Facultad 10

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

Título: Guía para la obtención de la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL

Autores:

Hansel Laza Jiménez

Yoanni Navarro Frómeta

Tutor:

Ing. Nadia Porro Lugo

Co-tutor: Ing. Michaell González Jorrín

Ciudad de La Habana, Junio 2010

DECLARACIÓN DE AUTORÍA

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

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

YOANNI NAVARRO FROMETA

_____

Firma del Autor

HANSEL LAZA JIMÉNEZ

_____

Firma del Tutor

Síntesis de la Tutora:

Especialidad de graduación: Ingeniería en Ciencias Informáticas.

Categoría docente: Instructor.

Categoría Científica: -----

Años de graduado: 3 años.

Síntesis del Co-Tutor:

Especialidad de graduación: Ingeniería en Informática.

Categoría docente: Asistente.

Categoría Científica: Máster en Gestión de Proyectos Informáticos.

Años de graduado: 7 años.

La Industria del Software en la actualidad dada las características del mercado, exige la mejora constante de los procesos productivos, para crear así, productos con calidad. Por tal razón en el proceso de desarrollo de soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL del centro de datos DATEC es necesario que exista la mayor calidad posible.

Una evaluación integral y detallada de la calidad de los productos de software es un factor clave para asegurar que la calidad de estos sea la adecuada. Esto se puede lograr definiendo de manera apropiada las características de calidad; este trabajo se centrará en la portabilidad y dentro de esta, en dos de sus subcaracterísticas como son la reemplazabilidad y la conformidad de la portabilidad, teniendo en cuenta el propósito del uso del producto de software en la institución.

Es importante especificar y evaluar cada característica relevante de la calidad de los productos de software, cuando esto sea posible, utilizando mediciones validadas o de amplia aceptación, que hagan técnicamente transparente esta actividad.

Por tanto, el **objetivo** de este trabajo consiste en realizar una guía para la obtención de la reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

PALABRAS CLAVE

Calidad, portabilidad, reemplazabilidad, conformidad de la portabilidad, PostgreSQL, Data Warehouse, soluciones informáticas de apoyo a la toma de decisiones.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El enfoque tradicional de la ingeniería del software ha puesto sus esfuerzos en la definición de metodologías, lenguajes de programación, modelos de desarrollo y herramientas. Sin embargo, actualmente, la ingeniería del software, en su enfoque orientado a procesos, recomienda prestar más atención a la forma de realizar los productos, sin dejar a un lado su documentación. Además, argumenta que la calidad del producto depende de la calidad del proceso que se sigue para obtenerlo. Es decir, que depende fuertemente de las personas, la organización y los procedimientos utilizados para crearlo, entregarlo y mantenerlo. Por lo tanto, la calidad de un producto no puede ser asegurada simplemente con la simple inspección, llevando a cabo controles estadísticos o centrando sus programas de calidad únicamente al producto.

En la actualidad está siendo usado el ISO/IEC 9126 que es un estándar internacional que permite evaluar la calidad del producto software. Este estándar actualmente se encuentra siendo guiado y supervisado por el ISO/IEC 25000:2005 y una serie de estándares agrupados dentro del proyecto SQuaRE (Software product Quality Requirement and Evaluation). El ISO/IEC 9126 contiene un modelo de medición y un modelo de calidad que permiten llevar a cabo la evaluación de la calidad de los productos de software. Actualmente el estándar está conformado de 4 partes que dirigen: un modelo de calidad, métricas externas, métricas internas y calidad en las métricas de uso. La primera parte agrupa y define un conjunto de características y subcaracterísticas para determinar la calidad de un producto. Ellas son: Funcionalidad , Fiabilidad ,Usabilidad , Eficiencia , Mantenibilidad y Portabilidad en la cual haremos referencia principalmente en dos de sus subcaracterísticas que son la reemplazabilidad y la conformidad de la portabilidad.

La Industria Cubana del Software por supuesto no está exenta de problemas de calidad, por tal motivo y ante la necesidad de convertirla en un sector competitivo en el mundo, surge la exigencia de tener en cuenta estas características en el proceso de desarrollo de software. La universidad como parte de la industria cuenta con numerosos proyectos y dentro de ellos se encuentra el centro de datos DATEC, que entre otras cosas se encarga de crear soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL, las cuales presentan problemas de calidad de este tipo.

Por lo antes dicho estamos en presencia de una **situación problémica** en el centro de datos DATEC donde existe pobre conocimiento acerca de la portabilidad y sus subcaracterísticas asociadas, reemplazabilidad y conformidad de la portabilidad, esto se traduce en:

- Uso indebido de patrones de diseño y de arquitectura.
- Incorrecta selección del lenguaje de programación.
- Proceso complejo en la obtención de actualizaciones.

Por tanto, el **problema a resolver** queda formulado a modo de interrogante de la siguiente forma: ¿Cómo obtener la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL?

Con esta investigación/trabajo los aportes prácticos esperados del trabajo son:

- Los factores para obtener la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- Una guía para diseñar, configurar y obtener la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

El **objeto de estudio** lo constituye **l**a reemplazabilidad y la conformidad de la portabilidad.

De aquí se deriva que **el campo de acción** abarque la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

El **objetivo general del trabajo** es: construir una guía para diseñar, configurar y obtener la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

De acuerdo con esta propuesta se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la Investigación.
- Obtener una guía para diseñar, configurar y obtener la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- Validar la propuesta.

Para cumplir con los objetivos y resolver la situación problémica planteada, se proponen las siguientes **acciones o tareas**:

- Seleccionar y revisar bibliografías para actualizar los logros y limitaciones existentes sobre la evaluación de la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- Analizar modelos de calidad tales como ISO/IEC 9126, McCall, Boehm, etc. Análisis de las métricas relacionadas con la reemplazabilidad y la conformidad de la portabilidad.
- Analizar tecnologías similares a PostgreSQL Realización de un diagnóstico de los métodos de pruebas que se realizan para evaluar la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- Identificar los involucrados potenciales en el diagnóstico y caracterización de su marco de actuación respecto al tema de investigación. Realización de entrevistas a personas especializadas en el tema de la reemplazabilidad y la conformidad de la portabilidad.
- Analizar los tipos y métodos de pruebas que se realizan para medir la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

- Analizar riesgos potenciales asociados con las subcaracterísticas de calidad estudiadas, los métodos y tipos de pruebas y las herramientas.
- Evaluar la información obtenida y definir la posición como investigador.
- Aplicar el método de evaluación de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- Validar el método aplicado.

Por tanto, durante la investigación se **defiende la idea**:

Si se construye una guía se podrá diseñar, configurar y obtener la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

Introducción

La industria de software cubana se plantea la necesidad de crear software con la mayor calidad posible. En este sentido, la universidad y dentro de esta, el centro de datos DATEC, desempeñan un papel trascendental como desarrolladores de software. Esta última prestándole mayor atención específicamente a las soluciones informáticas que utilizan Data Warehouse basadas en PostgreSQL. Por este motivo resulta conveniente introducir factores que favorezcan a la calidad de estos productos haciendo mayor énfasis en la portabilidad y dentro de esta en sus subcaracterísticas reemplazabilidad y conformidad de la portabilidad.

Los modelos de calidad existentes junto con el estándar ISO 9126 aportan gran cantidad de datos en cuanto a calidad y sus características se refiere por lo que su estudio resulta fundamental para entender y llevar la calidad a estas soluciones informáticas. Resulta provechoso también el estudio de todo lo referente a inteligencia de negocio, dígase sistemas de apoyo a la toma de decisiones, Data Warehouse, Datamarts y otros, para lograr un mayor entendimiento dentro de este contexto. Los patrones y estilos ya sean de diseño como de arquitectura aportan gran peso en cuanto a calidad se refiere, pudiendo beneficiar y a su misma vez perjudicar a varias de sus características, por tal motivo es importante un detallado estudio de estos para potenciar la característica que se desee.

La selección de las herramientas para la confección de estas soluciones informáticas dígase sistemas gestores de bases de datos, procesos ETL (*Extraction, Transformation, Load*) entre otros, son de mucha importancia debido a que pueden beneficiar la calidad del producto en sí, por lo que su estudio va a resultar beneficioso.

1.1 ¿Qué es la Calidad de Software?

En toda empresa grande o pequeña o en cualesquiera de sus áreas se manejan entornos que tienen diferentes niveles de actuación, desde la parte administrativa hasta las áreas donde se desarrolla o proporciona el producto o servicio, procesos que van enfocados a ser parte de una comunidad que también exige resultados que sean al menos igual a sus expectativas. “Tanto en los medios de comunicación escritos y audiovisuales como en las revistas técnicas el tema de la calidad tiene una presencia continuada; incluso los políticos y gobernantes incluyen el término en sus discursos y

proyectos". (1) Vivimos rodeados de esa mágica palabra llamada calidad y la exigimos, pero en definitiva, ¿sabemos de verdad lo que significa calidad?

De acuerdo con la definición del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) "La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario".

Pressman define la calidad del software como:

"la concordancia con los requerimientos 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".

1.1.1 Modelos de Calidad

El diccionario de la Real Academia Española (RAE) reseña diferentes usos y significados del término modelo (del italiano modello). Entre ellos, se destaca que un modelo es un arquetipo o punto de referencia para imitarlo o reproducirlo, o un ejemplar que se debe seguir e imitar por su perfección.

"Un modelo, en sentido amplio, es un arquetipo, una referencia a imitar o reproducir. En la ingeniería de software un modelo nos proporcionará el objetivo a alcanzar, donde debemos llegar, pero no nos indicará como".(2)

Un modelo de calidad es, por lo tanto, un conjunto de prácticas vinculadas a los procesos de gestión y el desarrollo de proyectos. Este modelo supone una planificación para alcanzar un impacto estratégico, cumpliendo con los objetivos fijados en lo referente a la calidad del producto o servicio. Es un conjunto de buenas prácticas para el ciclo de vida del software, enfocado en los procesos de gestión y desarrollo de proyectos.

1.1.1.1. Modelo de McCall

El modelo fue propuesto por McCall en 1977 y está orientado a los desarrolladores de sistemas, para ser utilizado durante el proceso de desarrollo.

Abarca tres áreas de trabajo:

Operación del Producto

Requiere que pueda ser comprendida rápidamente, operada eficientemente y que los resultados sean aquellos requeridos por el usuario.

- **Corrección:** Hasta dónde satisface un programa su especificación y logra los objetivos de la misión del cliente.
- **Fiabilidad:** Hasta dónde se espera que un programa lleve a cabo su función pretendida con la exactitud requerida.
- **Eficiencia:** Cantidad de recursos informáticos y códigos necesarios para que un programa realice su función.
- **Integridad:** Hasta dónde se controla el acceso al software o a los datos por personas no autorizadas.
- **Usabilidad (Facilidad de manejo):** Esfuerzo necesario para aprender, operar, preparar los datos de entrada e interpretar las salidas (resultados) de un programa.

Revisión del Producto

Está relacionada con la corrección de errores y la adaptación de los sistemas. Esto es importante porque es generalmente considerada como la parte más costosa en el desarrollo de software.

- **Mantenibilidad (Facilidad de mantenimiento):** Esfuerzo necesario para localizar y arreglar un error del programa (definición limitada).
- **Flexibilidad:** Esfuerzo necesario para modificar un programa operativo.
- **Facilidad de prueba:** Esfuerzo necesario para probar un programa para asegurarse que realiza su función pretendida.

Transición del Producto

Puede que no sea muy importante en todas las aplicaciones. Sin embargo, la orientación en cuanto a procesamiento distribuido y el rápido cambio en el hardware es probable que incremente su importancia.

- **Portabilidad:** Esfuerzo necesario para transferir el programa de un entorno de sistema de hardware y/o software a otro.
- **Reusabilidad (capacidad de reutilización):** Hasta dónde se vuelve a emplear un programa o partes de un programa en otras aplicaciones, en relación al empaquetamiento y alcance de las funciones que realiza el programa.
- **Interoperabilidad:** Esfuerzo necesario para acoplar un sistema con otro.

Figura 1. Los tres ejes o puntos de vista de MacCall

1.1.1.2. Modelo de Boehm

Fue creado por Barry Boehm en 1978, este modelo introduce características de alto nivel, características de nivel intermedio y características primitivas, cada una de las cuales contribuye al nivel general de la calidad.

Características de Alto Nivel

Representan requerimientos generales de uso, pueden ser:

- **Utilidad per-se:** cuan usable, confiable, eficiente es el producto en sí mismo.
- **Mantenibilidad:** cuán fácil es modificarlo, entenderlo y resetearlo.
- **Utilidad general:** si puede seguir usándose si se cambia el ambiente.

Características de Nivel Intermedio

Representan los factores de calidad de Boehm:

- Portabilidad (utilidad general)
- Confiabilidad (utilidad per-se)
- Eficiencia (utilidad per-se)
- Usabilidad (utilidad per-se)
- Testeabilidad (mantenibilidad)
- Facilidad de entendimiento (mantenibilidad)
- Modificalidad o Flexibilidad (mantenibilidad)

Características Primitivas

El nivel más bajo corresponde a características directamente asociadas a una o dos métricas de calidad.

De portabilidad:

- consistencia
- estructuración
- concisidad
- legibilidad

De confiabilidad:

- auto-contención
- exactitud
- completitud
- consistencia
- robustez/integridad

De eficiencia:

- accesibilidad
- eficiencia de uso de dispositivos

De usabilidad:

- robustez/integridad
- accesibilidad
- comunicación

De testeabilidad:

- comunicación
- auto descripción
- estructuración

De entendibilidad:

- consistencia
- estructuración
- concisidad
- legibilidad

De modificalidad:

- estructuración
- aumentabilidad

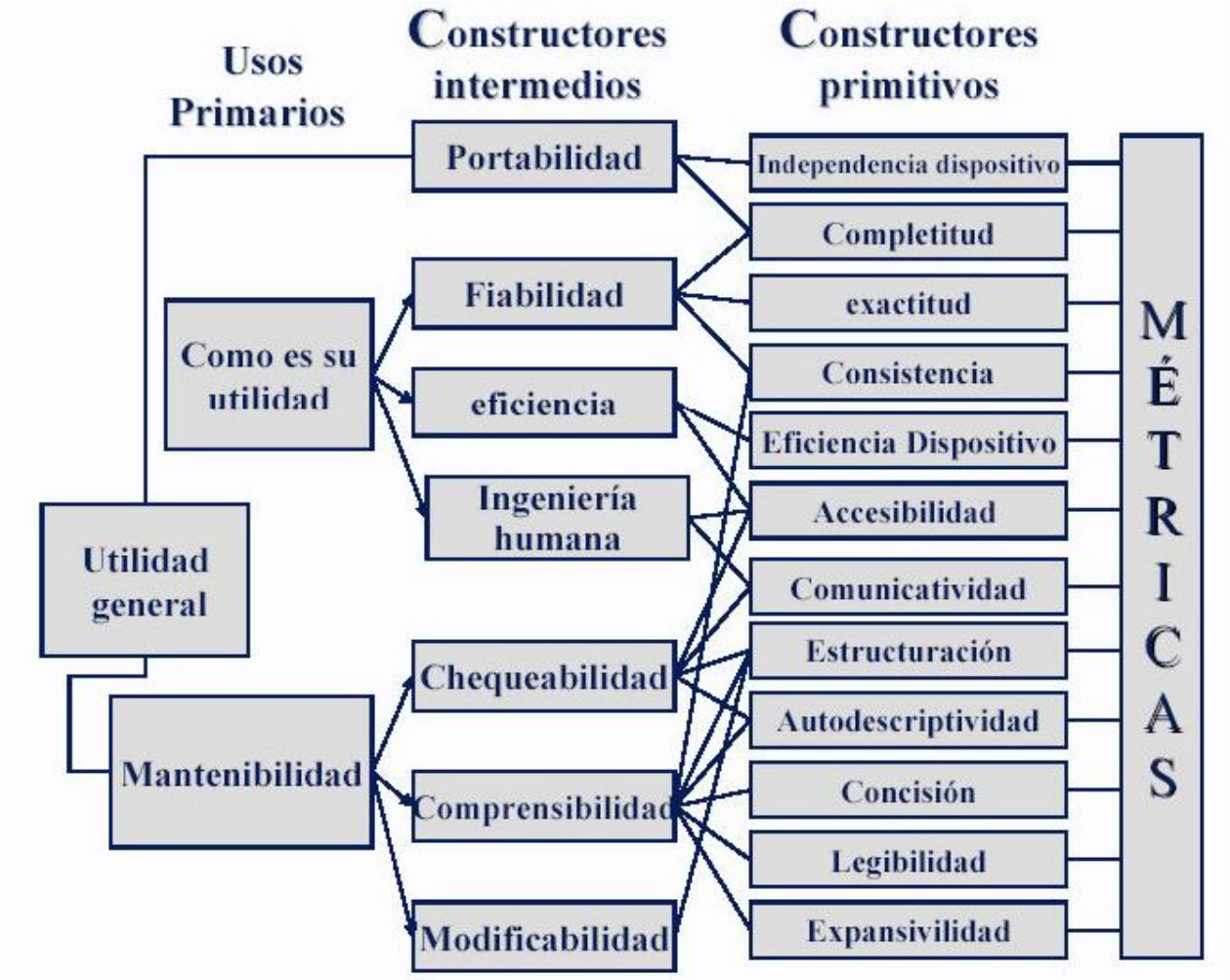


Figura 2. Modelo de Boehm

1.1.1.3. Características de Calidad del Estándar ISO 9126

Este modelo surge como respuesta a la necesidad de establecer criterios sólidos de evaluación del software en virtud del incremento de la calidad del mismo. Para ello se parte de la identificación y evaluación de cinco atributos básicos por parte del usuario final como son: Comprensibilidad, Facilidad de Aprendizaje, Atractividad, Operabilidad y Conformidad. (3)

Es un estándar internacional para la evaluación del Software. Está supervisado por el proyecto SQuaRE, ISO 25000:2005, el cual sigue los mismos conceptos.

El estándar está dividido en cuatro partes las cuales dirigen, respectivamente, lo siguiente: modelo de calidad, métricas externas, métricas internas y calidad en las métricas de uso.

El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas. La segunda parte del estándar define las métricas externas necesarias para medir con respecto a cada una de las características y subcaracterísticas agrupadas en la primera parte. La tercera parte del estándar define las métricas internas necesarias para estimar las características de calidad de un software que se encuentre en ejecución o desarrollo. Por último, la cuarta parte, define las métricas para establecer la calidad en uso de un producto software. Los atributos de calidad se clasifican según seis características:

- Funcionalidad
- Confiabilidad
- Eficiencia
- Facilidad de uso
- Facilidad de mantenimiento
- Portabilidad

1.1.1.3.1 Portabilidad

La portabilidad de un software se define como, el grado de dependencia de la plataforma en la que corre el software. La portabilidad es mayor cuanto menor es el grado de dependencia entre el software y la plataforma. Si un software puede ser compilado en plataformas diversas (x86, iA64, amd64, entre otras.), entonces se dice que el software es multiplataforma.

En algunos casos el software es “independiente” de la plataforma y puede ejecutarse en plataformas diversas sin necesidad de ser compilado específicamente para cada una de ellas , a este tipo de software se le llama interpretado , porque necesita de un intérprete para ser ejecutado en las diferentes plataformas. La capacidad que tiene un producto de software para ser transferido de un ambiente a otro. Está compuesta por las siguientes subcaracterísticas:

- Adaptabilidad
- Instalabilidad
- Coexistencia
- Reemplazabilidad
- Conformidad de la portabilidad

La portabilidad implica que un programa puede correr en distintas plataformas, no sólo de sistemas operativos sino diferentes versiones, diferentes ambientes y esquemas. De esta se realizará un estudio más profundo de dos de sus principales subcaracterísticas.

¿Cómo Medir la Portabilidad?

El grado de portabilidad puede ser medido a través de la siguiente fórmula:

DP (su) = 1 - (Cport (su, e2) / Crdev (req, e2)).

Donde DP es el grado de portabilidad por unidad de software, Cport es el costo de la portabilidad por unidad de software en el nuevo ambiente y Crdev es el costo de volver a desarrollar el sistema con los mismos requerimientos en el nuevo ambiente.

Donde si el valor de DP es mayor que 0 el costo de portabilidad es más factible que el de volver a desarrollar el software para la nueva plataforma. Más aún, el costo de portabilidad es inversamente

proporcional al grado de portabilidad. El valor del grado de la portabilidad si es 1 representa la perfecta portabilidad (4).

Conformidad de la Portabilidad

- La capacidad del software para adherirse a estándares o convenciones relacionados a la portabilidad.

Reemplazabilidad

- La capacidad del producto de software para ser utilizado en lugar de otro producto de software, para el mismo propósito y en el mismo entorno. Por ejemplo, la reemplazabilidad de una nueva versión de un producto de software es importante para el usuario cuando dicho producto de software es actualizado (actualizaciones, upgrades). Se utiliza en lugar de compatibilidad de manera que se evitan posibles ambigüedades con la interoperabilidad. Puede incluir atributos de ambos, inestabilidad y adaptabilidad. El concepto ha sido introducido como una sub característica por sí misma, dada su importancia.

1.1.1.4. Características de la Norma ISO 12207

Resulta fundamental entender el mantenimiento como uno de los procesos principales dentro del contexto del ciclo de vida del software, para lo que conviene examinar los principales estándares internacionales relativos a este tema, y en particular, la norma ISO 12207.

Este estándar "establece un marco de referencia común para los procesos del ciclo de vida software, con una terminología bien definida, que puede ser referenciada por la industria software".

La estructura del estándar ha sido concebida de manera flexible y modular de manera que pueda ser adaptada a las necesidades de cualquiera que lo use. Para conseguirlo, el estándar se basa en dos principios fundamentales: Modularidad y responsabilidad. Con la modularidad se pretende conseguir procesos con un mínimo acoplamiento y una máxima cohesión. En cuanto a la responsabilidad, se busca establecer un responsable para cada proceso, facilitando la aplicación del estándar en proyectos en los que pueden existir distintas personas u organizaciones involucradas.

Los procesos se clasifican en tres tipos: Principales, de soporte y de la organización. Los procesos de soporte y de organización deben existir independientemente de la organización y del proyecto ejecutado. Los procesos principales se instancian de acuerdo con la situación particular.

