ASEGURAMIENTO DE LA CALIDAD EN LOS SISTEMAS DE INFORMACIÓN

Autor: JOSÉ GILDARDO HERRERA SÁNCHEZ

Tesina presentada para obtener el título de:

LICENCIADO EN SISTEMAS COMPUTARIZADOS [SIC]

Nombre del asesor:

SERGIO FRANCISCO BARRAZA IBARRA

UNIVERSIDAD VASCO DE QUIROGA

LICENCIATURA EN SISTEMAS COMPUTARIZADOS

"ASEGURAMIENTO DE LA CALIDAD EN LOS SISTEMAS DE INFORMACIÓN"

T E S I N A

Que para obtener el Título de:
LICENCIADO EN SISTEMAS COMPUTARIZADOS

Presenta:
JOSÉ GILDARDO HERRERA SÁNCHEZ

Asesor:
ING. Y M.A. SERGIO FRANCISCO BARRAZA IBARRA

CLAVE 16PSU0014Q
ACUERDO 952006

Morelia, Mich. Diciembre 1998

UNIVERSIDAD VASCO DE QUIROGA

LICENCIATURA EN SISTEMAS COMPUTARIZADOS

"ASEGURAMIENTO DE LA CALIDAD EN LOS SISTEMAS DE INFORMACIÓN"

T E S I N A

Que para obtener el Título de:
LICENCIADO EN SISTEMAS COMPUTARIZADOS

Presenta:
JOSÉ GILDARDO HERRERA SÁNCHEZ

Asesor:
ING. Y M.A. SERGIO FRANCISCO BARRAZA IBARRA

CLAVE 16PSU0014Q
ACUERDO 952006

Morelia, Mich. Diciembre 1998

Dedico este esfuerzo a J. Gildardo Herrera A.,
quien en vida supo ser un padre amoroso
y en su ausencia física
es luz que guía mi grata existencia.

CONTENIDO

INTRODUCCIÓN

El planeta tierra se originó hace 4500 millones de años y el hombre apareció en ella 2 millones de años después.

La edad de piedra se extiende durante muchos milenios y se subdivide generalmente en dos grandes etapas: *paleolítico y neolítico.*

La antigua edad de piedra constituye el período fundamental en que la especie humana se transformó física y mentalmente hasta alcanzar el estadio del Homo Sapiens. Durante esta era de transformación evolutiva, el hombre descubrió el fuego y utilizó las primeras herramientas hechas con piedra, y tal vez podamos encontrar algunas otras pocas manifestaciones de progreso.

La edad de los metales comprende los períodos de cobre, bronce y hierro, también abarca varios milenios. En esta era las manifestaciones evolutivas de la vida humana se expresaron consistentemente. El hombre empezó a sistematizar su conocimiento, lo que dio paso a la construcción de las bases científicas; también empezó a extenderse el uso de tecnologías.

Cabe hacer notar que en cada uno de los períodos en que está dividida la evolución, han transcurrido miles de años. Estos períodos tan largos así los comprende la historia, porque las transformaciones eran escasas hasta el momento en que el hombre empieza a utilizar la escritura; a partir de allí, las transformaciones se han venido presentando en la sociedad humana con una velocidad logarítmica; de tal modo, que en el estadio de la vida moderna los avances científicos y tecnológicos ya no se pueden cuantificar.

Es muy común escuchar en los medios de información, que vivimos en la era de la cibernética; pero resulta lógico pensar que si las transformaciones en la actualidad son tan vertiginosas, los períodos históricos habrán de ser de una brevedad proporcional.

Tal vez lo que cause más asombro para el hombre moderno sea el progreso logrado en el campo de la comunicación y la informática. Estos avances han "reducido" el mundo a una aldea.

En esa perspectiva, el universo del software en general y los sistemas de información en particular, son la punta de lanza de la ciencia y tecnología de la vida moderna. Así que las aplicaciones computarizadas ante el despertar del segundo milenio reclaman un componente que se ha vuelto indispensable "CALIDAD".

Con esa preocupación se ha construido la siguiente propuesta, aplicando el mejor esfuerzo, una breve experiencia profesional, una intensa búsqueda en obras especializadas; pero sobre todo, el invaluable aprendizaje que me ha dejado mi entrañable Universidad.

En el capítulo de los Antecedentes se expone brevemente la manera en que ha venido evolucionando el software desde un punto de vista histórico. Se caracterizan cuatro etapas; se destacan los problemas más relevantes y que inciden en todas las áreas de desarrollo del software.

La identificación de ese núcleo problemático, es lo que fundamenta la necesidad de crear un software de alta calidad para iniciar, tal vez no solo una nueva etapa evolutiva del software, sino una nueva era en la evolución de la sociedad humana. "*La era de la Cibernética y de la Excelencia*".

