Universidad de Ciencias Informáticas

Facultad 10

Título: "Guía para la obtención de la adaptabilidad y la coexistencia de las soluciones de software de apoyo a la toma de decisiones".

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

Autores:
Danay Oropeza Gálvez.
Hector Delgado Anaya.

Tutor:
Ing. Nadia Porro Lugo.

Co-Tutor:
Msc. Michaell González Jorrín.

Ciudad de la Habana, junio 2010.

"Año del 52 Aniversario de la Revolución"

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de junio del año 2010.

Danay Oropeza Gálvez. **Hector Delgado Anaya.**

_____ _____

Firma de los Autores.

Ing. Nadia Porro Lugo. **Msc. Michaell González Jorrín.**

_____ _____

Firma de los Tutores.

"(…) la lucha por la calidad del producto es una lucha revolucionaria y de vanguardia. Y nunca se equivoque en pensar, que por el hecho de ser revolucionario se puede dar al pueblo un producto de mala calidad, eso sería atentar contra la Revolución."

Ernesto Che Guevara.

AGRADECIMIENTOS DE "DANAY"

A mi familia por haber luchado a mi lado y permitirme superarme para alcanzar todas mis metas a lo largo de estos años.

A mi mamá, por darme la oportunidad de existir, por confiar en mi, por su sacrificio en los tiempos más difíciles, por su ejemplo de superación incasable, por su amor y amistad incondicional y sobre todo porque sin su apoyo no hubiera sido posible alcanzar este sueño.

A mi papá por el ejemplo que me ha brindado y por permitir realizarme como persona.

A mis tíos, en especial a mi tío Lando por estar siempre presente para mí y por ser mi segundo papá.

A mis abuelos por su preocupación y apoyo durante toda mi vida, y muy especialmente a mis abuelos maternos Ramona y Antonio por ser los mejores del mundo.

A mis primos, en especial a Lisbety, Heriberto y Erisbel que más que mis primos son mi vida.

A todas mis amigas en especial a mis nenas: Lizi a la que considero como a una madre, Nana por ser mi conciencia, Ceci a la que le debo muchas cosas entre ellas mi forma de ver la vida. A Yordania, Yailen, Dayani, y Elianne por aguantar mi mal carácter y todas mis malcriadeces, además por compartir conmigo los buenos y malos momentos, por ser las amigas que quedan para toda la vida.

A todos mis amigos, en especial a: Ronny, Yuri, Fidel y Victor por formar parte de mi vida durante estos cinco años.

A Virginia, María Antonia y María Eugenia por abastecernos de papita rica y por permitirme conocer a estas grandes amigas.

A mi compañero de tesis por todos los sueños que hemos logrado juntos, por la confianza que depositó en mí, por haber sido un gran amigo y por ampliar mis niveles de paciencia.

A mi tutora por la dedicación que brindó en la realización de esta tesis.

A todos mis profesores por haber dedicado parte de su tiempo a formarme como una mejor profesional.

Y por ultimo agradecer al Comandante en Jefe Fidel Castro por permitirme realizarme como una profesional y poder ejercer como Ingeniera en Ciencia Informáticas.

AGRADECIMIENTOS DE "HECTOR"

A mi abuela Ángela por iluminarme y protegerme siempre.

A mi gran madre Esmérida Maldonado Tamé, nuestra mimi, por cuidarme y darme tanto amor.

A mi madre Milagros, por darme la vida y permitirme crecer.

A mi abuelo Miguel Anaya Vega, por darme el mejor ejemplo de padre que he tenido jamás.

A mis hermanos Lisbeth, Javier Miguel y Dianelys.

A mis tíos Clara, Daniel y a su esposa Olaida, los que aprecio mucho.

A mis primos Frank, Yaniel y Yunita por las peleas constantes y los buenos momentos al final del día.

A Mercedes, Eusebio, Yanelis Yarenis y Yahilys, los que amo mucho.

A mi amigo Dayron, por demostrarme ser un amigo, fiel y leal, aunque en ocasiones pelee demasiado.

Al resto de mi familia por su cariño sincero y por su ayuda prestada.

A Yayneris y Yaimé, por pasar conmigo grandes momentos de felicidad.

A mis compañeros del IPCVE: Vila, Abelito, Norlis, Rubén, Guille, Yunior, Yasser, Iosmelis con los que pase momentos inolvidables.

A mis compañeros de la Universidad los cuales me ayudaron a conocer mejor la vida, las personas y el mundo.

A mis amigas del salón de impresión Isabel, Clarita, Merci, Yahumara, Dania y Salomé quienes me han soportado tanto pero tanto que con agradecimiento no basta lo que han hecho por mí.

A Danay Oropeza Gálvez mi compañera de Tesis, por haber compartido conmigo estas páginas de sacrificio y de noches de insomnio. La cual admiro y respeto. Le deseo mucho éxito en la vida.

A mi tutora la Ing. Nadia Porro Lugo por confiarnos la realización de esta Tesis.

A la Ing. Eneybis García Soto nuestra oponente, que descubrimos que sin ella, nuestra tesis no hubiese sido perfecta.

A mis profesores todos, por darme de ellos la mejor de las enseñanzas y ayudar en nacer de mí un profesional.

A Fidel Castro por brindarme la posibilidad de graduarme en la Universidad de Ciencias Informáticas.

DEDICATORIA.

A mi mamá Griselda, por enseñarme con su ejemplo y ser la luz que me ha guiado en todos los momentos de mi vida.

A mi papá Leonardo por ser más que mi padre, por ser mi amigo.

A mi abuela Pelusa por quererme tanto y por ser lo máximo en mi vida.

A mi tío Lando por ser mi segundo papá.

Danay.

A mis abuelos Miguel Anaya Vega y Esmérida Maldonado Tamé.

Hector.

Resumen

Algunos desarrolladores de software consideran que la calidad de los productos sólo se debe tener en cuenta después que se ha creado el código. Raramente los problemas que aparecen en fases posteriores admiten soluciones, poniendo en riesgo la profesionalidad del equipo de desarrollo y la calidad del producto final.

El presente trabajo se centra en actividades específicas del proceso de desarrollo que permiten a empresas o entidades, garantizar que el producto final se lleve a cabo de forma correcta y en el momento justo.

Siguiendo la línea de la investigación dentro del proceso de desarrollo de software, se persigue obtener la adaptabilidad y la coexistencia, por lo que se propone la elaboración de una guía para buenas prácticas orientada a optimizar la característica de portabilidad. La misma incluye el empleo de normas y modelos a fin de lograr todos los requerimientos que se necesitan, para crear un producto con la calidad necesaria.

La propuesta a desarrollar brinda aspectos fundamentales que deben estar presentes en el proceso de elaboración de un software, es ahí donde se cometen la mayoría de los errores. Se detallaron los factores que favorecen la adaptabilidad y la coexistencia, precisando con claridad los puntos de vista de cada uno, ofreciendo criterios y planteando una solución, la cual fue avalada por varios especialistas, dándole a la investigación un alto valor científico-técnico para mejorar el proceso de producción de software.

Palabras Clave: Normas y Modelos de Calidad, Adaptabilidad, Coexistencia, Portabilidad, Proceso de Desarrollo de Software.

Índice

Introducción.

El desarrollo de las TICs[1] ha provocado un aumento en el conocimiento, el cual se ha convertido en un papel importante en el progreso de la humanidad. Debido a la creación de nuevos productos donde la reducción del coste y el tiempo de producción se tornan un factor decisivo para el aumento de las ganancias y el posicionamiento en el mercado, las nuevas empresas surgen y luchan porque el software que producen sea lo que realmente necesitan los clientes. Consecuentemente, la industria del software se extiende sobre lugares cimeros, y producir con calidad y eficiencia es el principal objetivo de nuestra sociedad.

Las empresas desarrolladoras de software trabajan día a día dirigiendo sus estudios a las mejoras continuas que se realizan en el proceso de desarrollo del software, para que el producto final cumpla con las nuevas normas y modelos de calidad. Gracias a los servicios y requerimientos que estos plantean, la evaluación del software se hace más exigente y cumplir con los requisitos descritos en estos es de vital necesidad para el desarrollo de cualquier producto software.

Iniciar el desarrollo de un software depende de la puesta en práctica de una sucesión de acciones que se denominan planificación del proyecto. Antes de aplicar estas actividades se debe llevar a cabo una estimación del trabajo a realizar por el equipo de desarrollo, donde se tendrán en cuenta los recursos necesarios y el tiempo que transcurrirá desde el inicio hasta el final de su realización.

Cuando se planifica un proyecto se tienen que obtener estimaciones del costo y esfuerzo humano requerido, por medio de las mediciones de software que se utilizan para recolectar los datos cualitativos acerca del software y sus procesos, para aumentar su calidad. [1]

Como parte del proceso de informatización del país, la industria cubana del software ha ido avanzando satisfactoriamente, existiendo diferentes empresas como Softel, CITMATEL, Desoft, Centersoft donde profesionales con la capacitación necesaria producen software y brindan servicios informáticos con la mayor calidad posible.

La tecnología se actualiza continuamente, influyendo en las diferentes ramas de las ciencias y sectores sociales. Entre las transformaciones educacionales y el novedoso proceso de desarrollo social, surge

[1] Tecnologías de la Información y las Comunicaciones.

como nuevo programa de la Batalla de Ideas en aras de formar ingenieros informáticos capacitados la UCI[2], para contribuir con el desarrollo de la economía cubana.

La universidad desarrolla proyectos donde millones de bits de información se generan y circulan a través de la red. En la misma existe un centro de desarrollo nombrado DATEC[3], donde se gestionan e implementan soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL, a las cuales es necesario controlar y certificar su calidad. Estudios realizados determinaron la siguiente

Situación problémica:

- Carencia de conocimientos sobre las normas para controlar la portabilidad en las soluciones de software.
- Inexistencia de un proceso bien definido que permita controlar la portabilidad de las soluciones de software.
- Falta de experiencia por parte de los evaluadores de calidad en el uso de modelos y normas de calidad asociadas al proceso de desarrollo del software.
- Se ignora el esfuerzo que requiere el usuario para adaptar las soluciones de software a cualquier plataforma.
- Poco conocimiento relativo al comportamiento del software al coexistir con otros en un determinado ambiente.

Todo lo antes expuesto evidencia ausencia de interacción con el cliente de las soluciones de software en el proceso de las pruebas de aceptación, lo que afecta en gran medida la credibilidad como productores de software.

Para el desarrollo de una adecuada investigación e implementación, se hizo necesario expresar estas ideas de forma conjunta en el siguiente **problema científico a resolver**: ¿Cómo obtener la adaptabilidad y la coexistencia de las soluciones de software de apoyo a la toma de decisiones basadas en bases de datos relacionales gestionadas con PostgreSQL?

El **objeto de estudio** de la investigación se expone en la adaptabilidad y la coexistencia de las soluciones de software, delimitando el **campo de acción** en la adaptabilidad y la coexistencia de las

[2] Universidad de la Ciencias Informáticas.

[3] Centro de Tecnologías de Gestión de Datos.

soluciones de software de apoyo a la toma de decisiones basadas en bases de datos relacionales gestionadas con PostgreSQL.

La presente investigación tiene como **objetivo general:** Elaborar una guía para diseñar, configurar y lograr la adaptabilidad y la coexistencia de las soluciones de software de apoyo a la toma de decisiones basadas en bases de datos relacionales gestionadas con PostgreSQL.

Con el propósito de darle cumplimiento al mismo se trazan los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Obtener una guía para diseñar, configurar y lograr la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.
- Validar la propuesta a la cual se arribó, luego de concluida la investigación.

Posibles resultados:

- Parámetros para obtener la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.
- Situación actual de la obtención de la adaptabilidad y la coexistencia en bases de datos relacionales gestionadas con PostgreSQL.
- Guía para diseñar, configurar y lograr la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.
- Análisis de los resultados luego de aplicada la guía propuesta.

El trabajo queda sustentado en la siguiente **idea a defender:** confeccionar una guía para la obtención de la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL, permitirá una mayor calidad y una mejor obtención de estas subcaracterísticas durante el proceso de desarrollo del software.

Con el propósito de dar validez a lo anteriormente planteado se formulan las siguientes **tareas de investigación:**

- Selección y revisión bibliográfica para actualizar los logros y limitaciones existentes sobre la obtención de la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.

- Análisis de modelos de calidad tales como McCall, Boehm y el estudio de la norma ISO/IEC 9126, y otros, con el propósito de determinar los parámetros para obtener la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.
- Comparación con tecnologías similares a PostgreSQL.
- Identificación de los involucrados potenciales de DATEC en cuanto a la obtención de la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL, para la realización de un diagnóstico que permita conocer el estado actual del tema.
- Determinación de buenas prácticas para diseñar, configurar y lograr la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.
- Obtención de los riesgos potenciales asociados a las subcaracterísticas de calidad estudiadas para la determinación de los problemas más frecuentes.
- Aplicación de la guía a soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL para determinar si ha cambiado el estado inicial de la obtención de la adaptabilidad y la coexistencia.

Se empleó el **método científico de investigación**, ya que permite abordar la realidad, estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. [2]

Los métodos científicos de investigación proponen un procedimiento para llevar a cabo cualquier investigación, ya que este establece una mejor organización del trabajo y exige el establecimiento de un conjunto de reglas con objetivos bien definidos. Dentro de las clasificaciones que se le dan los métodos científicos están: métodos teóricos y métodos empíricos, y aunque difieren en el nombre estos están dialécticamente relacionados.

Método teórico: *Permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, contribuyendo al desarrollo de las teorías científicas y para su ejecución se apoyan en el proceso de análisis y síntesis.* [2]

Método empírico: *Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional*. [3]

Los métodos teóricos utilizados fueron:

Método Analítico–sintético: este método separa en pequeñas partes todos los factores para ser analizados y luego se sintetizan obteniendo relaciones y características de estos factores. Se aplicó este método en el análisis de modelos de calidad como el Modelo de McCall y el Modelo de Boehm, además para estudiar la norma ISO/IEC 9126, obteniendo las descripciones y evaluaciones de las características de calidad y determinando así, los parámetros que permitan alcanzar la adaptabilidad y la coexistencia de las soluciones de software. También se utilizó con el objetivo de comprobar si las tecnologías similares a PostgreSQL cumplen con estas subcaracterísticas.

Los métodos empíricos utilizados fueron:

Método Encuesta: este método se emplea a través de preguntas o cuestionarios previamente elaborados con el fin de obtener una información crucial para la investigación. En este proceso se encuestará a un grupo de individuos del proyecto de DATEC, para validar la existencia de factores que favorecen la adaptabilidad y la coexistencia de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.

Método Entrevista: la entrevista es una técnica para obtener información mediante una conversación profesional entre un entrevistado y el entrevistador. La misma se lleva a cabo a través de un diálogo o un cuestionario previamente elaborado. Su uso atribuye conocimiento cualitativo a la investigación, puede ser individual o colectiva. El investigador debe tener bien claro lo que desea preguntar ya que esta puede requerir de mucho tiempo. En la investigación se usa la entrevista para validar la propuesta solución, se entrevistarán una serie de expertos que darán su criterio, obteniéndose los resultados luego del análisis de sus respuestas.

El presente trabajo se divide en 3 capítulos, a continuación se presenta el nombre del capítulo y su objetivo desde un contexto global:

Capítulo #1:“FUNDAMENTACIÓN TEÓRICA”. El objetivo de este capítulo es abordar conceptos generales y básicos que permitan comprender temas relacionados con los sistemas de apoyo a la

toma de decisiones, los generadores de reportes dinámicos y las bases de datos relacionales gestionadas con PostgreSQL. De igual forma se exponen los aspectos que determinan la necesidad de la implementación de la propuesta solución.

Capítulo #2:"CARACTERÍSTICAS DE LA PROPUESTA SOLUCIÓN". Este capítulo tiene como principal objetivo proponer una solución para corregir las dificultades existentes.

Capítulo #3:"ANÁLISIS DE LOS RESULTADOS OBTENIDOS". El objetivo del vigente capítulo es mostrar un conjunto de resultados y conclusiones generales de la aplicación de la propuesta planteada.

Capítulo 1: Fundamentación Teórica.

En el presente capítulo se expondrán los principales conceptos que sustentan esta investigación, lo que permitirá un mejor entendimiento de la misma. El estudio estará enfocado en las subcaracterísticas de adaptabilidad y coexistencia de las soluciones de software de apoyo a la toma de decisiones basadas en base de datos relacionales que son gestionadas con PostgreSQL.

Se analizarán Modelos de Calidad de Software como McCall y Boehm. Además la norma ISO/IEC 9126, con el propósito de analizar el enfoque dado en cada uno de ellos y comparar las subcaracterísticas en los mismos. Se abordarán temas como sistemas de apoyo a la toma de decisiones, generador de reporte dinámico, base de datos, sistemas gestores de base de datos, y cómo se ven reflejadas las subcaracterísticas de portabilidad en los mismos.

