso que encierran las funcionalidades básicas del sistema. A continuación se representa la vista de casos de uso:

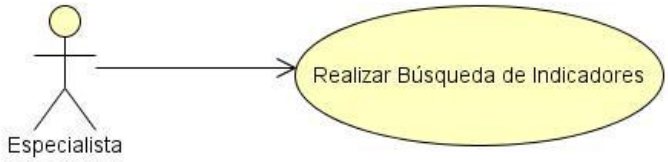

Figura 8. Diagrama de Casos de Uso arquitectónicamente significantes.

A continuación se describen los casos de uso:

Tabla 2. Descripción del CU Realizar Búsqueda de Indicadores.

Caso de Uso:	Realizar Búsqueda de Indicadores
Actores:	Especialista
Resumen:	El caso de uso se inicia cuando el especialista accede al componente de búsqueda, para realizar la misma.
Precondiciones:	
Referencias	
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Accede al componente de búsqueda de la aplicación.	
	2. Visualiza la Interfaz de usuario solicitada.
3. Selecciona un indicador de perforación.	
	4. Visualiza la Interfaz de usuario del indicador seleccionado.
5. Introduce un criterio de búsqueda y ejecuta la acción buscar.	

	6. Visualiza los resultados de la búsqueda.	
Flujos Alternos		
Acción del Actor	**Respuesta del Sistema**	
-	-	
Relaciones	CU Incluidos	
	CU Extendidos	
Poscondiciones		

3.4.2. Vista lógica

En esta vista se representa los elementos de diseño más importantes para la arquitectura del sistema. Se describen las clases más importantes, su organización en paquetes y/o subsistemas, así como la relación que existen entre ellos y la organización de estos a su vez en capas.

Las capas son agrupaciones horizontales lógicas de componentes de software que forman la aplicación o el servicio. Nos ayudan a diferenciar entre los diferentes tipos de tareas a ser realizadas por los componentes, ofreciendo un diseño que maximiza la reutilización y, especialmente, la mantenibilidad. El dividir una aplicación en capas separadas que desempeñan diferentes roles y funcionalidades, nos ayuda a mejorar el mantenimiento del código; nos permite también diferentes tipos de despliegue y, sobre todo, nos proporciona una clara delimitación y situación de dónde debe estar cada tipo de componente funcional e incluso cada tipo de tecnología. (César de la Torre Llorente, Noviembre 2010)

Con el estudio realizado se llegó a la conclusión de que el estilo arquitectónico que más se adapta a la solución, es la Arquitectura en Capas, ya que la utilización de dichas capas proporciona al sistema niveles de abstracción que resuelven muchos problemas de desarrollo, proporcionando reutilización, optimización y refinamiento.

Este estilo asegura que los componentes se estructuren en niveles o capas donde los componentes de una capa pueden interactuar solo con componentes de la misma capa o bien con otros componentes de capas inferiores. Esto ayuda a reducir las dependencias entre componentes de diferentes niveles. Normalmente hay dos aproximaciones al diseño en capas: Estricto y Laxo.

Un **"diseño en Capas estricto"** limita a los componentes de una capa a comunicarse solo con los componentes de su misma capa o con la capa inmediatamente inferior, el cuál se pone de manifiesto en esta arquitectura (César de la Torre Llorente, Noviembre 2010).

Un **"diseño en Capas laxo"** permite que los componentes de una capa interactúen con cualquier otra capa de nivel inferior (César de la Torre Llorente, Noviembre 2010).

A continuación se representa la organización en capas de la arquitectura:

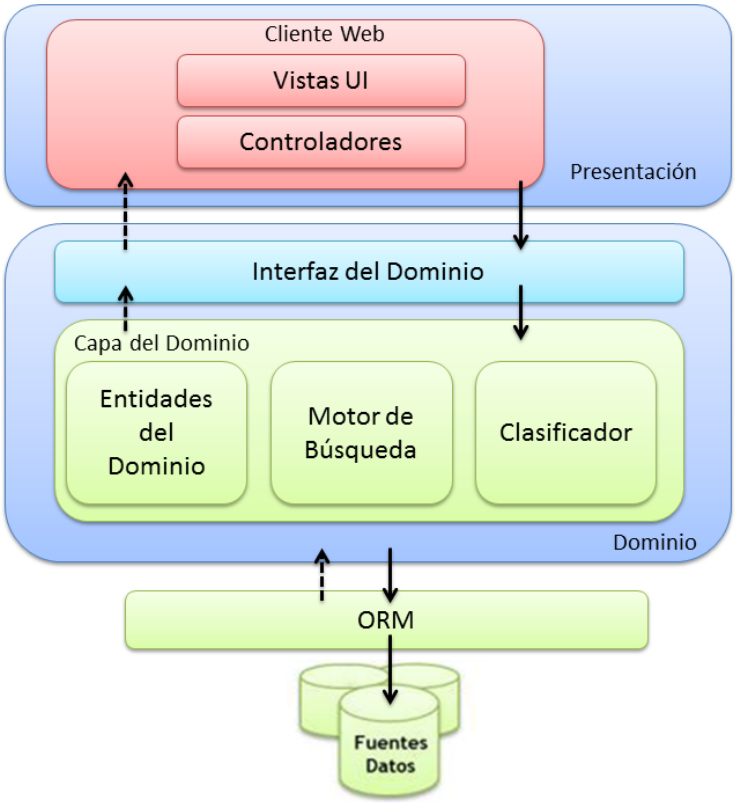

Figura 9. Vista Lógica de la Arquitectura.

La capa de Presentación, Dominio y Acceso a datos que componen esta arquitectura, están compuestas por componentes que se describen a continuación:

Capa de Presentación.

Esta capa es responsable de mostrar información al usuario e interpretar sus acciones. Los componentes de esta capa de presentación implementan la funcionalidad requerida para que los usuarios interactúen con el sistema.

Vistas de Interfaz de Usuario (UI).

En este paquete se encapsulan todas las vistas o interfaces que permiten la interacción del usuario con el sistema. Son componentes que formatean datos en cuanto a tipos de letras y controles visuales, y también reciben datos proporcionados por el usuario.

Controladores de Interfaz.

Para ayudar a sincronizar y orquestar las interacciones del usuario, puede ser útil conducir el proceso utilizando componentes separados de los componentes propiamente gráficos. Esto impide que el flujo de proceso y lógica de gestión de estados esté programada dentro de los propios controles y formularios visuales y permite reutilizar dicha lógica y patrones desde otros interfaces o vistas. También es muy útil para poder realizar pruebas unitarias de la lógica de presentación.

Capa de Dominio.

Esta capa es responsable de representar conceptos de negocio, información sobre la situación de los procesos de negocio e implementación de las reglas del dominio. Esta capa es el corazón del software. Sus componentes implementan la funcionalidad principal del sistema y encapsulan toda la lógica de negocio relevante (genéricamente llamado lógica del negocio). Básicamente suelen ser clases en el lenguaje seleccionado que implementan la lógica del dominio dentro de sus métodos y esta capa tiene que ignorar completamente los detalles de persistencia de datos.

Interfaz del dominio.