Procesos Principales:

- Adquisición
- Suministro
- Desarrollo
- Operación
- Mantenimiento

Procesos de Soporte:

- Documentación
- Gestión de la configuración
- Aseguramiento de calidad
- Verificación
- Validación

- Revisión conjunta
- Auditoría
- Resolución de problemas

Procesos de la Organización:

- Gestión
- Infraestructura
- Mejora
- Recursos humanos

1.2. La Inteligencia de Negocio como estrategia para la toma de decisiones

En la actualidad en cualquier organización se hace necesario la toma decisiones, en ocasiones lo suficientemente complejas, necesitándose una buena estrategia para lograr un desarrollo satisfactorio. Generalmente estas decisiones, están basadas en enormes volúmenes de información registrada en bases de datos operacionales o de otros tipos de fuentes de datos. La recopilación y análisis de esta información, dado su carácter heterogéneo y su volumen se convierten usualmente en un problema para las organizaciones y es aquí donde interviene la Inteligencia de Negocio, mediante los Sistemas de Apoyo a la Toma de Decisiones.

Se define por **Inteligencia de Negocio o *Business Intelligence* (BI)** el conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de los datos existentes en una organización o empresa. Este conjunto de herramientas y estrategias tienen en común las siguientes características: (5)

- **Accesibilidad a la información:** Los datos son la fuente principal. Lo primero que debe garantizar este tipo de herramientas o técnicas será el acceso de los usuarios a los datos con independencia de su procedencia.
- **Apoyo en la toma de decisiones:** Se busca ir más allá en la presentación de la información, de manera que los usuarios tengan acceso a herramientas de análisis que les permitan seleccionar y manipular sólo aquellos datos que les interesen.
- **Orientación al usuario final:** Se busca independencia entre los conocimientos técnicos de los usuarios y su capacidad para utilizar estas herramientas.

Para que una organización sea competitiva, las personas que toman las decisiones necesitan acceder de forma rápida y fácil, a la información, y esto se realiza por medio de los Sistemas de Apoyo a la Toma de Decisiones, o en inglés: ***Decision Support Systems*** (DSS).

Dichos sistemas se basan en crear modelos informáticos del negocio de modo que este pueda funcionar más eficientemente; y permitiéndole a la dirección de la organización tomar decisiones semiestructuradas o no estructuradas, mediante la combinación de datos y modelos de análisis. Su propósito es ayudar a la dirección para que “marque tendencias, señale problemas y tome decisiones inteligentes” (6).

Como sistema informático, los DSS consisten habitualmente en varios componentes: bases de datos fuentes, sistemas de Extracción-Transformación-Carga de datos, Data Warehouse (DW) o Almacenes de Datos, herramientas de Procesamiento Analítico en Línea (OLAP), bases de datos multidimensionales y otras herramientas de análisis de información (7).

Tomar decisiones implica aceptar un riesgo, lo que es indudable es que el objetivo es minimizarlo. Aquí es donde intervienen las herramientas de Inteligencia de Negocio. Estas son las encargadas de transformar los datos del sistema en información útil para la toma de decisiones empresariales. La organización que carezca de estos sistemas tendría que realizar la transformación y análisis de la información manualmente y solo por el factor tiempo la solución no sería factible, sin contar con que el análisis de los datos sería superficial y por tanto, se perdería información valiosa.

Los datos en un proceso de toma de decisiones, evolucionan por varias etapas. Se comienza con la extracción, limpieza e integración de los datos provenientes de diversas fuentes y se transita por diversos procesos de análisis y búsqueda de conocimiento (Figura 3).

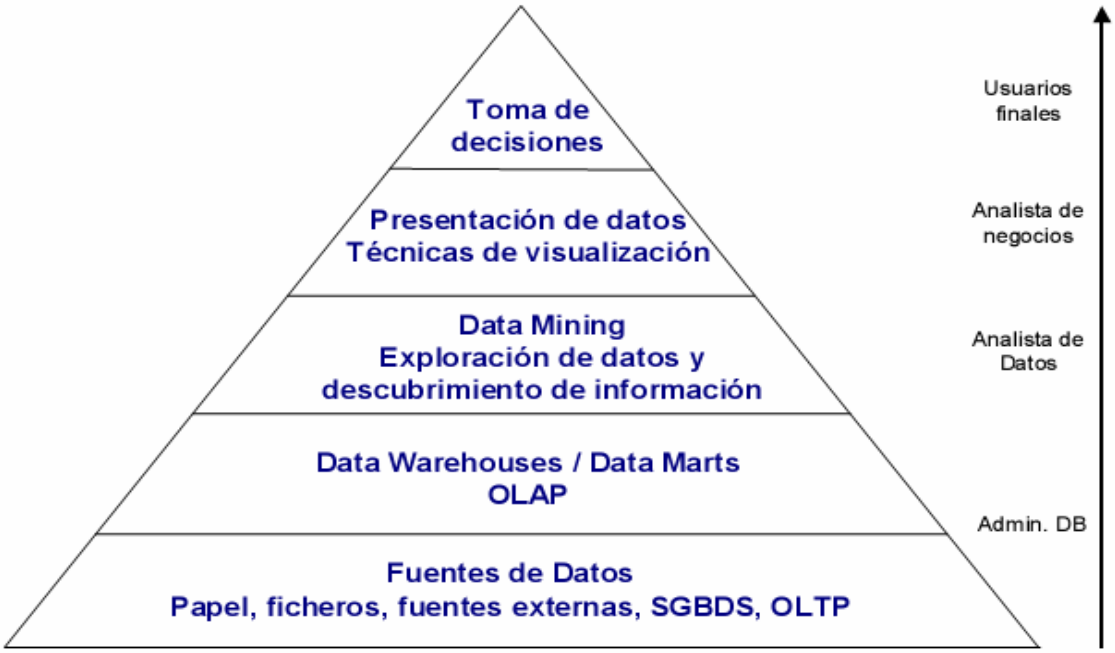


Figura 3. Etapas por las que transitan los datos en un proceso de toma de decisiones

Tipos de Sistemas de Información:

- **Sistema Táctico:** Están diseñados para soportar las tareas de coordinación de actividades y manejo de documentación, definidos para facilitar consultas sobre la información almacenada en el sistema, proporcionar informes; en resumen, facilitar la gestión independiente de la información en los diferentes niveles de la organización.

- **Sistema Técnico-Operativo:** Son los que cubren la parte fundamental de las operaciones tradicionales de captura masiva de datos (data entry) y servicios básicos de tratamiento de datos, con tareas predefinidas. Son los llamados Sistemas Operacionales.

- **Sistema Global:** Este nivel de sistemas de información está surgiendo recientemente, y es consecuencia del desarrollo organizacional orientado a un mercado de carácter global, el cual obliga a pensar e implementar sistemas de comunicación más estrechos entre la organización y el mercado. Se convierte en el vehículo de comunicación entre la organización y el mercado con alcance nacional y global.

- **Sistema Estratégico:** Estos sistemas facilitan la labor de la dirección, proporcionándole un soporte básico, mostrando la información de forma apropiada para la toma de decisiones. El objetivo fundamental es generar información que pueda ser trabajada y analizada de forma intuitiva en tiempo real, y con la posibilidad de integrar diferentes fuentes de datos para ofrecer una visión global que pueda ser compartida y distribuida (5).

Los sistemas de Inteligencia de Negocio tienen características muy peculiares que los diferencian de los sistemas operacionales. Estos están optimizados para consultar y divulgar la información recogida. Esto significa que los datos están desnormalizados para apoyar preguntas de alto rendimiento, soportados sobre nuevas estructuras de análisis (básicamente multidimensionales). Mientras que, en los sistemas operacionales los datos se normalizan completamente y se basan por lo general en modelos relacionales. Los procesos de ETL (Extracción, Tratamiento y Limpieza de los datos) son los encargados de traducir de un tipo de sistema a otro.

El apoyo para la toma de decisiones, no es parte de la tecnología de base datos por sí misma, sino que resulta de la combinación de varias aplicaciones de esta técnica (Figura 4). Las aplicaciones reciben los nombres de Data Warehouse, Datamart o Mercado de datos, OLAP (Procesamiento Analítico en Línea), Minería de Datos, entre otros (6).

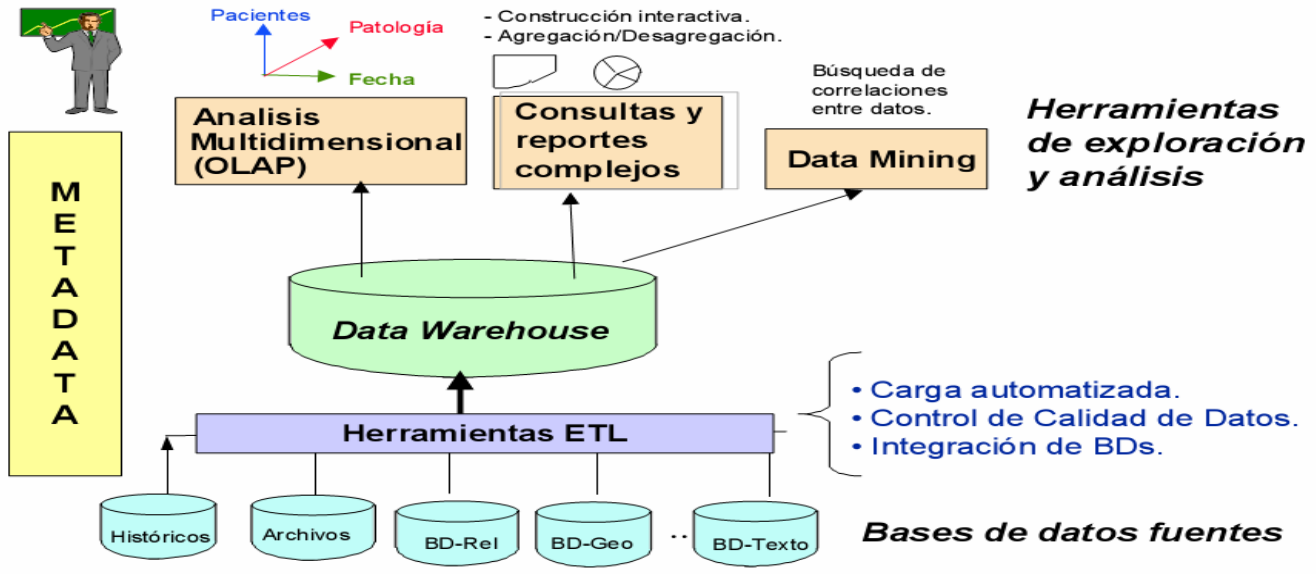


Figura 4. Sistema de Apoyo a la Toma de Decisiones.

1.2.1. Data Warehouse

La primera condición para poder analizar y extraer información valiosa de un conjunto determinado de datos es disponer de ellos. En algunos casos esto puede parecer trivial pero en ocasiones la recopilación de la información y su tratamiento en el tiempo pueden convertirse en procesos complejos, donde se debe decidir, entre otras cosas, de qué fuentes internas o externas se obtendrán los datos y cómo deben organizarse (8).

Generalmente, las organizaciones tienen almacenada la información diaria en sistemas operacionales, los que están diseñados para satisfacer determinados requerimientos en el funcionamiento cotidiano de la organización, con tareas muy bien definidas y una carga de procesamiento determinada. Sin embargo, resulta ineficiente utilizar estos sistemas para la toma de decisiones, debido principalmente a su diseño físico y a la imposibilidad de dar respuesta a las complejas consultas que dicho proceso conlleva. Estas divergencias hacen que sea inadecuado solapar en un mismo sistema el procesamiento operacional con la toma de decisiones de la empresa, fundamentalmente por restricciones de rendimiento, planeación del almacenamiento y administración de recursos; dando paso al surgimiento de los Almacenes de Datos o *Data Warehouse* (DW). (6)

Los "almacenes de datos" son una tecnología relativamente reciente, encaminada a proporcionar metodologías para recopilar e integrar los datos históricos de una organización, cuyo fin es el análisis, la obtención de resúmenes e informes complejos y la extracción de conocimiento. Esta tecnología está diseñada especialmente para organizar grandes volúmenes de datos de procedencia generalmente estructurada (por ejemplo bases de datos relacionales).

Existen dos paradigmas en el campo del *Data Warehousing*, el de Bill Inmon y el de Ralph Kimball, ambos conocidos como los padres del Data Warehouse.

- **El paradigma de Inmon:** La tecnología Data Warehouse, forma parte de los sistemas de inteligencia de negocio. Una empresa debe tener un Data Warehouse y varios Data Marts que se nutran de la información del Data Warehouse. En un Data Warehouse la información puede estar almacenada en 3ra Forma Normal.

- **El paradigma de Kimball:** Es la unión de todos los Datamarts de las diferentes áreas de una empresa. La información se almacena siguiendo un modelo dimensional (9).

Ambos paradigmas son válidos pero se considera al de Ralph Kimball como el más ajustado a la evolución de esta tecnología dado que la mayoría de las organizaciones por diversos motivos, casi siempre tiempo y costo de producción, comienzan por la implementación de varios Data Marts que

posteriormente se integran en un Data Warehouse, y el modelo dimensional se ha convertido en un patrón de diseño muy difundido en esta tecnología.

La siguiente tabla describe las diferencias que se deben tener en cuenta para estructurar y diseñar almacenes de datos en comparación con las bases de datos transaccionales (10).

Tabla 1. Diferencias entre las bases de datos transaccionales y los almacenes de datos.

Parámetros	Base de Datos Transaccional	Almacén de Datos
Propósito	Operaciones diarias. Soporte a las aplicaciones.	Recuperación de información, informes, análisis y minería de datos.
Tipo de datos	Datos de funcionamiento de la organización.	Datos útiles para el análisis, la sumarización, etc.
Características de los datos	Datos de funcionamiento, cambiantes, internos, incompletos.	Datos históricos, datos internos y externos, datos descriptivos.
Modelo de datos	Datos normalizados.	Datos en estrella, en copo de nieve, parcialmente desnormalizados, multidimensionales.
Número y tipo de usuarios	Cientos/miles: aplicaciones, operarios, administrador de la base de datos.	Decenas: directores, ejecutivos, analistas.
Acceso	SQL. Lectura y escritura.	SQL y herramientas propias (slice & dice, drill, roll, pivot). Lectura.

Los Data Warehouse poseen una estructura de la información con un almacenamiento homogéneo y fiable, basado en la consulta y en el tratamiento jerárquico de los datos.

Esta tecnología está caracterizada por 4 aspectos fundamentales:

- **La organización de la información es temática:** los datos están agrupados por temas para facilitar su acceso y entendimiento al usuario final.
- **Integración consistente de los datos:** todas las inconsistencias (nomenclaturas, formatos y tipos de datos, etc.) deben de ser eliminadas.
- **Variante en el tiempo:** la información es exacta para un momento del tiempo, por tal motivo, no deben realizar actualizaciones salvo en raras ocasiones. Mantienen un registro histórico de la información.
- **La información es no volátil:** la información existe para ser leída y no modificada, por lo que, cuando se habla de actualización se está haciendo referencia a una carga incremental sin realizar ningún tipo de acción sobre la información que ya existía (10).

La organización y el mantenimiento de la información en un Almacén de Datos requiere tomar en cuenta cuestiones técnicas referentes a su ciclo de vida como: selección de esquemas de diseño, definición de los procesos de carga, ciclos de mantenimiento y definición de reglas para preservar la consistencia de los datos. Un DW pasa a ser un integrador o recopilador de información de diferentes fuentes.

1.2.1.1. Datamarts

La tecnología Data Warehouse se caracteriza por poseer un enorme poder de extracción de información, y está preferentemente orientada al perfil de toda una organización. En ocasiones se hace necesario aplicar esta técnica a negocios más pequeños que no cuentan con tanta información y la implementación de un DW sería costosa e innecesaria. Por otra parte en algunas organizaciones que ya cuentan con DW surge la necesidad de hacer extracciones de datos de forma departamental y sin mucho gasto de recursos, que solo involucran un subconjunto de la información del DW, por lo que surge la idea de construir un almacén

más limitado que facilite el acceso y esté sincronizado con el DW. Esta problemática trajo consigo que surgiera la idea de los Datamarts.

Existen dos definiciones fundamentales del término Datamart:

- Es un almacén de datos especializados en un área específica, orientado a un tema, integrado y volátil. El término volátil significa que los datos pueden ser actualizados e incluso borrados.
- Es un almacén de datos departamental, o sea, orientado a un sector dentro de la organización.

Los Datamarts al poseer un propósito más específico y orientado a un sector determinado gozan de innumerables ventajas entre las que se encuentran, su bajo costo de realización y operación, el corto tiempo de diseño y su rapidez para satisfacer los pedidos de información ya que manejan datos resumidos (11).

1.2.1.2. Procesos de Extracción, Transformación y Carga de Datos (ETL)

La recopilación de los datos que poblarán los almacenes de datos debe ir acompañada de una limpieza, transformación e integración de los mismos, para que estos estén en condiciones de someterse al análisis y aporten resultados válidos. La validez de los resultados en el proceso de extracción de conocimiento depende en gran medida de la calidad de los datos analizados, debido a que estos constituyen la materia prima. Para poseer buena calidad en los datos estos deben estar completos, íntegros y consistentes.

Los procesos ETL (*Extraction, Transformation, Load*) son los encargados de ejecutar periódicamente tareas que extraen los datos, los transforman, los integran y los carga al Data Warehouse. Las fuentes de datos que pueden estar vinculadas suelen ser muy diversas, entre estas se pueden encontrar: bases de datos relacionales, ficheros textos, ficheros binarios, ficheros XML (*Extensible Markup Language*), datamarts, hojas de cálculo, entre otros.

Los procesos ETL se encargan de realizar las siguientes tareas:

- Lectura de las fuentes de datos, ya sean bases de datos operacionales o fuentes externas (extracción): se trata, generalmente, de obtener los datos mediante consultas SQL sobre bases de datos operacionales. Esto se realiza preferiblemente en horario de poca carga transaccional. En caso de otro tipo de fuente se utilizan herramientas que generan XML o tablas de bases de datos que puedan integrarse en el almacén de datos.

- Creación de claves: se recomienda crear nuevas claves primarias.

- Integración de los datos: al obtener datos de fuentes heterogéneas es necesario definir las restricciones para conectarlos.

- Limpieza y transformación: se trata de evitar datos redundantes e inconsistentes, estandarizar medidas, formatos, fechas, tratar valores nulos, etc.

- Creación y mantenimiento de metadatos: en todo proceso de ETL es necesario guardar la información del propio proceso, así como las descripciones de las tareas a desarrollar, cuáles se deben realizar y en qué momento.

- Identificación de cambios: se puede realizar por carga total, comparación de instancias, análisis de archivos log, marcas de tiempo, empleo de disparadores o técnicas mixtas.

- Planificación de carga y mantenimiento.

- Prueba de calidad: se realiza mediante la aplicación de métricas de calidad a los datos almacenados.

- Carga de datos: es la tarea final del proceso, donde se insertan los datos en el almacén. En este momento se recomienda que el almacén deje de funcionar, por lo que, podría usarse un almacén espejo.

1.2.1.3. Modelo Multidimensional

El modelo de datos más extendido para la implementación de almacenes de datos es el multidimensional donde los datos se organizan en torno a los hechos (datos históricos) y tienen atributos o medidas que son detalladas en las dimensiones (10).

Las relaciones entre las tablas de hechos y las dimensiones están representadas en la llave primaria de la tabla de hechos, la que está compuesta por las llaves de todas las tablas de dimensiones con las que tiene relación (llaves foráneas).

Dadas las características de los almacenes de datos es ideal la utilización en su diseño de un Modelo Multidimensional (MMD). Este tipo de diseño tiene como ventajas para la toma de decisiones, sobre el Modelo Entidad-Relación (MER), ya que es muy flexible, está desnormalizado y orientado a los intereses de un usuario final, aunque esto no significa que existan inconsistencias en los datos. Mediante la utilización de un MMD se disminuyen la cantidad de tablas y relaciones entre ellas, lo que agiliza el acceso a los datos (5).

- **Tablas de Hechos:** Representan la ocurrencia de un determinado proceso dentro de la organización y no tienen relación entre sí. Generalmente, almacenan medidas numéricas, las que representan valores de las dimensiones, aunque en ocasiones estas no están presentes y se les denominan “tablas de hechos sin hechos”.

- **Tablas de Dimensiones:** Contienen, generalmente, una llave simple y atributos que la describen. En dependencia del esquema de diseño que se asuma pueden contener llaves foráneas de otras tablas de dimensión. Existe una dimensión fundamental en todo DW, *la dimensión tiempo*. Esto ocurre porque todo registro que se incluya constituye la ocurrencia de un fenómeno en un instante de tiempo definido. Dicha dimensión es la que establece uno de los objetivos fundamentales de la construcción de un DW, la conservación de un “histórico”. Para la implementación de un MMD existen varios esquemas o patrones de diseño, pero los más utilizados son los siguientes:

- **Esquema de Estrella:** La tabla de hechos está en el centro de la estrella y están relacionadas con ella de forma radial todas las tablas de dimensiones, las cuales no se relacionan entre sí. No existen caminos alternativos en las dimensiones.

- **Esquema de Copo de Nieve:** Es parecido al de estrella pero existen jerarquías en las dimensiones. Las tablas de dimensiones pueden estar relacionadas, o sea, existen caminos alternativos en ellas (6).

1.2.2. Procesamiento Transaccional en Línea y Procesamiento Analítico en Línea (OLTP y OLAP)

El desarrollo de los sistemas de información sustentados sobre bases de datos, ha traído como consecuencia la proliferación de herramientas de consultas cada vez más complejas. Por tanto, es necesario distinguir los diferentes tipos de procesamiento existentes: el procesamiento transaccional y el procesamiento analítico.

- OLTP (*On-Line Transactional Processing* o Procesamiento Transaccional en Línea), trabajo primario en un sistema de información. Consiste en realizar transacciones, es decir, actualizaciones y consultas a la base de datos con un objetivo operacional: hacer funcionar las aplicaciones de la organización, proporcionar información sobre el estado del sistema y permitir actualizarlas.

- OLAP (*On-Line Analytical Processsing* o Procesamiento Analítico en Línea), reúne un gran número de operaciones (solamente de consulta), en las que se cruzan gran cantidad de información con el objetivo final de crear informes y resúmenes que sean útiles en la toma de decisiones. Los algoritmos que utiliza están implementados para optimizar los tiempos de respuesta a las consultas, logrando eficiencia y almacenando los datos en estructuras especializadas.

OLAP fue creado bajo las siguientes ideas:

- **Lograr rapidez de respuesta:** entregar la información a los usuarios finales en el menor tiempo posible, de 0 a 5 segundos.

- **Posibilitar el análisis:** Ofrecer análisis numérico y estadístico de los datos, con valores agregados. Esto permite analizar tendencias, causas, detectar variables de interés y descender hasta los niveles más bajos de la información, lo que se complementa con la ayuda de los motores de reportes y gráficos que se incluyen. También incluye vistas personalizadas.

- **Compartir Datos:** Incluye los mecanismos de seguridad necesarios para compartir la información entre los usuarios que se definan.