1.1. Calidad de Software.

El desarrollo de software genera una serie de actividades en las que encontrar un error humano es enorme. Los errores pueden verse desde el primer momento del proceso, en que la recopilación de requisitos del software son seleccionados de forma incorrecta o imperfecta. Debido a que los seres humanos no son perfectos cada paso debe ir acompañado de una labor para garantizar la calidad.

El objetivo no es necesariamente alcanzar una calidad perfecta, sino la necesaria y suficiente para cada contexto a la hora de la entrega y del uso por parte de los clientes. Es necesario comprender las necesidades reales de los usuarios con tanto detalle como sea posible (requisitos).

Según Roger Pressman el concepto de Calidad de Software puede ser modificado o ampliado, por lo que se expone de la siguiente manera:

Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente.[4]

Otras definiciones relacionadas con del tema son:

La calidad del software es el conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas.[4]

La calidad del software es el grado con el que un sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario.[5]

Según la norma ISO/IEC 9126 *la calidad del software en un conjunto estructurado de características y subcaracterísticas que le confieren calidad al producto de software.*[6]

Calidad de Datos: aquellas características que deben tener los datos como materia prima, para que utilizando un proceso de producción adecuado se pueda generar un producto de información.

Calidad de Información: características que debería poseer un Producto de Información (PI) para que su utilización sea adecuada, es decir, que cumpla con los requisitos del usuario.

Según Eppler la Calidad de información *es la característica de la información para satisfacer los requisitos funcionales, técnicos, cognitivos y estéticos de los productores, consumidores, administradores y expertos*. [7]

Métrica: *valor numérico o nominal asignado a características o atributos de un ente computado a partir de un conjunto de datos observables y consistentes con la intuición. Una métrica puede ser directa o indirecta, interna o externa, objetiva o subjetiva.*[8]

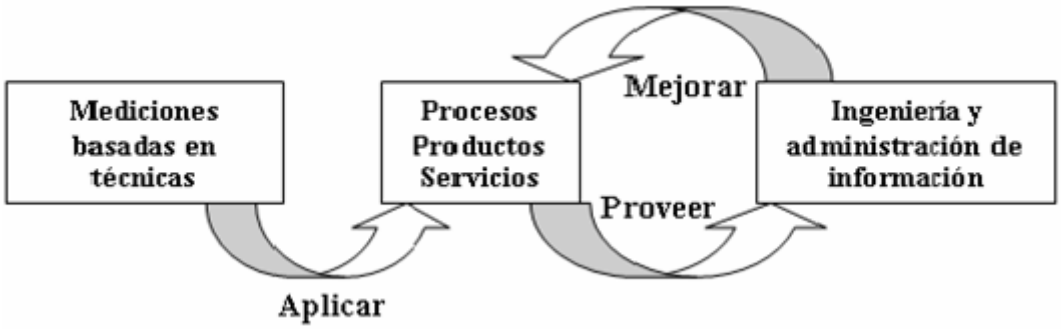

Fig. 1 Concepto de métrica.

Métricas de calidad: son todas las métricas de software que definen de una u otra forma la calidad del software como por ejemplo: la exactitud, la estructuración o modularidad, las pruebas, el mantenimiento y la reusabilidad del software.

1.1.1. Modelos en la Gestión de Calidad.

Los modelos de calidad del software ayudan a poner en práctica los conceptos de calidad que se han venido citando hasta el momento, ofreciendo una definición más operacional. Unos de los modelos de calidad más antiguos y extendidos es el de McCall, y de él se han derivado otros, como el de Boehm o el SQM[4].

En los modelos de calidad, la misma se define de forma jerárquica. Es un concepto que se deriva de un conjunto de sub-conceptos, cada uno de los cuales se van a evaluar a través de un conjunto de indicadores o métricas.

Los modelos de calidad tienen una estructura, por lo general, en tres niveles:

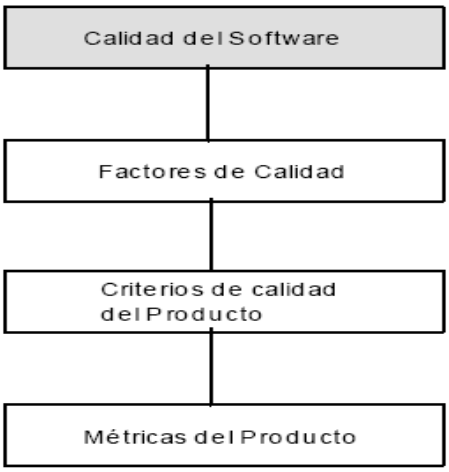

Fig. 2 Estructura de un modelo de calidad.

Los factores de calidad se pueden encontrar en el nivel más alto de la jerarquía. Para medir la calidad de un software, en estos modelos se tienen en cuenta los atributos de calidad externos del software ya que ellos plantean la medición desde el punto de vista del usuario, siendo estas las características que componen la calidad.

Sin embargo a los atributos de calidad internos también se les atribuye gran popularidad a la hora de la medición, ya que estos son antecesores de los atributos externos. Se trata de una visión de la calidad desde el punto de vista del producto software. Siendo de gran importancia ya que los atributos internos

[4] Software Quality Management.

están disponibles para medición desde más temprano y son más fáciles de medir. Los factores de calidad se descomponen en un conjunto de criterios de calidad.

En cada uno de los criterios de calidad son necesarias las métricas, para definir el nivel en que el producto posee un atributo de calidad, aportándole al mismo un valor cuantitativo. Los modelos son de gran importancia, ya que convierten la calidad en algo concreto, se pueden definir, medir y sobre todo planificar. Ayudando así a comprender las relaciones que existen entre las diferentes características de un producto de software.

En ocasiones se hace un tanto difícil comprender los conceptos expuestos por los diferentes modelos, ya que estos difieren en plantearnos la distinción entre características, atributos y métricas. También se suma que estos modelos no han demostrado su validez absoluta, siendo las manifestaciones planteadas por ellos resultados de la experiencia, consecuencia de esto es la existencia de múltiples modelos de calidad.[9]

Modelo de McCall.

El modelo de McCall fue el primero en ser presentado en 1977. Se enfoca en el producto final, identificando atributos claves desde el punto de vista del usuario. Este modelo organiza los factores en tres ejes o puntos de vista desde los cuales el usuario puede contemplar la calidad de un producto. Se basa en 11 factores de calidad, los que se organizan en torno a los tres ejes, cada uno de estos factores se descomponen, a su vez en criterios, McCall define un total de 23 criterios.

La relación Punto de Vista - Factores - Criterios que establece el modelo queda modelada en la siguiente tabla: [10]

Punto de Vista	Factores	Criterios
Operación de Producto.	Facilidad de uso.	- Facilidad de operación. - Facilidad de Comunicación. - Facilidad de Aprendizaje.
	Integridad.	- Control de Accesos. - Facilidad de Auditoria.
	Corrección.	- Completitud.

		- Consistencia. - Trazabilidad.
	Fiabilidad.	- Precisión. - Consistencia. - Tolerancia a fallos. - Modularidad. - Simplicidad.
	Eficiencia.	- Eficiencia en ejecución. - Eficiencia en almacenamiento.
Revisión de Producto.	Facilidad de mantenimiento.	- Simplicidad. - Consistencia. - Concisión. - Auto descripción.
	Facilidad de Prueba.	- Simplicidad. - Auto descripción. - Instrumentación.
	Flexibilidad.	- Auto descripción. - Capacidad de expansión. - Generalidad. - Modularidad.
	Reusabilidad.	- Generalidad. - Modularidad. - Independencia entre Sistema y Software. - Independencia del hardware.

Transición de Producto.	Interoperabilidad.	- Modularidad. - Compatibilidad de comunicaciones. - Compatibilidad de datos.
	Portabilidad.	- Auto descripción. - Independencia entre Sistema y Software. - Independencia del hardware.

Tabla 1: Relación Punto de Vista - Factores - Criterios que establece el modelo de McCall.

El modelo McCall propuso 41 métricas, de tipo subjetivo, es decir, métricas que evaluadas por personas diversas lograrían dar como resultado valores diferentes. Hoy en día no existen métricas formales y objetivas que cubran todos los criterios planteados por este modelo.

La Transición del Producto es uno de los tres puntos de vista que plantea este modelo, puede que no sea muy importante en todas las aplicaciones; sin embargo, la orientación a procesamiento distribuido y el rápido cambio en el hardware es probable que incremente su importancia.

Uno de los factores que se especifican en este eje es la portabilidad, que se puede definir como el esfuerzo necesario para transferir el programa de un entorno de sistema de hardware y/o software a otro. La 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*. [11]

Según McCall la portabilidad se puede medir gracias a dos indicadores principales el esfuerzo para portar y el esfuerzo para implementar.

Modelo de Boehm.

El segundo modelo de calidad más conocido es el presentado por Barry Boehm en 1978. Es un modelo fijo sin posibilidad de ser modificado o adaptado por el usuario, este modelo añade características de alto nivel (Usos Primarios), nivel intermedio (Constructores intermedios) y características primitivas (Constructores primitivos), cada una de las cuales contribuye al nivel general de calidad.

Las características de *alto nivel* representan requerimientos generales de uso que pueden ser:

- Utilidad per-se cuán usable, confiable, eficiente es el producto en sí mismo.
- Mantenibilidad cuán fácil es modificarlos, entenderlos y re-testearlos.
- Utilidad general si puede seguir usándose si se cambia el ambiente.

Las características de *nivel intermedio* representan los factores del modelo de calidad de Boehm. El *nivel más bajo* corresponde a características directamente asociadas a una o dos métricas de calidad.

El modelo de Boehm constituye uno de los más bien definidos entre los modelos de calidad. Este modelo es de naturaleza jerárquica y los criterios de calidad se presentan en tres grandes grupos. El primer grupo está integrado por los servicios que el sistema ofrece. El segundo se compone de acuerdo a la operación del producto y el último se hace de acuerdo a la mantenibilidad del producto software. En la siguiente figura se hace referencia a la estructura planteada por dicho modelo:

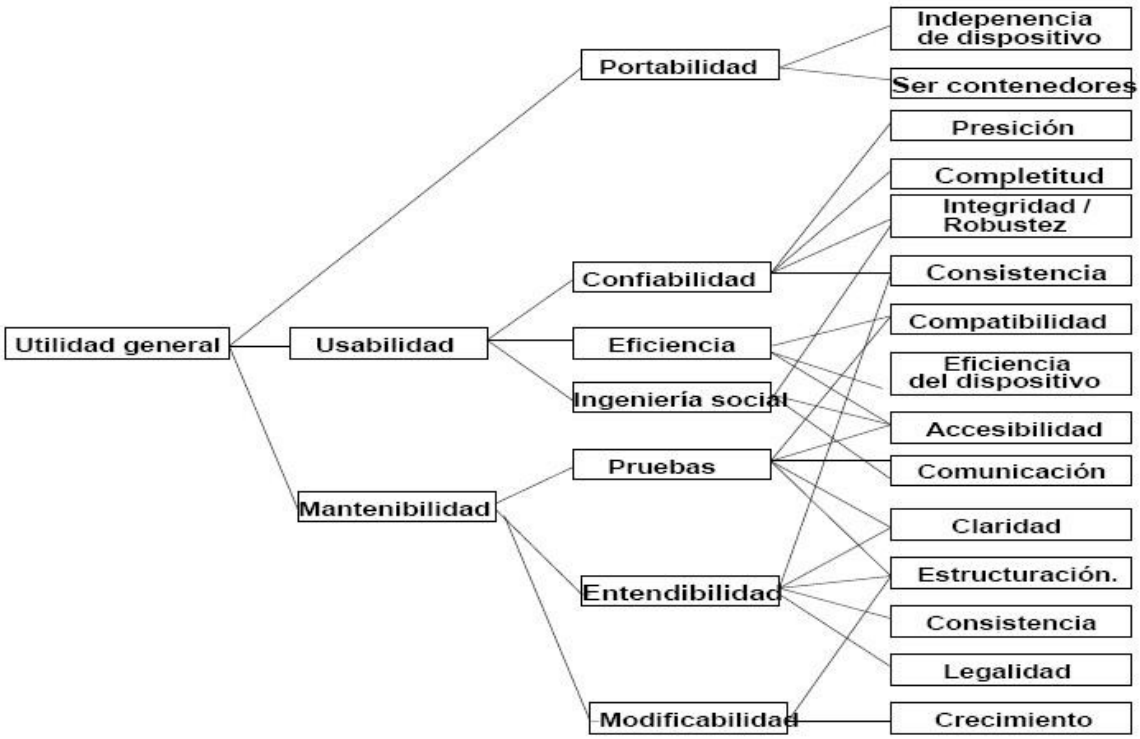

Fig. 3 Estructura planteada por el modelo de Boehm.

Norma ISO/IEC 9126.

Durante muchos años se buscó en la Ingeniería de Software un modelo único para expresar la calidad, la ventaja era clara, poder comparar productos entre sí era el principal logro en la calidad del software. En 1992, una variante del modelo de McCall fue propuesta como estándar internacional para medición de calidad de software la norma ISO/IEC 9126 Software Product Evaluation: Quality Characteristics and Guidelines for their Use, es el nombre formal con el que se le conoce.

La norma ISO/IEC 9126, investiga la dificultad existente para comparar y entender la calidad del software por parte de los usuarios-clientes. Aún a pesar de los significativos intentos por definir este concepto y conseguir valorarlo, no existe una calidad perfecta o absoluta, existe solamente una calidad necesaria y suficiente para un contexto dado.
La norma ISO/IEC 9126 posee cuatro partes:

- Parte 1: Modelo de la calidad.
- Parte 2: Métricas internas.
- Parte 3: Métricas externas.
- Parte 4: Calidad en las métricas de uso.

Solo la primera parte, *modelo de calidad*, es un estándar aprobado y publicado siendo el resto de la norma, informes que se encuentran en la fase denominada Technical Report (TR).
En la siguiente tabla se muestra un conjunto estructurado de características y subcaracterísticas de la primera parte del estándar, la norma ISO/IEC 9126-1:

Características de calidad	Subcaracterísticas de calidad	Descripción
Funcionalidad: Es la capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas e	Adecuación.	Atributos que determinan si el conjunto de funciones son apropiadas para las tareas especificadas.
	Exactitud.	Atributos que determinan que los efectos sean los correctos o los esperados.
	Seguridad.	Atributos que miden la habilidad para prevenir accesos no

implícitas cuando el software se usa bajo las condiciones especificadas.		autorizados, ya sea accidental o deliberado, tanto a programas como a datos.
	Interoperabilidad.	Atributos que miden la habilidad de interactuar con sistemas especificados.
	Conformidad con la Funcionalidad.	Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones o regulaciones legales.
Confiabilidad: La capacidad del producto software para mantener un nivel de ejecución especificado cuando se usa bajo las condiciones especificadas.	Madurez.	Atributos que se relacionan con la frecuencia de fallas por defectos en el software.
	Tolerancia a las fallas.	Atributos que miden la habilidad de mantener el nivel especificado de performance en caso de fallas del software.
	Recuperación.	Atributos que miden la capacidad de restablecer el nivel de performance y recuperar datos en caso de falla, y el tiempo y esfuerzo necesario para ello.
	Conformidad con la Confiabilidad.	Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones o regulaciones legales.
Usabilidad:	Entendimiento.	Atributos que miden el esfuerzo del usuario en reconocer el concepto lógico del software y su aplicabilidad.

Capacidad del producto software de ser comprendido, aprendido, utilizado y de ser atractivo para el usuario, cuando se utilice bajo las condiciones especificadas.	Aprendizaje.	Atributos que miden el esfuerzo del usuario en aprender la aplicación (control, operación, entrada, salida).
	Operabilidad.	Atributos que miden el esfuerzo del usuario en operar y controlar el sistema.
	Conformidad con la usabilidad.	Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones o regulaciones legales.
Eficiencia: Capacidad del producto software para proporcionar una ejecución o desempeño apropiado, en relación con la cantidad de recursos utilizados bajo condiciones establecidas.	En tiempo.	Atributos que miden la respuesta y tiempos de procesamiento de las funciones.
	En recursos.	Atributos que miden la cantidad de recursos usados y la duración de tal uso en la ejecución de las funciones.
	Conformidad con la eficiencia.	Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones o regulaciones legales.
Mantenibilidad: Capacidad del producto software de ser modificado. Las modificaciones pueden	Analizabilidad.	Atributos que miden el esfuerzo necesario para el diagnóstico de deficiencias o causas de fallas, o para identificación de las partes que deben ser modificadas.
	Facilidad para el cambio.	Atributos que miden el esfuerzo necesario para realizar modificaciones, remoción de

incluir las correcciones, mejoras o adaptaciones del software a cambios en el ambiente, así como en los requisitos y las especificaciones funcionales.		fallas o cambios en el contexto.
	Estabilidad.	Atributos que se relacionan con el riesgo de efectos no esperados en las modificaciones.
	Testeabilidad.	Atributos que miden el esfuerzo necesario para validar el software modificado.
	Conformidad con la mantenibilidad.	Atributos que hacen que el software adhiera a estándares relacionados con la aplicación, y convenciones o regulaciones legales.
Portabilidad: Capacidad de producto software de ser transferido de un ambiente a otro.	Adaptabilidad.	Capacidad del producto software de ser adaptado a los ambientes especificados sin aplicar acciones o medios de otra manera que aquellos suministrados con el propósito de que el software cumpla sus fines.
	Instalabilidad.	Capacidad del producto software de ser instalado en un ambiente especificado.
	Coexistencia.	Capacidad del producto software de coexistir con otro software independiente en un ambiente común y compartir los recursos comunes.
	Reemplazabilidad.	Capacidad del producto software de ser usado en lugar de otro producto software especificado para los mismos fines y en el

		mismo ambiente
	Conformidad con la Portabilidad.	Capacidad del producto software de adherirse a las normas o convenciones relativas a la portabilidad

Tabla 2: Calidad del Software (interna y externa).