Define los trabajos que la aplicación como tal debe de realizar y redirige a los objetos del dominio y de infraestructura (persistencia, etc.) que son los que internamente deben resolver los problemas. Esta capa no debe contener reglas del dominio o conocimiento de la lógica de negocio, simplemente debe realizar tareas de coordinación. El trabajo final se delegará posteriormente a los objetos de las capas inferiores, es decir, hará de fachada del modelo de Dominio.

Entidades del dominio.

Estos objetos son entidades desconectadas (datos + lógica) y se utilizan para alojar y transferir datos de entidades entre las diferentes capas, contienen también la lógica del dominio relativo a cada entidad y representan las entidades de negocio del mundo real.

Motor de búsqueda.

Este paquete de clases conforma el núcleo principal del buscador. Es el encargado de realizar las búsquedas sobre las diferentes fuentes de datos, por lo cual, se considera el corazón del sistema.

Clasificador de resultados.

Los criterios de ordenación de resultados determinan en la calidad de la respuesta de estos sistemas, están directamente relacionados con la pertinencia y la relevancia. Si las primeras referencias que devuelve son poco significativas, el usuario tendrá una visión negativa de la herramienta o no estar dispuesto a seguir recorriendo resultados, desistir de esa demanda e

incluso del uso del buscador. Este componente se encarga de la correcta clasificación de los resultados.

Capa de Acceso a datos.

Esta capa proporciona la capacidad de persistir datos así como lógicamente acceder a ellos. Pueden ser datos propios del sistema o incluso acceder a datos expuestos por sistemas externos (Servicios Web externos, etc.). También esta capa de persistencia de datos expone el acceso a datos a las capas superiores, normalmente las capas del dominio. Esta exposición deberá realizarse de una forma desacoplada.

Mapeo objeto relacional. Object/Relational Mapping frameworks (O/RM).

Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

3.4.3. Vista de implementación.

La vista de implementación ilustra la descomposición del software en componentes, subsistemas de implementación y las dependencias que se establecen entre ellos. Un componente de software es una parte física de un sistema como puede ser un módulo, una base de datos, un programa ejecutable, varias clases en un mismo archivo, entre otros. A continuación se presenta la vista de implementación del sistema.

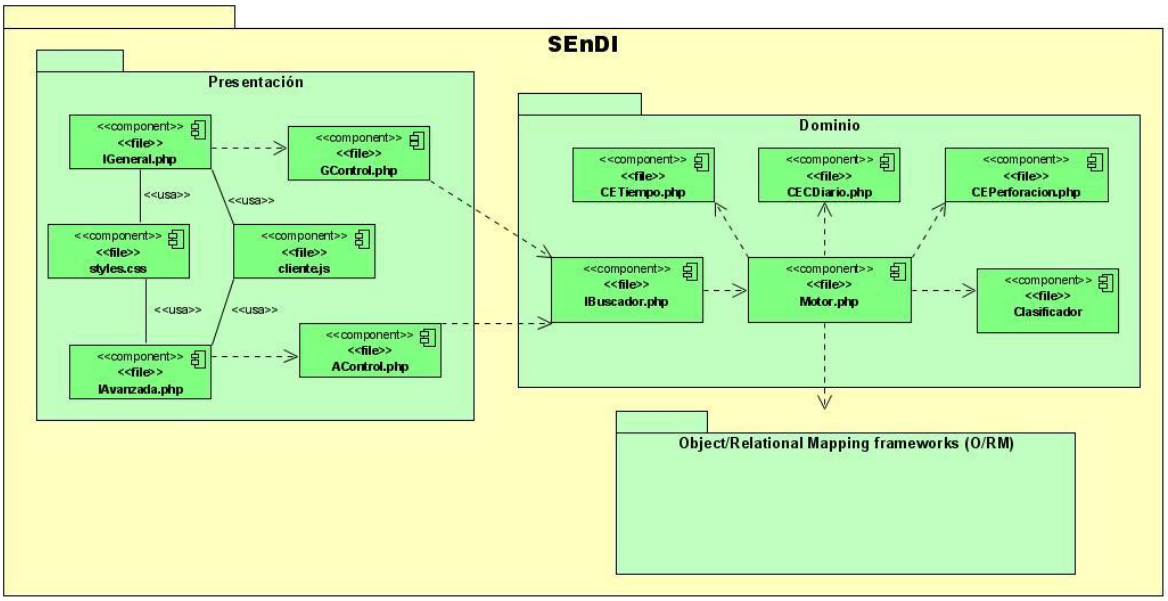

Figura 10. Vista de implementación.

IGeneral.php: en este componente se encuentra toda la interfaz de usuario para la realización de búsquedas generales en el sistema. Es el encargado también de visualizar al usuario los resultados de la búsqueda realizada.

IAvanzada.php: en este componente se encuentra toda la interfaz de usuario para la realización de búsquedas avanzadas en el sistema. Es el encargado también de visualizar al usuario los resultados de la búsqueda realizada. Esta interfaz posee muchas más opciones de entradas para el usuario, haciendo la búsqueda avanzada ya que este puede ser más específico en la entrada de criterios de búsquedas.

styles.css: este componente contiene todas las reglas de estilos que poseen las distintas interfaces de usuario del sistema.

cliente.js: este componente contiene todas las funciones de validación del lado del cliente, así como las clases que hacen la interfaz más interactiva con los usuarios.

GControl.php: este componente contiene las clases que controlan la interfaz de búsqueda general, además se encargan de la validación de los datos, así como de la correcta interpretación de los mismos, para luego ser enviados hacia las capas inferiores.

AControl.php: este componente contiene las clases que controlan la interfaz de búsqueda avanzada, además se encargan de la validación de los datos, así como de la correcta interpretación de los mismos, para luego ser enviados hacia las capas inferiores.

IBuscador.php: este componente es uno de los más importantes de la capa de dominio, ya que es el encargado de permitir la comunicación entre las interfaces y el negocio. Es considerado como la fachada del dominio.

Motor.php: este componente es el más importante de todos, contiene las clases y métodos que rigen todo el funcionamiento del sistema, es donde se realiza todo el proceso lógico de búsqueda

de indicadores sobre la fuente de datos. Este componente incluye otro componente denominado clasificador y juntos desarrollan todo el proceso de recuperación de información.

Clasificador.php: este componente, es utilizado por el motor de búsqueda para realizar un proceso de clasificación de resultados según los parámetros de filtrado entrados por el usuario. Este proceso permite que los resultados visualizados, posean un orden y organización que facilita la interpretación de los mismos.

CETiempo.php: clase entidad que posee todos los atributos y métodos necesarios que conforman un indicador de tiempo para el negocio.

CECDiario.php: clase entidad que posee todos los atributos y métodos necesarios que conforman un indicador de costo diario para el negocio.

CEPerforacion.php: clase entidad que posee todos los atributos y métodos necesarios que conforman un indicador de perforación para el negocio.

3.4.4. Vista de despliegue

La vista de despliegue muestra un conjunto de nodos unidos por conexiones de comunicación. Ésta vista suministra además una base para comprender como queda la distribución física del sistema y cómo estarán distribuidos los componentes de la aplicación en ella. Para realizar el despliegue de la aplicación se requiere de un servidor Web Apache. Se deberá contar también con un servidor de BD, donde se estará ejecutando el Sistema Gestor de Base de Datos (SGBD) PostgresSQL y se requiere además que las PC Clientes tengan un navegador Web. Es importante aclarar que el SGBD, puede estar en el mismo nodo donde se encuentra el Servidor Web Apache, es común, que en la mayoría de los sistemas pequeños, que no manipulan mucha información, se den estos casos.