- **Basado en un Estructura Multidimensional:** Haciendo sencilla la selección y navegación de los datos.

- **Recuperación de Información:** Acceso a los datos y recuperación de información valiosa (solo lectura) para las diferentes aplicaciones clientes.

Existen tres tipos de OLAP:

- **ROLAP (*Relational OLAP*)**: se construye el almacén de datos directamente sobre un gestor de bases de datos relacional, todas las tablas (hechos y dimensiones) son almacenadas en tablas relacionales.

- **MOLAP (*Multidimensional OLAP*)**: es la forma clásica de OLAP. Se construye el almacén de datos directamente sobre estructuras matriciales multidimensionales. Se almacenan las agregaciones y una copia de los datos bases. Una vez realizada la carga, el motor de MOLAP se encarga de brindar la información detallada y agregada. Solo se requiere la participación del servidor de bases de datos cuando se vuelvan a procesar los datos.

- **HOLAP (*Hybrid OLAP*)**: usa tablas relacionales para almacenar la información base y estructuras multidimensionales para las agregaciones, o sea, es una combinación de los dos anteriores. Cada uno de los tipos de OLAP tiene beneficios en dependencia del problema en que se aplique. MOLAP requiere de menor espacio de almacenamiento y es más rápido calculando las agregaciones y devolviendo las respuestas, aunque se recomienda emplear para pequeños volúmenes de datos. ROLAP es considerado el más escalable, pero es más lento en el

preprocesamiento y rendimiento de las consultas. HOLAP es rápido en el preprocesamiento y rendimiento de las consultas, aunque más lento que MOLAP y es escalable. HOLAP es ideal para grandes fuentes de datos.

OLTP y OLAP son dos tipos de procesamiento bien distintos, aunque ambos se realizan en tiempo real. Hasta hace poco tiempo en las organizaciones ambos se realizaban sobre las mismas bases de datos transaccionales, lo que traía consigo grandes problemas, debido a que las complejas consultas OLAP que involucraban muchas tablas y agregaciones perturbaban el trabajo de estos sistemas, causando demoras innecesarias y consumiendo muchos recursos. Además, los diseños que se asumen en las bases de datos transaccionales no son los más adecuados para realizar consultas OLAP. Actualmente, esto dejó de ser un problema debido al surgimiento de los DW (10).

1.3 Estilos y Patrones

Bosch (2000) establecen que la imposición de ciertos estilos arquitectónicos mejora o disminuye las posibilidades de satisfacción de ciertos atributos de calidad del sistema (12). Con esto afirman que cada estilo propicia atributos de calidad, y la decisión de implementar alguno de los existentes depende de los requerimientos de calidad del sistema. De manera similar, plantean el uso de los patrones arquitectónicos y los patrones de diseño para mejorar la calidad del sistema. Se afirma que un criterio importante del éxito de los patrones – tanto arquitectónico como de diseño - es la forma en que estos alcanzan de manera satisfactoria los objetivos de la ingeniería de software (13). Los patrones soportan el desarrollo, mantenimiento y evolución de sistemas complejos y de gran escala.

1.3.1 Estilo Arquitectónico

Shaw y Garlan (1996) definen estilo arquitectónico como una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes (14).

Buschmann (1996) definen estilo arquitectónico como una familia de sistemas de software en términos de su organización estructural. Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Así mismo, se considera como un tipo particular de estructura fundamental para un sistema de software, conjuntamente con un método asociado que especifica cómo construirlo. Éste incluye información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación (13).

La tabla 2 resume los principales estilos arquitectónicos, los atributos de calidad que propician y los atributos que se ven afectados negativamente (15). Tabla 2. Estilos Arquitectónicos y Atributos de Calidad.

Tabla 2: Estilos Arquitectónicos y Atributos de Calidad

Estilo	Descripción	Atributos Asociados	Atributos en Conflicto
Datos Centralizados	Sistemas en los cuales cierto número de clientes accede y actualiza datos compartidos de un repositorio de manera frecuente.	• Integrabilidad • Escalabilidad • Modificabilidad	Desempeño
Flujo de Datos	El sistema es visto como una serie de transformaciones sobre piezas sucesivas de datos de entrada. El dato ingresa en el sistema, y fluye entre los componentes, de uno en uno, hasta que se le asigne un destino final (salida o repositorio).	• Reusabilidad • Modificabilidad • Mantenibilidad	Desempeño

Máquinas Virtuales	Simulan alguna funcionalidad que no es nativa al hardware o software sobre el que está implementado.	• Portabilidad	Desempeño
Llamada y Retorno	El sistema se constituye de un programa principal que tiene el control del sistema y varios subprogramas que se comunican con éste mediante el uso de llamadas.	• Modificabilidad • Escalabilidad • Desempeño	Mantenibilidad Desempeño
Componentes Independientes	Consiste en un número de procesos u objetos independientes que se comunican a través de mensajes.	• Modificabilidad • Escalabilidad	Desempeño Integrabilidad

1.3.2. Patrón Arquitectónico

En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede utilizarse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas[1].

Un patrón es una regla que consta de tres partes, la cual expresa una relación entre un contexto, un problema y una solución. En líneas generales, un patrón sigue el siguiente esquema: (13)

- **Contexto:** Es una situación de diseño en la que aparece un problema de diseño.
- **Problema:** Es un conjunto de fuerzas que aparecen repetidamente en el contexto.
- **Solución:** Es una configuración que equilibra estas fuerzas.

[1] La notación formal de los patrones nació con los patrones arquitectónicos de Christopher Alexander. En los años 80, Kent Beck y Ward Cunningham hicieron su aplicación al software.

El punto solución del esquema de un patrón abarca:

Estructura con componentes y relaciones

Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos, etc.

La colección de patrones arquitectónicos debe ser estudiada en términos de factores de calidad e intereses, en anticipación a su uso. Esto quiere decir que un patrón puede ser analizado previamente, con la intención de seleccionar el que mejor se adapte a los requerimientos de calidad que debe cumplir el sistema. La tabla 3 presenta algunos patrones arquitectónicos, además de los atributos que propician y los atributos en conflicto.

Tabla 3. Patrones Arquitectónicos y Atributos de Calidad

Patrón Arquitectónico	Descripción	Atributos asociados	Atributos en conflicto
Layers	Consiste en estructurar aplicaciones que pueden ser descompuestas en grupos de subtareas, las cuales se clasifican de acuerdo con un nivel particular de abstracción.	• Reusabilidad • Portabilidad • Facilidad de Prueba	Desempeño Mantenibilidad
Pipes and Filters	Provee una estructura para los sistemas que procesan un flujo de datos. Cada paso de procesamiento está encapsulado en un componente filtro (filter). El dato pasa a través de conexiones (*pipes*),	• Reusabilidad	

	entre filtros adyacentes.	• Mantenibilidad	Desempeño
Blackboard	Aplica para problemas cuya solución utiliza estrategias no determinísticas. Varios subsistemas ensamblan su conocimiento para construir una posible solución parcial ó aproximada.	• Modificabilidad • Mantenibilidad • Reusabilidad • Integridad	Desempeño Facilidad de Prueba
Broker	Puede ser usado para estructurar sistemas de software distribuido con componentes desacoplados que interactúan por invocaciones a servicios remotos. Un componente *broker* es responsable de coordinar la comunicación, como el reenvío de solicitudes, así como también la transmisión de resultados y excepciones.	• Modificabilidad • Portabilidad • Reusabilidad • Escalabilidad • Interoperabilidad	Desempeño
Presentation- Abstraction- Control	Define una estructura para sistemas de software interactivos de agentes de cooperación organizados de forma jerárquica. Cada agente es responsable de un aspecto específico de la funcionalidad de la aplicación y consiste de tres componentes: presentación, abstracción y control.	• Modificabilidad • Escalabilidad • Integrabilidad	Desempeño Mantenibilidad
	Aplica para sistemas de software que deben estar en capacidad de adaptar los		

Microkernel	requerimientos de cambio del sistema. Separa un núcleo funcional mínimo del resto de la funcionalidad y de partes específicas pertenecientes al cliente.	• Portabilidad • Escalabilidad • Confiablidad • Disponibilidad	Desempeño
Reflection	Provee un mecanismo para sistemas cuya estructura y comportamiento cambia dinámicamente. Soporta la modificación de aspectos fundamentales como estructuras tipo y mecanismos de llamadas a funciones.	• Modificabilidad	Desempeño

1.3.3. Patrón de Diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular.

Son menores en escala que los patrones arquitectónicos, y tienden a ser independientes de los lenguajes y paradigmas de programación. Su aplicación no tiene efectos en la estructura fundamental del sistema, pero sí sobre la de un subsistema, debido a que especifica a un mayor nivel de detalle, sin llegar a la implementación, el comportamiento de los componentes del subsistema. La tabla 4 presenta algunos patrones de diseño, junto a los atributos de calidad que propician y los atributos que entran en conflicto con la aplicación del patrón (13).

Tabla 4. Patrones de Diseño y Atributos de Calidad

Patrón de Diseño	Descripción	Atributos asociados	Atributos en conflicto
Whole-Part	Ayuda a constituir una colección de objetos que juntos conforman una unidad semántica.	• Reusabilidad • Modificabilidad	Desempeño
Master-Slave	Un componente maestro *(*master*)* distribuye el trabajo a los componentes esclavos *(*slaves*)*. El componente maestro calcula un resultado final a partir de los resultados arrojados por los componentes esclavos.	• Escalabilidad • Desempeño	Portabilidad
Proxy	Los clientes asociados a un componente se comunican con un representante de éste, en lugar del componente en sí mismo.	• Desempeño • Reusabilidad	Desempeño
Command Procesor	Separa las solicitudes de un servicio de su ejecución. Maneja las solicitudes como objetos separados, programa sus ejecuciones y provee servicios adicionales como el almacenamiento de los objetos solicitados, para permitir que el usuario pueda deshacer alguna solicitud.	• Funcionalidad • Modificabilidad • Facilidad de • Prueba	Desempeño
	Ayuda a manejar todas las vistas que provee un sistema de software. Permite a		

View Handler	los clientes abrir, manipular y eliminar vistas. También coordina dependencias entre vistas y organiza su actualización.	• Escalabilidad • Modificabilidad	Desempeño
Forwarder-Receiver	Provee una comunicación transparente entre procesos de un sistema de software con un modelo de interacción punto a punto *(*peer to peer*).*	• Mantenibilidad • Modificabilidad • Desempeño	Configurabilidad
Client-Dispatcher-Server	Introduce una capa intermedia entre clientes y servidores, es el componente despachador *(dispatcher)*. Provee una ubicación transparente por medio de un nombre de servicio, y esconde los detalles del establecimiento de una conexión de comunicación entre clientes y servidores.	• Configurabilidad • Portabilidad • Escalabilidad • Disponibilidad	Desempeño Modificabilidad
Publisher-Subscriber	Ayuda a mantener sincronizados los componentes en cooperación. Para ello, habilita una vía de propagación de cambios: un editor (publisher) notifica a los suscriptores (suscribers) sobre los cambios en su estado.	• Escalabilidad	Desempeño

1.4. Herramientas de Inteligencia de Negocio "Open Source"

La comunidad "*Open Source*" hoy en día ha incursionado prácticamente en todas las áreas de la informática y existen algunas donde su supremacía es indudable. En el área de la Inteligencia de Negocio también se ha producido un despegue en el desarrollo de soluciones.

Una solución basada en Inteligencia de Negocio, de acuerdo con sus características debe de estar soportada por un conjunto de herramientas donde se establece una cooperación entre ellas para transitar por las diferentes etapas del proceso de análisis de los datos, desde la adquisición hasta la visualización de los resultados.

Entre las principales herramientas disponibles están:

- Herramientas ETL: Kettle, Clover, Enhydra Octopus, etc.
- Desarrollo OLAP: Mondrian y JPivot.
- Minería de Datos: WEKA, YALE y otras herramientas con versiones libres limitadas como: Tiberius, WizWhy, CART y See5 / C5.0.
- Motores de Reportes y Gráficos: JFreeReport, BIRT, JasperReport y JFreeChart.
- Entorno de desarrollo para Cuadros de Mando (*Dashboards*): JetSpeed y JBoss Portal.
- Gestores de Bases de Datos: MySQL y PostgreSQL.
- Soluciones completas: Pentaho y SpagoBI.

Herramientas ETL

En la siguiente tabla se muestra el análisis comparativo realizado entre las herramientas más significativas y populares disponibles para realizar procesos de ETL. (16) (17) (18)

Tabla 5. Análisis comparativo entre herramientas ETL

Herramientas	Descripción	Fuentes de Datos	Multiplataforma
Enhydra	Permite crear bases de datos, tablas, índices y relaciones. Permite insertar y actualizar datos de acuerdo con varios criterios, así como generar llaves artificiales.	Conexión mediante JDBC con cualquier bases de datos que posea dicho driver. Ficheros de Texto: CSV (CSV-JDBC), MS-SQL (FreeTDS), XML (XML-JDBC) y archivos de propiedades	Sí (basado en Java)

Octopus	Las transformaciones son descritas en ficheros XML.	(i18n-JDBC).	
Talend	Permite hacer transformaciones y mapeos complejos. Posee un entorno gráfico aunque las transformaciones deben ser programadas íntegramente (Perl), no es muy intuitivo.	Conexión mediante JDBC con cualquier base de datos que posea dicho driver. Gran integración con MySQL.	Sí (basado en Java e integrable con Eclipse)
CloverETL	Las transformaciones son descritas en ficheros XML y ejecutas de forma independiente (un hilo de ejecución para cada una). Permite una amplia variedad de transformaciones.	Conexión mediante JDBC con cualquier base de datos que posea dicho driver como: Oracle, MySQL, MS Excel, etc. Acceso a ficheros como: dBase, FoxPro y FlashFile. Posee componentes para la conexión con LDAP.	Sí (basado en Java)
Kettle	Permite realizar una amplia variedad de transformaciones y posee soporte para el trabajo con almacenes de datos. Posee un entorno gráfico muy intuitivo. Permite el monitoreo de las tareas ejecutadas. Las tareas son ejecutadas en hilos diferentes y su prioridad puede ser administrada.	Puede establecer conexión mediante: JDBC, ODBC y JNDI, con alrededor de 25 tipos de gestores de bases de datos. Permite acceder a ficheros: CSV, XML, de texto personalizado, Excel, etc. Permite conexión con Servicios Web.	Sí (basado en Java)
	Las transformaciones son descritas en ficheros XML. Las tareas son ejecutadas en hilos diferentes permitiendo el paralelismo en su	Conexión mediante JDBC con cualquier base de datos que posea dicho driver.	Sí (basado en Java)

KETL	realización.		
OpenAdaptor	La plataforma es altamente integrable y sus componentes pueden utilizarse de forma independiente. Las transformaciones son descritas en ficheros XML.	Conexión mediante JDBC con cualquier base de datos que posea dicho driver. Acceso a ficheros XML.	Sí (basado en Java)
Scriptella	Las transformaciones son descritas en ficheros XML. El rendimiento y el bajo uso de memoria son sus objetivos principales.	Conexión mediante JDBC con cualquier base de datos que posea dicho driver. Acceso a ficheros: Texto, CSV y XML. Posee componentes para conexión con LDAP.	Sí (basado en Java)

Es válido señalar que aunque estas herramientas poseen muchos puntos de divergencia también presentan aspectos semejantes entre los que se puede señalar: la independencia de la plataforma debido a que todas se basan en Java, la posibilidad de conexión mediante JDBC con las diferentes fuentes de datos y la descripción de las transformaciones en ficheros XML.

Servidor OLAP

En la actualidad no existen muchos proyectos pertenecientes a la Comunidad *"Open Source"* encaminados al desarrollo de servidores OLAP. Existen varias herramientas orientadas a propósitos muy bien definidos y específicos, como es el caso de PALO (OLAP al revés), que trabaja fundamentalmente con hojas de cálculo tipo Excel y posee una alta eficiencia. Es válido destacar que se han dado grandes pasos en el desarrollo de herramientas para la visualización y manipulación de los resultados de consultas OLAP, como es el caso de: OpenI, JPivot, etc (5).

En la comunidad sobresale el proyecto del Servidor OLAP Mondrian registrado en el año 2001, que constituye el paso más serio dado al respecto por sus prestaciones, estabilidad y escalabilidad (19).

Mondrian

Es un servidor OLAP libre y multiplataforma pues se basa en Java. Permite el acceso a bases de datos relacionales y emplea como lenguaje de consulta MDX. Cumple con la arquitectura ROLAP.

Mondrian está estructurado en 4 capas con tareas bien definidas y que cooperan entre sí, ellas son: capa de presentación, capa dimensional, capa principal y capa de almacenamiento.

La capa de presentación determina qué información será mostrada al usuario y de qué manera (tablas interactivas, gráficos de pastel, barras o líneas, etc.) para que pueda interactuar con ella y generar nuevos resultados. La capa dimensional es la encargada de analizar, validar y ejecutar las consultas MDX, lo que se realiza en varios pasos. La capa principal es la encargada de administrar la memoria caché de agregaciones. Su tarea consiste en recibir las peticiones de la capa dimensional y verificar si está cargada en caché; en caso contrario, enviar la petición a la capa de almacenamiento. La capa de almacenamiento es un gestor de bases de datos responsable de cargar los datos almacenados y agregados (hechos y dimensiones) de un gestor relacional.

Los componentes de estas capas pueden estar todos distribuidos o desplegados en la misma máquina. Las capas dimensional y principal deben estar desplegadas juntas, pero la capa de almacenamiento podría estar en otra máquina y acceder a ella por vía remota JDBC. En sistemas multi-usuarios la capa de presentación podría estar desplegada en cada estación.

Mondrian sigue la siguiente estrategia de agregación:

- Los hechos son almacenados en un gestor de bases de datos relacional. Los desarrolladores plantean que no debe desarrollarse un manipulador de datos almacenados si el gestor de bases de datos puede realizar esta tarea.

- La lectura de los datos agregados almacenados en la memoria caché se realiza mediante consultas que utilizan GROUP BY. De igual forma plantean que no es necesario desarrollar un manipulador de agregaciones si el gestor de bases de datos puede realizar esta tarea.

- Si el gestor de bases de datos soporta la ejecución de vistas y el administrador escoge crear vistas para agregaciones particulares, Mondrian las usará de forma implícita.

- La idea general es delegar a la base de datos todas las tareas que por naturaleza le corresponden, de esta manera, se carga en cierta medida el servidor pero los clientes se benefician, tornándose ligeros.

- Mondrian proporciona una API (*Application Programming Interface*) para aplicaciones clientes destinada a ejecutar consultas. La API no goza de una amplia aceptación debido a que utiliza un lenguaje distinto para especificar las consultas llamado MDX (*Multi-Dimensional Expressions*). MDX es un lenguaje de consultas para bases de datos con un diseño multidimensional, en la misma manera que SQL es usado para consultar bases de datos relacionales.

- Para la visualización y manipulación de los resultados de las consultas OLAP se cuenta con JPivot, basada en la tecnología Java Server Page (JSP). JPivot es una biblioteca personalizada de etiquetas JSP que permite visualizar tablas OLAP y realizar sobre ellas operaciones típicas como: slice, drill down, roll up, etc (19).

Conclusiones

En este capítulo se realizó un estudio de algunos aspectos referentes a la calidad de software en general, recopilando ideas y conceptos de diferentes bibliografías y autores que ayudaron en gran medida a ampliar los conocimientos acerca del tema, conociendo a mayor profundidad sobre las subcaracterísticas de portabilidad como son reemplazabilidad y conformidad con la portabilidad. Se llevó a cabo un estudio sobre todo lo referente a Data Warehouse para entender más sobre este tema y así poder llevar la reemplazabilidad y la conformidad de la portabilidad a las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL desarrolladas en el centro de datos DATEC. Se realizó también un estudio de las herramientas utilizadas en la confección de sistemas informáticos de apoyo a la toma de decisiones para realizar una correcta selección de la más adecuada si se quiere alcanzar los atributos de calidad que se desean. De esta manera, se puede proceder al desarrollo de la propuesta que dará solución al problema planteado por lo que en el capítulo siguiente se abordara sobre la estructura que poseerá la misma.

Introducción

En el presente capítulo se plasma la estructura y principales componentes de la guía propuesta, con el objetivo de facilitar y enriquecer el desarrollo de software en la Universidad de las Ciencias Informáticas en especial el centro de datos DATEC. Se plantean las diferentes fases de la metodología de desarrollo utilizada en DATEC así como los grupos que en ella laboran. Se describen los factores de portabilidad que se ven afectados, en que fases de la metodología tienen mayor peso y se explican las buenas prácticas para alcanzarlos.

2.1. Objetivos de la Guía Propuesta

La Guía está dividida en varias partes, teniendo en cuenta las diferentes fases de la metodología utilizada en DATEC, que es una unión de Kimball con la experiencia lograda en el desarrollo de proyectos similares. El objetivo general de la guía es describir los principales factores de portabilidad principalmente los que tengan que ver con sus subcaracterísticas reemplazabilidad y conformidad de la portabilidad en una solución informática de apoyo a la toma de decisiones que utilice Data Warehouse basada en PostgreSQL.

Los objetivos de la presente Guía son los siguientes:

- Brindar una descripción exacta de los principales factores de portabilidad que están presentes en cada fase de desarrollo del software.
- Ofrecer una guía formal de técnicas en la que se logre la reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- Mejorar la calidad de software enfocándonos en la portabilidad de los proyectos productivos de la UCI.

Los objetivos anteriormente mencionados están en correspondencia con las necesidades existentes en la actualidad del entorno productivo UCI, pues no existe una guía por la cual los desarrolladores de software

de cada proyecto puedan apoyarse para valorar las diferentes alternativas que se le presentan y tomar decisiones de forma correcta. Por tanto, surge la necesidad de la creación de la presente.

2.2. Descripción de la Metodología de Desarrollo Utilizada en DATEC

La metodología empleada es Kimball junto con la experiencia alcanzada en la realización de proyectos similares. Esta cubre todas las fases por las que pasa la construcción de un almacén de datos, desde el levantamiento de información inicial hasta la implementación de la herramienta de Inteligencia de Negocio. Es una metodología mixta que reúne elementos de varios procesos de desarrollo de proyectos de integración de datos.