La norma ISO/IEC 9126 provee 3 conjuntos de métricas, para medir respectivamente las características externas (ISO/IEC 9126-2), las internas (ISO/IEC 9126-3), y las de uso (ISO/IEC 9126-4), las cuales se definen a continuación:

Las **Métricas Internas** permiten atender los aspectos de la calidad desde las etapas más tempranas antes de que el producto software devenga ejecutable.[6]

Las **Métricas Externas** permiten evaluar la calidad del producto software durante el ensayo o la operación.[6]

Las **Métricas de Calidad en el Uso** miden hasta qué punto un producto satisface las necesidades de usuarios específicos para lograr las metas especificadas con la eficacia, productividad, seguridad y satisfacción en un contexto determinado de uso.[6]

Las métricas internas son utilizadas internamente para que el equipo de desarrollo evalúe la calidad de su producto mientras que las métricas externas son utilizadas por un evaluador externo que certifica la calidad del producto antes de ser entregado al cliente. Idealmente, la calidad interna determina la calidad externa y esta a su vez la calidad en el uso.

Durante el desarrollo de esta investigación es de importancia apoyarse en lo planteado por los modelos de calidad con respecto a la característica de Portabilidad; se tomó como norma base del estudio la norma ISO/IEC 9126, por ser en ella donde más ampliamente se abarca el tema. La norma ISO/IEC 9126 fue publicada con el objeto de promover un entorno que permita la evaluación de la calidad del software, además se introducen por primera vez los conceptos de calidad interna y calidad externa.

1.2. Sistemas de Apoyo a la Toma de Decisiones.

No es una exageración decir que el mundo está cruzando rápidamente de una sociedad basada en la industria a una que se centra en la información. El aumento de las capacidades de la computadora y el desarrollo de la inteligencia artificial, seguramente favorecerán la introducción de ajustes importantes en la forma en que trabajan los administradores y en que actúan las organizaciones.

La aplicación de la tecnología de la computación para la administración de la información y sistemas de apoyo de decisiones ciertamente ha tenido un efecto en la manera en que los administradores desempeñan sus labores. La capacidad de tomar decisiones debe encontrar su máxima expresión en la capacidad de solucionar problemas. Una decisión no es tal mientras no se enuncie en una acción. Todo el proceso de solución de problemas es un ejercicio de toma de decisiones.

Un DSS[5], en términos muy generales, es *un sistema basado en computador que ayuda en el proceso de toma de decisiones*.[12]

En términos ya más específicos, es *un sistema de información basado en un computador interactivo, flexible y adaptable, especialmente desarrollado para apoyar la solución de un problema de gestión no estructurado para mejorar la toma de decisiones. Utiliza datos, proporciona una interfaz amigable y permite la toma de decisiones en el propio análisis de la situación.[13]*

Los sistemas de apoyo a la toma de decisiones, ayudan a las empresas con su objetivo de crear una ventaja competitiva, debido a que *son sistemas basados en computadoras que brindan a los usuarios la información necesaria para la toma de decisiones*.[14]

A través de estos sistemas, el usuario puede responder de una manera más específica en la toma de decisiones, ya que le brinda una panorámica sobre el problema y una serie de alternativas propias para la resolución del mismo. Así mismo, los beneficios que se pueden obtener de los mismos son: calidad de decisión, mejora de comunicación, reducción de costos e incrementos en la productividad.

[5] *Decision support system (en Español: Sistema de Soporte de Decisiones).*

1.2.1. ¿Cómo tomar una Decisión?

La única manera de tomar una buena decisión es a través de la aplicación de un buen procedimiento, o modelo de toma de decisiones, el cual ahorrará tiempo, esfuerzo y energía. Existen seis criterios fundamentales para tomar una decisión eficaz estos son reconocidos como:

- Concentrarse en lo que realmente es importante.
- Realizar el proceso de forma lógica y coherente.
- Considerar tanto los elementos objetivos como los subjetivos y utilizar una estructura de pensamiento analítica e intuitiva.
- Recoger la información necesaria para optar o elegir.
- Recopilar las informaciones y opiniones, que se han formado en torno a la elección realizada.
- Ser directos y flexibles antes, durante y después del proceso.

1.2.2. Tipos de Sistemas de Apoyo a la Toma de Decisiones.

Orientado a modelos: Sistemas independientes aislados de los principales sistemas de información organizacionales que utilizan algún tipo de modelos para realizar análisis “que pasaría si”, y de otros tipos.

Orientados a datos: Analizan las grandes concentraciones de datos que se encuentran en los principales sistemas organizacionales. Apoyan la toma de decisiones permitiendo a los usuarios extraer información útil oculta previamente en enormes bases de datos.

1.2.3. Características de los DSS.

Existen ciertas características y capacidades que deben tener estos sistemas para ayudar a los usuarios en el proceso de toma de decisión, estas particularidades hacen de los DSS, una herramienta tecnológica que ayuda considerablemente a la organización. Estas características como las menciona Turban son:

- Proporciona soporte para tomadores de decisiones principalmente en situaciones semiestructuradas y no estructuradas.
- Provee soporte para diferentes niveles administrativos.
- Se puede aplicar para grupos e individuos.
- Suministra apoyo para decisiones interdependiente o secuenciales.

- Soporta todas las fases del proceso de toma de decisión.
- Tolera una variedad de procesos y estilos de decisión.
- Son adaptables sobre el tiempo.
- Fácil uso para interactuar.
- Son efectivos y no eficientes.
- El DSS ayuda pero no reemplaza al humano.
- Fácil de construir por usuarios finales.
- Utiliza modelos y análisis.
- Facilita el acceso a datos.

1.2.4. Componentes de un DSS.

- **TPS:** Sistemas de procesamiento de transacciones.
- **Bases de datos del DSS:** Conjunto de datos de varias aplicaciones o grupos. Puede ser una pequeña base de datos en una PC, o un enorme almacén de datos.
- **Sistema de Software del DSS**: Conjunto de herramientas del software que se usan para el análisis de datos, como las herramientas de OLAP, las herramientas de la extracción de datos o un conjunto de modelos matemáticos y analíticos.
- **Modelo**: Representación abstracta que ilustra los componentes o las relaciones de un fenómeno.
- **Herramientas OLAP**: Procesamiento analítico en línea.
- **Interfaz de usuario de los DSS**: Permite la fácil interacción entre los usuarios del sistema y las herramientas de software de los DSS. Una interfaz grafica de usuario, fácil de usar y flexible apoya el dialogo entre el usuario y los DSS.

1.3. Sistema Generador de Reporte Dinámico.

Los generadores de reportes son herramientas complementarias de los sistemas de formación. Utilizan una especie de lenguaje transparente para el usuario por medio del cual este realiza consultas a la base de datos y obtiene información de ella en forma de reporte. A diferencia de las consultas tradicionales, con los generadores dinámicos se tiene mayor control sobre el aspecto que tendrá la salida de la información, es decir, el usuario puede definir el aspecto que desea que tenga el reporte a generar.[15]

Son programas para crear informes sobre un diseño con una amplia variedad de formatos, que no son creados por un sistema de información. Extraen datos de los archivos o de las bases de datos y crean reportes de acuerdo con los formatos especificados, proporcionan más control, ya que pueden manejar datos de cálculos y lógica compleja antes de darles la salida.

1.3.1 Características.

- Imprime toda la tabla o los registros seleccionados.
- Permite determinar las columnas a imprimir.
- Permite establecer los encabezados y pie de página al estilo MS-Excel
- Permite modificar completamente la apariencia del reporte.
- Permite imprimir reportes anchos a través de múltiples páginas horizontales.

1.3.2. Diferentes Generadores Dinámicos.

Agata Report es un generador de reportes multi-plataforma, una herramienta de asesoramiento y creación de gráficos que se enlaza a diferentes bases de datos, como PostgreSQL, MySQL, Oracle, InterBase, Sybase, o Frontbase y permite exportar los reportes en formatos como PostScript, plain text, HTML, XML, PDF o CSV. Permite definir niveles de datos, subtotales y totales, además se pueden establecer una gran variedad de documentos como por ejemplo cartas.

FastReport es un componente complementario que le da a las aplicaciones la capacidad de generar reportes rápida y eficientemente. El mismo brinda todas las herramientas que se necesitan para desarrollar reportes de forma dinámica. Todas las variantes de este contienen un diseñador visual de reportes con reglas, guías y ampliación de asistentes para reportes tipo base, filtros de exportación a html, tiff, bmp, jpg, xls, pdf, admisión de reportes tipo matriz de puntos y admisión de los motores de bases de datos más populares. Está diseñado para ser utilizado solo en plataforma Windows.

Crystal Report es una aplicación de inteligencia empleada para diseñar y generar informes desde un amplio repertorio de fuente. Es el generador de reporte por excelencia de Visual Basic. Es un producto creado pensando en el usuario final, es decir, cualquier persona pueda crear sus propios informes sin necesidad de la asistencia de un desarrollador.

Los usuarios al instalar Crystal Reports en un equipo y utilizarlo para seleccionar filas y columnas específicas de una tabla de datos compatibles, pueden organizar los datos en el informe en el formato que deseen. Una vez que el diseño está completo, el informe se puede guardar/salvar como un archivo

con extensión rpt. Ese archivo especifica el modo en que se hará la presentación del reporte, pero no incluye los datos propiamente dichos, estos se precisan en tiempo de ejecución.

Crystal Reports se basa en un concepto muy común, los reportes se componen de "secciones", y cada sección es un espacio horizontal en la página.

Smart Report Maker es un generador de reportes PHP/MySQL de fácil uso, que le permite establecer y controlar un número indefinido de Reportes MySQL apoyados en tablas o consultas, cuenta con una interfaz amigable que permite destacar solamente los campos que se desean que estén presentes en el reporte, configura múltiples niveles de agrupamiento, ordena los datos de forma ascendente o descendente y tiene la capacidad de exportar los reportes a archivos en formatos PDF, XML o CSV.

Los reportes generados serán actualizados automáticamente, siempre y cuando se agreguen nuevos registros a su base de datos. La apariencia de los reportes generados es totalmente personalizable, de forma tal que el usuario puede elegir el diseño y el estilo de hoja de su preferencia.

Dynamic Report es una avanzada aplicación que provee una función de interfaz para su aplicación. Es un programa que le permite generar consultas a la base de datos en formato HTML y diseñar su propia página de reporte. Con el mismo, los usuarios pueden construir de una manera sencilla una aplicación interactiva o sitio web y mejorar su comunicación comercial. Utilizar Dynamic Report es sencillo, ya que no es necesario complicarse con páginas de código para controlar el diseño, simplemente se ingresan sus preferencias en los campos del formulario del programa, y el generará el código por usted.

A través de las distintas funciones y procedimientos que trae el Dynamic Report podrás preseleccionar el diseño que deseas para el informe, así como la base de datos a la que deseas hacer referencia. Todo el código fuente se genera automáticamente, lo que supondrá un gran ahorro de tiempo y esfuerzo para el programador. Trabaja con aplicaciones como Visual Basic, Visual Basic C++, Delphi, y Borland C++. Se instala automáticamente en los sistemas operativos WIndows 98 y XP.

Jasper Reports es una poderosa y flexible solución de código abierto para la generación y gestión de informes. El principal objetivo de este es facilitar la construcción de documentos con contenido dinámico y su visualización en diferentes formatos (PDF, HTML, RTF, XLS, CSV y XML). Cualquier salida de los informes está debidamente paginada y es posible navegar entre ellas. También existe la posibilidad de exportar los informes a una hoja de cálculo Excel o a un documento Word.

Es una aplicación web en Java que no requiere de grandes esfuerzos para ponerla en marcha. La única configuración que es necesario tener en cuenta, es la que conecte al módulo con la base de datos que se está utilizando.

1.4. Base de Datos.

A lo largo del desarrollo de la tecnología algunos autores han expuesto su criterio sobre este término, algunas de estas definiciones quedan expuestas a continuación:

Las Base de datos son aplicaciones informáticas para manejar información. La mayoría de las bases de datos actuales permiten hacer listados, consultas, crear pantallas de visualización de datos, controlar el acceso de los usuarios, etc. También es cada vez más frecuente que las consultas se puedan hacer en un lenguaje estándar conocido como Structured Query Language (SQL)[6]. [16]

Una base de datos es el lugar donde se guardan los datos en reposo y al cual acceden las diferentes aplicaciones (sistemas o programas) de una organización dada. [17]

Desde el punto de vista de la informática*, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos*. [18]

Concluyendo las bases de datos son una fuente para guardar elevadas cantidades de información se pueden utilizar para realizar muchas actividades como por ejemplo: escoger los datos más importantes a la hora de resolver un problema, estudiar y relacionar los datos, y así obtener conclusiones a partir de dicho estudio. Se puede decir que una base de datos es una colección de datos interrelacionados.

La función fundamental de una base de datos es almacenar y recuperar la información necesaria, para que las personas puedan tomar decisiones. Es por esta razón que las bases de datos son esenciales para la supervivencia de cualquier organización; pues los datos organizados son un recurso básico indispensable. Según sea la capacidad de almacenamiento y procesamiento del hardware, se puede contar con una única base de datos, o con múltiples.

[6] Lenguaje Estructurado de Consultas.

1.4.1. Clasificación de las Bases de Datos.

Las bases de datos pueden clasificarse de varias formas, de acuerdo al criterio escogido para su clasificación: [19]

Según la forma en que cambia la información almacenada:

Bases de datos estáticas: Son aquellas en las que la información que está guardada no puede ser modificada, es decir, son de sólo lectura. Solo se pueden consultar, son utilizadas principalmente para guardar datos históricos con los que posteriormente se podrán realizar estudios sobre su comportamiento en el transcurso del tiempo, lo cual ayudará a la toma de decisiones.

Bases de datos dinámicas: Son aquellas donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta.

Según la información que almacenan:

Bases de datos bibliográficas: Almacenan la información necesaria para poder localizar la fuente primaria. Un registro típico de una base de datos bibliográfica contiene información sobre el título, fecha de publicación, autor, editorial, de una determinada publicación, etc. Puede contener un resumen o extracto, un fragmento de la publicación original, pero nunca el texto completo.

Bases de datos de texto completo: Almacenan las fuentes primarias, es decir, guardan toda la información completa, no fragmentos, ni referencias. Por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

Directorios: Un ejemplo son las guías telefónicas en formato electrónico. Este es el caso de directorios digitales que edita la empresa de ETECSA[7] con los números telefónicos de sus usuarios, donde se puede encontrar el nombre al que pertenece dicho número y la dirección dónde se encuentra.

Banco: Son bases de datos que almacenan diferentes tipos de información estos pueden ser audio, video, imágenes, multimedia, etc.

[7] Empresa de Telecomunicaciones de Cuba.

1.4.2. Tipos de Bases de Datos.

Además de las clasificaciones antes planteadas, las bases de datos también se pueden clasificar de acuerdo al modelo de administración de datos, algunos de los más frecuentemente utilizados son:

Bases de datos Orientada a Objetos.

Incorporan el paradigma de la Orientación a Objetos (OO) a las bases de datos. La base de datos está compuesta por objetos, que pueden ser de muy disímiles tipos, y sobre los cuales se hallan definidas operaciones. Las bases de datos orientadas a objetos pueden manipular información binaria (como objetos multimedia) de una forma eficiente. Su principal limitación se encuentra en su especialización, ya que suelen estar diseñadas para un tipo específico de objetos.

Hay varias maneras de buscar los objetos de la base de datos. Una de las vías consiste en dar nombres a los objetos, igual que se hace con los archivos. Este enfoque resulta cómodo con un número de objetos pequeños, pero no resulta fácil para millones de ellos. Una segunda vía radica en exponer los punteros persistentes de los objetos, que pueden guardarse de manera externa. A diferencia de los nombres, estos punteros no tienen por qué ser fáciles de recordar y pueden ser incluso punteros físicos dentro de la base de datos. La tercera forma es guardar las listas de objetos y permitir que los programas iteren sobre las mismas para hallar los objetos deseados.

Bases de datos Jerárquica.