En los Antecedentes se aborda también la evolución de la calidad en la productividad de bienes o servicios. Se explica el origen de la Organización Internacional para la Normalización (ISO); la manera en que funciona; y de cómo está integrada.

Analizar el proceso evolutivo nos sirve para comprender de donde proviene la necesidad de ofrecer una mejor calidad del producto o servicio; pues la calidad es actualmente factor indispensable en la supervivencia de cualquier empresa. Después se hace un deslinde conceptual de Aseguramiento de la Calidad y Sistema de Calidad. Se concluye este espacio destacando la creciente demanda de la calidad, que por cierto abarca todo el entorno del desarrollo científico y tecnológico.

Al plantearnos un Objetivo para este trabajo, también nos hemos planteado una Expectativa.

Se ha pretendido construir un modelo metodológico que facilite los procesos evaluatorios para la producción de software, con la firme certeza de que si se

mejora la calidad de la evaluación; incuestionablemente que se mejorará la calidad del producto.

El capítulo cuarto abarca el enfoque filosófico que le da sustentación a la búsqueda de la excelencia en la productividad en general.

Como se puede apreciar en su despliegue, se ha optado por un enfoque participativo, que requiere relaciones horizontales en su estructura orgánica; esto dependiendo de las dimensiones del proyecto de trabajo de que se trate.

Desde el punto de vista teórico se llega a la conclusión de que el número de elementos de cualquier equipo de trabajo, no limita las posibilidades de lograr productividad con calidad, si uno se adhiere a dicho enfoque.

En el capítulo quinto se define la calidad del software, refiriéndose al conjunto de cualidades que lo conforman y que determinan su utilidad y existencia; lo que implica una serie de factores como son: *eficiencia, flexibilidad, corrección, confiabilidad, portabilidad, usuabilidad, seguridad e integridad*, entre otros.

Con la convicción de que la calidad del software es medible; en este espacio se busca caracterizar un modelo que pueda garantizarnos la obtención de un software con calidad. Lo fundamental en este afán es tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software. Desde luego que es necesario utilizar una metodología o procedimientos estándares para el análisis, diseño, implementación y auditoría. También se recomienda definir una política de trabajo que comprenda tres aspectos básicos: *tecnológico, administrativo y ergonómico*.

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura. Para el aseguramiento de la calidad es necesario su control y evaluación.

En cuanto al control, es indispensable seleccionar los índices de calidad y estructurar el proceso de control.

Respecto a la evaluación, se puede decir que es la columna vertebral de este trabajo, consecuentemente el momento más consistente para el aseguramiento de la calidad del software. Así que aquí se dan respuestas a preguntas, tales como:

- *¿Qué comprende la evaluación del software?*
- *¿En qué momento del proceso de desarrollo del software es conveniente evaluar ?*
- *¿Es viable verificar cada una de las combinaciones posibles de entradas, estados y salidas?*
- *¿Qué tantas pruebas deben hacerse y dónde se deben aplicar?*
- *¿Qué tanto tiempo y dinero debe invertirse a la aplicación de las pruebas?*

Es importante en cualquier trabajo técnico precisar la perspectiva en que son empleados los términos, por lo que resulta necesario presentar los conceptos alrededor de los cuales gira la Evaluación de la Calidad de Software: así que, el capítulo sexto se ha destinado para el Marco Conceptual y los términos que se deslindan son: *proceso de software, producto de software, proyecto, versión, nueva versión, versión por cambios de funcionalidad, versión de mantenimiento, regla, estándares, estilo, factor de calidad y métricas.*

Ya situados en el terreno de la sección propositiva de este documento recepcional, resulta prudente cuidar hasta el más mínimo detalle, por lo tanto, en el séptimo capítulo se estructura una Guía de Evaluación para el Aseguramiento de la Calidad del Software, cuyo propósito es presentar los lineamientos para facilitar la evaluación, se proponen las preguntas que deberán contestarse, clasificadas en cada una de las etapas del proceso de producción del software y también se explica en qué consisten dichas etapas.

El capítulo octavo comprende el cuestionario de evaluación para el aseguramiento de la calidad del software con las preguntas ordenadas en cada una de las etapas que rigen el proceso productivo del software: *Análisis, Diseño, Implementación y Auditoría*. Cada etapa incluye sus interfaces y por consiguiente sus preguntas concretas.

En el capítulo nueve se agrupan los factores de calidad y las métricas para evaluarlos. Los once factores de calidad son presentados con su correspondiente referencia. Por otro lado, las veintiún métricas definidas para evaluar los factores de calidad se ordenan con las preguntas signadas en el cuestionario de evaluación para el aseguramiento de la calidad del Software. Este espacio se cierra precisando el procedimiento de evaluación que comprende seis pasos:

1) Contestar las preguntas del cuestionario.
2) Evaluar cada una de las métricas.
3) Ponderar cada uno de los factores de calidad.