En una primera fase contempla el levantamiento de información a nivel de negocio para identificar los posibles indicadores y aspectos a medir en los análisis, que luego de algunas transformaciones se convierten en los requerimientos de información de entrada y de salida para la solución de integración. De forma paralela a esta actividad se lleva a cabo un estudio de las fuentes de datos que soportan los datos a cargar. Finalizadas estas dos tareas, se corrobora que la información levantada sobre las necesidades de los clientes esté realmente almacenada en las fuentes correspondientes, teniendo los requerimientos informativos correctamente definidos, se procede a diseñar la solución de BD. Una vez diseñada la estructura de BD, se realiza la carga de los datos desde las fuentes y posteriormente se implementan los requerimientos de inteligencia de negocio identificados en el levantamiento de información inicial. Las actividades y artefactos de la solución son realizados por 4 grupos que conforman la línea, especializados en componentes específicos de la solución.

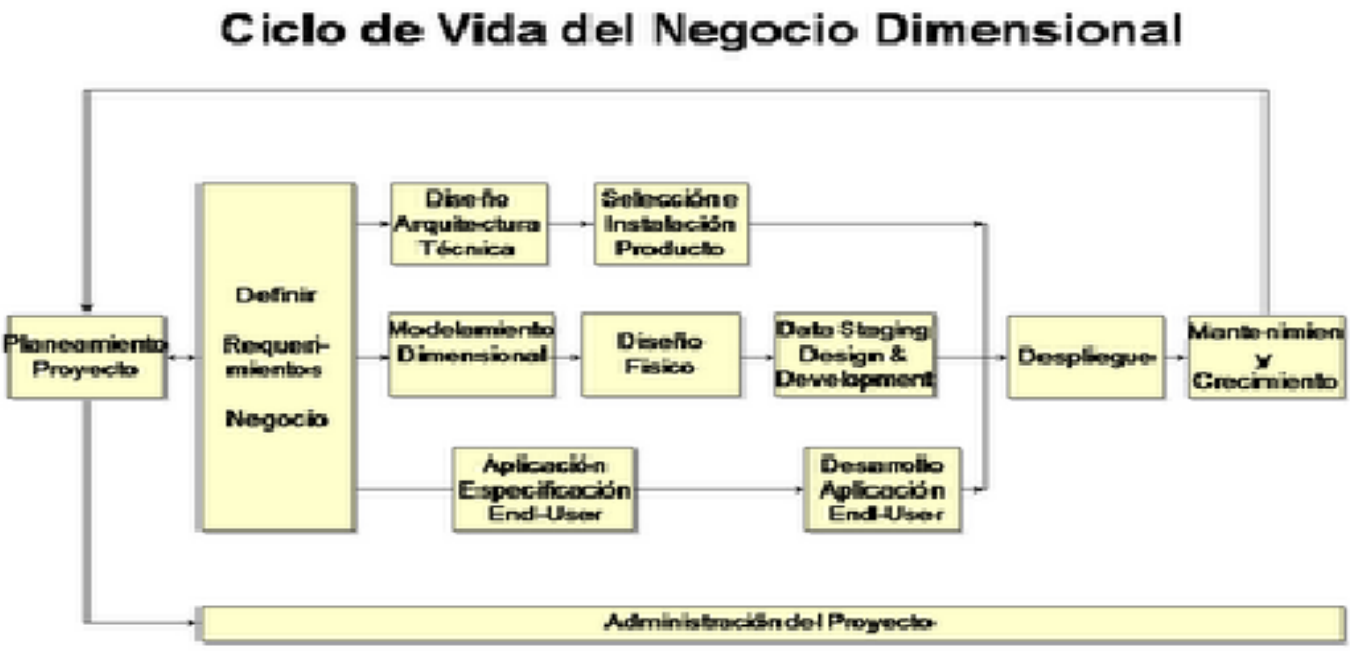


Figura 5. Ciclo de vida propuesto por Kimball

2.2.1. Grupos de desarrollo de la metodología de desarrollo utilizada en DATEC

Se presentan los objetivos específicos de cada grupo de desarrollo establecido en la metodología de desarrollo Kimball.

Grupo de Análisis

- Identificar todos los requerimientos de la solución estableciendo una adecuada comunicación con el cliente.
- Validar durante todo el desarrollo del proyecto que se satisfagan las necesidades de los usuarios (clientes).

Grupo de Almacenes

- Modelar, implementar, documentar y mantener los componentes del Repositorio de Datos en particular.

Grupo de ETL

- Extraer los datos de los sistemas fuentes, limpiarlos e integrarlos a un formato consistente y cargados al almacén.

Grupo de BI

- Diseñar y/o Implementar Soluciones de BI.

2.2.2. Fases de la Metodología de Desarrollo utilizada en DATEC

Estudio Preliminar

En la fase de Estudio Preliminar se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel y la legalización del mismo. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto y realizar estimaciones de tiempo, esfuerzo y costo.

Fase Requerimientos

Durante la fase de requerimientos el grupo de análisis tiene un papel protagónico pues es el máximo responsable de realizar todas las acciones para la gestión de los requisitos. Los artefactos que se desarrollan dentro de esta fase son los siguientes:

- Evaluación de Áreas de la Organización
- Plan de Gestión de Requisitos
- Cronograma de Entrevistas

- Resumen y Resultados de las Entrevistas
- Cuestionarios

Estos documentos pueden ir desarrollándose simultáneamente pues no hay dependencia entre ellos. Una vez desarrollados, se elaboran los siguientes documentos teniendo en cuenta la información obtenida. Ambos artefactos pueden elaborarse simultáneamente

1. Diseño conceptual.

Seguidamente se realizan los que se enuncian a continuación:

2. Especificación de Requisitos
3. Modelo de integración de datos
4. Diseño Conceptual
5. Plan de Pruebas
6. Diseño de Casos de Prueba

Fase de Arquitectura y Diseño

En la fase de arquitectura y diseño se concentra el mayor volumen de desarrollo en proyectos de soluciones integrales, dentro de esta fase todos los grupos tienen participación pero desempeñan un papel fundamental el grupo de almacenes y ETL que serán los encargados del diseño y modelado del almacén y el perfilado de los datos respectivamente. Durante esta fase se comienza la elaboración de los siguientes documentos que son realizados a partir de la información obtenida en la fase de requisitos:

- Modelo de integración de datos
- Informe Levantamiento de Arquitectura de Información

Posterior a la realización de estos artefactos se puede desarrollar la Arquitectura de información y una vez concluida se elaboran los siguientes artefactos:

- Documento de Arquitectura del Sistema
- Arquitectura de información.
- Plan de Infraestructura

A partir de los documentos anteriores se puede pasar a la elaboración del modelo dimensional y perfil de los datos:

1. Modelo de datos

Como últimos artefactos que se generan en esta fase tenemos:

2. Estimación del Tamaño. Planeación de evolución.
3. Estrategia de Respaldo y Recuperación
4. Diseño de pruebas de integración
5. Actualización de plan de pruebas

Durante la fase de arquitectura y diseño además de generar los anteriores artefactos también se refinan la especificación de reglas del negocio y la especificación de requisitos. Una vez concluido el diseño físico así como el perfilado de datos se puede comenzar con la implementación y pruebas de la solución.

Fase de Implementación

En la fase de Implementación se realizan, los procesos de extracción, carga y limpieza de datos para el almacén, se implementa el almacén y la aplicación BI. Durante esta fase se generan los siguientes artefactos:

- Diseño de Casos de Pruebas

- Manual de Usuario
- Especificaciones del usuario final
- Diseño de pruebas de integración
- Código fuente.
- Plan de pruebas

Fase de Prueba

Durante esta fase se realizan las pruebas de calidad a cada una de las unidades implementadas así como pruebas de integración y del sistema de manera general.

- Actualización de plan de pruebas
- Pruebas de Unidad e integración
- Pruebas del sistema

Fase de Despliegue

Durante la fase de despliegue se procede a implantar la solución y a la capacitación al cliente en las herramientas a utilizar y se le deja todo listo para la explotación del almacén. Durante esta fase se crea el siguiente artefacto:

- Guía de Implantación
- Ficheros del Almacén

Fase de Soporte

La fase de soporte es el periodo donde la solución está implantada, los usuarios finales están utilizándolos y se generan una serie de no conformidades de las misma.

1. Desarrollar las Condiciones de Soporte

Con la terminación de la fase de despliegue se le da cierre al proyecto desarrollado.

2.3 Guía Para la Obtención de la Reemplazabilidad y la Conformidad de la Portabilidad de las Soluciones Informáticas de Apoyo a la Toma de Decisiones que Utilizan Data Warehouse Basadas en PostgreSQL

La Guía para la obtención de la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL, está formada por la descripción de los factores de portabilidad en cada una de las fases de la metodología utilizada.

De cada uno de los factores se expone brevemente los objetivos que se persiguen, así como los elementos básicos que los componen y los principios básicos para su uso. Una gran parte de estos factores se pueden resolver sin la ayuda de herramientas por su facilidad de uso.

La guía está estructurada de la siguiente forma:

Tabla 6. Estructura que tiene la guía respecto a los factores y sus respectivas fases

Fases de la Metodología Utilizada en DATEC	Factores de Portabilidad
Fase requerimientos	• Correcta selección de los requisitos para lograr portabilidad. • Requisito de actualización del software. • Documentación de la reemplazabilidad y conformidad de la portabilidad.
	• Selección de las herramientas para diseñar y desarrollar el software.

Fase arquitectura y diseño	• Interfaz de actualización o upgrades del software. • Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software. • Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.
Fase implementación	• Correcta selección del lenguaje de programación.
Fase de soporte	• Realizar proceso de soporte.

2.3.1 Factores de Portabilidad que intervienen en el Centro de Tecnologías de Análisis de Datos (DATEC)

A continuación, presentaremos los principales factores encontrados en el desarrollo de soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL. Estos factores están vinculados a la portabilidad de los softwares específicamente para sus subcaracterísticas reemplazabilidad y conformidad de la portabilidad.

Correcta Selección de los Requisitos de Portabilidad

En ocasiones se cometen errores en la captura de los requisitos por parte del equipo de desarrollo. El desarrollo de un buen software debe comenzar con un cuidadoso proceso de análisis de requerimientos.

Requisito de la Actualización del Software

En muchos proyectos a la hora de desarrollar un software no se tiene en cuenta un requisito que involucre la actualización del producto, trayendo como consecuencia que no exista una funcionalidad para actualizar. Se deben captar requisitos que permitan la posterior actualización del sistema en el tiempo.

Documentación de la Reemplazabilidad y la Conformidad de la Portabilidad

Muchas veces no se tiene en cuenta la documentación de los requisitos no funcionales en la fase de requerimientos, propiciando que en fases posteriores no se desarrolle de forma correcta. Se debe documentar lo referente a reemplazabilidad y conformidad de la portabilidad de manera clara y precisa.

Selección de las Herramientas Para Diseñar y Desarrollar el Software

Si no se realiza un análisis de la herramientas con las cuales se va a realizar el software, se puede caer en un error de selección y escoger una herramienta que no sea la ideal para lo que se quiera. Se debe realizar un análisis de las herramientas a utilizar que ayuden a potenciar las características de calidad con que se desea que quede el software desarrollado.

Interfaz de Actualización o Upgrade del Software

Cuando existe una funcionalidad actualizar software su interfaz por lo general no está diseñada de la manera más conveniente para los usuarios. Las interfaces deben ser intuitivas, amigables, sencillas; hay que tener presente que el sistema está pensado para que sea usado por usuarios que no sean necesariamente expertos en informática.

Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.

Cuando existe una funcionalidad actualizar software en la mayoría de los casos el grado de dificultad para realizarle la actualización es alto, trayendo como consecuencia que el usuario se pierda en el proceso o se sienta incómodo.

Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.

En el desarrollo de un software en ocasiones se emplean patrones sin la menor idea de lo que estos propician, cada patrón o estilo favorece algún elemento en particular. Se recomienda usar patrones o estilos ya sean de diseño como de arquitectura para lograr bajo acoplamiento, alta cohesión y modularidad en el software, características estas importantes para alcanzar grados altos de portabilidad.

Correcta Selección del Lenguaje de Programación

Este factor se refiere primeramente a una correcta selección del lenguaje de programación para después hacer un buen uso de él siguiendo las normas de portabilidad asociadas al mismo logrando con esto llevar la portabilidad al producto final.

Realizar Proceso de Soporte

Muchos proyectos terminan cuando se entrega el software al cliente, trayendo como consecuencia que los clientes una vez probado el sistema generen una serie de no conformidades, sugerencias y hasta nuevas funcionalidades que no son aprovechadas por el grupo de desarrollo, pudiendo con estas sugerencias mejorar el software en nuevas versiones; por eso la importancia de un proceso donde se recojan todos estos elementos.

2.3.2. Buenas Prácticas de los Factores de Portabilidad en su Correspondiente Fase de la Metodología

Fase Requerimientos

Correcta Selección de los Requisitos de Portabilidad

Es evidente que el desarrollo de un buen software debe comenzar con un cuidadoso proceso de análisis de requerimientos. El propósito de la especificación es identificar las funcionalidades y otras propiedades se esperan en el producto a desarrollar.

Hay muchas estructuras propuestas para evaluar la completitud de las condiciones que se plantean en los requerimientos, que van desde informales a plenamente formales como notaciones matemáticas.

Estas notaciones formales están confeccionadas para la forma de expresar los requerimientos funcionales de un producto de software, pero no están diseñados para expresar los no funcionales, la fiabilidad, el rendimiento o la portabilidad.

Si estos requisitos deben existir se expresan por medios menos formales. Hay muy poco de común acuerdo sobre la mejor forma u organización de los documentos de especificaciones.

Una serie de alternativas son, por ejemplo:

- Evitar los obstáculos de portabilidad
- Especificar los objetivos de la portabilidad de forma explícita

Evitar los obstáculos de portabilidad

Evitar los obstáculos significa que las declaraciones en la recogida de los requisitos deben ser evaluadas para verificar que no contengan dependencias innecesarias del sistema.

Por ejemplo, la siguiente declaración está cargada de dependencias:

Un objeto puede ser seleccionado por dos clics del botón derecho del ratón estos dos clics no deben pasar más de 1 / 2 segundo de diferencia, dentro de los 10 píxeles de el centro del objeto.

Probablemente no hay razón para insistir en que la pantalla deba tener una cierta resolución, que el ratón tenga dos botones, o incluso que el dispositivo de señalización deba ser necesariamente un ratón.

Una declaración más adecuada, evitando detalles innecesarios, podría ser:

Un objeto puede ser seleccionado por el uso de un adecuado dispositivo.

La mayoría de las dependencias del sistema, es probable que se refieran a la interfaz de usuario, pero no todas. Por ejemplo, no hay necesidad de especificar que todos los enteros deban estar representados en el rango de 32 bits, si usted requiere solamente que algunos valores puedan variar entre cero y un millón.

Especificar los objetivos de la portabilidad de forma explícita

La recogida de requisitos proporciona una oportunidad para identificar la portabilidad como un objetivo explícito.

No sirve de nada una instrucción como:

Este programa deberá ser fácilmente portado a nuevas plataformas.

Este es un requisito sin sentido que no se puede probar. En su lugar, debe caracterizar a la portabilidad de un modo que se puede lograr y medir.

La portabilidad sólo tiene sentido con respecto a los particulares como entornos de destino. Uno de los aspectos de una especificación para portátiles debe ser una declaración de ambientes en un objetivo esperado. Esta no debe ser una lista de sistemas específicos (Windows NT, Solaris, Apple Newton, entre otros). En cambio, el objetivo ambiente debe ser caracterizado como una clase. Ejemplo, la siguiente declaración podría caracterizar la clase adecuada para un programa de visualización:

Este software puede ser portado a cualquier ordenador personal o estación de trabajo entorno que soporte al menos 16 bits de color en una pantalla de 15 pulgadas, consiguiendo un punto de referencia SPECfp95 calificación mínima de 5,0, y con una capacidad de almacenamiento de datos de al menos 8 MB.

Aquí se identifican los requisitos esenciales, es importante no especificar los detalles de la arquitectura o sistema operativo.

Un segundo aspecto de los objetivos de la portabilidad es determinar cuánto se requiere de la portabilidad. El objetivo de la portabilidad puede estar limitado por objetivos contradictorios, como un alto rendimiento o la explotación de las características de un entorno específico. Los costos de desarrollo y los horarios se verán afectados por esta elección. No hay método comúnmente aceptado para la cuantificación de portabilidad, pero el grado de portabilidad discutido anteriormente ofrece un enfoque posible. Dada una clase de destino de entornos, es posible especificar un valor objetivo según las métricas.

Una unidad de software portátil que no esté bien afinada a un determinado ambiente puede presentar un bajo rendimiento o un mal uso de los recursos que una aplicación en sintonía.

Este es un ejemplo donde se desglosan los requerimientos de portabilidad

Figura 6. Requerimientos de portabilidad

Requisito de Actualización del Software

Se recomienda que en la fase de requisitos cuando se realicen las entrevistas a los clientes explicarles la importancia que tiene una funcionalidad actualizar sistema. ¿Por qué es importante actualizar nuestro software? Se le va a dar respuesta a esta interrogante con otras preguntas. ¿Un hombre necesita cepillarse todos los días? ¿Es necesario ir al médico cuando se está enfermo de gravedad? La respuesta por supuesto es si a todo lo anterior, un producto de software como el cuerpo mismo necesita de mantenimiento de vez en cuando. Es necesario que el software presente un sistema de parches o de actualizaciones para que pueda operar con seguridad y eficacia.

Estas solo son algunas de las funciones que pueden realizar los sistemas de actualizaciones:

- Mejorar la seguridad
- Aumentar la compatibilidad del software
- Optimizar la forma en que el sistema maneja los recursos de la máquina

- Añadir funcionalidades
- Quitar funciones y herramientas anticuadas

Por tal motivo si se logra en las conversaciones y las entrevistas con los clientes convencer a estos de la importancia de implementar una funcionalidad capaz de actualizar el software tanto manual como automáticamente se dará un gran paso hacia la reemplazabilidad en el software a desarrollar.

Documentación de la Reemplazabilidad y Conformidad de la Portabilidad

Se recomienda que en esta fase quede todo documentado y recopilado en cuanto a reemplazabilidad y conformidad de la portabilidad se refiere. Una vez en etapas más adelantadas los desarrolladores pueden guiarse por la documentación y pueden así cumplir con todas estas tareas al pie de la letra. Suele suceder que los desarrolladores en ocasiones no prestan la debida atención a esto, entonces el software puede sufrir de problemas de calidad, por eso es preciso cumplir con la documentación establecida.

Si lo antes expuesto se logra y se asegura que los desarrolladores cumplan con todo lo que está documentado podemos asegurar que el software a desarrollar obtendrá un buen nivel de calidad y con esta la reemplazabilidad y la conformidad de la portabilidad.

Fase Arquitectura y Diseño

Correcta Selección de las Herramientas Para Diseñar y Desarrollar el Software.

Es de suma importancia hacer una buena selección de las herramientas con las cuales vamos a desarrollar las soluciones informáticas de apoyo a la toma de decisiones pues no todas las herramientas tienen las mismas características. Si queremos potenciar en nuestro software la portabilidad y más específicamente la reemplazabilidad y la conformidad de la portabilidad, debemos realizar un cuidadoso análisis para esto.

La herramienta ETL seleccionada es Kettle debido a la diversidad de fuentes de datos que soporta (mediante JDBC, ODBC y JNDI), incluyendo la comunicación con Servicios Web. Debe señalarse también aspectos de gran importancia como: la amplia variedad de transformaciones incluyendo el trabajo con

almacenes de datos (implementación de *"Slowly Changing Dimensions"*), la ejecución en paralelo de las tareas permitiendo la asignación de prioridades, el monitoreo de las tareas en ejecución y la posibilidad de diseñar todo el proceso de forma visual haciendo uso de una interfaz de usuario muy amigable e intuitiva.

Interfaz de Actualización o Upgrades del Software

El encargado de diseñar la interfaz debe emplear siempre que pueda representaciones del mundo real en la interfaz para que el usuario se desarrolle mejor. Permitir con un buen diseño visual que los objetos sean afines a otros de la realidad cotidiana. Hacer todos los objetos disponibles de forma que el usuario pueda usar todos sus objetos en cualquier secuencia y en cualquier momento logrando mayor interacción con esta. Evitarle errores al usuario proporcionándole diferentes tipos de ayuda bien de forma automática o a petición del propio usuario. Soportar diversas técnicas de interacción, de forma que el usuario pueda seleccionar el método de interacción más apropiado para su situación. Permitir a los usuarios adaptar la interfaz a sus necesidades para que este se sienta más identificado con la interfaz.

Una interfaz bien diseñada debe facilitar el trabajo de los usuarios. Para ello es preciso entender el modelo mental del usuario y sus habilidades psíquicas, físicas y psicológicas. Realizar la interfaz consistente entre los distintos navegadores para eliminar posibles dependencias. Diseñar esta interfaz accesible e intuitiva lo cual es básico en el diseño de cualquier interfaz. Crear una sensación de progreso y logro en el usuario para que este no se sienta abrumado o confundido. Permitir manipular los objetos de la interfaz logrando así que sea más sencilla y manuable. Importante no sacrificar la usabilidad por la funcionalidad del sistema.

A continuación, se van a establecer aspectos necesarios en la construcción de la interfaz de las actualizaciones:

- Los textos no contengan faltas de ortografía y la construcción de las frases debe ser correcta.
- Que cuente con un sistema de navegación. Entorno transparente que permita al usuario tener el control. Que sea eficaz pero sin llamar la atención sobre sí mismo.
- La velocidad entre el usuario y el programa (animaciones, lectura de datos…) resulte adecuada.

- En el uso del teclado los caracteres escritos tienen que verse en la pantalla para poder corregir los errores.
- El análisis de respuestas. Que sea avanzado y por ejemplo, ignore diferencias no significativas (espacios superfluos) entre lo tecleado por el usuario y las respuestas esperadas.
- Multiplataforma. Es de vital importancia que la interfaz no dependa de ningún sistema operativo.
- Mapa de navegación. Buena estructuración del programa que permite acceder bien a los contenidos, actividades, niveles y prestaciones en general.
- La ejecución del programa sea fiable, no tenga errores de funcionamiento y detecte la ausencia de los periféricos necesarios.
- Proximidad: Los diseños sencillos son más fáciles de entender y favorecen el uso inmediato y la exploración exhaustiva de los recursos del diseño.
- Inmediatez: Los diseños sencillos tienen un impacto mayor precisamente porque su facilidad de comprensión los hacen inmediatamente reconocibles con un esfuerzo consciente mínimo.
- Usabilidad: Por todo lo anterior suelen ser también los más fáciles de usar.