Como su nombre lo indica, almacenan su información en una estructura jerárquica. En este modelo la información se organiza en una forma similar a un árbol, donde un nodo padre puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas.

Este modelo es muy útil en el caso de aplicaciones que manejan un gran volumen de información, permitiendo crear estructuras estables y de gran rendimiento. Una de las limitaciones de este modelo es que es incapaz de representar eficazmente la redundancia de datos.

Bases de datos de Red.

Este modelo representa los datos mediante colecciones de registros y sus relaciones se representan por medio de líneas. El Modelo de Datos de Red es levemente diferente al modelo jerárquico; en este modelo se permite que un hijo tenga varios padres, evento no aprobado en el modelo jerárquico.

Fue una gran mejora con respecto al modelo anterior, ya que ofrece una solución al problema de redundancia de la información; pero aún así, la dificultad que implica administrar tantos datos ha significado que sea un modelo más fácil de utilizar por programadores que por los usuarios finales.

Bases de datos Multidimensionales.

Una base de datos multidimensional, es aquella que almacena sus datos con varias dimensiones, es decir que en vez de un valor, se encuentran varios dependiendo de los "ejes" definidos.
Las bases de datos multidimensionales se utilizan principalmente para crear aplicaciones On-Line Analytical Processing (OLAP)[8] y pueden verse como bases de datos de una sola tabla, su peculiaridad es que por cada dimensión tienen un campo (o columna), y otro campo por cada métrica o hecho, es decir estas tablas almacenan registros cuyos campos son de la forma: (d_1,d_2,d_3,...,f_1,f_2,f_3,...).

Donde los campos 'd_i' hacen referencia a las dimensiones de la tabla, y los campos 'f_i' a las métricas o hechos que se quiere almacenar, estudiar o analizar.

Bases de datos Relacional.

Es la base de datos más empleada en la actualidad para modelar problemas reales y administrar datos dinámicamente. En este modelo, la forma de almacenar los datos no es de mucha importancia, esto proporciona la ventaja de que es más fácil de entender y de utilizar para un usuario cualquiera. La información puede ser recuperada o almacenada mediante consultas que hacen más cómodo el poder administrar la información. Debido a que los datos consultados en cada operación se encuentran en forma de tabla, existe la posibilidad de concatenar operadores unos con otros. Este modelo además está capacitado para hacer procesamiento de conjunto.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es el Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. La base de datos Relacional posee dos componentes básicos, las tablas y las relaciones entre ellas.

Tabla: Organiza la información en filas y columnas las que se relacionan entre sí para hacer que el acceso sea más fácil. Las filas son conocidas como registros, los que son dispositivos de almacenamiento dentro de una tabla. Las columnas son llamadas campos, son los elementos que no

[8] Procesamiento analítico en línea.

se pueden dividir en el contenido de un registro. Es posible que la información de los registros se repita, por esta razón es necesario adicionar una clave, conocida como campo clave, dicha clave identificará a cada registro como único.

Relación: Conjunto de tuplas bien definidas con una cantidad de atributos fijos donde cada valor asignado a cada atributo de cada tupla pertenece al dominio de valores para dicho atributo. Las relaciones pueden ser de 1 a 1, de 1 a mucho y de mucho a mucho.

El número de campos (*columnas*) que se pueden tener en una base de datos varía según las necesidades en cuanto a gestión de datos, de forma tal que después se pueda explotar la información de forma más ordenada y segura.

Características de la base de datos Relacional.

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoria.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.
- No pueden existir dos tablas con el mismo nombre.
- Cada tabla es un conjunto de registros, filas o tuplas.

Ventajas y Desventajas.

Ventajas

- Provee herramientas que garantizan evitar la duplicidad de registros.
- Garantiza la integridad, así al eliminar un registro elimina todos los registros relacionados dependientes.
- Favorece la normalización por ser más comprensible y aplicable.
- Control sobre la redundancia de datos.
- Independencia de los datos y los programas y procesos.
- Mayor integridad de los datos.

- Mayor seguridad en los datos. Al limitar el acceso a ciertos usuarios.
- Datos más documentados.
- Acceso a los datos más eficiente.

Desventajas

- Presentan deficiencias con datos gráficos, multimedia y sistemas de información geográfica.
- El control y administración requiere de un software y un hardware más poderoso.
- Requiere personal calificado.
- Implantación larga y difícil.
- Pueden ser de instalación compleja, usando muchas tablas.
- Es difícil entender cómo se relaciona una parte con la otra.

1.5. Sistemas gestores de Bases de datos.

Con el transcurso del tiempo y la evolución de las tecnologías el trabajo con la información ha dado un giro total en los sistemas de archivos, puesto que se limitaba a la estructuración del almacenamiento físico de los datos. Los primeros sistemas manejadores de archivos se conocen como ISAM[9] y VSAM[10] estos usaban la técnica de archivos separados, convirtiéndose en los primeros sistemas de base de datos. Desde los pasados treinta años y con la actualización de estas tecnologías los sistemas manejadores de archivos fueron integrados en una única colección de los datos y a partir de ese momento la manipulación de éstos fue llevada a cabo por los sistemas gestores de bases de datos.

Algunos autores han expuesto su criterio sobre este término, a continuación se presentan algunas definiciones del tema:

El sistema de gestión de la base de datos es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma. [21]

Los SGBD son a*plicaciones o Suite cuya finalidad es la de controlar la entrada y salida de datos de una base de datos, manteniendo la integridad de la misma*. [22]

[9] Método de Acceso Secuencial Indexado.

[10] Método de Acceso Virtual de Almacenamiento.

Conjunto coordinado de programas, procedimientos, lenguajes, etc. que suministra a los distintos tipos de usuarios los medios necesarios para describir y manipular los datos almacenados en la base de datos, garantizando su integridad, confidencialidad y disponibilidad.[23]

Un SGBD consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. El objetivo primordial de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente. [24]

Con lo anterior expuesto se define que los SGBD permiten el buen manejo de los datos, posibilitando que toda información necesaria y relevante para cualquier empresa sea manejada de una forma correcta, ordenada y lista para que cualquier usuario pueda acceder a la misma.

Los SGDB están compuestos por tres lenguajes:[25]

I. **Data Definition Language (DDL):** Lenguaje de definición de datos. Por medio de este, el DBMS[11] identifica las descripciones de los elementos de los esquemas y almacena la descripción del esquema en el catálogo del DBMS, además especifica el esquema conceptual e interno (Base de datos Almacenada).

II. **Data Manipulation Language (DML):** Lenguaje de manipulación de datos. Permite la manipulación de las operaciones de inserción, eliminación y modificación.

III. **Structured Query Language (SQL):** Lenguaje de consulta estructurado. Permite especificar todas las operaciones sobre la base de datos como por ejemplo: Inserción, Delete, Actualización. Utiliza características del álgebra y el cálculo relacional permitiendo de esta forma realizar consultas a la base de datos de forma sencilla.

1.5.1. Funcionalidades de un SGBD.

Algunas de las principales funcionalidades que presentan los sistemas gestores de bases de datos se mencionan seguidamente:

Abstracción de la información. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

[11] Sistemas manejadores de base de datos.

Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación. Proporcionan una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

1.5.2. Clasificación de los SGBD.

Criterio	Clasificación de los SGBD
Según modelo lógico en el que se basan.	- Modelo Relacional. - Modelo Redes. - Modelo Jerárquico. - Modelo Orientado a Objetos.
Según el número de usuarios.	- Monousuario. - Multiusuario.
Según el número de sitios.	- Centralizado. - Distribuido.
Según el uso de Software.	- Homogéneo. - Heterogéneo.
En cuanto al propósito.	- De propósito general. - De propósito específico.

Tabla 3: Clasificación de los SGBD.

1.5.3. Características de los SGBD.

A través del desarrollo de los diferentes lenguajes de programación se han desarrollados diferentes Sistema Gestores de Base de Datos. La elección de ellos por cualquier empresa no se debe realizar de manera aleatoria, hay que tener en cuenta que con esto se va a llevar a cabo una gran inversión, tanto de software como de hardware además de que hay que tener las condiciones necesarias para la interacción con el propio gestor.

Un SGBD crea un entorno operativo que depende directamente de sus características, y en la mayoría de los casos, se convierte en el centro del entramado informático de la empresa.

Generalmente las compañías capitalistas por su amplio nivel adquisitivo usan en sus empresas los SGBD propietarios ya que estos ofrecen grandes prestaciones, entre los que se pueden mencionar: Oracle, SQL Server.

Oracle: *Oracle es un sistema de administración de base de datos o RDBMS[12], fabricado por Oracle Corporation, básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Oracle es un sistema gestor de base de datos robusto, tiene muchas características que garantiza la seguridad e integridad de los datos; que las transacciones se ejecuten de forma correcta, sin causar inconsistencias; ayuda a administrar y almacenar grandes volúmenes de datos; estabilidad, escalabilidad y es multiplataforma.* [25]

SQL Server: *Es un buen producto, probablemente de los mejores acabados por Microsoft, basado en el lenguaje Transact-SQL. Su instalación es sencillísima, su interfaz es clara e intuitiva y viene acompañado de una suite de utilidades bastante completa. La herramienta de administración de Microsoft SQL Server (Microsoft SQL Server Enterprise Manager) muestra la habitual disposición de este tipo de aplicativos de Microsoft, permite acceder fácilmente a cualquier objeto de la base de datos, detener y re-iniciar el servicio.* [26]

Las amplias ventajas que ofrece el trabajo con el software libre, ha llevado a nuestro país a capacitar el personal y a crear instituciones que investiguen y desarrollen programas de código abierto, uno de los SGBD con los que nuestra Universidad se adentra y con los cuales se ha tenido buenos resultados han sido MySQL y PostgreSQL.

MySQL: Es el sistema de gestión de bases de datos SQL Open Source más popular, lo desarrolla, distribuye y soporta MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Hoy en día es desarrollado por la empresa Sun Mycrosystems.

El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido. MySQL es una marca registrada de MySQL AB. Posee una doble licencia, los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU o pueden adquirir una licencia comercial estándar de MySQL AB. [27]

PostgreSQL: *Es un servidor de base de datos relacional orientada a objetos de software libre, liberado bajo la licencia BSD[13]. El desarrollo de PostgreSQL no es manejado por una sola compañía sino que*

[12] Relational Data Base Management System.

[13] Berkeley Software Distribution.

es dirigido por una comunidad de desarrolladores y organizaciones comerciales, la que se denominada como PGDG[14]. PostgreSQL está considerado como la más avanzada base de datos de código abierto en el mundo, tiene una gran cantidad de características que, normalmente, sólo se encuentran en las bases de datos comerciales. [28]

PostgreSQL resulta la alternativa más obvia para quienes deseen librarse completamente de las licencias comerciales del software propietario, y optar por una solución 100% libre y gratuita. Este gestor ofrece la posibilidad de obtener su código fuente, usar el programa y modificarlo sin los límites planteados por un software propietario. Esta posibilidad de ser una fuente abierta, permite que se pueda acceder de forma clara a la evaluación de su rendimiento y estadísticas, cosas que empresas como Oracle prohíbe. PostgreSQL brinda la alternativa de que se pueda cambiar su código y que cada cual lo acomode según sean sus necesidades.

PostgreSQL está referenciado por los lenguajes de programación más importantes, incluyendo C, Perl, Python, Tel, Java, PHP, ODBC, JDBC, entre otros. Es multiplataforma y está disponible en casi cualquier Unix, (34 plataformas en su última versión estable).

Cumple con los factores que determinan la calidad del software (ISO 9126-1) como son:

- Características operativas: Corrección, Fiabilidad, Eficiencia, Integridad, Facilidad de uso.
- Capacidad para soportar cambios: Facilidad de mantenimiento, Flexibilidad, Facilidad de prueba.
- Adaptabilidad a nuevos entornos: Portabilidad, Reusabilidad, Interoperabilidad.

En cuanto a las características de hardware que necesita PostgreSQL para su instalación en el sistema operativo Linux por ejemplo se destaca que no es muy exigente ya que los requerimientos mínimos para instalar PostgreSQL son:

- 8 megabytes de RAM.
- 30 megabytes de espacio en disco para el cogido fuente.
- 5 megabytes de espacio en disco para la instalación de los ejecutables.
- 1 megabyte extra para las bases de datos básicas.
- 3 megabytes de espacio en disco rígido para el trabajo con el cogido fuente.

[14] PostgreSQL Global Development Group.

Para poder compilar e instalar PostgreSQL se necesitan una serie de programas. La mayoría de los cuales suelen instalarse por defecto si se instalan los paquetes relacionados con "Desarrollo y compiladores". Algunos de los que se necesitan:

- GNU make (gmake).
- Un compilador ISO/ANSI C. (GCC el compilador por defecto en Linux funciona perfectamente).
- Compactadores tar, gzip o bzip2 para desempaquetar las fuentes.
- Biblioteca GNU Readline.
- Biblioteca de compresión zlib.

Seguidamente se muestra un resumen de las principales características de algunos de los gestores que se mencionan anteriormente, para mejorar la comprensión del trabajo de los mismos en la actualidad.

SGBD	Características
Server SQL	• Soporte de transacciones. • Escalabilidad, estabilidad y seguridad. • Soporta procedimientos almacenados. • Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. • Permite trabajar en modo cliente-servidor, donde la información y los datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información. • Permite administrar información de otros servidores de datos.
Oracle	Se considera a Oracle como uno de los sistemas de BD más completos, destacando su: • Soporte de transacciones. • Estabilidad. • Escalabilidad. • Soporte multiplataforma. Sus últimas versiones han sido certificadas para poder trabajar bajo Linux.
	• Sistema de gestión de base de datos relacional, multihilo y multiusuario.

MySQL	• Esquema de licenciamiento dual. • Soporte a multiplataforma. • Procedimientos almacenados. • Triggers. • Cursores. • Vistas actualizables. • Extremadamente rápido y fácil de personalizar. • Consume muy pocos recursos. • Ofrece estabilidad y facilidad de despliegue.
PostgreSQL	• Alta concurrencia. • Amplia variedad de tipos nativos. • Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys). • Disparadores (Triggers). • Vistas. • Integridad transaccional. • Herencia de tablas. • Tipos de datos y operaciones geométricas.

Tabla 4: Sistemas gestores de BD más utilizados.

En este capítulo se vieron reflejados los principales temas asociados al objeto de estudio, así como los aspectos necesarios para definir la propuesta de solución, en este caso la creación de una guía para la obtención de la adaptabilidad y la coexistencia de las soluciones de software de ayuda a la toma de decisiones basadas en las bases de datos relacionales que se basan en PostgreSQL.

Como resultado de la investigación realizada se obtuvieron los conocimientos teóricos necesarios para continuar el desarrollo de la investigación. Se identificaron las principales deficiencias de las métricas dirigidas a la evaluación de la portabilidad en las soluciones de software con lo cual se ratificó la importancia que tiene la medición y evaluación de las soluciones de software basadas en bases de datos relacionales gestionadas con PostgreSQL.

Capítulo 2: Características de la Propuesta Solución.

En el presente capítulo se expondrán las características que tendrá la propuesta solución, referenciándose las subcaracterísticas de adaptabilidad y coexistencia de las soluciones de software de apoyo a la toma de decisiones, descritas por los modelos de calidad en el capítulo anterior. Además se realizará una descripción de la situación que posee DATEC en cuanto a la portabilidad de las soluciones de software de apoyo a la toma de decisiones.

Se definen los riesgos fundamentales en el proceso de desarrollo de software, identificando los factores que favorecen la característica de portabilidad principalmente las subcaracterísticas de adaptabilidad y coexistencia. Por último se desarrollará una guía para garantizar la obtención de la adaptabilidad y la coexistencia de las soluciones de software de apoyo a la toma de decisiones durante el proceso de desarrollo del software.

2.1. Metodología utilizada en la línea de Herramientas y Soluciones Integrales.

Para la implementación de los productos software es imprescindible que se utilice una metodología según las características del proceso de desarrollo de este, por la cual debe guiarse el equipo de desarrollo para una mejor elaboración y calidad de la solución de software.

Fue necesario para la investigación hacer un estudio de la metodología utilizada por DATEC en su línea de Herramientas y Soluciones Integrales, para ubicar los factores que favorecen la adaptabilidad y la coexistencia de las soluciones de software en el proceso de desarrollo del software.

Es empleada para el desarrollo de productos software la “Metodología para el Desarrollo de Productos Basada en el Modelo de Producción de Líneas de Productos de Software”. Esta metodología define las fases, actividades y artefactos que se deben generar durante el ciclo de vida del software. Teniendo en cuenta su aplicación en un proceso de desarrollo basado en líneas de productos de software.

Está basada en la adaptación de metodologías para el desarrollo de software en equipo como son SCRUM y OpenUP. Se centra en la organización propuesta para la aplicación del Modelo de Líneas de Productos de Software en DATEC. Actualmente se aplica en la línea de Herramientas y Soluciones Integrales del Centro de Tecnologías de Gestión de Datos.