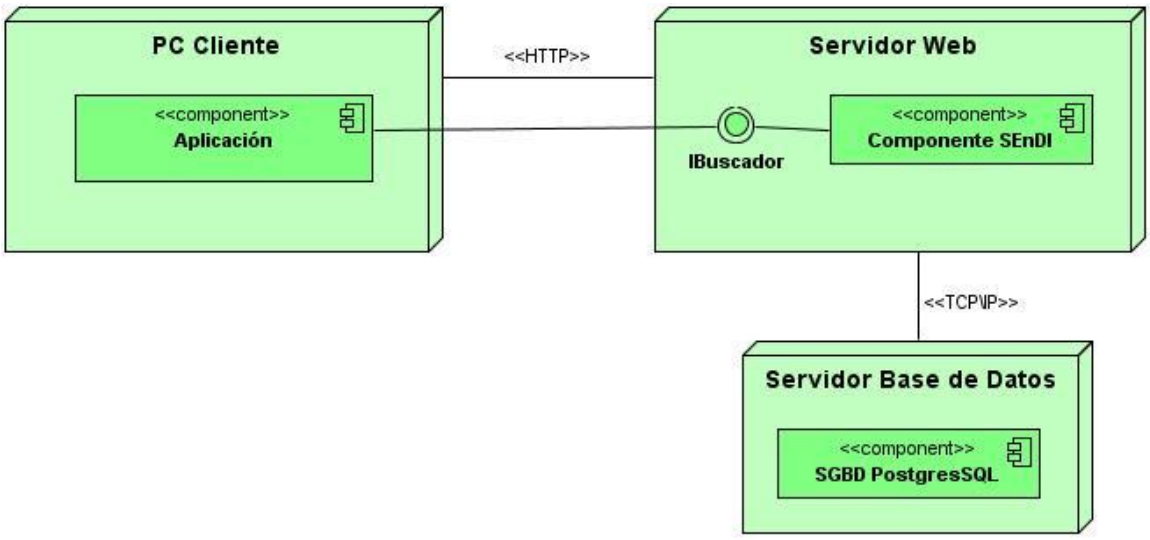

Figura 11. Vista de despliegue.

3.4.5. Vista de Procesos

Esta vista suministra una base para la comprensión de la organización de los procesos de un sistema, ilustrados en el mapeo de las clases y subsistemas en procesos e hilos. Solo suele usarse cuando el sistema presenta procesos concurrentes o hilos. En este caso la solución propuesta no presenta procesos concurrentes por tanto no se hace representación de esta vista.

3.5. Descripción del diseño del sistema.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. (Larman, 2000)

Incluye la siguiente información: (Larman, 2000)

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

A diferencia del Modelo Conceptual, un Diagrama de Clases de Diseño muestra definiciones de entidades de software, más que conceptos del mundo real. El diagrama de clases del diseño es el diagrama más importante de la fase de diseño de un software, este brinda una visión general para comenzar con la implementación del producto. Para la creación del diagrama de clases del diseño primeramente se identifican las entidades software necesarias para desarrollar el sistema que se pretende realizar, se definen las relaciones existentes entre ellas, los atributos y métodos según la necesidad del sistema. Es importante destacar que en el sistema a desarrollar al igual que en la mayoría de estos, el diagrama de clases del diseño se actualiza constantemente, pues aparecen nuevos elementos, a medida que avanza el desarrollo del sistema hasta obtener la versión final que se corresponde con el diseño definitivo del sistema.

Seguidamente se muestra el diagrama de clases del diseño propuesto, correspondiente al componente de búsqueda de indicadores que se modeló.

3.5.1. Diagrama de clases del sistema.

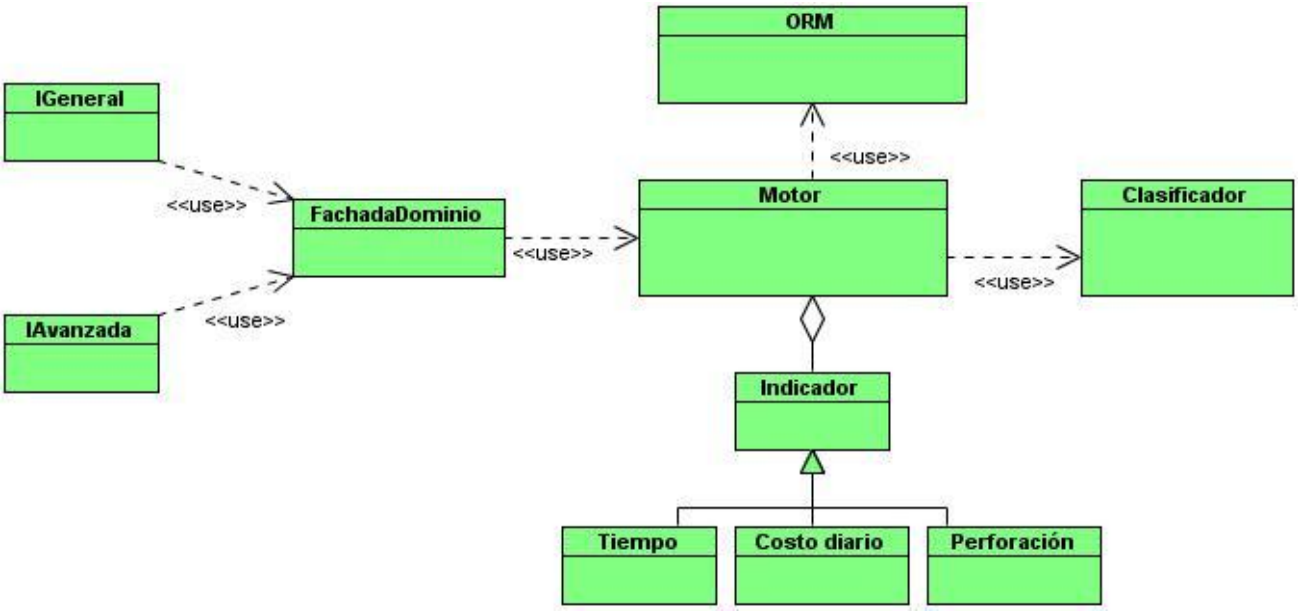

Figura 12. Diagrama de clases.

3.5.2. Patrones de Diseño Utilizados.

Un patrón de diseño es un conjunto de reglas que describen cómo afrontar tareas y solucionar problemas que surgen durante el desarrollo de software (Pérez, 2011), es la solución recurrente para un problema típico y en un contexto determinado. La solución se refiere a la respuesta al problema, por lo que ayuda a resolver las dificultades de diseño en problemas similares. Los patrones comunican soluciones de diseño a los desarrolladores y arquitectos que los leen y los utilizan.

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

Modelo-Vista-Controlador

Modelo-Vista-Controlador (MVC), como se muestra en la *figura 13*, separa el modelo de datos, la interfaz y la lógica de negocio de un sistema en tres componentes distintos. Esta división de las responsabilidades posibilita que sea fácil de modificar ante algún cambio que ocurra en el sistema. A continuación se describe cada uno de los componentes (Kiccillof, et al., 2004):

Vistas: Se encarga de presentar la información necesaria para interactuar con los usuarios.