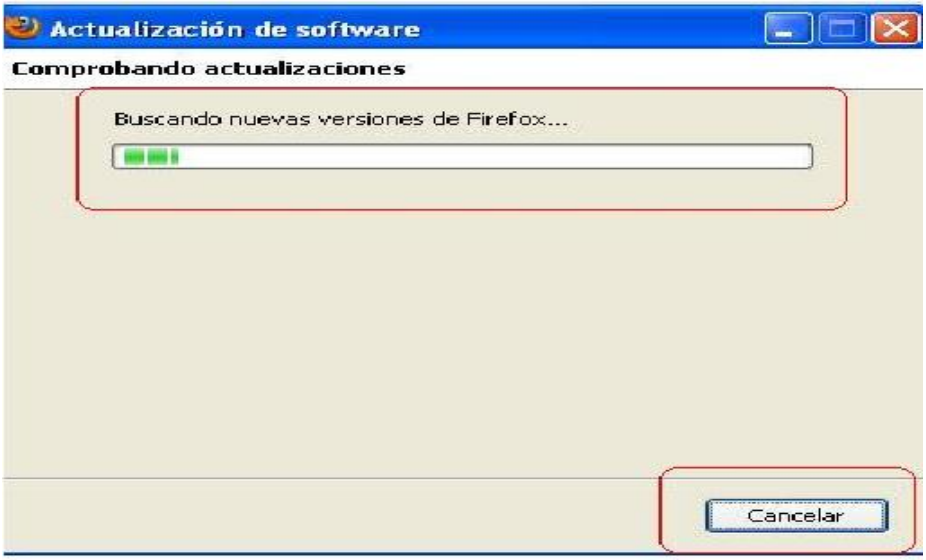


Figura 7. Ejemplo de aspecto de interfaz de actualización

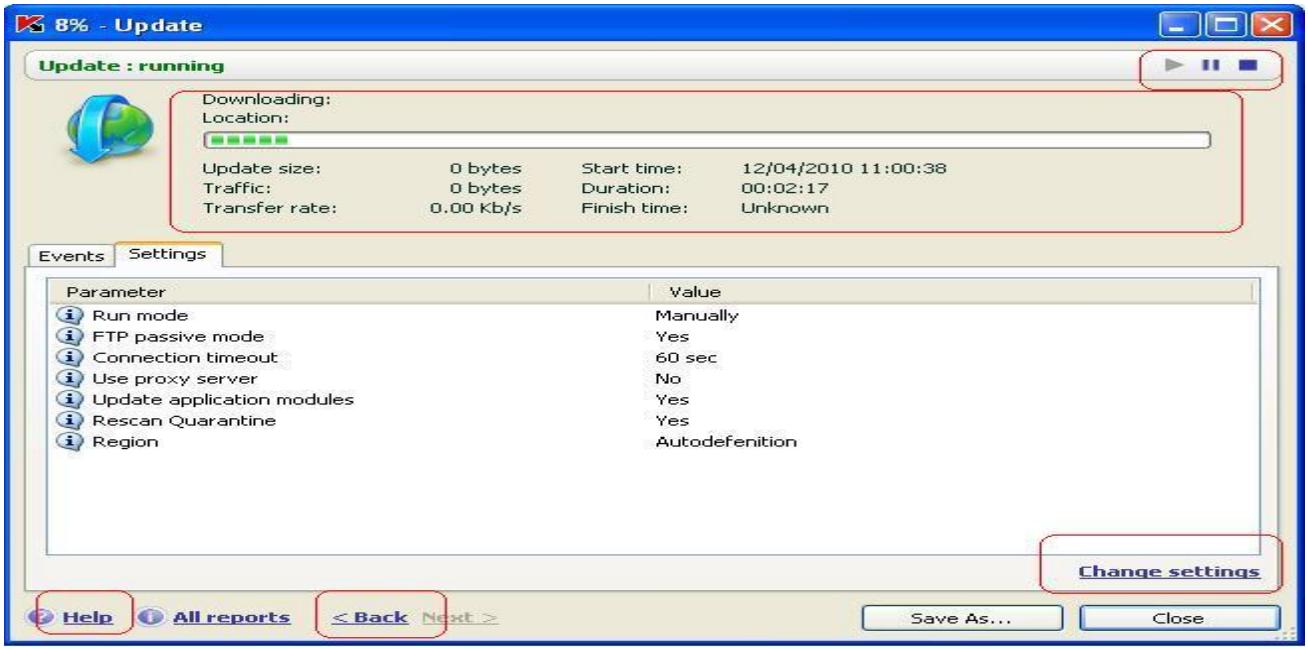


Figura 8. Ejemplo de aspecto de interfaz de actualización

Reducir el Grado de Dificultad al Realizar el Proceso de Actualizaciones o Upgrades en el Software.

Referido al grado de dificultad del usuario al realizarle una actualización al software. Para erradicar este problema se colocan a los usuarios con el control de la interfaz, permitiendo el uso del teclado y el mouse, mostrando mensajes y textos descriptivos. Acomodar a los usuarios con diferentes niveles de habilidad logrando con esto que no le sea difícil o complejo al usuario realizar este proceso. Proporcionar acciones inmediatas, reversibles y realimentación, facilitando el manejo de la actualización.

El diseñador tiene que tener en cuenta los siguientes aspectos para lograr el objetivo deseado, lo primero es reducir la carga de memoria de los usuarios hasta la forma más simple posible. Proporcionar pistas visuales al usuario para lograr mayor comprensibilidad de lo que se está realizando y así reducir la carga cognoscitiva, proporcionar opciones por defecto para que el usuario no tenga que pararse o pensar demasiado en una opción determinada, estando implementada por defecto ahorra trabajo y tiempo, proporcionar atajos logrando así no recorrer todo el flujo de trabajo para realizar la actualización del software. Aplicar técnicas de ingeniería para resolver la problemática del error humano haciendo uso de la consistencia y claridad del proceso. Tener en cuenta hacer las acciones previsibles y reversibles. Las acciones de los usuarios deberían producir los resultados que ellos esperan. Es muy importante la cantidad de pasos para realizar una actualización, tiene que ser la menor posible. El diseñador tiene que proporcionar el control sobre el sistema al usuario y suministrarle asistencia para facilitar la realización de las tareas. Proporcionar ayuda contextual para cada opción y objeto sobre el que pueda posicionarse el cursor.

Los principios generales a considerar en el desarrollo de la interfaz del software para facilitar su uso al cliente son:

- Identificar a quién va dirigido el software.
- Conocer los requerimientos de los usuarios.
- Identificar el grado de conocimiento de los usuarios que van a interactuar con el software.
- Identificar atributos o componentes que logren mayor comprensión de la actividad.

Puntos establecidos para lograr que el grado de dificultad del proceso de actualización sea mínimo:

- Los pasos para realizar la actualización se deben realizar de manera que los usuarios puedan utilizarlos inmediatamente sin tener que realizar una exhaustiva lectura de los manuales ni largas tareas previas de configuración.
- En cada momento el usuario debe conocer el lugar del programa donde se encuentra y tener la posibilidad de moverse según sus preferencias: retroceder, avanzar, etc.
- Contar con un sistema de ayuda on-line solucionará las dudas que puedan surgir.
- Los textos no contengan faltas de ortografía y la construcción de las frases es correcta.
- El uso del teclado. Los caracteres escritos se ven en la pantalla y pueden corregirse errores.
- El análisis de respuestas. Que sea avanzado y por ejemplo, ignore diferencias no significativas (espacios superfluos) entre lo tecleado por el usuario y las respuestas esperadas.
- La gestión de preguntas, respuestas y acciones.
- Adecuación a los usuarios y a su ritmo de trabajo. Los buenos programas tienen en cuenta las características iníciales de los usuarios a los que van dirigidos (desarrollo cognitivo, capacidades, intereses, necesidades).

Esta adecuación se manifestará en tres ámbitos principales:

- **Actividades:** tipo de interacción, duración, elementos motivacionales, mensajes de corrección de errores y de ayuda, niveles de dificultad, itinerarios, progresión y profundidad de los contenidos.
- **Contenidos:** extensión, estructura y profundidad, vocabulario, estructuras gramaticales, ejemplos, simulaciones y gráfico. Los contenidos deben ser significativos para los usuarios y estar relacionados con situaciones y problemas de su interés.
- **Entorno de comunicación:** Pantallas, sistema de navegación, mapa de navegación.

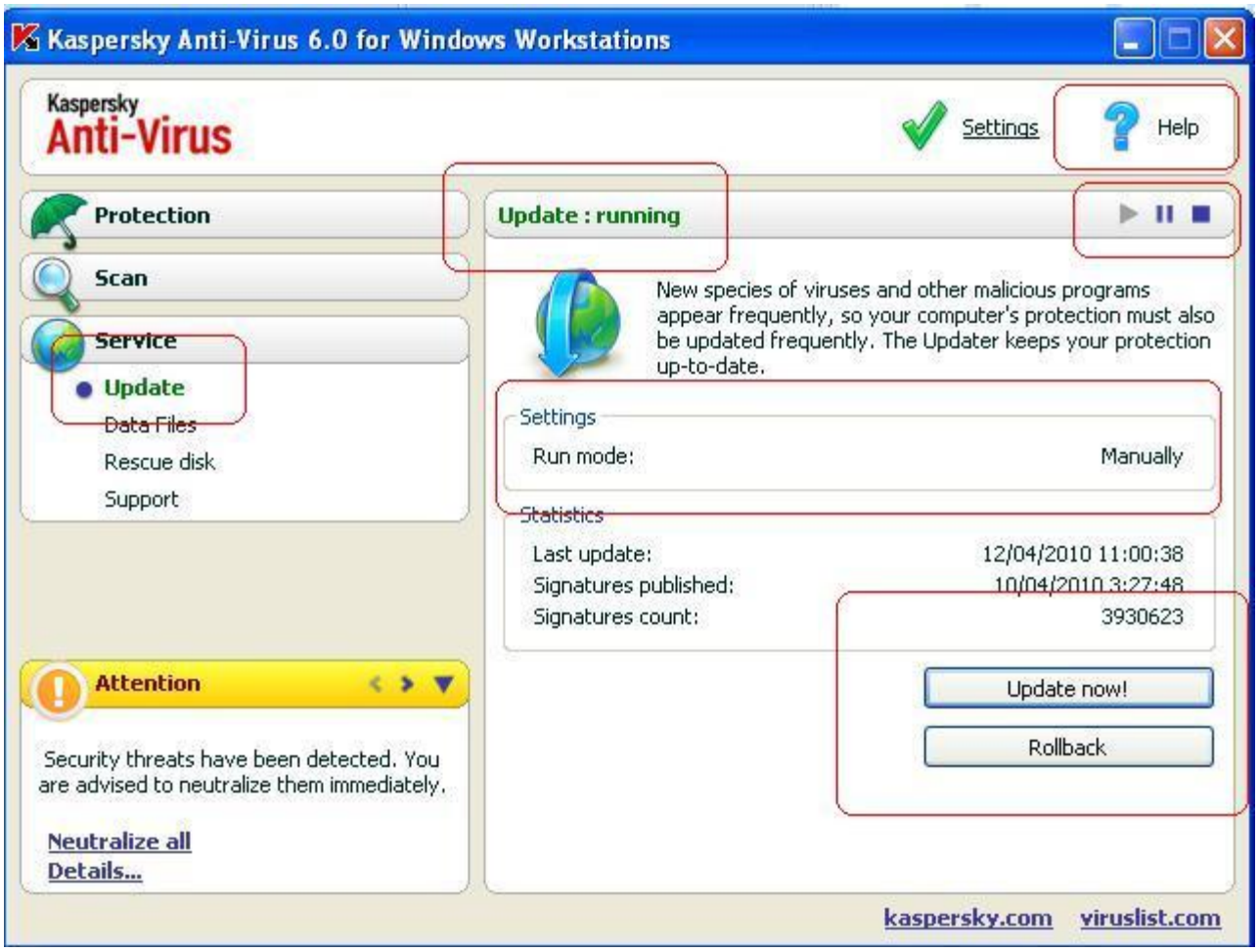


Figura 9. Ejemplo de la ayuda y las pistas que debe dar una interfaz para actualizaciones

Uso de Patrones y Estilos de Diseño y Arquitecturas Para Lograr Bajo Acoplamiento, Alta Cohesión y Modularidad.

Tanto el bajo acoplamiento, como la alta cohesión y la modularidad desempeñan un papel fundamental en la portabilidad, pues son de gran importancia para el mantenimiento, control, reutilización de código, corrección de errores y modificación para futuras versiones de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

Se recomienda utilizar cualquiera de las siguientes propuestas o en combinación pues todas favorecen de alguna manera a la aparición de bajo acoplamiento, alta cohesión y modularidad en el diseño y en la arquitectura de las soluciones.

Patrones recomendados:

Grasp. La mayoría de los patrones que lo conforman dígase experto, creador, controlador, alta cohesión y bajo acoplamiento.

El patrón Grasp o patrón de los principios generales para asignar responsabilidades se encarga como su nombre lo dice de asignar responsabilidades, y realizar correctamente esta tarea es muy importante en el diseño orientado a objetos pues favorece al bajo acoplamiento y a la alta cohesión (20).

- Ejemplo de Experto

En una aplicación de un punto de venta, alguna clase necesita conocer el gran total de la venta. ¿Quién es el responsable de conocer el gran total de venta? Desde el punto de vista del patrón Experto, deberíamos buscar la clase de objetos que posee la información necesaria para calcular el total. Analice la siguiente figura.

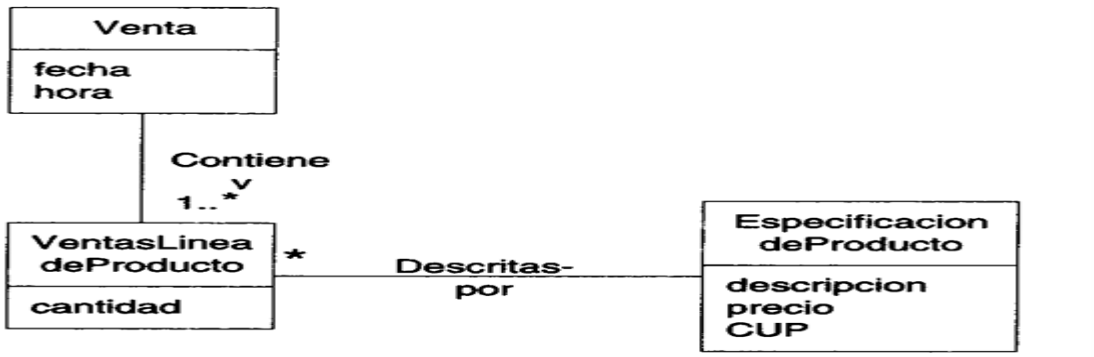


Figura 10. Segmento de un diagrama de clases de un sistema de ventas

¿Qué información hace falta para calcular el gran total de la venta? Hay que conocer todas las instancias *VentasLineadeProducto* de una venta y la suma de sus subtotales. Y esto lo conoce únicamente la

instancia *Venta*; por tanto, desde el punto de vista del Experto, *Venta* es la clase correcta de objeto para asumir esta responsabilidad; es el *experto en información*.

En conclusión, para cumplir con la responsabilidad de conocer y dar el total de ventas, se asignaron tres responsabilidades a las tres clases de objetos así:

Tabla 7. Responsabilidades de las clases

Clase	Responsabilidad
Venta	Conoce el total de la venta
VentaLineadeProducto	Conoce el subtotal de la línea de producto
EspecificaciónDeProducto	Conoce el precio del producto

Este patrón nos brinda como beneficios que se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Programación por Capas

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

Figura 11. Ejemplo de programación por capas

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado (modularidad). Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos.

Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En esta arquitectura a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

Modelo-Vista-Controlador

Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- **Controlador:** Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

El empleo de este patrón presenta muchos beneficios entre ellos muestran diferentes perspectivas de un componente o subsistema. La representación interna puede ser diferente a la que se visualiza en la GUI. Facilita adaptar o reutilizar un componente en diferentes circunstancias con un mínimo de recodificación. El modelo vista controlador favorece el encapsulamiento y con esto a un bajo acoplamiento y a cierta modularidad.

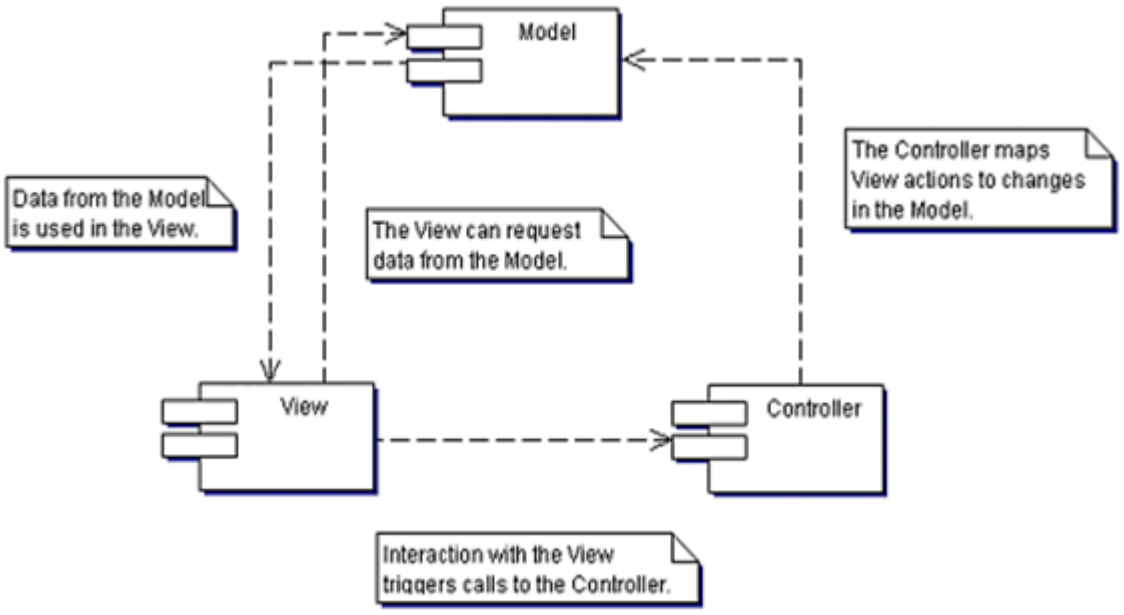


Figura 12. Modelo-Vista-Controlador

Los patrones anteriormente propuestos contribuyen en cierta manera a la aparición del bajo acoplamiento, la alta cohesión y la modularidad, factores estos de suma importancia para que exista reemplazabilidad en las soluciones de software en que se apliquen.

Fase Implementación

Correcta Selección del Lenguaje de Programación.

¿Qué es un estándar? De acuerdo con la definición de la Real Academia Española, “estándar es aquello que sirve como tipo, modelo, norma, patrón o referencia”.

Todos los lenguajes de programación que se presentan son altamente portables en el sentido de que las implementaciones compatibles están disponibles en todos los sistemas Unix modernos, para la mayoría, las implementaciones en Windows y MacOS están disponibles también. Los problemas de portabilidad tienden a surgir no en los idiomas principales, pero sí en las bibliotecas de soporte y en el grado de integración con el entorno local.

Esta sección contiene un resumen de las normas de portabilidad de los principales lenguajes de programación:

Lenguaje de Programación C

El lenguaje de programación C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

El estándar que define este lenguaje fue conocido vulgarmente como ANSI C o como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portátil entre plataformas y/o arquitecturas. En la práctica, los programadores suelen usar elementos no-portátiles dependientes del compilador o del sistema operativo. Uno de los objetivos de diseño del lenguaje C es que sólo sean necesarias unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que haga falta un soporte intenso en tiempo de ejecución.

En consecuencia, el lenguaje C está disponible en un amplio abanico de plataformas (seguramente más que cualquier otro lenguaje).

Un programa escrito cumpliendo los estándares e intentando que sea portátil puede compilarse en muchos computadores.

ANSI C está soportado hoy en día por casi la totalidad de los compiladores. La mayoría del código C que se escribe actualmente está basado en ANSI C. Cualquier programa escrito sólo en C estándar sin código que dependa de un hardware determinado funciona correctamente en cualquier plataforma que disponga de una implementación de C compatible.

Sin embargo, muchos programas han sido escritos de forma que sólo pueden compilarse en una cierta plataforma, o con un compilador concreto, esto puede ser debido a diversos motivos:

- La utilización de bibliotecas no estándar, como interfaces gráficas de usuario.
- El uso de compiladores que no cumplen las especificaciones del estándar.
- El uso de tipos de datos suponiendo que tendrán el mismo tamaño u orden de los bits en todas las plataformas.

Ventajas del uso del lenguaje de programación C:

- Es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas.
- A pesar de su bajo nivel es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas conocidos.

- Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.

- Las bibliotecas de C son colecciones de rutinas utilizadas en el lenguaje de programación C. Las bibliotecas más comunes son la biblioteca estándar de C, la ISO y estándar ANSI C provee las especificaciones de los estándares, las cuales son ampliamente compartidas entre bibliotecas. La biblioteca ANSI C estándar incluye rutinas para la entrada y salida de archivos, alojamiento de memoria y operaciones con datos comunes como funciones matemáticas, funciones de cadenas y funciones de hora y fecha.

Lenguaje de Programación Java

La portabilidad del lenguaje de programación Java es excelente después de todo, fue diseñado con el lema de: "escribir una vez, ejecuta en todas partes" como objetivo primordial. Java es un lenguaje de programación muy potente por la portabilidad que ofrece los programas que uno puede crear con él.

Una de las características que supuestamente han llevado a Java a la fama es su portabilidad "Universal", por lo que algunos programadores asumen que si implementan una aplicación en Java, esta será "perfectamente" portable a través de todas las plataformas de Java, de forma automática desafortunadamente no siempre es el caso.

La máquina virtual Java suele estar instalada en la gran mayoría de máquinas, ya sean Windows, Linux, etc. Indudablemente Java tiene a favor la capacidad de crear aplicaciones independientes.

Una versión del entorno de ejecución Java JRE (Java Runtime Environment) está disponible en la mayoría de equipos de escritorio. Sin embargo, Microsoft no lo ha incluido por defecto en sus sistemas operativos. En el caso de Apple, éste incluye una versión propia del JRE en su sistema operativo, el Mac OS. También es un producto que por defecto aparece en la mayoría de las distribuciones de GNU/Linux. Debido a incompatibilidades entre distintas versiones del JRE, muchas aplicaciones prefieren instalar su propia copia del JRE antes que confiar su suerte a la aplicación instalada por defecto. Los desarrolladores de applets de Java o bien deben insistir a los usuarios en la actualización del JRE, o bien desarrollar bajo una versión antigua de Java y verificar el correcto funcionamiento en las versiones posteriores.

Esto se logra mediante la compilación del código Java a una representación intermedia denominada bytecode de Java, en lugar de directamente de la plataforma de código máquina. En el bytecode de Java las instrucciones son análogas a código máquina, pero está destinado a ser interpretado por una máquina virtual (VM), escrito específicamente para el hardware de acogida. Los usuarios finales suelen usar un Java Runtime Environment (JRE) instalado en su propia máquina para aplicaciones Java, o en un explorador Web para Java applets.