2.1.1. Ciclo de vida.

Para los proyectos del alcance del Programa de mejora se define el siguiente ciclo de vida básico:

- Estudio Preliminar.
- Modelación del Negocio. (*)

- Requisitos.
- Análisis y Diseño.
- Implementación.
- Pruebas Internas.
- Pruebas de Liberación.
- Despliegue.
- Soporte. (*)

(*) Notas:
En el ciclo de vida pueden realizarse iteraciones según las necesidades del proyecto a partir de la fase de negocio. El tiempo de soporte debe ser de tiempo limitado.

a) **Estudio Preliminar:** Se planea el proyecto en un alto nivel y la legalización del mismo. En esta fase se estudia inicialmente la organización del cliente. Permitiendo obtener información del alcance del proyecto así como realizar estimaciones de tiempo, esfuerzo y costo.

b) **Modelación del Negocio:** Comprende el proceso de negocio en una organización, se plantea que se desea automatizar para garantizar que el software desarrollado cumpla con su propósito. Para la descripción y modelado de negocio se pueden utilizar técnicas como el Modelado de Casos de Uso del Negocio y Business Process Modeling Notation (BPMN)[15]. Para las herramientas solo se realiza Modelo de Dominio.

c) **Requisitos:** Desarrolla un modelo del sistema que se desea construir. Contiene una serie de casos de usos y servicios que describen todas las interacciones envuelve además el Plan de pruebas de Aceptación.

d) **Análisis y Diseño:** Son refinados y estructurados los requisitos descritos en la fase anterior para obtener una comprensión más precisa de los mismos y una descripción que sea fácil de entender. Es modelado el sistema y la arquitectura para que soporten todos los requisitos, incluyendo los no funcionales. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la implementación. Es desarrollado el documento de la arquitectura y el diagrama de despliegue entre otros.

e) **Implementación:** Se implementa el sistema en términos de componentes es decir, ficheros de código fuente, scripts, ejecutables y similares a partir de los resultados de la actividad anterior. Además se realiza el manual de usuario.

[15] Notación del proceso del modelado del negocio.

f) **Pruebas Internas:** Se comprueban los resultados obtenidos de la implementación probando según sea necesario en cada construcción. Se deben desarrollar artefactos de prueba como: Diseños de casos de prueba, Listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas. Se incluyen además las pruebas del sistema.

g) **Pruebas de Liberación:** Son diseñadas e implementada para todos los entregables de los proyectos antes de ser mostrados al cliente para su aceptación. Son elaboradas por el Laboratorio Industrial de Pruebas de Software.

h) **Despliegue:** Se procede a la entrega de la solución, así como a la instalación, configuración, prueba y puesta en marcha del software en el ambiente real del cliente. Se incluyen pruebas de aceptación y pruebas piloto. Además se debe capacitar y acompañar al cliente durante este tiempo para garantizar que obtengan los conocimientos necesarios para interactuar con el software.

i) **Soporte:** El proyecto ofrece servicios para resolver conflictos y problemas de usabilidad y rendimiento del software, facilitándole actualizaciones y parches a errores, todo esto por un tiempo limitado luego de a ver entregado el software.

2.1.2. Roles y responsabilidades.

Roles.	Responsabilidades.
Líder de Proyecto	• Análisis de indicadores, disciplina, cumplimiento de cronogramas y costes. • Análisis de riesgos. • Visión del producto. • Gestión integral del proyecto. • Chequeo del avance de cada una de las áreas de la gestión del proyecto. • Dirigir el equipo de desarrollo.
	• Disciplina técnica del desarrollo. • Necesidades tecnológicas. • Línea base. • Integración y vista de sistema del producto. • Reutilización. • Hitos tecnológicos.

Arquitecto Principal	• Construcción tecnología del producto. • Servicios tecnológicos al desarrollo. • Participar en las discusiones y decisiones arquitectónicas tanto de sistema como tecnológico, que puedan comprometer o tengan alto impacto en la arquitectura y por ende en su desarrollo.
Arquitecto de Sistema e Integración	• Disciplina técnica del desarrollo. • Línea base. • Integración y vista de sistema del producto. • Construcción tecnología del producto. • Participar en las discusiones y decisiones arquitectónicas de sistema.
Arquitecto de Datos	• Vista de datos del sistema. • Integración y trazabilidad entre los componentes lógicos y los componentes de datos. • Mantenimiento y rendimiento de las bases de datos, replicación y salvas.
Analista	• Participa en la definición del proyecto. • Interviene en la modelación del negocio. • Interactúa con el usuario final en la definición de los requisitos de la aplicación. • Crea el modelo de casos de uso del sistema. • Define el prototipo de interfaz de usuario elemental. • Responsable de traducir la comunicación entre usuarios finales y desarrolladores. • Gestiona los requisitos adicionales que aparezcan durante el desarrollo del software.
Diseñador	• Interpreta la información resultado del análisis y traduce al lenguaje de los programadores (interfaz, negocio y acceso a datos). • Define los elementos de diseño a tener en cuenta para la implementación de los casos de uso. • Diseña la implementación sobre la arquitectura definida. • Integra los componentes de la solución y define las interfaces. • Dirige el trabajo de los programadores.
	• Usa el diagrama entidad-relación para generar el diseño físico de la base de datos. • Crea y mantiene el ambiente de la base de datos para el funcionamiento de la aplicación.

Administrador de base de datos	• Interviene en el ajuste del desempeño de la aplicación. • Ayuda a los desarrolladores de lógica de negocio a desarrollar elementos relativos al acceso a datos. • En general el administrador de base de datos desempeña otros roles como especialista en migración de datos y desarrollador de lógica de negocio.

Tabla 5: Roles y responsabilidades de la metodología usada por DATEC.

2.2. Factores en el desarrollo del software a tener en cuenta para concebir las buenas prácticas.

En el estudio del proceso de software se encontraron importantes factores que favorecen las subcaracterísticas de adaptabilidad y coexistencia mencionadas en el capítulo anterior, para garantizar una mejor portabilidad en las soluciones de software.

Se hizo necesario el desglose de estos factores teniendo en cuenta la subcaracterística que afectan, para reconocer una serie de problemas que están presentes y de esta forma elaborar una encuesta, (Ver Anexo #1) la cual muestra el estado real de dificultad que presenta la Línea de Herramientas y Soluciones Integrales de DATEC en relación con estos factores.
Los factores encontrados son:

Adaptabilidad:
Garantizar que el software sea multiplataforma.
Se presenta cuando se elabora una aplicación y esta solo puede ser instalada en una computadora que posea la misma plataforma que donde fue creada. Por ejemplo la plataforma IBM-PC, x86, IA64 y la AMD64. La dependencia de una plataforma hardware/software supone una molesta atadura para el usuario y crea ligaduras que dificultan el paso a una arquitectura diferente.

Flexibilidad en el diseño.
La rigidez es un concepto análogo a la poca flexibilidad, el sistema debe ser flexible para poder garantizar que en caso de ser necesario, se puedan realizar cambios para lograr adaptarlo al ambiente que se desea. [29]

Implementación poco compleja (viscosidad).
Los programadores encuentran muchas formas de implementar un cambio, pero esta implementación es muy compleja para lograr mantener la filosofía del diseño original. Logrando que la implementación sea sencilla se garantiza que sea más fácil de adaptar el software al ambiente. [29]

Estandarización de los datos.

El usuario puede percibir la calidad del software desde el primer momento en que se enfrenta a la aplicación. En ocasiones la compatibilidad de aplicaciones en ciertos sistemas operativos no es la misma, ya que requieren de elementos necesarios (**plug-in)** para que estos puedan operar en distintos entornos y que no varié el formato de los datos. Mostrar los datos en un navegador u otro sin que sufran cambios o distorsiones ha sido uno de los objetivos que se persigue durante el desarrollo de un software. [30]

Establecer las condiciones mínimas de hardware.

Características mínimas que debe poseer la PC para soportar la aplicación, (Memoria RAM, velocidad del micro, capacidad). Requerimientos de rendimiento y de interfaz. ¿Cuántas veces ha pasado que el servidor ha quedado muy chico o solamente tuvo la capacidad suficiente para un año? Muchas veces por no sé qué motivo se declara la capacidad al mínimo, luego existen PCs con 256 Mb de memoria, en las que el usuario tendrá que correr el sistema operativo, su correo, las noticias, la información de la compañía, nuestra aplicación y los otros programas que el usuario posea, y entonces aparece el problema de que no se tiene suficiente espacio para soportar todas las aplicaciones al mismo tiempo.

Establecer protocolo de comunicación óptimo.

Los protocolos de comunicaciones definen las normas que posibilitan que se establezca una comunicación entre varios equipos o dispositivos, ya que estos equipos pueden ser diferentes entre sí. Debido a la gran complejidad que conlleva la interconexión de ordenadores, se ha tenido que dividir todos los procesos necesarios para realizar las conexiones en diferentes niveles. Cada nivel se ha creado para dar una solución a un tipo de problema particular dentro de la conexión. Cada nivel tendrá asociado un protocolo, el cual entenderán todas las partes que formen parte de la conexión.

Establecer la Topología de Red adecuada.

Se debe tener en cuenta la topología de red que exista en el ambiente organizacional en el que será instalada la aplicación, para garantizar que al ser implantada, la aplicación se adapte al entorno con facilidad, de forma tal que no sobre cargue al servidor y se evite la caída de los servicios constantemente.

Capacitar a los usuarios finales.

Actualmente muchos usuarios sufren de desconocimiento tecnológico o como se le suele llamar “Analfabetismo Tecnológico”, esto no es más que la incapacidad para utilizar nuevas tecnologías tanto en la vida diaria como en el mundo laboral. La falta de conocimiento del usuario ha provocado problemas para adaptarse e interactuar con tecnologías de punta, la escasa capacitación es un factor importante que influye de manera decisiva en el proceso de adaptabilidad, ya que ahí es donde el usuario va a interactuar

con el software, va a aprender a usarlo, y es el equipo de desarrollo o una persona capacitada en el tema quien debe darle esta capacitación a los usuarios. [31]

Coexistencia:

Uso mínimo de recursos.

Elemento importante a tener en cuenta debido a que puede provocar fallos en el sistema y lentitud en las operaciones ya que la PC tiene que ocuparse de una gran cantidad de solicitudes diferentes, y cada una de ellas consume parte de la energía de procesamiento de la PC. Un ejemplo sería una aplicación que esté utilizando el microprocesador pero necesite usar la impresora para poder seguir y otro que esté usando la impresora pero necesite el micro para poder continuar, pues uno impide la culminación de la ejecución del otro. Esto provoca ciclos infinitos o fallos en el sistema.

2.2.1. Resultados de la encuesta.

La encuesta se le aplicó a 6 especialistas que laboran con el Generador de Reportes Dinámicos. Una vez procesada se obtuvieron diversos resultados, en dependencia de la importancia que los encuestados le otorgaron a cada elemento.

Pregunta #1: ¿Se puede presentar el caso en que el usuario no pueda instalar la solución de software porque sea incompatible con la plataforma que presenta en su computadora?

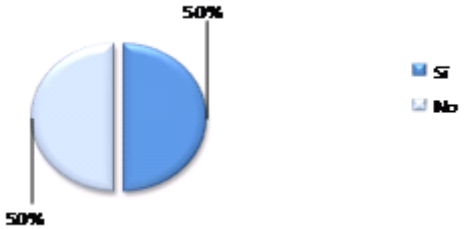

Fig. 5 Pregunta #1 de la encuesta.

Pregunta #2: ¿En el proceso de elaboración del software si no se tiene una total comprensión de lo que debe hacer el mismo, esto puede provocar un atraso en la entrega del producto?

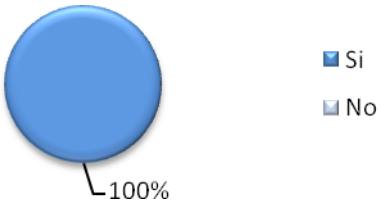

Fig. 6 Pregunta #2 de la encuesta.

Pregunta #3: La rigidez es un concepto análogo a la poca flexibilidad y muy asociado a la dependencia funcional. ¿Cambios en una parte de la solución de software pueden provocar cambios en otras partes de la misma?

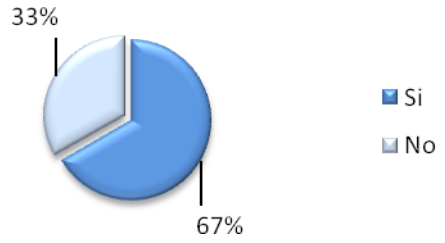

Fig. 7 Pregunta #3 de la encuesta.

Pregunta #4: ¿Se ha hecho necesario el uso de una programación compleja (Parches) para mantener la filosofía del diseño original?

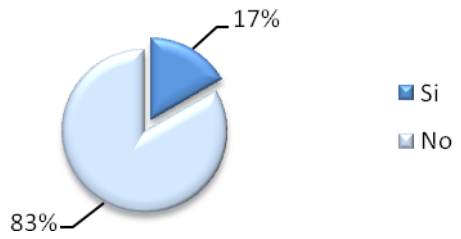

Fig. 8 Pregunta #4 de la encuesta.

Pregunta #5: ¿Cuándo se muestran los resultados obtenidos del sistema estos se presentan en ocasiones distorsionados o fuera del formato predeterminado con anterioridad?

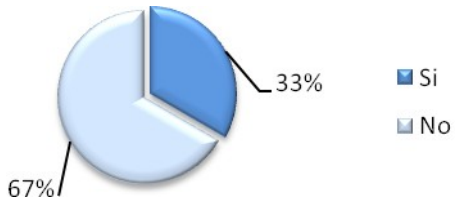

Fig. 9 Pregunta #5 de la encuesta.

Pregunta #6: ¿Es posible que pasado cierto tiempo de estar instala la solución de software, está presente problemas de capacidad en la Base de datos?

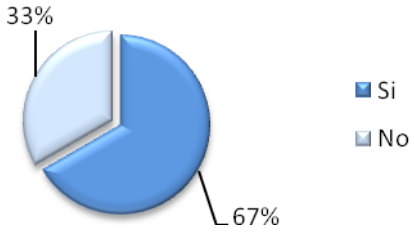

Fig. 10 Pregunta #6 de la encuesta.

Pregunta #7: ¿Es posible que si no se tiene en cuenta como será la comunicación que se va a establecer entre la solución de software y el resto las aplicaciones, esto provoque un mal funcionamiento, ya que no se podrían conectar adecuadamente?

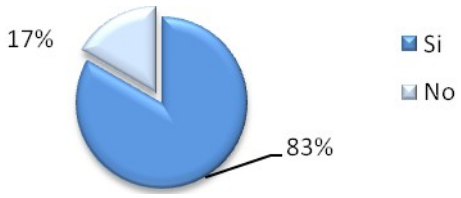

Fig. 11 Pregunta #7 de la encuesta.

Pregunta #8: ¿Es necesario conocer como esta estructurada la red en el lugar donde será implementada la solución de software?

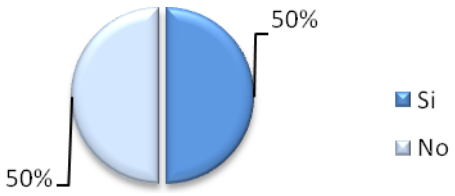

Fig. 12 Pregunta #8 de la encuesta.

Pregunta #9: ¿Es posible que al instalar la solución de software junto con otras, estas utilicen más memoria RAM de la que deberían, haciendo así el trabajo más lento para que el usuario interactué con la solución?

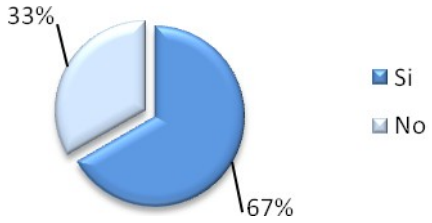

Fig. 13 Pregunta #9 de la encuesta.

Pregunta #10: ¿En caso de que dos soluciones de software sean instaladas en la misma computadora y estas utilicen un mismo recurso, esto podría provocar que al usuario interactuar con la aplicación la computadora procese más lento la información?

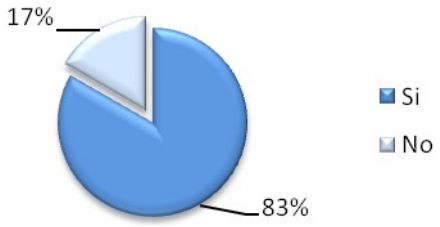

Fig. 14 Pregunta #10 de la encuesta.

Después de analizar los resultados obtenidos a través de la encuesta se llegó a la conclusión que estos factores favorecen la adaptabilidad y la coexistencia de las soluciones de software en mayor o menor grado.

2.3. Propuesta Solución.

Teniendo en cuenta todo lo establecido durante el transcurso de la presente investigación se puede establecer que se está en condiciones de crear una propuesta para darle solución al problema científico planteado para este trabajo de diploma, ¿Cómo obtener la adaptabilidad y la coexistencia de las soluciones de software de apoyo a la toma de decisiones basadas en bases de datos relacionales gestionadas con PostgreSQL?