Controlador: Responde a eventos, usualmente acciones del usuario que pueden provocar cambios en el modelo y probablemente en la vista.

Modelo: El modelo encapsula la manipulación de los datos y es independiente de las representaciones específicas o del comportamiento de entrada. En él la lógica de datos garantiza y asegura la integridad de los mismos permitiendo derivar nuevos datos. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

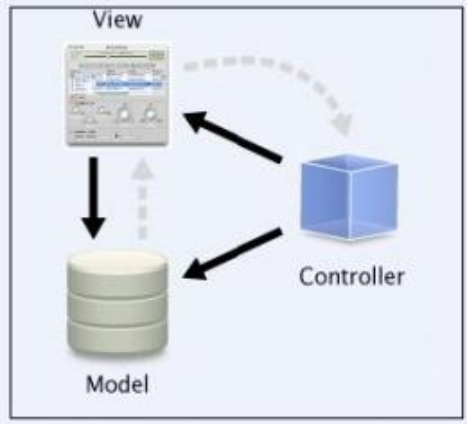

Figura 13. Modelo-Vista-Controlador.

Este patrón se evidencia en la arquitectura propuesta, ya que la misma está estructurada en 3 capas fundamentales: capa de presentación, capa de dominio y capa de persistencia de datos. Lo que permite un trabajo más estructurado y desacoplado.

Fábrica Simple (*Simple Factory*).

Patrón **Fábrica Simple** en inglés ***Simple Factory*** es un patrón de creación que se encarga de crear instancias de objetos de manera que ya no se tendrán que instanciar directamente proporcionando a los programas una mayor flexibilidad para decidir qué objetos usa. Su objetivo es devolver una instancia de múltiples tipos de objetos, normalmente todos estos objetos provienen de una misma clase padre mientras que se diferencian entre ellos por algún aspecto de comportamiento. (Pérez)

Este patrón se pone de manifiesto en la capa de dominio, ya que esta contiene la clase indicador y de ella heredan los diferentes tipos de indicadores de perforación. Con el empleo de este patrón

solo se tendría una instancia de la clase indicador y se podrá trabajar con las funcionalidades de todos los indicadores.

Patrones de Asignación de Responsabilidades.

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

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

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. La nueva instancia deberá ser creada por la clase que: tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase. Este patrón brinda soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases.

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

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

Bajo acoplamiento: Plantea la idea de tener las clases lo más desacoplada que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión

posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases, este patrón se evidencia en la mayoría de las clases pues no dependen de ninguna otra clase para realizar sus responsabilidades.

Controlador: Este patrón funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que esta recibe los datos y los envía a las distintas clases según el método llamado, la utilización de este patrón se evidencia en las clases de interfaz del dominio y en las controladoras de las distintas interfaces de usuario en la capa de presentación.

Patrones GoF

Los patrones GoF fueron descritos en el libro ¨Design Patterns: Elements of Reusable Object-Oriented Software¨, también conocido como el libro GoF (Gang-Of-Four o "pandilla de los 4") escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Según el libro GoF estos patrones se clasifican según su propósito en creacionales, estructurales y de composición. En este libro fueron recopilados y documentados 23 patrones, a continuación se describen algunos (Gamma, et al., 1998).

Fachada: Este patrón sirve para proveer una interfaz unificada que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Es utilizado para reducir la dependencia entre clases, ya que ofrece un punto de acceso al resto de las clases, si éstas cambian o se sustituyen por otras solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones clientes. Fachada no oculta las clases sino que ofrece una forma más sencilla de acceder a ellas y se pone de manifiesto con la clase interfaz del dominio en la capa del dominio de la arquitectura propuesta.

Singleton: Este patrón garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella. Se usa cuando debe haber exactamente una instancia de una clase y ésta debe ser accesible a los clientes desde un punto de acceso conocido. La única instancia debería ser extensible mediante herencia y los clientes deberían ser capaces de usar una instancia extendida sin modificar su código. Este patrón puede implementarse entre otras clases, en la interfaz del dominio, para que exista una sola instancia de la misma en todo el sistema.

3.6. Conclusiones Parciales.

En el presente capítulo se realizó una descripción de la propuesta de solución a través de la modelación del dominio y el planteamiento de los requisitos funcionales y no funcionales, describiendo el entorno donde trabajará el sistema. Con la aplicación de patrones al diseño

propuesto, se logró un diseño de mayor calidad. La aplicación del modelo 4 + 1 vista, posibilitó realizar una propuesta de arquitectura, que se ajustara a las necesidades de los requisitos funcionales.

CAPÍTULO 4. Validación de la propuesta de solución.

4.1. Introducción.

En el presente capítulo se aborda sobre la evaluación de la arquitectura, enfatizando en el por qué se hace necesario su realización, los beneficios que brinda. Se describen además las técnicas, métodos y atributos de calidad con el objetivo de seleccionar uno para realizar la evaluación de la propuesta de solución.

4.2. Evaluación de la Arquitectura de Software.

La evaluación es un estudio de factibilidad que pretende detectar posibles riesgos, así como buscar recomendaciones para contenerlos. El objetivo de evaluar una arquitectura es saber si puede habilitar los requerimientos, atributos de calidad y restricciones para asegurar que el sistema a ser construido cumple con las necesidades de los interesados o *stakeholders* (Brey, et al., 2005).

Debido a lo importante que es para el desarrollo de software tener una arquitectura que logre satisfacer los requerimientos no funcionales y de calidad es conveniente realizar actividades de verificación de la misma, con el fin de identificar posibles problemas que podría resultar muy costoso eliminar posteriormente. Según (Kazman, et al., 2001) el primer paso para la evaluación de una arquitectura es conocer qué es lo que se quiere evaluar. De esta forma, es posible establecer la base para la evaluación, puesto que la intención es saber qué se puede evaluar y qué no. Esta evaluación no define si una arquitectura es buena o mala, solo identifica donde están los riesgos, las fortalezas y debilidades.

La evaluación es una tarea no trivial, puesto que se pretende medir propiedades del sistema en base a especificaciones abstractas, como por ejemplo los diseños arquitectónicos. Por ello, la intención es más bien la evaluación del potencial de la arquitectura diseñada para alcanzar los atributos de calidad requeridos. Las mediciones que se realizan pueden tener distintos objetivos, dependiendo de la situación en la que se encuentre el arquitecto y la aplicabilidad de las técnicas que emplea (Bosch, 2000).

La arquitectura puede ser evaluada en dos momento, Kazman (Kazman, et al., 2001) define las siguientes variantes: la *evaluación temprana,* para su realización no es necesario que se haya especificado completamente la arquitectura, ya que esta evaluación abarca desde las fases tempranas de diseño y a lo largo del desarrollo; la *evaluación tardía* se realiza antes del comienzo de la implementación cuando ya fue especificada completamente la arquitectura.

4.3. Atributos de calidad.

Los atributos de calidad hacen referencia a requerimientos adicionales que brinda el sistema para lograr una mayor satisfacción del cliente, estos definen las propiedades de los servicios brindados. La calidad del software está dada por el grado de combinación que puedan tener sus atributos. Según un estudio plasmando en el libro "Arquitecturas de Software" por (Camacho, et al., 2004) los atributos están divididos en dos grupos:

Observables vía ejecución: aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución, éstos son descritos en el Anexo 2.