4) Obtener porcentajes para cada uno de los factores de calidad.
5) Evaluar los factores de calidad.
6) Obtener la evaluación final de calidad del producto de software.

Finalmente, el capítulo diez se ha destinado para concentrar el reporte de los resultados del proceso de aseguramiento de la calidad en los productos de software, y comprende ocho puntos importantes que hay que considerar para tal efecto:

1) Para quién y cuándo se requiere.
2) Objetivos del reporte .
3) Recomendaciones para el responsable.
4) Formato del reporte.
5) Incluir referencias a los papeles de trabajo.
6) Revisión previa del reporte.
7) Revisión externa del reporte.
8) Preparación del reporte final.

1. ANTECEDENTES

1.1. Evolución Histórica del Software

En la actualidad el Software juega un doble papel; es un producto y al mismo tiempo, el vehículo para hacer entrega de un producto. Como producto hace entrega de la potencia informática del hardware informático. Como vehículo utilizado para hacer entrega del producto, el software actúa como base de control de la computadora (sistemas operativos), la comunicación de información (redes), y la creación y control de otros programas (herramientas de software y entornos).

Sin embargo, es muy importante voltear la mirada hacia el marco histórico de su desarrollo para explicarnos con más claridad la manera en que se ha venido dando, hasta jugar ese doble papel que ya mencionamos.

El proceso evolutivo del software se inicia prácticamente a partir de la segunda mitad del siglo XX. Los distintos especialistas nos hablan de cuatro o cinco generaciones; pero casi todos coinciden en caracterizarlas de la siguiente manera:

- *1950-1965*. En los primeros años las características fundamentales eran: orientación por lotes (batch), diseño a la medida de cada aplicación y tenía una distribución relativamente pequeña. El software como producto estaba saliendo a la luz pública; se hacia a medida y la mayoría del software se desarrollaba y era utilizado por la misma persona u organización.
- *1965-1975*. La segunda era se caracterizó por la multiprogramación y los sistemas multiusuarios, lo que derivó nuevos conceptos de interacción hombre-máquina. Los sistemas de tiempo real podían recoger, analizar y transformar datos de múltiples fuentes, controlando así los procesos y produciendo salidas en milisegundos en lugar de minutos. Los avances en los dispositivos de almacenamiento en línea condujeron a la primera generación de sistemas de gestión de base de datos. Es importante destacar que en esta era el software se establece como producto; empezaba a tener una amplia distribución en un mercado multidisciplinar. Los programas se distribuían para computadoras grandes y para minicomputadoras. No menos importante es señalar que los productos de software comprados al exterior incorporaban cientos de miles de nuevas sentencias y todos esos programas, todas esas sentencias fuente tenían que ser corregidos cuando se

detectaban fallas, modificados cuando cambiaban los requisitos de los usuarios o adaptados a nuevos dispositivos hardware.

A esas actividades se les llamo "el mantenimiento del software". El esfuerzo por mantener un software actualizado comenzó a absorber recursos excesivos.

- *1975-1983.* En la tercera era se distinguen las redes de área local y área global. Las comunicaciones digitales y la demanda de acceso "instantáneo" a los datos, iba creciendo.

Esta etapa tocó su fin con la llegada y el extenso uso de los microprocesadores. Los microprocesadores han generado un extenso grupo de productos "inteligentes" (en automóviles, hornos de microondas, robots industriales, etc.); pero más importante han sido los aplicados a la computadora personal.

Es notable cómo el software se vuelve así, en pocos años un producto accesible a todo el público.

- *1987-1998.* La cuarta era de la evolución del software se aleja de las computadoras individuales y de los programas de computadoras, dirigiéndose al impacto colectivo de las computadoras y del software. Se presentan las siguientes características: sistemas personales potentes, tecnologías orientadas a objetos, sistemas expertos, redes neuronales artificiales, computación en paralelo y redes de computadoras.

Las redes de información en todo el mundo proporcionan una infraestructura que juega el rol de "súper autopista de información" y una "conexión del ciberespacio"; de hecho Internet se puede ver como un "software" al que pueden acceder usuarios individuales.

Los Sistemas de Información ya son una herramienta indispensable en la economía del mundo. No existe una industria ó empresa, por pequeña que sea, que no recurra a la aplicación de los sistemas de información. Sin embargo el desarrollo de los sistemas de información se encuentran en un cuello de botella; hay cientos de aplicaciones basadas en software en una situación critica y necesitan ser renovadas:

- Las aplicaciones de sistemas de información escritas, hace veinte años han sufrido 40 generaciones de cambios y ahora son virtualmente imposibles de

mantener; incluso, ahora, la más pequeña modificación puede hacer que falle todo el sistema.