Se propone la realización de una guía en la cual quedarán establecidas una serie de buenas prácticas que le permitirán a los grupos de desarrollo lograr implementar soluciones de software con una mayor portabilidad, ya que se estará construyendo la adaptabilidad y la coexistencia durante el proceso de desarrollo del software.

Para alcanzar diseñar la guía se hizo necesario el estudio del ciclo de vida de la Línea de Herramientas y Soluciones Integrales del proyecto DATEC, para ser más específicos el proyecto Generador de Reportes Dinámicos.

La misma está estructurada según las principales actividades de todo ciclo de desarrollo; Requisitos, Análisis y Diseño, Implementación, Prueba, Despliegue y Soporte. (Ver Anexo #2).

Concluyendo el presente capítulo, se evidencia la adquisición de conocimientos sobre la metodología usada en DATEC, especialmente en la Línea de Herramientas y Soluciones Integrales, se destacan las principales actividades así como los factores que favorecen la adaptabilidad y la coexistencia. Además fue corroborada la existencia de los mismos en el proceso de elaboración de un software mediante una encuesta aplicada a los involucrados, se determinó un conjunto de buenas prácticas para darles solución, construyéndose de esta forma una guía la cual será llevada a una posterior validación para verificar su buen uso.

Capítulo 3: Análisis de los resultados obtenidos.

Después de ser identificados los factores que favorecen la adaptabilidad y la coexistencia de las soluciones de software de apoyo a la toma de decisiones y luego de haberse realizado una cuidadosa investigación sobre las posibles buenas prácticas que puedan emplearse en este tipo de aplicaciones, se definirá como aporte a la mejora de la gestión de la calidad de las soluciones de software un método para la validación de la propuesta solución. Para valorar y aceptar la propuesta solución planteada en el objetivo general, se tomó como criterio las opiniones de un conjunto de especialistas. Se analizarán los resultados del método utilizado para confirmar que la guía cumple con los objetivos planteados.

3.1. Método Delphi.

El método Delphi es un método que se desarrolla mediante una técnica grupal donde se analizan opiniones, sigue la filosofía de que el criterio de un grupo de personas en iguales condiciones es más fiable que el de un individuo en particular, caracterizándose también por usar e investigar la opinión de expertos.

Como técnica empleada, la inteligencia colectiva es el principio por el cual se rige este método. Se logra además un grado de conformidad con las opiniones expresadas individualmente por las personas que se seleccionaron, clasificándose estos como expertos en el tema.

3.1.1. Características del método.

El método Delphi presenta entre sus características: [32]

Anonimato: No debe existir contacto entre los participantes, pero el administrador/gestor de la encuesta sí puede identificar a cada participante y sus respuestas.

Iteración: Se pueden manejar tantas rondas como sean necesarias.

Retroalimentación Controlada: Los resultados totales de la ronda previa no son entregados a los participantes, sólo una parte seleccionada de la información circula.

Resultados Estadísticos: La respuesta del grupo puede ser presentada estadísticamente (promedios y grado de dispersión).

3.2. Validación de la propuesta solución.

Para darle validez a la propuesta solución se utilizó el método Delphi el cual suele dividirse en tres etapas o fases fundamentales:

3.2.1 Fase preliminar.

Durante esta primera fase se delimita el contexto, el objetivo a alcanzar con la aplicación del método seleccionado, el diseño, los elementos básicos del trabajo y por último se produce la selección de los expertos que protagonizarán la evaluación de los resultados.

Selección de Expertos.

Se realizó como primer paso la consulta a 6 expertos y se tuvieron en cuenta varias características para su selección:

- Competencia.
- Creatividad.
- Disposición a participar en la entrevista.
- Capacidad de análisis.
- Honestidad.
- Experiencia en el trabajo.
- Dominio y conocimiento sobre el tema.

Para determinar algunas de estas características se realizó un cuestionario donde los expertos autoevalúan el nivel de información y argumentación que poseen sobre las soluciones de software de apoyo a la toma de decisiones.

Para ello primeramente se establece un valor en una escala creciente del 1 al 10, el cual corresponde con el grado de conocimiento o información que posean sobre el tema.

1	2	3	4	5	6	7	8	9	10

Fig. 15 Grado de conocimiento del experto.

Seguidamente los expertos deben realizar una autovaloración de sus niveles de argumentación o fundamentación sobre el tema, es decir grado de influencia de cada una de las fuentes siguiendo lo planteado en la siguiente tabla:

FUENTES DE ARGUMENTACIÓN	ALTO	MEDIO	BAJO
Análisis teórico realizado por usted.			
Experiencia de trabajo en esta enseñanza.			
Trabajos de autores nacionales.			
Trabajos de autores internacionales.			
Su propio conocimiento del estado del problema en el extranjero.			
Su intuición.			

Tabla 6: Grado de influencia de las fuentes de argumentación.

El resultado obtenido se interpreta en una escala que se plantea a continuación según el método Delphi:

Si 0.8<= k <= 1.0, el coeficiente de competencia es alto.

Si 0.5<= k <= 0.7, el coeficiente de competencia es medio.

Si k<= 0.4, el coeficiente de competencia es bajo.

De esta manera fue posible conocer algunas características como su disposición a participar en la encuesta, su capacidad de análisis y su competencia, luego de realizados los cálculos a cada uno de los expertos de la propuesta en cuestión, se tiene como resultado los datos que se muestran en la siguiente tabla:

Expertos	Coeficiente de conocimiento Kc.	Coeficiente de argumentación Ka.	Coeficiente de competencia K.
1	1	1	1
2	1	0.9	0.95
3	0.8	0.8	0.9
4	1	0.9	0.95
5	0.9	0.8	0.85
6	0.9	1	0.8

Tabla 7: Resultados del coeficiente de cada experto.

Como que el valor K queda entre 0.8 y 1 se puede decir que el coeficiente de competencia de los expertos seleccionados es alto, demostrando así que cada uno de ellos posee un alto nivel de capacitación en la materia tratada.

3.2.2. Fase exploratoria.

En esta fase las principales actividades son la elaboración y aplicación de los cuestionarios según sucesivas vueltas, para la realización de estas actividades se deben tener presente la longitud y el tipo de pregunta. Un planteamiento demasiado conciso provoca una excesiva variedad de interpretaciones y uno muy largo requiere administrar demasiados elementos de una sola vez, por lo que la forma en que se realiza el cuestionario es con planteamientos de mediana longitud. (Ver Anexo #3)

¿Cree usted que la propuesta es efectiva?

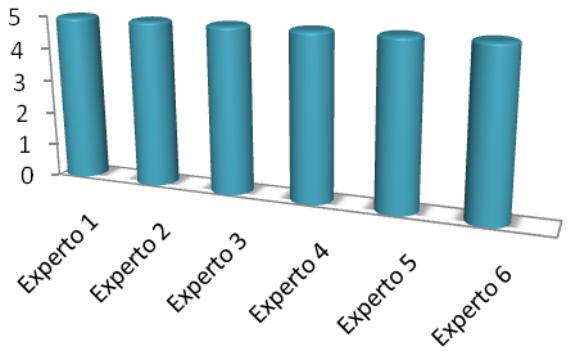

Fig. 16 Pregunta #1 de la entrevista.

Posibilidad de Aplicar los factores.

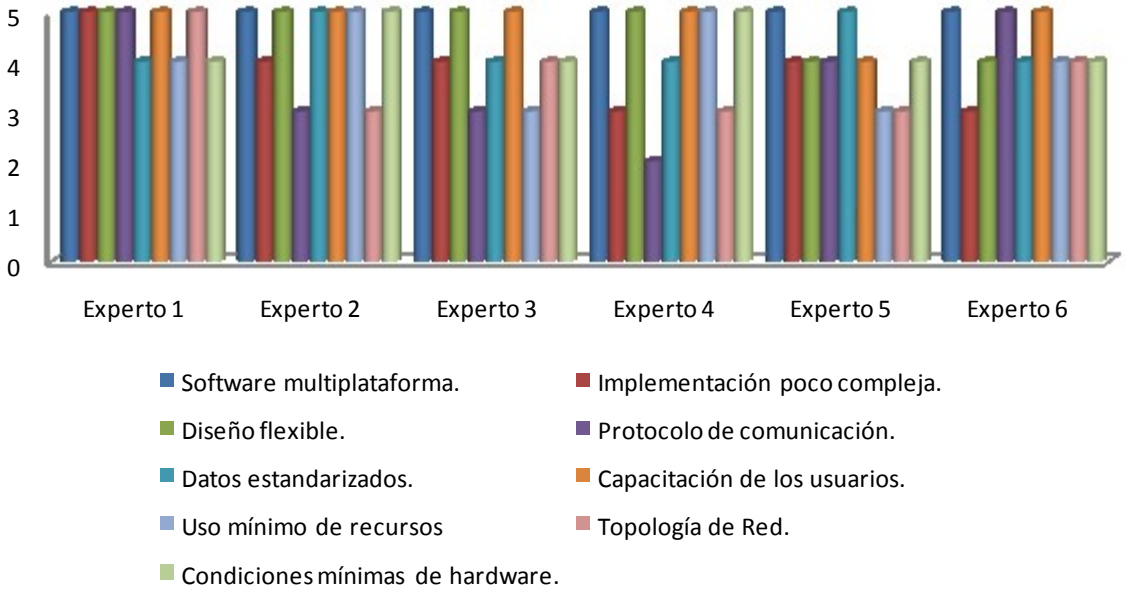

Fig. 17 Pregunta #2 de la entrevista.

Determinar el nivel de importancia de los factores.

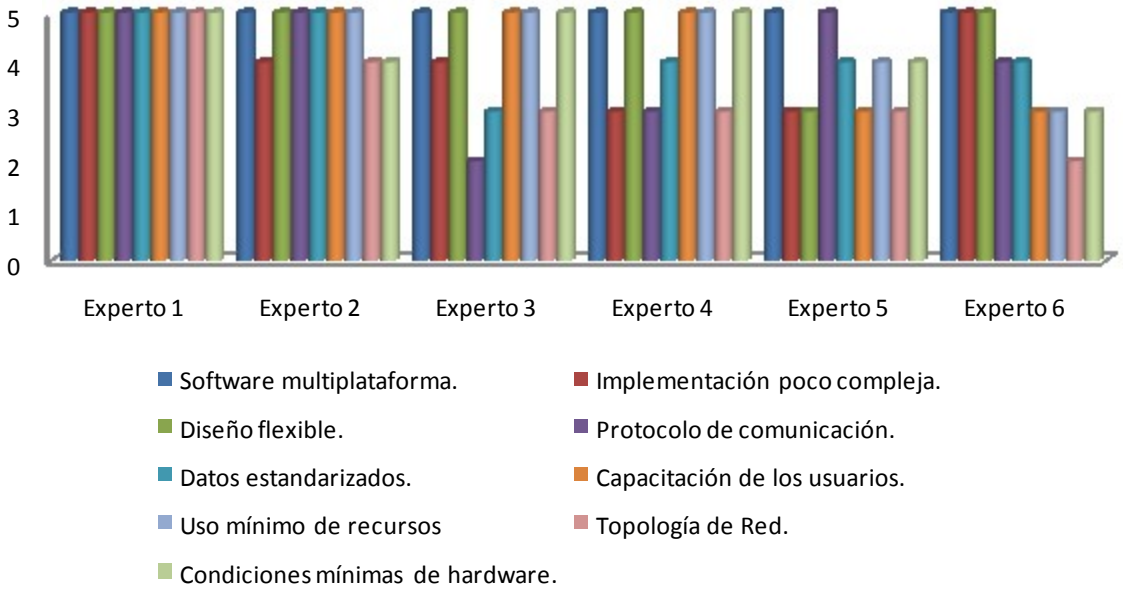

Fig. 18 Pregunta #3 de la entrevista.

Determinar la complejidad de los factores.

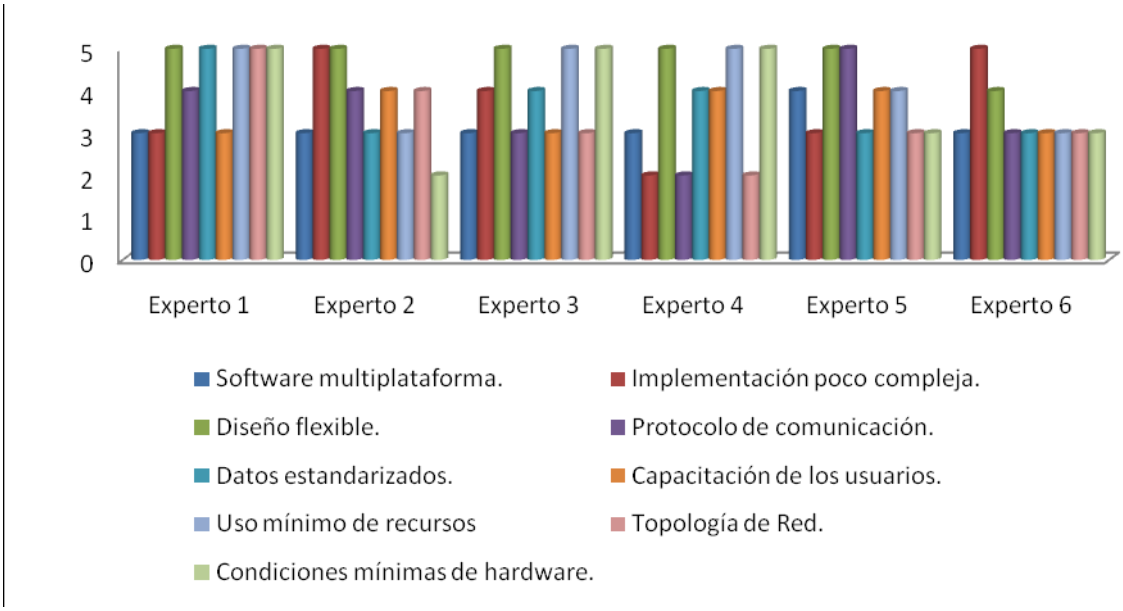

Fig. 19 Pregunta #4 de la entrevista.

3.2.3. Fase final.

En esta etapa se realiza un análisis estadístico, se procesa y estudia la información. Es importante calcular el porcentaje de las respuestas formadas por los especialistas, ya que este valor indicará cuan positivas o negativas fueron las respuestas en cada uno de los objetivos sometidos a evaluación.

El grado de concordancia de los especialistas luego de responder las preguntas es de 0.79, este valor fue calculado mediante el coeficiente de concordancia de Kendall utilizando el software "SPSS 13.0 for Windows", y como es un valor cercano a 1.00, se evidencia que los especialistas tienen un alto grado de concordancia a todas las preguntas.

Los resultados fueron positivos ya que con el establecimiento de la propuesta se mejora el proceso de desarrollo del software. Todo lo antes planteado se encuentra reflejado en la tabla de los porcientos que a continuación se muestra:

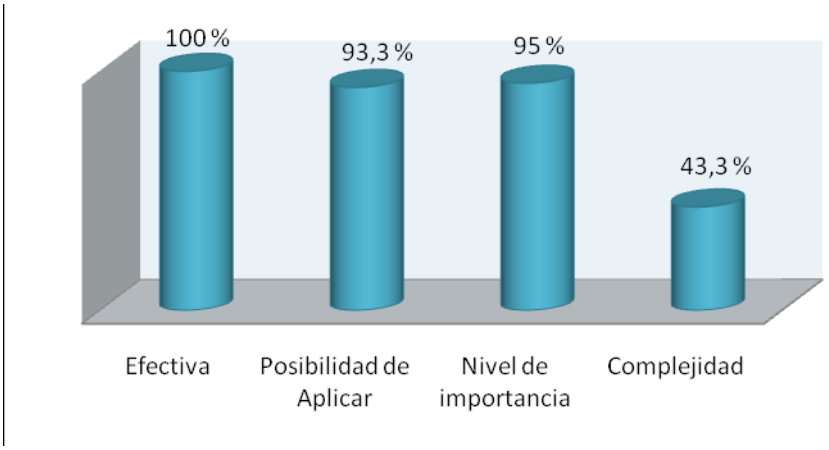

Fig. 20 Resultados de la entrevista.

Aplicando el método Delphi mediante el criterio de los expertos se arribó a la siguiente conclusión:

- La adecuada puesta en práctica de la guía va a mejorar la obtención de la adaptabilidad y la coexistencia en el proceso de desarrollo del software.

Con el desarrollo de este capítulo se evidencia la culminación de la presente investigación, donde se plantea el uso de un método de validación el cual fue muy de mucha importancia para obtener los resultados necesarios y adquirir la información suficiente sobre las valoraciones de los expertos.

Conclusiones.