No observables vía ejecución: aquellos atributos que se establecen durante el desarrollo del sistema, éstos son descritos en el Anexo 3.

En su mayoría, los atributos de calidad se pueden organizar y descomponer de maneras diferentes, en lo que se conoce como modelos de calidad. Los modelos de calidad de software facilitan el entendimiento del proceso de la ingeniería de software. Pressman indica que los factores que afectan a la calidad del software no cambian, por lo que resulta útil el estudio de los modelos de calidad que han sido propuestos en este sentido desde los años 70. Entre los modelos de calidad más importante se encuentra el establecido por el estándar ISO/IEC 9126 en el año 1991. El estándar ISO/IEC 9126 ha sido desarrollado en un intento de identificar los atributos clave de calidad para un producto de software. Este estándar es una simplificación del Modelo de McCall del año 1977, e identifica seis características básicas de calidad que pueden estar presentes en cualquier producto de software (Pressman, 2002), estas características y subccaracterísticas de calidad son mostradas en el Anexo 4.

4.4. Técnicas de evaluación de la Arquitectura de Software.

Existen varias técnicas para evaluar las arquitecturas, éstas se dividen en *cualitativas* y en *cuantitativas,* como se puede observar en la figura 14 dentro de estos dos grupos se encuentran varias técnicas, que dependiendo del momento en que se realiza la evaluación y el tipo de resultado que se espera se debe escoger la más acorde a lo que se desea. Por lo regular, las técnicas de evaluación cualitativas son utilizadas cuando la arquitectura está en construcción; mientras que las técnicas de evaluación cuantitativas, se usan cuando la arquitectura ya ha sido implantada.

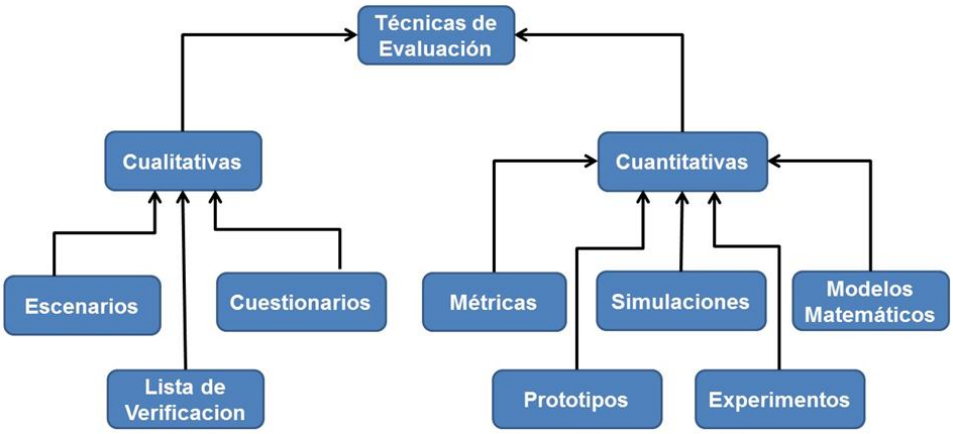

Figura 14. Técnicas de Evaluación de Arquitecturas.

4.4.1. Evaluación Basada en Escenarios.

Un escenario es una breve descripción de la interacción de alguno de los involucrados en el desarrollo del sistema con el propio sistema. Un escenario consta de tres partes: el estímulo, el contexto y la respuesta. El estímulo es la parte del escenario que explica o describe lo que el involucrado en el desarrollo hace para iniciar la interacción con el sistema. Puede incluir ejecución de tareas, cambios en el sistema, ejecución de pruebas, configuración, etc. El contexto describe qué sucede en el sistema al momento del estímulo. La respuesta describe, a través de la arquitectura, cómo debería responder el sistema ante el estímulo. Este último elemento es el que permite establecer cuál es el atributo de calidad asociado. Los escenarios proveen un vehículo que permite concretar y entender atributos de calidad (Kazman, et al., 2001).

Actualmente las técnicas basadas en escenarios cuentan con dos instrumentos de evaluación relevantes, a saber: el Utility Tree propuesto por (Kazman, et al., 2001) y los Profiles, propuestos por (Bosch, 2000).

Utility Tree

Un esquema en forma de árbol que presenta los atributos de calidad de un sistema de software, refinados hasta el establecimiento de escenarios que especifican con suficiente detalle, el nivel de prioridad de cada uno. La intención del uso del Utility Tree es la identificación de los atributos de calidad más importantes para un proyecto particular. No existe un conjunto preestablecido de atributos, sino que son definidos por los involucrados en el desarrollo del sistema al momento de la construcción del árbol. El Utility Tree contiene como nodo raíz la utilidad general del sistema. Los atributos de calidad asociados al mismo conforman el segundo nivel del árbol los cuales se refinan hasta la obtención de un escenario lo suficientemente concreto para ser analizado y otorgarle prioridad a cada atributo considerado. Cada atributo de calidad perteneciente al árbol contiene una

serie de escenarios relacionados y una escala de importancia y dificultad asociada, que será útil para efectos de la evaluación de la arquitectura.

Perfiles

Es un conjunto de escenarios, generalmente con alguna importancia relativa asociada a cada uno de ellos. El uso de perfiles permite hacer especificaciones más precisas del requerimiento para un atributo de calidad. Para la definición de un perfil es necesario seguir tres pasos: definición de las categorías del escenario, selección y definición de los escenarios para la categoría y asignación del peso a los escenarios. Cada atributo de calidad tiene un perfil asociado. Algunos perfiles pueden ser usados para evaluar más de un atributo de calidad. Esta técnica de evaluación basada en escenarios es dependiente de manera directa del perfil definido para el atributo de calidad que va a ser evaluado. La efectividad de la técnica es altamente dependiente de la representatividad de los escenarios.

4.4.2. Evaluación basada en simulación.

La evaluación basada en simulación utiliza una implementación de alto nivel de la AS. El enfoque básico consiste en la implementación de componentes de la arquitectura y la implementación a cierto nivel de abstracción del contexto del sistema donde se supone va a ejecutarse. La finalidad es evaluar el comportamiento de la arquitectura bajo diversas circunstancias. Una vez disponibles estas implementaciones, pueden usarse los perfiles respectivos para evaluar los atributos de calidad (Bosch, 2000).

4.4.3. Evaluación basada en modelos matemáticos.

La evaluación basada en modelos matemáticos se utiliza para evaluar atributos de calidad operacionales. Permite una evaluación estática de los modelos de diseño arquitectónico y se presenta como alternativa a la simulación, dado que evalúan el mismo tipo de atributos. Ambos enfoques pueden ser combinados utilizando los resultados de uno como entrada para el otro. Entre las desventajas que presenta esta técnica se encuentra la inexistencia de modelos matemáticos apropiados para los atributos de calidad relevantes y el hecho de que el desarrollo de un modelo de simulación completo puede requerir esfuerzos sustanciales (Bosch, 2000).