Pero la portabilidad de Java no está exenta de problemas. Las dificultades son en su mayoría problemas de riesgo, de la versión JDK 1.1 y entre los mayores AWT toolkit GUI (por un lado) y JDK 1.2 con el nuevo kit de herramientas GUI Swing.

Hay varias razones importantes para ellos:

- El Sun's AWT del diseño era tan deficiente que tuvo que ser sustituido por Swing.
- La negativa de Microsoft, a apoyar el desarrollo de Java en Windows y tratar de reemplazarlo con C #.
- La decisión de Microsoft para mantener Explorer un subprograma de apoyo a Internet en el JDK 1.1 nivel.
- Términos de licencia de Sun que hacen que las implementaciones de código abierto de JDK 1.2 sea imposible, retardando su implantación (especialmente en el mundo GNU\Linux).

Para los programas que involucran a interfaces de usuario gráficas, los desarrolladores de Java que buscan portabilidad, en un futuro próximo, se enfrentan a una elección: Estancia en el JDK 1.1/AWT con un conjunto de herramientas diseñadas para alcanzar una pobre portabilidad máxima (incluyendo a Microsoft Windows), o utilizar la caja de herramientas y capacidades del JDK 1.2 a costa de sacrificar cierta portabilidad.

Lenguaje de Programación Python

Alejándose de la parte sintaxis, creo que lo más importante a tener en cuenta es que normalmente cuando la gente piensa de Python, no podría pensar en todas las bibliotecas que lo componen.

Muchos paquetes de Python dependen de bibliotecas de C, que pueden o no ser compatibles con cualquier plataforma cruzada. Además, Python ejecuta en Java a través de Jython, y .Net a través de IronPython.

Si usted trata con los formatos de archivo binario en Python, tenga en cuenta que la struct y la array son módulos dependientes del tamaño de la máquina. struct se puede utilizar usando siempre < o > en la cadena de formato. array no puede. Probablemente será portátil para las matrices de bytes, pero la documentación no hace ninguna garantía. array todavía podría ser utilizado con un condicional <array instance>.byteswap() sobre la base de sys.byteorder , que es muy rápido.

Algunos módulos no son multiplataforma. Dos de ellos son curses (Linux) y msvcrt (Windows).

Lenguaje de Programación Perl

No debemos engañarnos pensando que es difícil crear código portable en Perl. Perl tiene un buen nivel de portabilidad, para cerrar las brechas entre lo que está disponible en diferentes plataformas, y todos los medios disponibles para usar estas características. Casi todo el código de Perl funciona en cualquier máquina sin modificación. Pero hay algunas cuestiones importantes en la escritura de código portable. No existen versiones ISO o ANSI de Perl.

Perl es generalmente portable; el mismo código puede llevarse de una arquitectura a otra sin problemas. Hay, por supuesto, algunas funciones que son dependientes de la plataforma como son los saltos de línea.

En la mayoría de sistemas operativos, las líneas en los archivos terminan con saltos de línea. Justo lo que se utiliza como una nueva línea puede variar de OS a OS. Unix tradicionalmente utiliza \012 , una especie de Windows utiliza \015\012 , y Mac OS utiliza \015.Perl usa \n para representar a la nueva línea lógica"``, donde lo que es lógico puede depender de la plataforma en uso. En Mac Perl, \n siempre

significa \015. En DOSish Perl \n por lo general significa \012 , pero cuando se accede a un archivo con el texto en el modo ``'', STDIO que se traduce en (o desde) \015\012 .

Debido a la traducción de texto ``'modo, Perl DOSish tiene limitaciones de la utilización de seek y tell cuando un archivo se tiene acceso en texto ``'modo. En concreto, si nos atenemos a seek -ing a los lugares que obtuvo de tell (y no otros), por lo general estamos libres de usar seek y tell incluso con el texto en el modo ``''. En general, utilizando seek o tell o archivo de otras operaciones que cuentan bytes en lugar de caracteres, sin tener en cuenta la longitud de la \n, puede ser no-portátil. Si utiliza binmode en un archivo, sin embargo, por lo general puede utilizar seek y tell con valores arbitrarios con bastante seguridad. Un error común en la programación socket es que \n eq \012 en todas partes del código. Cuando protocolos que utilizan como los protocolos comunes de Internet, \012 y \015 se llaman de forma específica y los valores de la lógica \n y \r no son fiables.

Ordenamiento de bytes

Diferentes integradores de CPU y el punto de números en coma flotante en distintos órdenes (llamado endianness) y anchos (32-bit y 64-bits es el más común).

Esto afecta a su programa si se intenta transferir un número en formato binario a partir de una arquitectura de CPU a otro a través de algún nivel: ya sea “en vivo "a través de conexiones de red o almacenamiento de los números para el almacenamiento secundario, como un archivo de disco.

Órdenes contradictorias de almacenamiento hacen lío total de los números: si un poco de acogida-endian (Intel, Alpha) almacena 0x12345678 (305419896 en decimal), un host grande-endian (Motorola, MIPS, Sparc, PA) se lee como 0x78563412 (2018915346 en decimal).Para evitar este problema en la red conexiones (socket) utilizan el pack () y unpack () formatos de "n" y "N" , la red'', `` órdenes, garantizando su portabilidad.

Archivos

La mayoría de las plataformas cuentan con una estructura de archivos de forma jerárquica. Por lo tanto, es razonable suponer que cualquier plataforma cuente con un camino para identificar de forma única un archivo en el sistema. Si bien son similares, el archivo de especificaciones de ruta de acceso difieren entre

Unix, Windows, Mac OS, OS / 2, VMS, RISC OS y probablemente otros. Unix, por ejemplo, es uno de los pocos sistemas operativos que tiene un directorio raíz. VMS, Windows y OS / 2 pueden trabajar de manera similar a Unix con / como separador de ruta, o en su idiosincrasia propias (tales como tener varios directorios raíz y `` diversos ficheros de dispositivos sin raíces"NIL tales: y LPT:). Mac OS como separador de ruta /. RISC OS Perl puede emular Unix con nombres de archivo / como separador de ruta, o nativas y el uso. Separador de ruta de acceso y a la señal de los sistemas de archivo y nombres de disco. Al igual que anteriormente el problema de nueva línea, hay módulos que pueden ayudar. El módulo File::Spec proporciona métodos para hacer lo correcto en cualquier plataforma pasa a estar en ejecución el programa.

Módulos estándar

En general, los módulos estándar de trabajo a través de plataformas. Excepciones notables son `CPAN.pm` (que actualmente realiza conexiones a programas externos que pueden no estar disponibles), módulos específicos de plataforma (como `ExtUtils::MM_VMS`), y módulos de DBM. No hay un módulo de que esté disponible en todas las plataformas. `SDBM_File` y los otros están generalmente disponibles en todos los Unix y los puertos DOSish, pero no en Mac Perl, donde sólo `NBDM_File` y `DB_File` están disponibles. La buena noticia es que al menos algún módulo de la PDD debe estar disponible, y `AnyDBM_File` uso será lo que el módulo se puede encontrar. Por supuesto, el código debe ser bastante estricto, que descendió hasta el mínimo denominador común (por ejemplo, que no exceda de 1 K para cada registro).

Juegos de caracteres y codificación de caracteres

No asuma nada acerca de los valores numéricos (ord () , chr ()) de caracteres. No asuma que los caracteres alfabéticos se codifican contiguos (en el sentido numérico). No suponer nada sobre el orden de los caracteres. Las letras minúsculas pueden venir antes o después de las letras mayúsculas, las minúsculas y mayúsculas pueden ser entrelazados para que tanto 'a' y 'A' antes de venir 'b', los otros caracteres internacionales y pueden ser entrelazadas de modo que una está antes la 'b'.

El lenguaje es uno de los elementos más importantes de cualquier desarrollo y tiene una influencia decisiva en la depuración y el mantenimiento de la aplicación. Debido a esto se debe tener en cuenta otros criterios para su selección como:

- Disponibilidad y entorno, hay que comprobar los compiladores existentes para la plataforma elegida. Estudio comparativo de eficiencia con un programa de prueba. Herramientas del entorno de desarrollo: editor, montador, depurador, control versiones, manejo de librerías, etc.

- Reusabilidad, organización de librerías que faciliten la búsqueda y almacenamiento de módulos reutilizables.

- Uso de varios lenguajes, no es aconsejable a no ser que las distintas partes sean más fáciles de desarrollar en lenguajes concretos. Hay que tener en cuenta la compatibilidad de los compiladores.

Fase de Soporte

Proceso de Soporte

Una vez terminado el software es entregado a los usuarios finales para que estos lo prueben y redacten una lista de no conformidades y sugerencias de nuevas funcionalidades que servirán para corregirlo y mejorar el software en futuras versiones. Aunque este proceso deberá empezar en fases anteriores.

Una no conformidad es un desvío o falta de cumplimiento sistemático de cualquiera de los requisitos especificados, o hechos concretos que por su importancia puedan comprometer el cumplimiento de la misión de los procesos. Para eliminar una no conformidad se realizan acciones correctivas que no son más que acciones tomadas para eliminar la causa raíz de una no-conformidad detectada u otra situación indeseable.

En cada proceso se establecerán los controles necesarios para la detección de aquellos resultados que no cumplan los requisitos especificados antes de su entrega a los clientes, para de esta manera minimizar la mayor cantidad posible de no conformidades.

Conclusiones

En este capítulo se realizó la descripción de la solución propuesta a la problémica planteada. Para ello se obtuvieron un grupo de factores que favorecen la portabilidad y más específicamente la reemplazabilidad

y la conformidad de la portabilidad. Estos factores fueron ubicados en la fase de la metodología de desarrollo utilizada en DATEC para después pasar a sugerir buenas prácticas a estos.

Introducción

En el presente capitulo se realiza la validación de la guía anteriormente propuesta teniendo en cuenta las valoraciones y criterios de un pequeño panel de especialistas en el tema y el empleo de una representación gráfica de los datos recogidos en la encuesta realizada a los mismos. Se utilizaron técnicas propuestas por el método Delphi, basado en el criterio de expertos para la obtención y modelado de estos criterios.

3.1. Método de Validación Delphi.

El método Delphi, cuyo nombre se inspira en el antiguo oráculo de Delphos, fue ideado originalmente a comienzos de los años 50 en el seno del Centro de Investigación estadounidense RAND Corporation por Olaf Helmer y Theodore J. Gordon, como un instrumento para realizar predicciones sobre un caso de catástrofe nuclear. Desde entonces, ha sido utilizado frecuentemente como sistema para obtener información sobre el futuro.

Linston y Turoff definen la técnica Delphi como: “Un método de estructuración de un proceso de comunicación grupal que es efectivo a la hora de permitir a un grupo de individuos, como un todo, tratar un problema complejo”.

Este método consiste en la selección de un grupo de expertos a los que se les encuesta su opinión sobre cuestiones referidas a acontecimientos del futuro. La capacidad de predicción del método se basa en la utilización sistemática de un juicio intuitivo emitido por un grupo de expertos. Es decir, el método Delphi procede por medio de la interrogación a expertos con la ayuda de cuestionarios sucesivos, a fin de poner de manifiesto convergencias de opiniones y deducir eventuales consensos.

La encuesta es realizada de forma anónima por lo que la calidad de los resultados depende del cuidado que se ponga en la confección del cuestionario y en la elección de los expertos a consultar. (21)

Dentro de las características que presenta se puede encontrar:

- **Anonimato:** Durante el Delphi ningún experto conoce la identidad de los otros que componen el grupo de debate.

- **Iteración y realimentación controlada:** La iteración se consigue al presentar varias veces el mismo cuestionario, lo que permite disminuir el espacio intercuartil, ya que se consigue que los expertos vayan conociendo los diferentes puntos y puedan ir modificando su opinión.

- **Respuesta del grupo en forma estadística:** La información que se presenta a los expertos no es solo el punto de vista de la mayoría sino que se presentan todas las opiniones indicando el grado de acuerdo que se ha obtenido.

- **Heterogeneidad:** Pueden participar expertos de determinadas ramas de actividad sobre las mismas bases.

Teniendo en cuenta las características de la investigación se decide utilizar el método Delphi, ya que mediante éste los expertos deben predecir los resultados a alcanzar con la propuesta elaborada para obtener información sobre el futuro, permitiendo a través de esto arribar a conclusiones acerca de la validez de la guía confeccionada. El procesamiento estadístico y matemático de la información es la característica más importante del método que lo diferencia del resto de los métodos, ya que la decisión final que se toma es un criterio fuertemente avalado por la experiencia y conocimiento del colectivo de expertos consultado (22).

A continuación, se detallan los pasos necesarios para la aplicación del método al problema en cuestión:

- Selección de los expertos.

- Elaboración del cuestionario para la validación de la propuesta.

- Cálculo de concordancia entre los expertos.

- Desarrollo práctico y explotación de los resultados.

3.1.1 Selección de los Expertos

La selección de expertos se realiza bajo las siguientes condiciones:

- Debe ser graduado de nivel superior.

- Debe estar vinculado a la investigación y al desarrollo de bases de datos desarrolladas en PostgreSQL.
- Debe contar con un año de experiencia como mínimo en el tema.
- Debe poseer conocimiento en el proceso de creación de bases de datos desarrolladas en PostgreSQL.
- Debe contar con conocimiento y habilidades en actividades de calidad del software.

La cantidad de expertos seleccionada es siete; a estos se les aplica la encuesta de autovaloración que se observa en el Anexo 1 para determinar el coeficiente de competencia.

Una metodología completa y sencilla para la determinación de la competencia de los expertos, la constituye la aprobada en febrero de 1971 por el comité estatal para Ciencia y la técnica de Rusia para elaboración de pronósticos científico técnico. En esta metodología la competencia de los expertos se termina por el coeficiente (**k**), el cual se calcula de acuerdo con la opinión del experto sobre su nivel de conocimiento acerca del problema que se está resolviendo y con las fuentes que le permiten argumentar sus criterios.

El **coeficiente de competencia** se calcula por la siguiente fórmula:

$K = ½ (kc + ka)$

Donde:

kc- es el **coeficiente de conocimiento o información** que tiene el experto acerca del problema, calculado sobre la valoración del propio experto en una escala del 0 al 10 y multiplicado por 0,1; de esta forma, la evaluación "0" indica que el experto no tiene absolutamente ningún conocimiento de la problemática correspondiente, mientras que la evaluación "10" significa que el experto tiene pleno conocimiento de la problemática tratada. El experto deberá marcar con una cruz (X) en la casilla que estime pertinente. La siguiente tabla muestra el nivel de conocimiento de los expertos seleccionados.

Tabla 8. Coeficiente de conocimiento de los expertos seleccionados

No. Experto	0	1	2	3	4	5	6	7	8	9	10
1							X				
2							X				
3								X			
4										X	
5							X				
6								X			
7									X		

ka- es el **coeficiente de argumentación o fundamentación** de los criterios del experto.

Para calcular este coeficiente el experto debe marcar, según su consideración, cuáles fueron sus fuentes para la obtención del conocimiento que le permite argumentar su evaluación del nivel de conocimiento que especificó anteriormente, en el anexo 1 se puede observar dicha tabla. En la siguiente tabla se muestra los valores obtenidos por cada uno de los expertos.

Tabla 9. Tabla para calcular el coeficiente de argumentació

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted.	4,7	1,2,3,5,6	
Su experiencia obtenida.	4,7	1,3,5,6	2
Trabajos de autores nacionales.			1,2,3,4,5,6,7
Trabajos de autores extranjeros.	1,3,4,5,6,7	2	
Su propio conocimiento del estado del problema en el extranjero.	7	1,4,6	2,3,5
Su intuición.	6	1,3,4,5	2,7

Para calcular el coeficiente de argumentación las respuestas de los expertos se traducen a puntos según lo que muestra la siguiente tabla patrón, este se calcula sumando los valores de la tabla patrón en concordancia con las respuestas dadas por los expertos.

Tabla 10. Tabla patrón para determinar el coeficiente de argumentación

FUENTES DE ARGUMENTACIÓN	Grado de influencia de cada una de las fuentes en sus criterios.		
	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted.	0.3	0.2	0.1
Su experiencia obtenida.	0.5	0.4	0.2
Trabajos de autores nacionales.	0.05	0.05	0.05
Trabajos de autores extranjeros.	0.05	0.05	0.05
Su propio conocimiento del estado del problema en el extranjero.	0.05	0.05	0.05
Su intuición.	0.05	0.05	0.05

El código de interpretación de tales coeficientes de competencias es:

- Si 0,8 < k < 1,0 el coeficiente de competencia es Alto.
- Si 0,5 < k < 0,8 el coeficiente de competencia es Medio.
- Si k < 0,5 el coeficiente de competencia es Bajo.

En la siguiente tabla se muestran los resultados obtenidos en la encuesta de autovaloración realizada a los expertos seleccionados:

Tabla 11. Resultados obtenidos en la encuesta de autovaloración

No. Experto	Kc Coeficiente de Conocimiento	Ka Coeficiente de Argumentación	K Coeficiente de Competencia	Grado
1	0.6	0.8	0.7	Medio
2	0.6	0.6	0.6	Medio
3	0.7	0.6	0.65	Medio
4	0.9	1	0.95	Alto
5	0.6	0.8	0.7	Medio
6	0.7	0.8	0.75	Medio
7	0.8	1	0.9	Alto

Teniendo en cuenta que los siete expertos estuvieron de acuerdo en responder las preguntas del cuestionario y que todos tuvieron un coeficiente entre medio y alto, se decidió que el número de expertos del Comité de Expertos en la investigación sería siete.

3.1.2. Elaboración del Cuestionario para la Validación de la Propuesta

Para la validación de la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL se utilizó la encuesta que se observa en el Anexo 1, entre sus objetivos se puede mencionar:

- Establecer el nivel de recomendación de los factores en la guía.

- Determinar la complejidad de los factores en la guía.
- Determinar la necesidad de los factores en la guía.
- Determinar la novedad de los factores de la guía.
- Determinar el nivel de efectividad de los factores de la guía.
- Posibilidad de aplicación de los factores de la guía.
- Relación entre los factores de la guía y su ubicación dentro del proceso de desarrollo de software.

3.1.3 Cálculo de la Concordancia entre los Expertos

Definido ya el grupo de expertos y enviado a estos la encuesta, se buscaron sus criterios sobre la guía de buenas prácticas para obtener reemplazabilidad y conformidad de portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL. Antes de realizar la explotación de los resultados es necesario calcular la concordancia entre los expertos para medir en qué magnitud, estos están de acuerdo en sus respuestas. Para esto es necesario calcular el *coeficiente de concordancia*. Este coeficiente fue posible calcularlo utilizando el software "*SPSS 13.0 for Windows*". El resultado obtenido fue 0.671, siendo este un valor muy cercano a 1.00, lo que evidencia que los especialistas tienen un alto grado de concordancia en todas las respuestas. En el Anexo 2 se puede observar el coeficiente de Kendall obtenido.

3.1.4 Desarrollo Práctico y Explotación de los Resultados

Para ir almacenando los resultados aportados por los expertos se confeccionan tablas utilizando el programa "*Excel 2007*". A continuación se muestra los gráficos y porcientos obtenidos por cada uno de los objetivos establecidos.

3.1.5 Datos relacionados con las preguntas de la encuesta realizada

Tabla 12. Número asociado al valor correspondiente

No. Factor	Factor asociado
Factor 1	Correcta selección de los requisitos de portabilidad.
Factor 2	Requisito de actualización del software.
Factor 3	Documentación de la reemplazabilidad y conformidad de la portabilidad.
Factor 4	Correcta selección de las herramientas para diseñar y desarrollar el software.
Factor 5	Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.
Factor 6	Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.
Factor 7	Correcta selección del lenguaje de programación.
Factor 8	Realizar proceso de soporte.

Calificación según importancia del factor evaluado.

Muy alto	**5**
Alto	**4**
Medio	**3**

Bajo	**2**
Muy Bajo	**1**

3.1.5.1 Establecer el nivel de recomendación de los factores en la guía

Es de vital importancia conocer que con los datos introducidos por los expertos en la encuesta realizada se calcula un promedio redondeado para evitar las comas en cuanto a la puntuación recibida para contar con una vista general sobre la validación, este proceso se realizara en todas las preguntas con el objetivo de poder validar la guía en cuantos a las preguntas realizadas.