En el transcurso de la elaboración de este trabajo de diploma se expuso la importancia y la necesidad de la obtención de nuevos productos que se caractericen por cumplir con los estándares de calidad. Obtener productos portables, garantiza el uso y la aceptación por parte de los clientes. Luego de terminado el desarrollo de este trabajo se arriba a las siguientes conclusiones:

- Los objetivos trazados fueron cumplidos satisfactoriamente, ya que se construyó una guía para la obtención de la adaptabilidad y la coexistencia en las soluciones de software.
- Se detectaron los principales factores que favorecen la adaptabilidad y la coexistencia en el proceso de desarrollo del software.
- Se elaboró la guía validándose satisfactoriamente
- La presente guía de buenas prácticas orientada a la calidad de las soluciones de software, constituye un modelo que sirve de apoyo para el proceso de desarrollo del software y sería muy efectiva su aplicación, teniendo en cuenta lo expresado por los especialistas encuestados.
- La guía es de fácil uso para cualquier persona con conocimientos mínimos en calidad de software.

Recomendaciones.

Luego de la investigación realizada en el presente trabajo y teniendo en cuenta las ideas que surgieron durante el progreso de la misma, se recomienda:

- Adaptar la guía propuesta a otro tipo de soluciones de software.
- Aumentar el valor de uso de la guía, identificando otros factores que favorezcan el logro de las características de calidad durante el proceso de desarrollo del software. (Funcionalidad, Eficiencia, Mantenibilidad, Confiabilidad, Usabilidad).

Referencias Bibliográficas.

1. Giraldo, Otoniel Pérez. April 29, 2008 [cited; Available from: http//www.willydev.net/descargas/willydev_planeasoftware.pdf.
2. Martinto, Pedro Carlos Pérez. (Mayo 2009) Diseno_metodol_de_la_invest-poblacion_y_muestra_Metodos_y_diseno_Experimental _Tema_3.
3. González, Sayda Cuello. León, Rolando Alfredo Hernández. *El Paradigma Cuantitativo De La Investigación Científica*. Noviembre del 2002, Editorial Universitaria EDUNIV.
4. Pressman, Roger S., Ingeniería del Software. Un enfoque práctico. Quinta ed. 2002: Mc Graw Hill.
5. IEEE. Standard Glossary of Software Engineering Terminology. 1990.
6. NC ISO/IEC 9126-1. 2005.
7. Eppler, M.J., *The Concept of Information Quality: An interdisciplinary Evaluation of recent Information Quality Frameworks*. Segunda ed. 2001.Berlin Heidelberg Germany.
8. Genero, Marcela. Jimenez, Luis. Piattini, Mario. *Measuring the Quality of Entity Relationship Diagrams.*2000.
9. Fillottrani, Pablo R., *Calidad en el Desarrollo de Software. Modelos de calidad de software*.2007.
10. ITEC “Leonardo DaVinci”. Calidad de Software.2006.
11. Kazman, R., Clements, P., Klein, M., *Evaluating Software Architectures. Methods and case studies.Addison Wesley*. 2001.
12. Finlay, P.N., *Introducing decision support systems*. Segunda ed. 1994, Oxford, UK Cambridge: NCC Blackwell; Blackwell Publishers.
13. Turban, E., *Decision support and expert systems: management support systems*. 1995, Englewood Cliffs, N.J.: Prentice Hall.
14. Turban, Efraim. Aronson, Jay E., *Decision Support systems and Intelligent systems*. Sexta ed. 2001: Prentice-Hall.
15. Hernández, Iliana Amabely Silva. *Generador Automático de Reportes Dinámicos*, *in Ingeniería Eléctrica*. Marzo del 2003.
16. Diccionario básico de Informática. [cited; Available from: http//usuarios.multimania.es/resve/diccioninform.htm
17. Rozic, Sergio Ezequiel. *Bases de Datos y su aplicación con SQL*. Mayo 2004, Buenos Aires, Argentina.: Editorial, MP Ediciones.
18. Slideshare.net. 2008 [cited; Available from:http//www.slideshare.net/rmonago/t1-gestion-de-datos-presentation.
19. ITPros-DC - SQL y YO. Para aprender con ITPros-DC y de SQL SERVER. 9 /11/2009 [cited; Available from: http://fransuavg.blogspot.com/2009/11/conceptos-de-bases-de-datos.html

20. Andrés, Maria Mercedes Marquez. 2001. [cited; Available from: http//www3.uji.es/~mmarques/f47/apun/node81.html
21. Andrés, Maria Mercedes Marquez. 2001. [cited; Available from: http//www3.uji.es/~mmarques/f47/apun/node4.html
22. Software Zone. [cited; Available from:http//www.softzone.es/glosario/s-t-y-u/.
23. LaBDA - Laboratorio de Bases de Datos Avanzadas 31/10/2008 [cited; Available from: http//basesdatos.uc3m.es/.
24. Silberschatz, Abraham. Korth, Henry F., Sudarshan, S., *Fundamentos de bases de datos*. Cuarta ed. 2002: Copyright © MMI, por McGraw-Hill Inc.
25. Proaño, Ing Diego Javier Burbano. *Analisis Comparativo De Bases De Datos De Código Abierto VS Código Cerrado*. Mayo del 2006.
26. Microsoft SQL Server, MySQL y PostgreSQL. Linux +.
27. MySQL. [cited; Available from: http://dev.mysql.com/doc/refman/5.0/es/introduction.html.
28. Cameron, N., PostgreSQL affiliates .ORG domain 2003.: COMPUTERWORLD.
29. Gracía, Joaquín. *Podredumbre del software.* 2003 [cited; Available from: http//www.ingenierosoftware.com/analisisydiseno/podredumbre.php.
30. LaWebera.es. 2010 [cited; Available from:http//www.lawebera.es/de0/compatibilidad-web-navegadores.php.
31. Kendall, K.E. Kendall.J.E., Análisis y diseño de sistemas. Sexta ed. 2005: Pearson educación.
32. Estévez, María de Lourdes Bravo. Gallastegui, José Joaquín Arrieta. *El método Delphi. Su implementación en una estrategia didáctica*. 2005. [cited; Available from: http//www.rieoei.org/deloslectores/804Bravo.PDF.

Bibliografía.

Andrés, Maria Mercedes Márquez. 2001[cited; Available from: http//www3.uji.es/~mmarques/f47/apun/node81.html.

Andrés, Maria Mercedes Márquez. 2001. [cited; Available from: http//www3.uji.es/~mmarques/f47/apun/node4.html.

Codazzi, IGAC Instituto Geográfico Agustín. 2004 [cited. Available from:http//www.igac.gov.co:8080/igac_web/UserFiles/File/ciaf/TutorialSIG_2005_26_02/paginas/ctr_sistemasdegestiondebasededatos.htm.

Cameron, N. PostgreSQL affiliates .ORG domain 2003: COMPUTERWORLD.

Diccionario básico de Informática. [cited; Available from: http://usuarios.multimania.es/resve/diccioninform.htm.

Eppler, M J., *The Concept of Information Quality: An interdisciplinary Evaluation of recent Information Quality Frameworks.* Segunda ed. 2001. Berlin Heidelberg Germany.

Fillottrani, Pablo R. *Calidad en el Desarrollo de Software. Modelos de calidad de software*. 2007.

Finlay, P.N., *Introducing decision support systems*. Segunda ed. 1994, Oxford, UK Cambridge: NCC Blackwell; Blackwell Publishers.

Giraldo, Otoniel Pèrez. April 29, 2008 [cited; Available from: http://www.willydev.net/descargas/willydev_planeasoftware.pdf.

García, Joaquín. Podredumbre del software. 2003 [cited; Available from: http://www.ingenierosoftware.com/analisisydiseno/podredumbre.php.

Hernàndez, Iliana Amabely Silva. *Generador Automático de Reportes Dinámicos*, in *Ingeniería Eléctrica*. Marzo del 2003.

IEEE. *Standard Glossary of Software Engineering Terminology.* 1990.

*ITEC "Leonardo DaVinci".Calidad de Software.*2006.

ITPros-DC - SQL y YO. Para aprender con ITPros-DC y de SQL SERVER. 9 /11/2009[cited; Available from: http://fransuavg.blogspot.com/2009/11/conceptos-de-bases-de-datos.html.

Kahn, Beverly K., Strong, Diana M. , *Information Quality Benchmarks: Product andService Performance.* Cuarta ed: ACM New York, 2002.

Kazman, R., Clements, P., Klein, M. *Evaluating Software Architectures. Methods and case studies. Addison Wesley.* 2001.

Kendall, K E., Kendall, J E., Análisis y diseño de sistemas. Sexta ed. 2005: Pearson educación.

León, Rolando Alfredo Hernández., González, Sayda Coello. *El Paradigma Cuantitativo De La Investigación Científica.* Noviembre del 2002, Editorial Universitaria EDUNIV.

LaBDA - Laboratorio de Bases de Datos Avanzadas 31/10/2008 [cited; Available from: http://basesdatos.uc3m.es/.

LaWebera.es. 2010 [cited; Available from: http://www.lawebera.es/de0/compatibilidad-web-navegadores.php.

M, Gonzalo Retamal. *Toma De Decisiones Y Solución De Problemas* 2000 [cited. Available from http://www.leonismoargentino.com.ar/INST283.htm

Mario Piattini, Coral Calero, Houari Sahraoui, Hakim Lounis. *Object-Relational Database Metrics* 2002 .

Mg Soluciones Informáticas 2004 [cited. Available from http://www.mginformatica.com.ar/modelo-de-calidad.htm.

Martinto, MSc: Pedro Carlos Perez. (Mayo 2009).*Diseno_metodol_de_la_invest poblacion_y_muestra_Metodos_y_diseno_Experimental _Tema_3*.

Marcela Genero, Luis Jimenez., Mario Piattini, *Measuring the Quality of Entity Relationship Diagrams* . 2000.

Microsoft SQL Server, MySQL y PostgreSQL. Linux +.

MySQL. [cited; Available from: http//dev.mysql.com/doc/refman/5.0/es/introduction.html.

NC ISO/IEC 9126-1. 2005.

Ortiz, Antonio Moreno. *Diseño E Implementación De Un Léxico Computacional Para La Lexicografía Y Traducción Automática.* 2000 [cited. Available from http://elies.rediris.es/elies9/.

Peter Rob, Carlos Coronel. *Sistemas De Bases De Datos: Diseño, Implementación Y Administración.*: Cengage Learning Editores.

Powell, Gavin. *Beginning Database Design*: Inc Wiley Publishing, 2006.

Programatium.Com [cited. Available from: http//www.programatium.com/bd/index.htm.

Pressman, Roger S., *Ingeniería del Software. Un enfoque práctico*. Quinta ed. 2002: Mc Graw Hill.

Proaño, Ing Diego Javier Burbano. Analisis Comparativo De Bases De Datos De Código Abierto VS Código Cerrado. Mayo del 2006.

Rozic, Sergio Ezequiel. *Bases de Datos y su aplicación con SQL*. Mayo 2004 Buenos Aires, Argentina.: Editorial, MP Ediciones.

Slideshare.net. 2008 [cited. Available from:http//www.slideshare.net/mvagila/bases-de-datos-presentation-719744.

Slideshare.net. 2008 [cited; Available from: http//www.slideshare.net/rmonago/t1-gestion-de-datos-presentation.

Software Zone. [cited; Available from: http//www.softzone.es/glosario/s-t-y-u/.

Silberschatz , Abraham., Korth, Henry F., Sudarshan, S.Fundamentos de bases de datos. Cuarta ed. 2002: Copyright © MMI, por McGraw-Hill Inc.

Tabares, Marta Silvia. *Técnicas Y Tecnologías De Las Bases De Datos.* [cited; Available from http://www.unalmed.edu.co/~mstabare/

Tutorial De Base De Datos 1 2006 [cited. Available from http//sistemas.itlp.edu.mx/tutoriales/basedat1/tema1_4.htm.

Turban, Efraim. *Decision support and expert systems: management support systems*. 1995, Englewood Cliffs, N.J.: Prentice Hall.

Turban, Efraim. Aronson, Jay E. *Decision Support systems and Intelligent systems*. Sexta ed. 2001: Prentice-Hall.

Valdés, Damián Pérez. *Maestros Del Web* Octubre, 2007. [cited. Available from: http//www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

Anexos.

Anexo # 1: Encuesta.

Siguiendo la línea de investigación, se hizo necesaria la realización de una encuesta la cual permitirá validar la existencia de problemas relacionados con las subcaracterísticas de adaptabilidad y coexistencia en el proceso de desarrollo de software de apoyo a la toma de decisiones. Dicha encuesta es aplicada en los laboratorios de DATEC (Centro de Tecnologías de Gestión de Datos), donde se centra el estudio.

PREGUNTAS :

Adaptabilidad: Capacidad del producto software de ser adaptado a los ambientes especificados sin aplicar acciones o medios de otra manera que aquellos suministrados con el propósito de que el software cumpla sus fines.

Software independiente de plataforma. (Se refiere a la necesidad de que las soluciones de software funcionen en diversas plataformas, es decir que no dependan de un Sistema Operativo (SO) en específico).

1. ¿Se puede presentar el caso en que el usuario no pueda instalar la solución de software porque está sea incompatible con la plataforma que presenta en su computadora?

   SI _____ NO _____

Flexibilidad en el diseño. (Hace referencia a la dependencia funcional, por lo que cambios en un módulo provocan cambios en otros módulos independientes, haciendo que el software sea frágil ante la más mínima modificación)

2. ¿En el proceso de elaboración del software si no se tiene una total comprensión de lo que debe hacer el mismo, esto puede provocar un atraso en la entrega del producto?

   SI _____ NO _____

3. La rigidez es un concepto análogo a la poca flexibilidad y muy asociado a la dependencia funcional. ¿Cambios en una parte de la solución de software pueden provocar cambios en otras partes de la misma?

   SI _____ NO _____

Implementación poco compleja. (Se refiere a que los programadores encuentran muchas formas de implementar un cambio, pero esta implementación es muy compleja para lograr mantener la filosofía del diseño original)

4. ¿Se ha hecho necesario el uso de una programación compleja (Parches) para mantener la filosofía del diseño original?

   SI _____ NO _____

Estandarización de los datos. (Capacidad que debe poseer la solución de software para mostrar la información sin que se vea afectada por la incompatibilidad de navegadores, es decir que sea capaz de mostrar las imágenes sin distorsionar y los datos con un formato estandarizado)

5. ¿Cuándo se muestran los resultados obtenidos del sistema estos se presentan en ocasiones distorsionados o fuera del formato predeterminado con anterioridad?

   SI _____ NO _____

Condiciones mínimas de hardware (Hace referencia a los requerimientos mínimos necesarios que debe poseer la computadora donde será instalada la solución de software, es decir se deben tener en cuenta estos requerimientos a la hora de desarrollar la solución de software)

6. ¿Es posible que pasado cierto tiempo de estar instala la solución de software, está presente problemas de capacidad en la Base de datos?

   SI _____ NO _____

Protocolo de comunicación óptimo. (Necesidad de establecer un protocolo capaz de garantizar la comunicación entre la solución de software y otras aplicaciones)

7. ¿Es posible que si no se tiene en cuenta cómo será la comunicación que se va a establecer entre la solución de software y el resto las aplicaciones, esto provoque un mal funcionamiento, ya que no se podrían conectar adecuadamente?

   SI _____ NO _____

Topología de red del ambiente. (Se debe tener en cuenta la topología de red para garantizar que al ser instalada la aplicación esta no sobrecargue el servidor y no provoque la caída constante de los servicios.)

8. ¿Es necesario conocer como está estructurada la red en el lugar donde será implementada la solución de software?

   SI _____ NO _____

Coexistencia: Capacidad del producto software de coexistir con otro software independiente en un ambiente común y compartir los recursos comunes.

Uso mínimo de recursos. (Presente cuando un programa X impide la culminación de la ejecución de un programa Y debido a que X está utilizando un recurso que Y necesita)

9. ¿Es posible que al instalar la solución de software junto con otras, estas utilicen más memoria RAM de la que deberían, haciendo así el trabajo más lento para que el usuario interactué con la solución?

   SI _____ NO _____

10. ¿En caso de que dos soluciones de software sean instaladas en la misma computadora y estas utilicen un mismos recursos, esto podría provocar que al usuario interactuar con la aplicación la computadora procese más lento la información?

    SI _____ NO _____

Anexo # 2: Guía de Buenas Prácticas.

Guía para la obtención de la adaptabilidad y la coexistencia de las Soluciones de software de apoyo a la toma de decisiones.

Introducción.

Cuando se habla de Ingeniería de Software existe un criterio generalizado que expresa que se deben construir sistemas **útiles** y de **calidad.**

Útiles porque cumplen con los requerimientos del usuario y la organización logra con ellos algo que no alcanzaría si no existieran. Y **calidad** porque verifican un conjunto de “áreas clave” en el desarrollo del sistema.