4.5. Métodos de evaluación de Arquitecturas de Software.

De los planteamientos de evaluación establecidos por (Bosch, 2000) y (Kazman, et al., 2001), se tiene que la evaluación de las AS puede ser realizada mediante el uso de diversas técnicas y métodos. En este sentido, resulta interesante estudiar las distintas opciones que existen en la actualidad para llevar a cabo esta tarea. Un estudio realizado por Kazman y sus colegas presenta

tres métodos de evaluación, estos son: *Architecture Trade-off Analysis Method (ATAM)*, *Software Architecture Analysis Method (SAAM)* y *Active Intermediate Designs Review (ARID)*.

4.5.1. Architecture Trade-off Analysis Method (ATAM).

Este método está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM. El nombre del método ATAM surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad y provee una visión de cómo los atributos de calidad interactúan con otros. Los análisis de atributos de calidad específicos deben realizarse teniendo en cuenta su dependencia con el resto de los atributos del sistema. Los atributos de calidad no son independientes, sino que interaccionan entre sí a través de las relaciones estructurales impuestas por la arquitectura (Camacho, et al., 2004).

El método de evaluación ATAM comprende nueve pasos, agrupados en cuatro fases:

Fase 1: Presentación.
Paso 1-Presentar el ATAM.
Paso 2-Controladores de negocios actual.
Paso 3-Presentar arquitectura.
Fase 2: Investigación y análisis.
Paso 4-Identificar los enfoques de arquitectura.
Paso 5-Generar árbol de utilidad de los atributos de calidad.
Paso 6-Analizar enfoques de arquitectura.
Fase 3: Prueba.
Paso 7-LLuvia de ideas y la prioridad de escenarios.
Paso 8-Analizar enfoques de arquitectura.
Fase 4: Reporte
Paso 9-Presentar los resultados.

4.5.2. Software Architecture Analysis Method (SAAM).

Este método fue el primero ampliamente promulgado y documentado. El método fue originalmente creado para el análisis de la modificabilidad de una arquitectura, pero en la práctica ha demostrado ser muy útil para evaluar de forma rápida distintos atributos de calidad, tales como modificabilidad, portabilidad, escalabilidad e integrabilidad. Se enfoca en la enumeración de un conjunto de escenarios que representan los cambios probables a los que estará sometido el sistema en el futuro. Como entrada principal, es necesaria alguna forma de descripción de la arquitectura a ser evaluada (Camacho, et al., 2004).

Las salidas de la evaluación del método SAAM son las siguientes:

- Una proyección sobre la arquitectura de los escenarios que representan los cambios posibles ante los que puede estar expuesto el sistema.
- Entendimiento de la funcionalidad del sistema, e incluso una comparación de múltiples arquitecturas con respecto al nivel de funcionalidad que cada una soporta sin modificación.

El método consta de seis pasos principales los cuales son:

Paso 1 - Desarrollar escenarios.
Paso 2 - Describir la arquitectura.
Paso 3 - Clasificar y priorizar escenarios.
Paso 4 - Evaluar individualmente escenarios indirectos.
Paso 5 - Evaluar la interacción de los escenarios.
Paso 6 - Crear una evaluación global.

4.5.3. Active Intermediate Designs Review (ARID).

ARID es un método de bajo costo y gran beneficio, el mismo es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo. Hacer revisiones en las etapas intermedias provee una valiosa visión de la viabilidad de la arquitectura a construir, además de que permite descubrir errores e inconsistencias en la misma desde el punto de vista de otras partes de la arquitectura.

ARID es un híbrido entre Active Design Review (ADR) y ATAM; del primer método toma la participación activa de los entrevistados, ya que los involucrados reciben documentación detallada y completan cuestionarios, cada uno por separado; del segundo método se queda con la idea de la generación de los escenarios por parte de todos los interesados (*stakeholders*), los usuarios le dirán a los diseñadores que es lo que ellos necesitan o que es lo que ellos esperan para que los diseñadores puedan demostrar en la pruebas que el diseño cumple con los requerimientos. Como resultado de la aplicación de dicho procedimiento se obtiene un diseño de alta fidelidad acompañado de una alta familiarización con el diseño de los *stakeholders* (Clements, 2000).

Este método consta de 9 pasos agrupados en dos fases, a continuación se describen:

Tabla 3. Pasos del método de evaluación ARID.

Fase 1: Actividades Previas	
Identificación de los encargados de la revisión	Ingenieros de software que van a usar el diseño.
Preparar el informe de diseño	El arquitecto prepara un informe que explica el diseño, el mismo deberá ser lo suficientemente detallado como para que una capacitada audiencia pueda usar el diseño.
Preparar los escenarios base	El arquitecto y el líder preparan los escenarios bases que sirven para ilustrar los conceptos a los revisores.
Preparar los materiales	Copias de la presentaciones, escenarios.
Fase 2: Revisión	
Presentación del ARID	El líder utiliza 30 minutos para explicar los pasos de la evaluación a los participantes.
Presentación del diseño	El arquitecto realiza una presentación del diseño mostrando los ejemplos.
Lluvia de ideas y establecimiento de prioridad de escenarios	Se proponen escenarios relevantes para solucionar problemas.
Aplicación de los escenarios	El líder del grupo de revisión es el encargado de probar que el diseño provea los escenarios, comenzando por los de mayor prioridad, hasta que concluya el tiempo estimado de la evaluación, se hayan analizado los escenarios de mayor prioridad, o la conclusión de la revisión satisfaga a los implicados.
Resumen	Al final, el facilitador recuenta la lista de puntos tratados, pide opiniones de los participantes sobre la eficiencia del ejercicio de revisión, y agradece por su participación.

4.6. Evaluación de la propuesta de solución.

Con el objetivo de evaluar la arquitectura propuesta y después de estudiar diferentes métodos y técnicas de evaluación se decide utilizar el método ARID junto a la técnica basada en escenarios con el instrumento perfiles, en el Anexo 5 se puede observar una comparación entre los métodos presentados. Entre las ventajas de este método se nombran su facilidad de uso, su aplicación poco costosa y por el gran beneficio que brinda para realizar una evaluación de la factibilidad de la

arquitectura. Para realizar la evaluación se seleccionaron los atributos de calidad más importantes en aquellos escenarios de acuerdo a un perfil específico. Cada escenario, aparece vinculado a varios atributos los cuales pueden ser afectados por decisiones en torno al diseño. Los atributos que se analizarán para esta estrategia de evaluación temprana son algunos de los propuestos por el estándar ISO 9126, fueron seleccionados según el equipo de arquitectos los de mayor importancia para realizar la evaluación de la propuesta de solución, esto son: Modificabilidad, Portabilidad, Eficiencia y Funcionalidad. El atributo Confiabilidad no fue evaluado pues no se puede medir el nivel de madurez, la tolerancia a fallos y la recuperación ante estos porque no ha sido implementada la capa, por este mismo motivo no se puede evaluar el esfuerzo que deberá invertir el usuario para utilizar el sistema que describe el atributo Usabilidad.

Debido a que en la presente investigación los encargados de llevar a cabo la evaluación de la arquitectura propuesta serán los integrantes del grupo de arquitectura que definieron la misma, no se siguieron estrictamente los pasos propuestos por el ARID. Los arquitectos del sistema tienen un dominio completo del diseño arquitectónico del mismo, y presentan conocimientos del método a aplicar para la evaluación, por lo que no se realizó la fase1, pasando directamente a aplicarse los tres últimos pasos de la fase 2.