A este objetivo se le da cumplimiento en la pregunta uno de la encuesta, donde los especialistas tienen que responder en qué nivel se deben recomendar los factores de la guía desarrollada. Los resultados alcanzados se muestran en el siguiente gráfico:

Tabla 13. Resultados obtenidos en la Encuesta Realizada por experto en cada factor

No. Experto	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8	Promedio obtenido
1	5	4	4	4	5	4	5	4	**4**
2	4	3	4	4	5	4	3	4	**4**
3	5	3	4	4	5	4	4	5	**5**
4	5	3	4	4	5	4	2	5	**4**

5	5	4	4	5	4	4	5	4	**5**
6	5	4	4	4	5	5	4	4	**5**
7	5	4	4	5	5	5	4	5	**5**

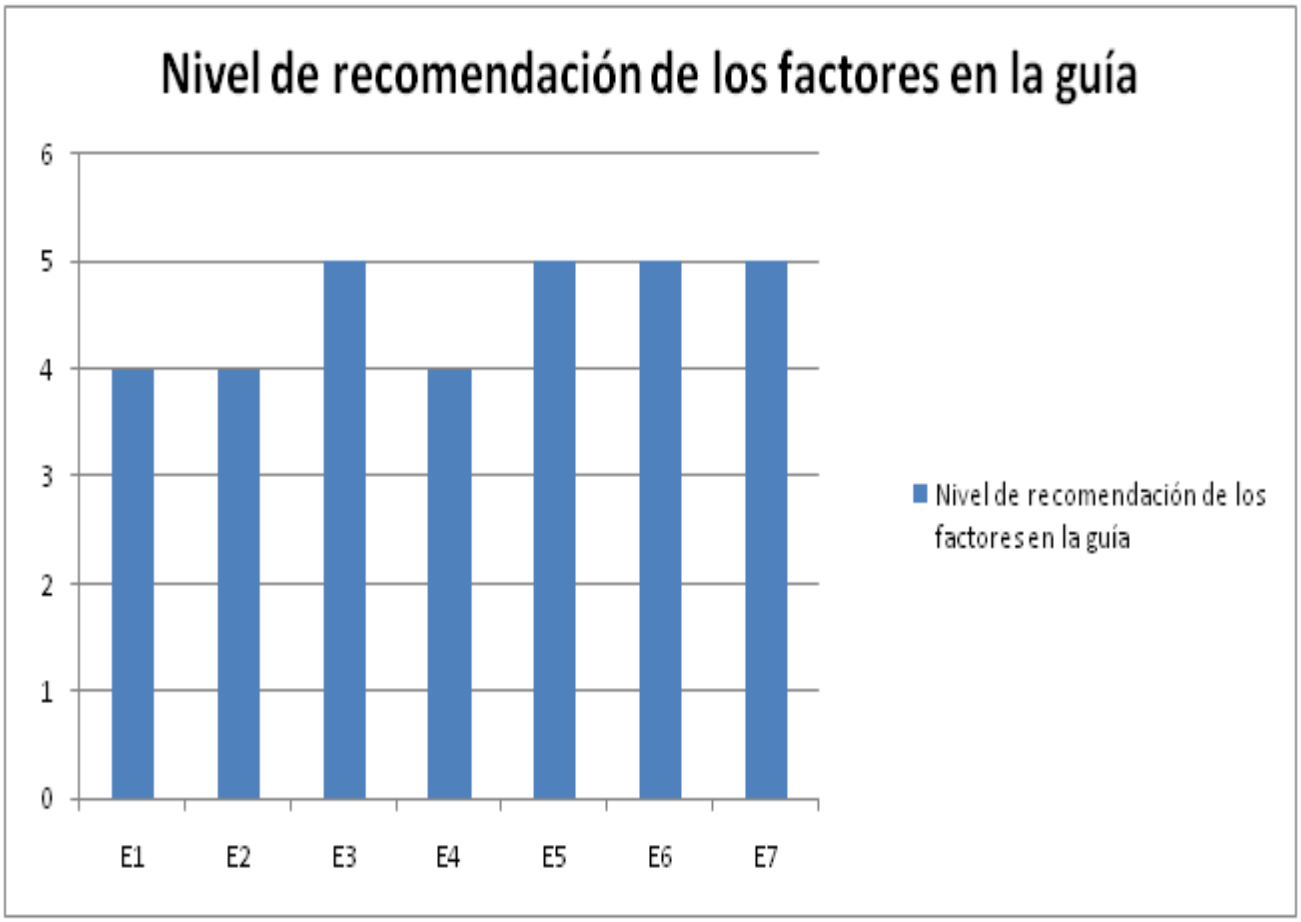


Figura 13. Resultados obtenidos en la Encuesta Realizada

Se puede observar que los expertos coinciden en que la guía de buenas prácticas presenta un 57.14% de especialistas que coinciden que este aspecto la guía cuenta con muy alto en cuanto a recomendar la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones

informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL, lo que constituye un buen grado de aceptación y opinan lo siguiente:

Estos aspectos hay que tenerlos en cuenta desde el mismo diseño y concepción del sistema. Para que los desarrolladores lo hagan de una manera uniforme y se eviten errores posteriores.

3.1.5.2 Determinar la complejidad de los factores en la guía

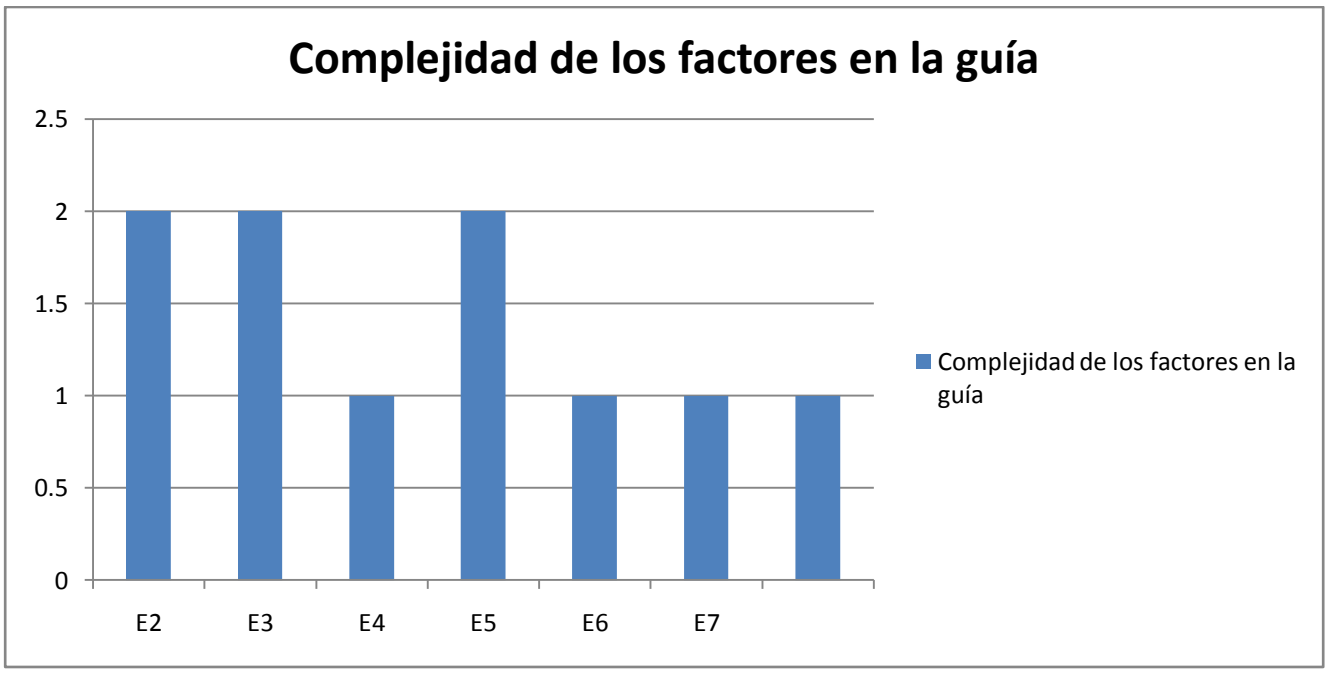


Figura 14. Resultados obtenidos en la Encuesta Realizada

En esta pregunta todos los especialistas concuerdan que la guía de buenas prácticas para obtener recomendar la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL no es compleja ni presenta dificultad para la aplicación de la misma y opinando lo siguiente:

- Es una guía sencilla, entendible por cualquier usuario.
- Está fácil de realizar.
- Es una guía fácil de seguir.

3.1.5.3 Determinar la necesidad de los factores en la guía

La pregunta tres responde a este objetivo, los especialistas deben seleccionar el grado de necesidad de emplear la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL, en el gráfico que se presenta a continuación se muestran los resultados logrados:

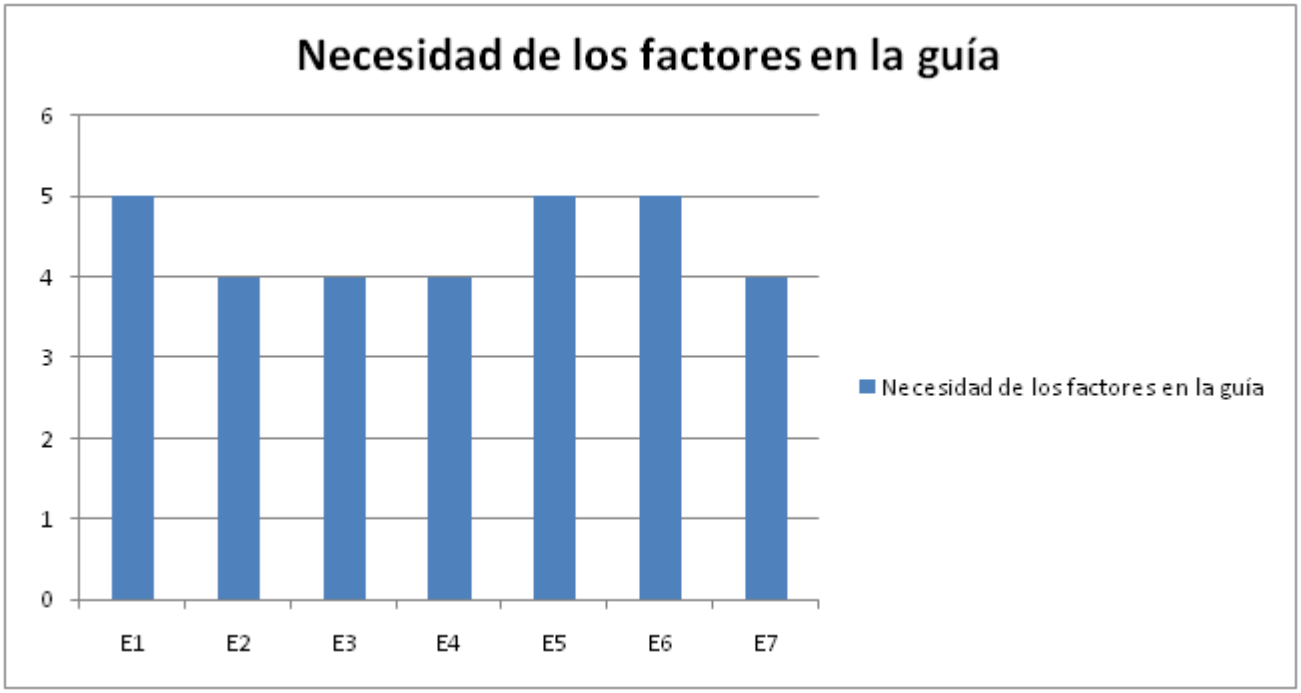


Figura 15. Resultados obtenidos en la Encuesta Realizada

Se puede observar que los expertos coinciden en que la guía de buenas prácticas presenta un 42.85% de especialistas que coinciden que este aspecto la guía cuenta con muy alto en cuanto a necesidad de empleo de la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

3.1.5.4 Determinar la novedad de los factores de la guía

La pregunta cuatro responde a este objetivo, comprueba la novedad de los factores en la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL, los especialistas deben seleccionar su respuesta lo que permitirá afirmar o negar el cumplimiento del objetivo, en el siguiente gráfico se puede observar los resultados logrados:

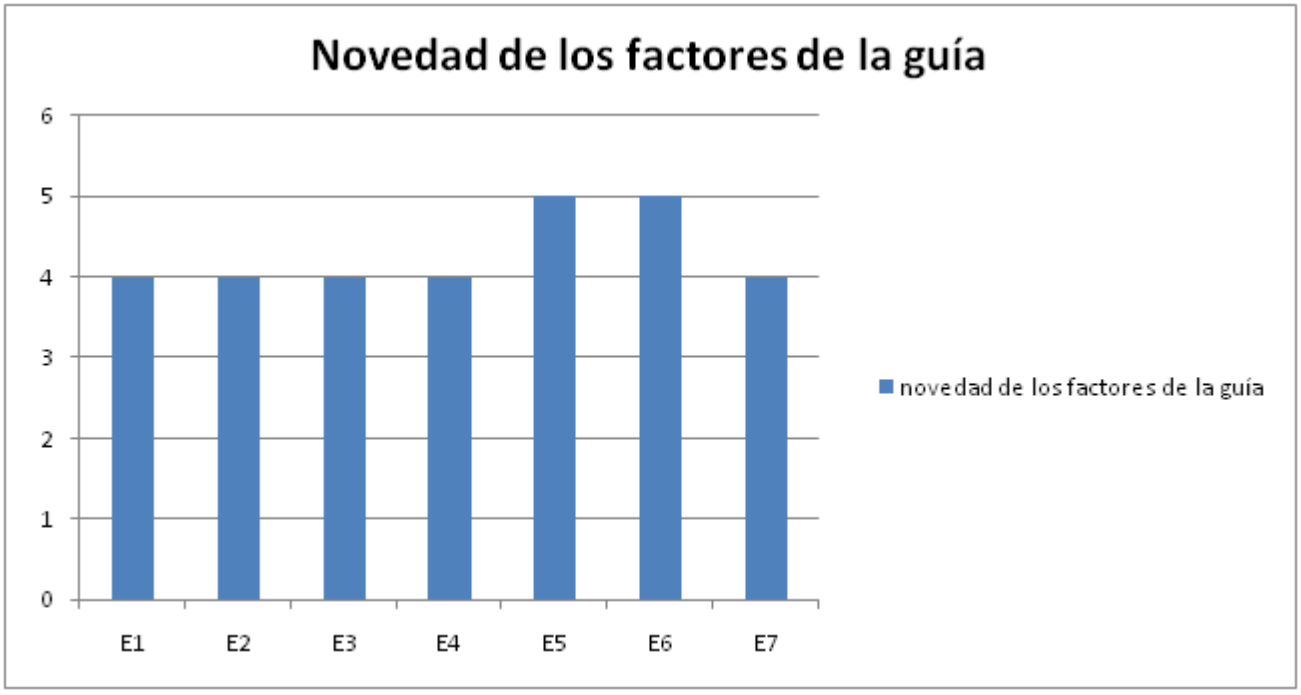


Figura 16. Resultados obtenidos en la Encuesta Realizada

Se puede observar que los expertos coinciden en que la guía de buenas prácticas presenta un 28.57% de especialistas que coinciden que este aspecto la guía cuenta con muy alto en cuanto a conocimientos novedosos y que aportan aspectos importantes los cuales son realmente importantes en el desarrollo de este tipo de aplicaciones opinando lo siguiente.

- Es una guía que presenta aspectos novedosos y muy importantes.

- Los proyectos que desarrollan este tipo de aplicaciones en la Universidad de las Ciencias Informáticas tienen que profundizar en los temas de calidad del producto software.

3.1.5.5 Determinar el nivel de efectividad de los factores de la guía

La pregunta cinco responde a este objetivo, comprueba la efectividad de la utilización de la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL. Los especialistas deben seleccionar su respuesta lo que permitirá afirmar o negar el cumplimiento del objetivo, en el siguiente gráfico se puede observar los resultados logrados:

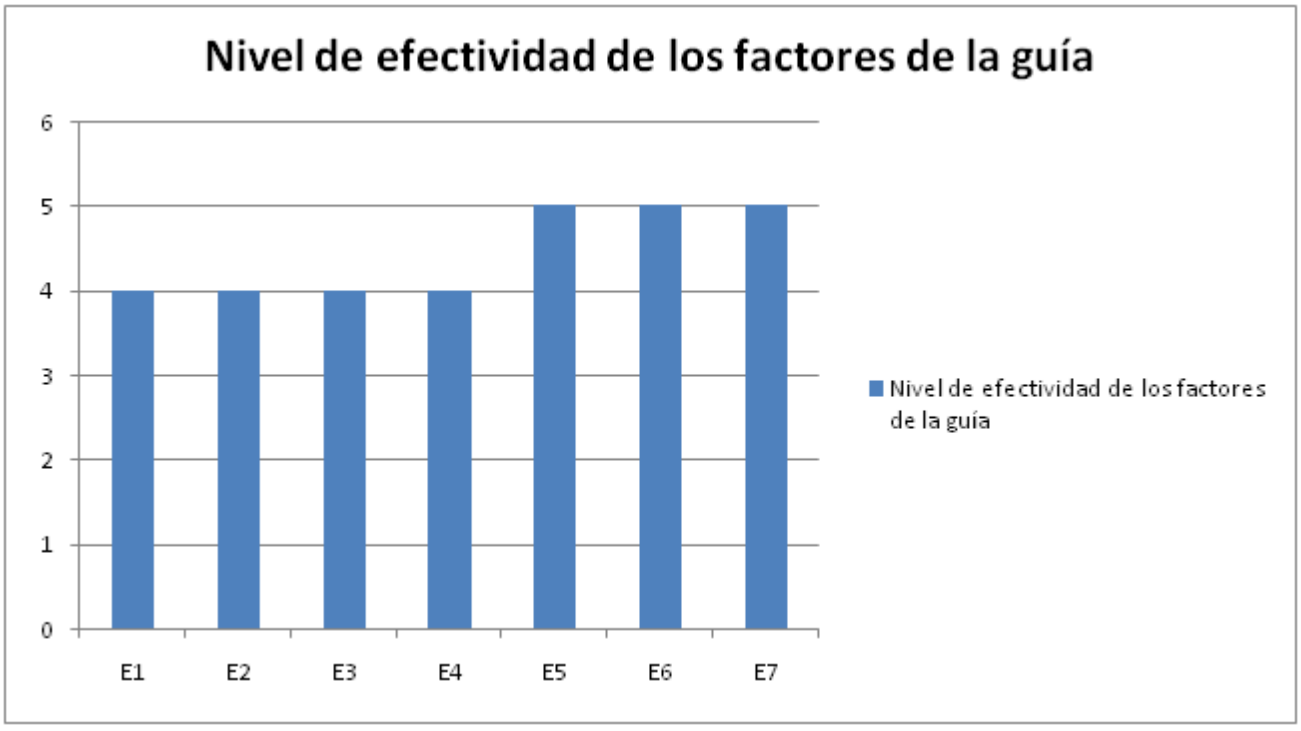


Figura 17. Resultados obtenidos en la Encuesta Realizada

Se puede observar que los expertos coinciden en que la guía de buenas prácticas presenta una efectividad en su utilización presentando mayormente una evaluación de alto y en algunos casos de muy alto.

3.1.5.6 Posibilidad de aplicación de los factores de la guía

La pregunta seis responde a este objetivo, comprueba el grado de posibilidad de aplicación de los factores de la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL en el desarrollo de estas aplicaciones. Los especialistas deben seleccionar su respuesta lo que permitirá definir el grado de cumplimiento del objetivo con muy bajo, bajo, medio, alto y muy alto, en el siguiente gráfico se puede observar los resultados logrados:

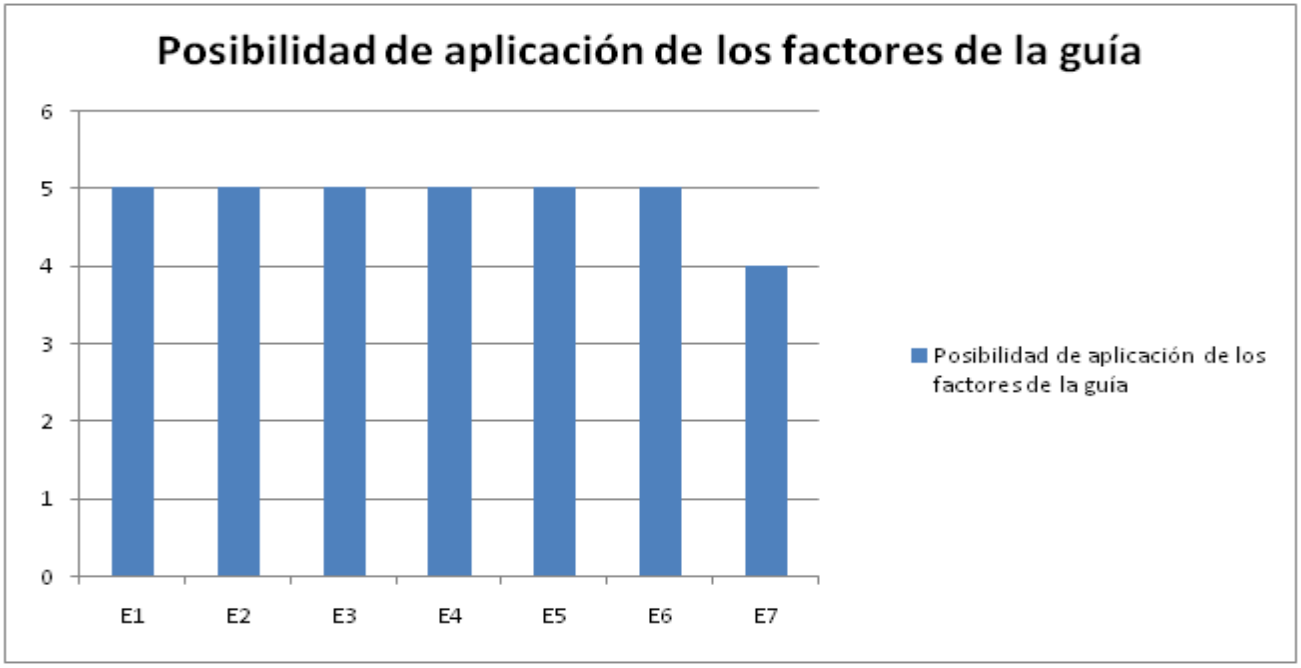


Figura 18. Resultados obtenidos en la Encuesta Realizada

Se puede observar que los expertos coinciden en que la guía de buenas prácticas presenta un 85.71% de especialistas que coinciden que este aspecto la guía cuenta con muy alto en cuanto a la aplicación de los

factores de la guía lo que nos lleva a concluir que este importante aspecto los resultados son satisfactorios.