Pero la calidad no es tan fácil de definir, incluso no existe una definición completa. La confiabilidad y la mantenibilidad son algunas de las cualidades que como criterios de calidad deben poseer las soluciones de software, pero no son las únicas, existen otras características de gestión no menos importantes como es la portabilidad. Esta característica se convertirá en un factor de gran importancia y que construyendo

correctamente sus subcaracterísticas (adaptabilidad y coexistencia) se garantizará que el software final sea lo más portable posible.

El objetivo principal de esta guía es lograr obtener, configurar y diseñar la adaptabilidad y la coexistencia mediante un conjunto de buenas prácticas, para de esta forma poder garantizar la portabilidad de las soluciones de software a lo largo del proceso de desarrollo. Por buenas prácticas entiéndase un conjunto coherente de acciones que permiten darle solución a un problema en determinado momento.

La guía será aplicada a las soluciones de software de apoyo a la toma de decisiones, es importante que se emplee correctamente pues con ella se logrará un software más usado por los clientes, ya que se podrá mover con facilidad y se adaptará a cualquier entorno obteniendo por ende un software más portable.

La presente guía está estructura según el ciclo de vida del sistema en construcción:

- Requisitos.
- Análisis y Diseño.
- Implementación.
- Pruebas.
- Despliegue.
- Soporte.

I. Requisitos.

En esta etapa se establecen los requisitos a tener en cuenta durante el proceso de desarrollo de la solución de software, los cuales son:

- Software multiplataforma.
- Diseño flexible.
- Implementación poco compleja.
- Memoria RAM de la PC.
- Capacidad del disco duro.
- Sistema operativo.
- Protocolo de comunicación.
- Velocidad de procesamiento del CPU.
- Datos estandarizados.
- Dependencias funcionales.
- Configuración de la red.
- Capacitación de los usuarios.

II. Análisis y Diseño.

En esta actividad aparecen factores a tener en cuenta, entre los que se pueden mencionar: que el software sea multiplataforma, presente un diseño flexible, la implementación sea poco compleja, los requerimientos sean los mínimos, el protocolo de comunicación sea el adecuado y por último la topología de red favorezca a las características de la aplicación.

Garantizar que el software sea multiplataforma.

La aplicación debe poder ser instalada en cualquier plataforma. Esto facilitará que se lleven a cabo migraciones de cargas de trabajo desde cualquier ubicación y a cualquier destino.

- Se recomienda establecer la arquitectura por capas, ya que esta favorece a la adaptabilidad y a la coexistencia aunque no de manera directa.
- Es importante diseñar un sistema que limite su dependencia a una plataforma, para ello se sugiere el uso de los patrones de diseño:

  Abstract Factory: Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

  Bridge o Puente: Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

Flexibilidad en el diseño.

El sistema debe ser flexible para poder garantizar que en caso de ser necesario, se puedan realizar cambios para lograr adaptarlo al ambiente que se desea.

- Se recomienda emplear los siguientes patrones de diseño ya que con ellos se puede lograr obtener un diseño más flexible.

  Adapter: Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

  Composite: Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.

  Observe: Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

  Interface: Define un comportamiento independiente de donde vaya a ser utilizado.

Implementación poco compleja. (Viscosidad)

Los programadores encuentran muchas formas de implementar un cambio, pero esta implementación es muy compleja para lograr mantener la filosofía del diseño original.

- Se exhorta a emplear los siguientes patrones de diseño ya que gracias a ellos se puede alcanzar una implementación poco compleja y por lo tanto más adaptable.

  Flyweight o Peso ligero: Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.

Establecer protocolo de comunicación óptimo.

Se debe determinar el protocolo de comunicación bajo el cual se pueda realizar una mejor comunicación entre la aplicación y las bases de datos, garantizando así estabilidad en las comunicaciones y evitando la caída de los servicios.

- Se sugiere utilizar la arquitectura o familia TCP/IP (Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP)) ya que esta sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN). Es la base de Internet, y tiene un grado muy elevado de fiabilidad, es adecuado para redes grandes y medianas, así como en redes empresariales. Proporciona una abstracción del medio haciendo posible el intercambio de información entre medios desiguales y tecnologías que un principio eran incompatibles.

Uso mínimo de recursos.

Se recomienda tener presente el mínimo de recursos que utilizará la solución de software para evitar que existan problemas a la hora que esta sea instalada en el ambiente real.

- Se recomienda impedir que más de un proceso acceda simultáneamente a las variables compartidas y esto se logra garantizando la exclusión mutua.
- Para lograr alcanzar la exclusión mutua se tienen tres tipos de soluciones:
  - ✓ Soluciones de software.

    Una manera es dejar la responsabilidad a los procesos que se deseen ejecutar concurrentemente, de esta manera los procesos deben coordinarse unos con otros para cumplir la exclusión mutua sin ayuda alguna, aunque estas soluciones son propensas a errores y a una fuerte sobrecarga de proceso (Algunos ejemplos: Algoritmo de Dekker y Algoritmo de Peterson).

  - ✓ Soluciones de hardware.
    - En caso de conflicto, mantienen la integridad del sistema descartando las actualizaciones.
    - Bloquean todo aquello que pueda interferir.

- Actualizan la variable.
- Desbloquean lo bloqueado al principio.

✓ Soluciones aportadas por el Sistema Operativo.

- Se utilizan los semáforos ya que resultan adecuados cuando hay que proteger un recurso que puede compartir varios procesos. El semáforo se inicializa con el número total de recursos disponibles (n), las operaciones de espera y señal se diseñan de modo que se impida el acceso al recurso protegido por el semáforo cuando el valor de éste es menor o igual que cero. Cada vez que se solicita y obtiene un recurso, el semáforo se decrementa y se incrementa cuando se libera uno de ellos. Si la operación de espera se ejecuta cuando el semáforo tiene un valor menor que uno, el proceso debe quedar en espera de que la ejecución de una operación libere alguno de los recursos.
- Los mensajes son otra solución del sistema operativo que permitirá la sincronización de procesos y la comunicación entre ambos.

Establecer las condiciones mínimas de hardware.

Se recomienda tener en cuenta los requerimientos mínimos de los generadores de reportes dinámicos así como los requerimientos mínimos de la PC servidora donde será instalada la solución de software.

- El servidor donde se instalará la Base de Datos del sistema debe cumplir con los siguientes requerimientos (puede ser el mismo donde estará la aplicación):
  - ✓ SO: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.
  - ✓ PostgreSQL versión 8.3 o superior.
  - ✓ PGAdmin III o algún administrador para PostgreSQL.
- El servidor donde se instalará la aplicación debe cumplir con los siguientes requerimientos:
  - ✓ SO: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.
  - ✓ Paquetes: apache2, php5, libapache2modphp5, php5cli, php5mysql, php5pgsql, php5sqlite, php5xsl, php5gd.
- Para lograr instalar PostgreSQL en Linux se deben tener en cuenta los siguientes requerimientos mínimos:
  - ✓ 8 Mb de RAM.
  - ✓ 30 Mb de espacio en disco para el código fuente.
  - ✓ 5 Mb de espacio en disco para la instalación de los ejecutables.
  - ✓ 1 Mb extra para las base de datos básicas.

- ✓ 3 Mb de espacio en disco rígido para el trabajo con el código fuente.

- Para instalar PostgreSQL en Windows se debe tener en cuenta que el Sistema Operativo (SO) sea estable y funcione bien. Además debe estar en la versión 8.2 o superior, ya que en esta se corrigen errores de código que tienen las versiones anteriores y así no se pone en riesgo la estabilidad del código. Se requiere para los Clientes en Windows, una memoria RAM de 128 MB como mínimo para Windows 2000 y de 256 MB para XP, para los puestos con carga de Impresión se recomienda una capacidad de 512 MB. Además los servidores deben tener la capacidad adecuada para soportar la cantidad de clientes que tendrá la Base de Datos.

Establecer la Topología de Red adecuada.

Existen diferentes tipos de topologías de red, según la que se necesite en la empresa, pero siempre se debe tener en cuenta que esta debe ser óptima o útil y sobre todas las cosas su funcionamiento debe impedir que se formen cuellos de botella. Se recomienda al análisis de las diferentes topologías de red y teniendo en cuenta las características del medio donde será instalada la aplicación final, seleccionar la más adecuada.

- En la topología **Malla** todos los nodos están conectados entre sí, de modo que si un nodo falla, siempre habrá otro camino para mandar la información. Esta es una de las topologías cuya funcionalidad trae muchas ventajas, ya que es de gran fiabilidad y es inmune a problemas de fallos o cuellos de botella. Sin embargo como todos los nodos están conectados entre sí, es muy costosa de instalar.
- En la topología **Anillo** los host están distribuidos en forma de anillo y cada uno establece conexión receptor/trasmisor con el siguiente host. Cada host tiene la función de repetidor para pasar la señal al siguiente host regenerando la señal y ampliando la red. La principal desventaja es que entre más nodos se agreguen será menos eficiente, además si un nodo falla, toda la red lo hará.

  En la topología **Estrella** todos los nodos están conectados a un nodo central, eliminando el problema que plantea la topología de anillo, si un nodo falla la red seguirá funcionando, pero si el central falla, toda la red lo hará. Debido a que la topología estrella utiliza un cable de conexión para cada computadora, es muy fácil de expandir, sólo dependerá del número de puertos disponibles en el switch. Es fácil de implementar, pero tiene una limitación de alcance de hasta 100 metros.
- La topología **Árbol jerárquico** es la más empleada y muy útil para implementar en grandes empresas, pues la señal puede viajar más lejos. Sin embargo su implementación es difícil de configurar y si se cae la rama principal toda la red lo hará.

III. Implementación.

En esta actividad se tiene presente la Estandarización de los datos como principal factor que de no tenerse en cuenta en el proceso de desarrollo podría provocar que el software no quedará todo lo adaptable o coexistente posible.

Estandarización de los datos.

La gran mayoría de las soluciones de software de apoyo a la toma de decisiones son aplicaciones web por las facilidades de uso que brindan, pero en ocasiones la compatibilidad de estas en diferentes sistemas operativos no es la misma, ya que requieren de elementos necesarios (**plug-in)** para que estás puedan operar en distintos entornos y que no varíe el formato de los datos. Mostrar los datos en un navegador u otro sin que sufran cambios o distorsiones en las imágenes es lo que se quiere lograr por lo que:

- Se recomienda estandarizar la salida de los datos mediante la validación del código de tu Web en base a los estándares del W3C. Este proceso consiste en buscar en los archivos HTML y/o CSS errores de programación y corregirlos, posibilitando aumentar la compatibilidad de navegadores y asegurando la calidad del código y su valor para futuras revisiones.
- Para ello se sugiere usar el validador de CCS del W3C, ya que esta entidad se encarga de crear los estándares de la Web. Utilizando esta herramienta online se elimina la ocurrencia de errores y se agiliza esta tarea que de hacerse manualmente resultaría muy tediosa.

IV. Pruebas.

En esta actividad del proceso de desarrollo del software no se encuentra presente ningún factor que logre las subcaracterísticas de adaptabilidad y coexistencia, sin embargo se debe tener presente porque es en ella donde se verifica que en el transcurso del proceso se haya logrado lo que se quería con la aplicación de la guía en las fases anteriores.

- Se recomienda la realización de pruebas para determinar si se ha logrado obtener hasta el momento la adaptabilidad y la coexistencia de la solución de software.

V. Despliegue.

En la presente actividad se tuvo en cuenta la necesidad de que los usuarios finales estuvieran capacitados de acuerdo a las necesidades y funcionalidades del sistema en construcción. Dándole cumplimiento con este factor se garantizará que los usuarios finales sean capaces de interactuar con la solución de software de forma más adaptable.

Capacitar a los usuarios finales.

Proporcionar al usuario final conocimientos, habilidades y actitudes para un mejor desempeño de su trabajo antes de enfrentarse a la aplicación, en cuanto al uso de herramientas similares, documentación

relacionada con el tema y funcionalidades del sistema es una vía racional que ni los *Manuales de Usuario* ni las *Ayudas* que posea el software les servirá para entenderlo.

- Se sugiere que la capacitación no sea con el fin de cumplir un mero requisito, sino que se obtenga a partir de un proceso continuo.
- Se recomienda capacitar al usuario final en un ambiente real, porque ahí es donde realmente el software funcionará, y se comprueba la calidad del mismo.
- Se propone que a la hora de realizar la capacitación los usuarios estén separados según los niveles de habilidades que posean para evitar que usuarios novatos se pierdan con las explicaciones y los expertos se aburran con los puntos básicos.
- Para lograr una capacitación lo más óptima posible se recomienda seguir los siguientes lineamientos:
    - Establecimiento de objetivos de capacitación.

Los objetivos de capacitación de cada grupo deben ser definidos con claridad ya que estos monitorizaran el proceso de capacitación y permiten llevar un control sistemático de las habilidades adquiridas por los capacitados.

- Uso de métodos de capacitación apropiados.

Se deben tener en cuenta los diferentes métodos para la capacitación, ya que existe para cada usuario varios que les facilita su auto aprendizaje. Algunos usuarios aprenden mejor viendo, otros oyendo y otros haciendo. Debido a esta variedad no es posible personalizar la capacitación, por lo que la mejor manera es combinarlos todos. De esta forma se llega a la mayoría de los usuarios por uno u otro método.

- Selección de lugares de capacitación adecuados.

La capacitación se puede dar en cualquier lugar. Siempre y cuando se tenga en cuenta que el ambiente sea adecuado y permita al usuario concentrarse en el aprendizaje del nuevo sistema. Si la capacitación se lleva a cabo en el lugar de la implantación del software, el usuario se familiarizará aún más con el mismo, y estará listo para usarlo cuando este sea completamente operacional.

- Empleo de materiales de capacitación comprensibles.

Es de vital importancia el uso de materiales para la capacitación. Se deben usar manuales escritos con claridad, es decir, estos manuales deben tener buenos índices, estar escritos para los usuarios adecuados, contener un vocabulario especial y disponible para cualquier usuario que lo necesite.

VI. Soporte.

En el proceso de la investigación no se obtuvieron factores a tener en cuenta durante esta actividad.

Conclusiones.

Con la correcta aplicación de esta guía se facilitará el trabajo del equipo de desarrollo, identificando los principales errores que se comenten durante el proceso de desarrollo de una solución de software y de esta forma poder erradicarlos. Se podrá asegurar que una vez terminado el proceso, la solución sea adaptable y coexistente en cualquier ambiente y con cualquier aplicación.

Anexo # 3: Entrevista.

Entrevista encaminada a validar la propuesta solución.

La presente entrevista forma parte de un estudio sobre los factores que apoyan la adaptabilidad y la coexistencia dentro de la portabilidad, en el proceso de desarrollo de las soluciones de software de apoyo a la toma de decisiones. La guía elaborada tiene como objetivo brindar factores que ayuden a decidir que buenas prácticas tener en cuenta durante el proceso de desarrollo del software para obtener la portabilidad requerida en este tipo de aplicaciones. Se necesita conocer del criterio de especialistas del tema ya que es de suma importancia obtener información cualitativa sobre el impacto de estos factores en dicho proceso.

Se espera de su total colaboración para obtener los mejores resultados.

Preguntas

Se estima que los siguientes factores pudieran ser importantes para garantizar la adaptabilidad y la coexistencia durante el proceso de desarrollo de las soluciones de software de apoyo a la toma de decisiones. Otorgue según su apreciación y orden de importancia a cada uno de estos factores una calificación entre 1 y 5 puntos, en correspondencia con la pregunta realizada.

Calificación según nivel del factor evaluado.

Muy alto	**5 pts.**
Alto	**4 pts.**
Medio	**3 pts.**
Bajo	**2 pts.**
Muy Bajo	**1 pts.**

1. **¿Cree usted que la propuesta es efectiva?**

2. **Posibilidad de Aplicar los factores.**
   [Referido a la capacidad de los factores de ser aplicados en el proceso de desarrollo de este tipo de soluciones de software.]

Factores	Peso Otorgado [1-5ptos]
Software multiplataforma.	
Implementación poco compleja.	
Diseño flexible.	
Protocolo de comunicación.	
Datos estandarizados.	
Capacitación de los usuarios.	
Uso mínimo de recursos.	
Topología de Red.	
Condiciones mínimas de hardware.	

3. **Determinar el nivel de importancia de los factores.**

[Referido a cuán efectivo es el factor si se tiene en cuenta en el proceso de desarrollo del software.]

Factores	Peso Otorgado [1-5ptos]
Software multiplataforma.	
Implementación poco compleja.	
Diseño flexible.	
Protocolo de comunicación.	
Datos estandarizados.	
Capacitación de los usuarios.	
Uso mínimo de recursos.	
Topología de Red.	
Condiciones mínimas de hardware.	

4. **Determinar la complejidad de los factores.**

[Referido al nivel de complejidad de cada uno de los factores en el proceso de desarrollo del software.]

Factores	Peso Otorgado [1-5ptos]
Software multiplataforma.	
Implementación poco compleja.	
Diseño flexible.	
Protocolo de comunicación.	
Datos estandarizados.	
Capacitación de los usuarios.	
Uso mínimo de recursos.	

Topología de Red.	
Condiciones mínimas de hardware.	