A continuación se muestran diferentes tablas que resumen los pasos finales de la fase última del método ARID formado por: el establecimiento de prioridad de escenarios, la aplicación de estos y el resumen de la relación entre los atributos de calidad y dichos escenarios. Estos escenarios han sido escogidos por ser primordiales para evaluar algunos de los atributos escogidos del estándar ISO 9126.

Tabla 4. Migración del Sistema Gestor de Base de Datos.

Escenario # 1	**Migración del Sistema Gestor de Base de Datos**
Atributos de Calidad	Modificabilidad
Perfil	Mantenibilidad
Relación Atributo-Escenario	
El diseño de clases propuesto está creado de tal manera, que si se hace necesario cambiar el SGBD solo habría que configurar un ORM que soporte ese nuevo gestor. El acceso a la BD es transparente a	

las demás clases del diseño propuesto, facilitando así que se tenga que realizar un mínimo esfuerzo para adaptar el software a un ambiente diferente.

Tabla 5. Migración de Sistema Operativo.

Escenario # 2	Migración de Sistema Operativo
Atributos de Calidad	Portabilidad
Perfil	Adaptabilidad
Relación Atributo-Escenario	
Para el desarrollo de este componente, se propuso como lenguajes de programación PHP, HTML, JavaScript, como servidor Web Apache y como SGBD PostgreSQL. Estos elementos son compatibles con la mayoría de los sistemas operativos más usados en la actualidad al ser considerado multiplataforma y libres. Por esto se puede concluir que el componente se puede adaptar a diferentes ambientes sin la necesidad de aplicar modificaciones.	

Tabla 6. Aumento del modelo de datos.

Escenario # 3	Aumento del modelo de datos
Atributos de Calidad	Eficiencia
Perfil	Comportamiento con respecto a Recursos
Relación Atributo-Escenario	
El SGBD PostgreSQL brinda la posibilidad de crear tantos clústeres de bases de datos como sea necesario. Esto permite aumentar la capacidad de almacenamiento y disminuir la sobrecarga en los servidores, en caso de concurrencia o el espacio en estos sea insuficiente. Por tanto el aumento del	

modelo de dato no dificulta el funcionamiento normal del sistema.

Tabla 7. Agregar nuevas interfaces de usuario al sistema.

Escenario # 4	Agregar nuevas interfaces de usuario al sistema
Atributos de Calidad	Funcionalidad
Perfil	Adecuación
Relación Atributo-Escenario	
El diseño de la arquitectura propuesta, posee una estructura de clases, que permite la escalabilidad de las interfaces de usuario. La propuesta de clases fue prevista para dos tipos de interfaces, una general y otra avanzada, pero se puede adicionar tantas como se necesite, solo será necesario agregar una nueva interfaz y una clase controladora que pueda manejar sus datos. Esto es posible porque el diseño propuesto es flexible.	

Tabla 8. Agregar nuevos indicadores de perforación.

Escenario # 5	Agregar nuevos indicadores de perforación
Atributos de Calidad	Funcionalidad
Perfil	Adecuación
Relación Atributo-Escenario	
Una de las recomendaciones de este trabajo, es continuar el estudio de nuevos indicadores de perforación, para posteriormente ser agregados al buscador de indicadores. En la propuesta de solución, se identificaron 3 indicadores, pero la arquitectura propuesta, permite que se puedan agregar nuevos indicadores de perforación, como entidades del negocio de manera fácil y flexible, esto se debe	

al bajo acoplamiento que existe entre sus componentes.

La evaluación cualitativa realizada refleja que el diseño propuesto cumple con los atributos de calidad de la norma ISO 9126 que fueron evaluados. Se demuestra además que la utilización de los patrones de diseño seleccionados para guiar el diseño de clases propuesto da cumplimiento a los atributos de calidad de Modificabilidad y Funcionalidad, mientras que las tecnologías propuestas satisfacen los atributos de Portabilidad y Eficiencia. A su vez se arrojan resultados que evidencian la solución de la situación problemática que da vida a esta investigación. Por lo anteriormente planteado se puede afirmar que el modelado de la arquitectura y diseño propuesta, es adecuado para desarrollar el componente de búsqueda de indicadores de perforación.

4.7. Conclusiones Parciales.

En el presente capítulo se han analizado los elementos fundamentales relacionados con la evaluación de la arquitectura y diseño del componente, se estudiaron los principales métodos y técnicas que se pueden utilizar en la evaluación, los atributos de calidad por los que se puede evaluar y las etapas en las que se puede realizar una evaluación. Este estudio posibilitó que se pudieran seleccionar los métodos, técnicas y atributos para evaluar la propuesta de solución. Se obtuvo un resultado satisfactorio ya que la arquitectura cumple con cada uno de los escenarios que se definieron, entre otros elementos.

CONCLUSIONES GENERALES

Luego de haber finalizado el presente trabajo se puede concluir que con el modelado de la arquitectura del componente de búsqueda de indicadores de perforación, se sentaron bases sólidas para su posterior implementación. Con el estudio del estado del arte de los componentes desarrollados con este propósito a nivel nacional e internacional, se pudo determinar que ninguno de ellos posee un componente como la propuesta de solución. El empleo de escenarios de evaluación, basados en las variables de calidad y en los perfiles descritos en el ISO 9126, permitió evaluar la buena calidad de la arquitectura propuesta y su flexibilidad. Además la presente investigación aporta una solución factible a la situación problemática que lo originó y su aplicación significará una mejora considerable en la calidad y eficiencia del desarrollo de sistemas para el área de la perforación de pozos de petróleos, cumpliéndose de esta forma los objetivos de la investigación.

RECOMENDACIONES

Se recomienda continuar el análisis del negocio para identificar nuevos indicadores de perforación y posibles relaciones entre ellos, así como la posterior implementación de dicho componente guiado por la arquitectura propuesta.

BIBLIOGRAFÍA

1. **Apache Foundation. 2011.** The Apache Software Foundation. [Online] 2011. http://www.apache.org/.

2. **Bosch, J. 2000.** *Design & Use of Software Architectures.* s.l. : Addison-Wesley, 2000.

3. **Brey, Gustavo Andrés, et al. 2005.** *Evaluación de Arquitecturas.* Universidad Tecnológica Nacional. Facultad Regional de Buenos Aires : s.n., 2005.

4. **Camacho, Erika, Cardeso, Fabio and Nuñez, Gabriel. 2004.** *Arquitectura de Software: Guía de Estudio.* 2004.

5. **Castellano., Portal de AJAX en.** Comunidad de AJAX para aprender desde cero. [Online] [Cited: 03 17, 2011.] http://www.ajaxfacil.com/.

6. **César de la Torre Llorente, Unai Zorrilla Castro , Javier Calvarro Nelson, Miguel Ángel Ramos Barroso. Noviembre 2010.** *Guía Arquitectura N-Capas Orientada al Dominio - Microsoft Architecture.* s.l. : 1a Edición, Noviembre 2010.

7. **Clements, Paul C. 2000.** *Active Reviews forIntermediate Designs.* 2000.

8. **Eguíluz Pérez, Javier. 2011.** LibrosWeb. [Online] 03 04, 2011. http://www.librosweb.es/ajax..

9. **Florencia. 2008.** Definicion ABC. [En línea] 31 de 12 de 2008. [Citado el: 01 de 03 de 2011.] http://www.definicionabc.com/general/buscar.php.