3.1.5.7. Relación entre los factores de la guía y su ubicación dentro del proceso de desarrollo de software.

La pregunta siete responde a este objetivo, comprueba si los factores de la guía de buenas prácticas para obtener reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL se encuentran dentro de la fase correspondiente en el proceso de desarrollo de software. Los especialistas deben seleccionar su respuesta lo que permitirá afirmar o negar el cumplimiento del objetivo, en el siguiente gráfico se puede observar los resultados logrados:

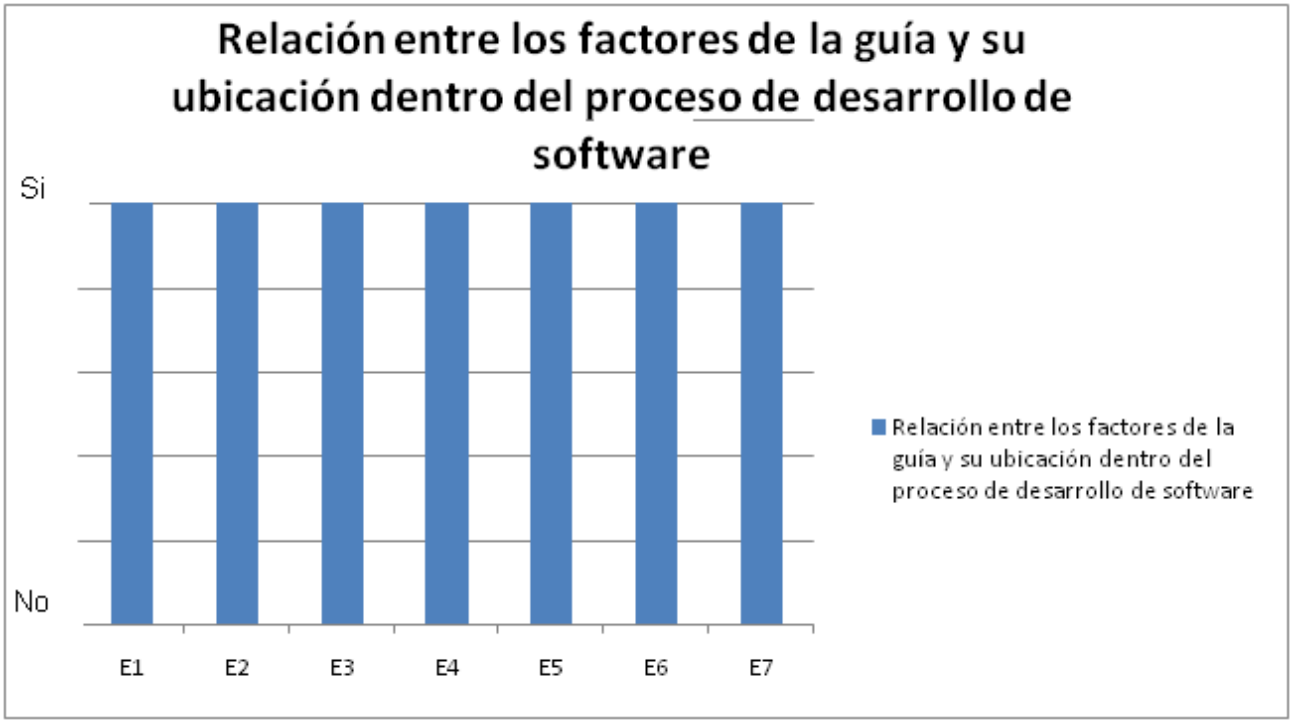
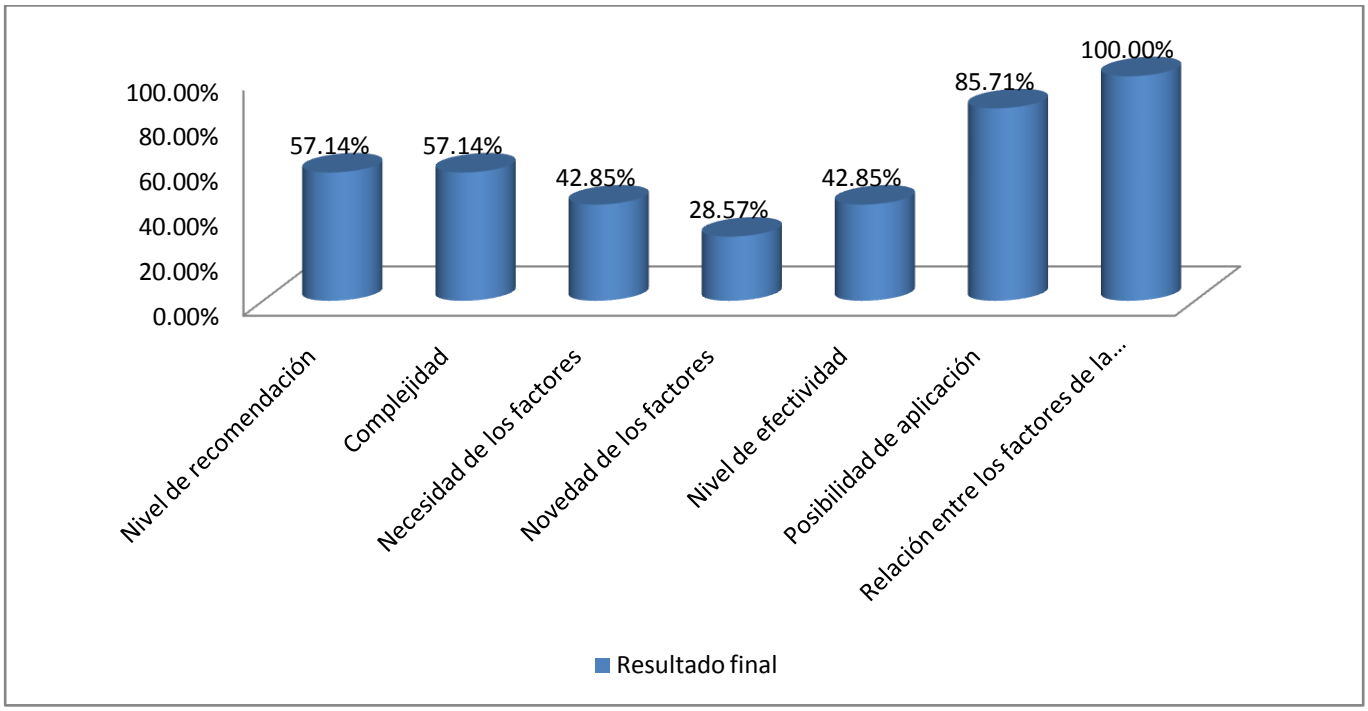


Figura 19. Resultados obtenidos en la Encuesta Realizada

Se puede observar que los expertos coinciden en que la guía de buenas prácticas presenta una buena puntuación en este aspecto teniendo un 100% de aceptación con respecto a los especialistas lo que nos lleva a concluir que este objetivo es satisfactorio.

3.1.5.8. Resultado Final



El porcentaje de respuestas de los especialistas a cada uno de los objetivos propuestos fue positivo, validando así la confección de esta guía, pero antes es necesario tener en cuenta los siguientes resultados:

- En el objetivo Necesidad de Empleo de la Guía dos de los especialistas plantearon que no existe una necesidad de este empleo, sin embargo, en el objetivo de Recomendación de la misma, todos plantean que se debe recomendar y que presenta una gran importancia, por lo que el porcentaje faltante no es significativo en la validación.

- Los especialistas plantearon como observación considero que la presente investigación podría ser muy útil para el proceso de desarrollo de este tipo de soluciones de DWH con PostgreSQL.

- Los especialistas platearon que para lograr una buena gestión de la modularidad en este tipo de soluciones es algo complicado por lo que se debería tener muy en cuenta a la hora de la captura de requisitos para la solución, ya que constituye un punto vital en el ciclo de vida de la misma: “De una captura excelente de requisitos, puede dar como resultado una solución viable, altamente modular, con un nivel de profesionalidad extremo”.

Conclusiones

Se escogió un grupo de especialistas en el tema tratado, la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL en específico la reemplazabilidad y la conformidad con la portabilidad, que permitieron certificar la guía confeccionada. Con la validación presentada se puede apreciar un resultado satisfactorio obtenido de la opinión recogida a través de las encuestas realizadas.

Con la culminación del presente trabajo de diploma los autores llegaron a las siguientes conclusiones:

- La realización de un estudio de varios modelos de calidad, así como todo lo referente a inteligencia de negocio demostró la inminente necesidad de tener estos modelos en el desarrollo de software.
- La realización un estudio de las diferentes métricas de reemplazabilidad y conformidad de la portabilidad demostró que estos aspectos pocas veces tenidos en cuenta son realmente necesarios en el desarrollo de este tipo de aplicaciones.
- Se demostró a través de pruebas documentadas la importancia de los factores propuestos en la guía demostrando que todos estos son necesarios para alcanzar una reemplazabilidad y una conformidad de la portabilidad en las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- La realización de la presente guía constituye un modelo que sirve de apoyo para poder desarrollar el proceso de diseño de software y podría resultar muy efectiva su aplicación.

Con el fin de lograr una mejor gestión de las imágenes digitales en los medios de prensa se recomienda:

- Aplicar la Guía para la obtención de la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.
- Continuar investigando acerca de la reemplazabilidad y la conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse.
- Continuar investigando acerca de los modelos de calidad ISO/IEC 9126, McCall, Boehm.

1. Ballesteros, Jesús Minguet Melián y Juan Francisco Hernández. *La Calidad del Software y su Medida.* Madrid : Editorial Universitaria Ramón Areces, 2003. 84-8004-611-2.

2. Ballesteros, Jesús Minguet Milián y Juan Francisco Hernández. *La Calidad del Software y su Medida.* Madrid : Editorial Universitaria Ramón Areces, 2003. 84-8004-611-2.

3. Gustavo Alberto Ruiz, Alejandro Peña,. *Modelo de Evaluación de Calidad de Software de Acuerdo con la Norma ISO/IEC 9126.* Medellín : s.n., 2006.

4. Mooney, James D. *Issues in the Specification and Measurement of Software Portability.* West Virginia : Dept. of Statistics and Computer Science, 1993.

5. BI. *Informe business intelligence, recopilación de los mejores artículos de inteligencia de negocio del 2006.* abril 2006.

6. Cedeño Trujillo, Alexis. *Data Warehouse para la Estadística Docente basado en software libre.* C.Habana : Tesis de Diplomado CEIS CUJAE, 2005.

7. Ruggia, Raúl. *Diseño de DSS y DW basado en reutilización de especificaciones: Análisis y propuesta de estrategia.* Instituto de computación, Facultad de Ingeniería, Universidad de la República : s.n., 2002.

8. Hedman, Fernando Arruza. *MODELO PARA EL ANÁLISIS DE LA CALIDAD DEL DESEMPEÑO DE ROLES EN LA CARRERA INGENIERÍA INFORMÁTICA.* C. Habana : Tesis de Diploma CEIS CUJAE, 2007.

9. Bill Inmon vs Ralph Kimball. [En línea] http://www.1keydata.com/datawarehousing/ inmon-kimball.html.

10. Hernández, J O, Ramírez, M J.Q y Ferri, C R. *Introducción a la Minería de Datos.* s.l. : Editorial Pearson, 2004.

11. Date, C. *Introducción a los Sistemas de Bases de Datos.* La Habana : Félix Varela, 2003.

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

13. Buschmann, F, y otros. *Pattern – Oriented Software Architecture. A System of Patterns.* Inglaterra : John Wiley & Sons, 1996.

14. Shaw, M y Garlan, D. *Introduction to Software Architectures. New perspectives on an emerging discipline. .* s.l. : Prentice Hall., 1996.

15. Bass, L, y otros. *Attribute Based Architectural Styles.* Pittsburgh : Carnegie Mellon Univesity, 1999.

16. Enhydra Octopus Application. *Enhydra Octopus Application.* [En línea] http://www.enhydra.org/tech/octopus/ index.html.

17. Talend, un nuevo ETL Open Source. *Talend, un nuevo ETL Open Source.* [En línea] http://todobi.blogspot.com/2007/ 01/talend-un-nuevo-etl-open-source.html.

18. Open Source ETL (Extraction, Transform, Load). [En línea] http://www.manageability.org.

19. Architecture Layers of a Mondrian system. *Architecture Layers of a Mondrian system.* [En línea] http://mondrian.pentaho.org..

20. Larman, Craig. *UML y Patrones.* Mexico : Prentice-Hall Hispanoamerica, 2004.

21. El método Delphi. *El método Delphi.* [En línea] http://www.unalmed.edu.co/~poboyca/documentos/documentos1/documentos-Juan%20Diego/Plnaifi_Cuencas_Pregrado/Sept_29/Metodo_delphi.pdf..

22. Izquierdo Moreno, Paula Cecilia y Pascual Plaza, Beatriz. El metodo Delphi. *El metodo Delphi.* [En línea] http://www.google.com.cu/url?sa=t&source=web&ct=res&cd=2&ved=0CCEQFjAB&url=http%3A%2F%2Fwww.uam.es%2Fpersonal_pdi%2Feconomicas%2Frmc%2Fprevision%2Fpdf%2FDELPHI.ppt&rct=j&q=caracteristicas+del+metodo+delphi&ei=C_DtS9q7EMH38AbKoOD9Cg&usg=AFQjCNHOAsdIY8xURJX.

BIBLIOGRAFÍA CONSULTADA

1. Ballesteros, Jesús Minguet Melián y Juan Francisco Hernández. La Calidad del Software y su Medida. Madrid : Editorial Universitaria Ramón Areces, 2003. 84-8004-611-2.

2. Ballesteros, Jesús Minguet Milián y Juan Francisco Hernández. La Calidad del Software y su Medida. Madrid : Editorial Universitaria Ramón Areces, 2003. 84-8004-611-2.

3. Gustavo Alberto Ruiz, Alejandro Peña,. Modelo de Evaluación de Calidad de Software de Acuerdo con la Norma ISO/IEC 9126. Medellín : s.n., 2006.

4. Mooney, James D. Issues in the Specification and Measurement of Software Portability. West Virginia : Dept. of Statistics and Computer Science, 1993.

5. BI. Informe business intelligence, recopilación de los mejores artículos de inteligencia de negocio del 2006. abril 2006.

6. Cedeño Trujillo, Alexis. Data Warehouse para la Estadística Docente basado en software libre. C.Habana : Tesis de Diplomado CEIS CUJAE, 2005.

7. Ruggia, Raúl. Diseño de DSS y DW basado en reutilización de especificaciones: Análisis y propuesta de estrategia. Instituto de computación, Facultad de Ingeniería, Universidad de la República : s.n., 2002.

8. Hedman, Fernando Arruza. MODELO PARA EL ANÁLISIS DE LA CALIDAD DEL DESEMPEÑO DE ROLES EN LA CARRERA INGENIERÍA INFORMÁTICA. C. Habana : Tesis de Diploma CEIS CUJAE, 2007.

Anexo 1. Encuesta realizada a los expertos

Encuesta sobre Reemplazabilidad y Conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL.

La presente encuesta forma parte de un estudio sobre los factores que apoyan la reemplazabilidad y conformidad de la portabilidad de las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse basadas en PostgreSQL para el desarrollo de una guía; dicho estudio tiene como punto de partida la portabilidad de este producto software. Esta guía tiene como objetivo establecer buenas prácticas durante el proceso de desarrollo de software para obtener la portabilidad del mismo. Se espera de su total colaboración para obtener los mejores resultados.

Nombre Encuestado:

Especialidad:

Años Experiencia:

Centro de Trabajo:

Conocimiento mínimo previo:

Reemplazabilidad: Según la ISO/IEC 9126 es la capacidad del producto de software para ser utilizado en lugar de otro producto de software especificado para el mismo propósito en el mismo ambiente. La reemplazabilidad puede incluir atributos tanto de instalabilidad como de adaptabilidad.

Conformidad de la portabilidad: Según la ISO/IEC 9126 es la capacidad del producto software para adherirse a estándares o convenciones relativas a la portabilidad.

Factores	Breve descripción

Correcta selección de los requisitos de portabilidad.	En ocasiones se cometen errores en la captura de los requisitos por parte del equipo de desarrollo. El desarrollo de un buen software debe comenzar con un cuidadoso proceso de análisis de requerimientos.
Requisito de actualización del software.	Consiste en recoger un requisito que sea que el software deba actualizarse de alguna manera. Ya que en ocasiones este proceso es muy complejo o en el peor de los casos no existe la manera de actualizar el software.
Documentación de la reemplazabilidad y conformidad de la portabilidad.	Consiste en documentar todo lo referente a lo que se va a realizar en el desarrollo del software con respecto a reemplazabilidad y conformidad de la portabilidad. Para que los desarrolladores se guíen cuando estén desarrollando el software.
Correcta selección de las herramientas para diseñar y desarrollar el software.	Seleccionar cuidadosamente las herramientas para desarrollar las soluciones informáticas de apoyo a la toma de decisiones que utilizan Data Warehouse. Ya que en ocasiones esto no se tiene en cuenta y se escoge una herramienta sin previo análisis que resulta errónea para el propósito que se persigue.
Interfaz de actualización o upgrades del	Estas son las características de la interfaz de actualización, ej.: intuitiva, sencilla, amigable. El

software.	manejo de las funcionalidades de las soluciones de informáticas debe ser lo más intuitivo posible, de manera que sean muy claras las posibles acciones a llevar a cabo y la manera de hacerlas. Esto es esencial, ya que el sistema está pensado para que sea usado también por usuarios no sean necesariamente expertos en informática.
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.	Este es el grado de al grado de dificultad del usuario al realizarle una actualización al software. El usuario se siente incómodo cuando el proceso de actualización del software es engorroso y complicado lo cual hace que el producto sea menos atractivo a los usuarios.
Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.	Tanto el bajo acoplamiento, como la alta cohesión y la modularidad juegan un papel fundamental en la portabilidad, ya que son de gran importancia para el mantenimiento, control, reutilización de código, corrección de errores y modificación para futuras versiones de las soluciones informáticas de apoyo a la toma de decisiones que utilizan data wharehouse basadas en PostgreSQL.
Correcta selección del lenguaje de programación.	Este factor se refiere al uso de los lenguajes de programación como darle un buen uso a este en dependencia del lenguaje seleccionado y siguiendo las normas de portabilidad asociadas a este

	lenguaje.
Realizar proceso de soporte.	Hace referencia a las forma de obtener las actualizaciones del producto pues esto puede llegar a ser muy complicado ya que en ocasiones es muy difícil obtener estas o no hay formas de obtenerlas. Con una forma establecida de alcanzar estas actualizaciones dejaría de ser un problema ya que el usuario no tendría que buscar complicadas alternativas con el objetivo de lograr una actualización.

Preguntas

1. Valore su conocimiento en el campo de la mantenibilidad en bases de datos, marque con una X en una escala del 0 al 10.

0	1	2	3	4	5	6	7	8	9	10

2. Seleccione el grado que usted crea que han influenciado las siguientes fuentes de conocimiento.

FUENTES DE ARGUMENTACION	Grado de influencia de cada una de las fuentes en sus criterios.

	A (alto)	M (medio)	B (bajo)
Análisis teóricos realizados por usted			
Su experiencia obtenida			
Trabajos de autores nacionales			
Trabajos de autores extranjeros			
Su propio conocimiento del estado del problema en el extranjero			
Su intuición			

Se estima que los siguientes factores pudieran ser importantes para garantizar la portabilidad de un producto software. Otorgue según su apreciación y orden de importancia a cada uno de estos factores una calificación entre 1 y 5 puntos.

Calificación según importancia del factor evaluado.

Muy alto **5 pts.**

Alto **4 pts.**

Medio **3 pts.**

Bajo **2 pts.**

Muy Bajo **1 pts.**

Preguntas:

1. **Establecer el nivel de recomendación de los factores**

(Referido al nivel de recomendación que usted le a cada uno de los factores de la guía confeccionada para otras investigaciones.)

Factores	Peso Otorgado [1 - 5]
Correcta selección de los requisitos de portabilidad.	
Requisito de actualización del software.	
Documentación de la reemplazabilidad y conformidad de la portabilidad.	
Correcta selección de las herramientas para diseñar y desarrollar el software.	
Interfaz de actualización o upgrades del software.	
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.	

Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.	
Correcta selección del lenguaje de programación.	
Realizar proceso de soporte.	

2. Determinar la complejidad de los factores

(Referido al nivel de complejidad de cada uno de los factores.)

Factores	Peso Otorgado [1 - 5]
Correcta selección de los requisitos de portabilidad.	
Requisito de actualización del software.	
Documentación de la reemplazabilidad y conformidad de la portabilidad.	

Correcta selección de las herramientas para diseñar y desarrollar el software.	
Interfaz de actualización o upgrades del software.	
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.	
Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.	
Correcta selección del lenguaje de programación.	
Realizar proceso de soporte.	

3. **Determinar la necesidad de los factores**

(Referido al grado con que se necesita los factores en la guía confeccionada ayudando esta a la solución de problemas en proyectos productivos.)

Factores	Peso Otorgado [1 - 5]
Correcta selección de los requisitos de portabilidad.	
Requisito de actualización del software.	
Documentación de la reemplazabilidad y conformidad de la portabilidad.	
Correcta selección de las herramientas para diseñar y desarrollar el software.	
Interfaz de actualización o upgrades del software.	
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.	
Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.	
Correcta selección del lenguaje de programación.	
Realizar proceso de soporte.	

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

(Referido a cuan novedosa usted considera que son los factores a partir del aporte de conocimientos para alcanzar los objetivos que se persiguen con esta.)

Factores	Peso Otorgado [1 - 5]
Correcta selección de los requisitos de portabilidad.	
Requisito de actualización del software.	
Documentación de la reemplazabilidad y conformidad de la portabilidad.	
Correcta selección de las herramientas para diseñar y desarrollar el software.	
Interfaz de actualización o upgrades del software.	
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.	
Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.	
Correcta selección del lenguaje de programación.	

Realizar proceso de soporte.	

5. Determinar el nivel de efectividad de los factores

(Referido al grado de efectividad que usted considera que poseen los factores de la guía a partir de las buenas prácticas establecidas para el proceso propuesto.)

Factores	Peso Otorgado [1 - 5]
Correcta selección de los requisitos de portabilidad.	
Requisito de actualización del software.	
Documentación de la reemplazabilidad y conformidad de la portabilidad.	
Correcta selección de las herramientas para diseñar y desarrollar el software.	
Interfaz de actualización o upgrades del software.	
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.	

Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.	
Correcta selección del lenguaje de programación.	
Realizar proceso de soporte.	

6. Posibilidad de Aplicación de los factores

(Referido a la capacidad que usted considere con que cuentan los factores de ser aplicados en el proceso de desarrollo de estos tipos de software.)

Factores	Peso Otorgado [1 - 5]
Correcta selección de los requisitos de portabilidad.	
Requisito de actualización del software.	
Documentación de la reemplazabilidad y conformidad de la portabilidad.	
Correcta selección de las herramientas para diseñar y desarrollar el software.	

Interfaz de actualización o upgrades del software.	
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.	
Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.	
Correcta selección del lenguaje de programación.	
Realizar proceso de soporte.	

7. Relación entre los factores y su ubicación dentro del proceso de desarrollo de software

(Referido a si usted considera que los factores se encuentren ubicados en su correspondiente fase del desarrollo de software.)

Factores	Ubicación dentro del PDS	Si	No
Correcta selección de los requisitos de portabilidad.			

Requisito de actualización del software.			
Documentación de la reemplazabilidad y conformidad de la portabilidad.			
Correcta selección de las herramientas para diseñar y desarrollar el software.			
Interfaz de actualización o upgrades del software.			
Reducir el grado de dificultad al realizar el proceso de actualizaciones o upgrades en el software.			
Uso de patrones y estilos de diseño y arquitecturas para lograr bajo acoplamiento, alta cohesión y modularidad.			
Correcta selección del lenguaje de programación.			
Realizar proceso de soporte.			

8. **Identificar aspectos erróneos y recomendaciones que permitan mejorar la propuesta. Para ello, en cada pregunta formulada los expertos podían realizar sus observaciones**

Anexo 2. Coeficiente de Kendall obtenido

Kendall's W Test

Ranks

	Mean Rank
nivel_recomendacion	4,57
complejidad	3,71
necesidad	4,14
novedad	4,57
nivel_efectividad	4,14
aplicacion	5,86
relacion	1,00

Test Statistics

N			7
Kendall's W[a]			,671
Chi-Square			28,174
df			6
Asymp. Sig.			,000
Monte Carlo Sig.	Sig.		,000[b]
	95% Confidence Interval	Lower Bound	,000
		Upper Bound	,000

a. Kendall's Coefficient of Concordance

b. Based on 10000 sampled tables with starting seed 299883525.

Figura 20. Resultado obtenido por el programa SPSS 13.0