10. **Gamma, Erich, et al. 1998.** *Design Patterns - Elements of Reusable Object-Oriented Software.* 1998.

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

12. **IBM, IBM.** [Online] [Cited: 02 13, 2011.] http://www-142.ibm.com/software/products/es/es/enterprise..

13. **ife.org.** [Online] [Cited: 03 12, 2011.] http://www.ife.org.mx/documentos/Reforma_Electoral/link_glosario.htm..

14. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

15. **Kazman, R., Clements, P. and Klain, M. 2005.** *Evaluating Software Architectures. Methods and case studies.* s.l. : Adison-Wesley Professional, 2005. ISSN: 978-0201704822 .

16. **Kazman, Rick, Clements, Paul and Klain, Mark. 2001.** *Evaluating Software Architectures. Methods and case studies.* s.l. : Adison-Wesley Professional, 2001. ISSN: 978-0201704822.

17. **Kiccillof, Nicolás and Reynoso, Carlos. 2004.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

18. **Kruchten, Philippe and Wesley, Addison. 2000.** *The Rational Unified Process: An Introduction. .* 2000. 0-201-70710-1.

19. **La biblioteca, escolar.** La biblioteca escolar com a eina educativa. [Online] [Cited: 03 25, 2011.] www.bibliotecaescolar.info/doc/documentos.

20. **Larman, Craig. 2000.** *UML y Patrones,Introducción al análisis y diseño orientado a objetos. .* Mexico : s.n., 2000.

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

22. **Línea, Estamos en. 2011.** Estamos en Línea . [Online] 2011. http://www.estamosenlinea.com.ve/2009/12/17/sun-microsystems-presenta-netbeans-ide-6-8/.

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

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

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

26. **Nielsen, Jakob. 2001.** useit.com. [En línea] 2001. http://useit.com.

27. **Paneque, Jorge Roberto Allen. 2010.** *Módulo Pozo del Sistema de Manejo Integral de Perforación de Pozos.* Ciudad de la Habana : s.n, 2010.

28. **Paradigma Visual**. [Online] 01 23, 2011. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%2.

29. **Pérez, Martín Mariñán. 2011.** *Patrones de Diseño.* 2011.

30. **Pressman, R. 2002.** *Ingeniería del software: un enfoque práctico.* 2002.

31. **Rodríguez, Milton Benito León. 2010.** *Diseño del Segundo Ciclo de Desarrollo del Sistema de Manejo Integral de Perforación de Pozos.* Ciudad de la Habana : s.n, 2010.

32. **Salvador Oliván, José A. 2008.** Recuperación de Información. [Online] Alfagrama, 2008. [Cited: 03 01, 2011.]

33. **SEI. 2011.** Software Engineering Institute. [Online] 2011. http://www.sei.cmu.edu/architecture/.

34. **Softonic, Softonic. 2009.** *Softonic. Softonic.* [Online] 2009. http://netbeans-ide.softonic.com/.

35. **Tecnologías. Tecnologías.** [Online] 11 25 , 2009. [Cited: 03 24, 2011.] http://www.geopalm.cl/tecnologia/uml.html .

36. **UP, Agile.** Agile UP. *Agile UP.* [Online] [Cited: 03 11 , 2010.] http://cgi.una.ac.cr/AUP/html/overview.html.

37. **Wilson, Leslie B. 1993.** *Comparative Programming Languages, Second Edition.* 1993.

ANEXOS

Anexo 1: Comparación entre Servidores Web.

	Apache	Internet Information Server
Sistemas Operativos	Multiplataforma	Windows
Lenguaje que soportan	PHP, Perl, Python, Ruby, Coldfusión, mysql, jsp, SQL, MSsql, asp, asp.net, xml, ajax	ASP, ASP.net, PHP, Perl, vbscript, Ajax, msSql, mysql, xml, Acsses
Licencia	Licencia Apache	Software privativo
Por ciento de uso en el mundo	61.13% (abril 2011)	18.83% (abril 2011)
Última versión	2.2.17 (19 de octubre de 2010)	7.5 (2008)
Facilidad de administración	Menor	Mayor

Anexo 2: Descripción de atributos de calidad observables vía ejecución.

Atributos de calidad	Descripción
Disponibilidad	Es la medida de disponibilidad del sistema para el uso.
Confidencialidad	Es la ausencia de acceso no autorizado a la información.
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido.
Desempeño	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones como velocidad, exactitud o uso de memoria.
Confiabilidad	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo.
Seguridad Externa	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información.

Seguridad Interna	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.

Anexo 3: Descripción de atributos de calidad no observables vía ejecución.

Atributos de calidad	Descripción.
Configurabilidad	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema.
Integrabilidad	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados.
Integridad	Es la ausencia de alteraciones inapropiadas de la información.
Interoperabilidad	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de integrabilidad.
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema.
Mantenibilidad	Es la capacidad de someter a un sistema a reparaciones y evolución.
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos.
Reusabilidad	Es la capacidad de diseñar un sistema de forma tal que su estructura o partes de sus componentes puedan ser reutilizados en futuras aplicaciones.
Escalabilidad	Es el grado con el que se puede ampliar el diseño arquitectónico, de datos o procedimental.
Capacidad de prueba	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su próxima ejecución de prueba.

Anexo 4: Características y subccaracterísticas de calidad – Modelo ISO/IEC 9126

Características	Subccaracterísticas
Funcionalidad	• Adecuación • Exactitud • Interoperabilidad • Seguridad
Confiabilidad	• Madurez • Tolerancia a fallas • Recuperabilidad
Usabilidad	• Entendibilidad • Capacidad de aprendizaje • Operabilidad
Eficiencia	• Comportamiento en tiempo • Comportamiento de recursos
Mantenibilidad	• Analizabilidad • Modificabilidad • Estabilidad • Capacidad de pruebas
Portabilidad	• Adaptabilidad • Instalabilidad • Reemplazabilidad

Anexo 5: Comparación entre 3 métodos de evaluación (Pressman, 2002).

	ATAM	SAAM	ARID
Atributos de Calidad contemplados	✓ Modificabilidad. ✓ Seguridad. ✓ Confiabilidad. ✓ Desempeño.	✓ Modificabilidad. ✓ Funcionalidad.	✓ Conveniencia del diseño evaluado.
Objetos analizados	✓ Estilos arquitectónicos. ✓ Documentación. ✓ Flujo de datos Vistas. ✓ Arquitectónicas.	✓ Documentación. ✓ Vistas arquitectónicas.	✓ Especificación de los componentes.
Etapas del proyecto en las que se aplica	✓ Luego de que el diseño de la arquitectura ha sido establecido.	✓ Luego de que la arquitectura cuenta con funcionalidad ubicada en módulos.	✓ A lo largo del diseño de la arquitectura.

Enfoques utilizados	✓ Utility Tree y lluvia de ideas para articular los requerimientos de calidad. ✓ Análisis arquitectónico que detecta puntos sensibles, puntos de balance y riesgos	✓ Lluvia de ideas para escenarios y articular los requerimientos de calidad. ✓ Análisis de los escenarios para verificar funcionalidad o estimar el costo de los cambios.	✓ Revisiones de diseños, lluvia de ideas para obtener ✓ Escenarios.