- Las aplicaciones de ingeniería que se utilizan para generar datos críticos de diseño, y que, sin embargo, a pesar de su edad y estado de conservación, realmente no se entienden. Nadie tiene un conocimiento detallado sobre la estructura interna de esos programas.
- Sistemas empotrados (usados para controlar plantas industriales, tráfico aéreo y fabricas, entre sus cientos de aplicaciones) que parecen extraños y a veces tienen un comportamiento inexplicable; pero que no se pueden poner fuera de servicio porque no hay nada para reemplazarlos.

No será suficiente "reparar" lo que esta mal y dar una imagen moderna a estas aplicaciones, pues los problemas no se limitan al software que no "funciona correctamente". El mal abarca los problemas asociados a cómo desarrollar el software, cómo mantener el volumen cada vez mayor de software existente y cómo esperar mantenernos al corriente de la demanda creciente de software.

Ante tal situación, donde los problemas inciden en todas las áreas de desarrollo del software, el reto mayor que se presenta en el umbral del siglo XXI para la ingeniería del software es proporcionar un marco de trabajo para construir software con mayor calidad.

1.2. Evolución Histórica de la Calidad.

La demanda de la calidad en la sociedad actual ha tenido un comportamiento evolutivo que parte de la Segunda Guerra Mundial y fue precisamente el sector de la industria militar quien se interesó primero, de manera formal, en una productividad de calidad.

La idea de producir bienes o servicios con calidad se extendió rápidamente en muchos países del mundo de manera paradigmática y en todos los sectores de la productividad humana.

En 1947 distintos países europeos aspiraban al reconocimiento mundial de la más alta calidad de sus productos, pues con ello buscaban facilitar el comercio exterior. Así que, por consenso, se integró en Ginebra, Suiza la International Standar Organization (ISO).

La Organización Internacional para la Normalización (ISO) es la institución responsable para establecer los estándares de calidad a nivel mundial, con una agrupación hasta la fecha de 91 países y concentran un total aproximado de 3500 empresas. ISO esta integrada por aproximadamente 180 comités técnicos, cada uno de los cuales es responsable de la normalización para cada área de especialidad, desde, por ejemplo, asbestos hasta zinc. El propósito de ISO es promover el desarrollo de la normalización para fomentar a nivel internacional el intercambio de bienes y servicios y para el desarrollo de la cooperación en actividades económicas, intelectuales, científicas y tecnológicas. El resultado del trabajo técnico dentro de ISO se publica en forma final como normas internacionales.

En México corresponde a la Secretaria de Comercio y Fomento Industrial (SECOFI), a través de la Dirección General de Normas (DGN) la representación de ISO. En nuestro país existen alrededor de 600 empresas que cuentan con la certificación de ISO 9000 y aproximadamente 1400 en proceso de certificación.

El común de los autores de libros especializados hablan de los cambios que ha sufrido el término calidad, descubriendo la acepción que se ha tenido en cada una de las etapas históricas de la productividad, y cuales han sido los objetivos a perseguir.

ETAPA	CONCEPTO	FINALIDAD
Artesanal	• Hacer las cosas bien independientemente del costo o esfuerzo necesario para ello.	• Satisfacer al cliente • Satisfacer al artesano, por el trabajo bien hecho • Crear un producto único
Revolución Industrial	• Hacer muchas cosas no importando que sean de calidad (se identifica producción con calidad)	• Satisfacer una gran demanda de bienes • Obtener beneficios
Segunda Guerra Mundial	• Asegurar la eficiencia del armamento sin importar el costo, con la mayor y más rápida producción. (Eficacia + Plazo = Calidad)	• Garantizar la disponibilidad de un armamento eficaz en la calidad y el momento preciso.
Postguerra (Japón)	• Hacer las cosas bien a la primera	• Minimizar costos mediante la calidad • Satisfacer al cliente • Ser competitivos
Postguerra (resto del mundo)	• Producir, cuanto más mejor	• Satisfacer la gran demanda de bienes causados por la guerra
Control de Calidad	• Técnicas de inspección en producción para evitar la salida de bienes defectuosos	• Satisfacer las necesidades técnicas del producto
Aseguramiento de la Calidad	• Sistemas y procedimientos de la organización para evitar que se produzcan bienes defectuosos.	• Satisfacer al cliente • Prevenir errores • Reducir costos • Ser competitivo
Calidad Total	• Teoría de la administración empresarial centrada en la permanente satisfacción de las expectativas del cliente.	• Satisfacer tanto al cliente externo como interno • Ser altamente competitivo • Mejora continua

Analizar el proceso evolutivo nos sirve para comprender de donde proviene la necesidad de ofrecer una mejor calidad del producto o servicio. La calidad no es únicamente un requisito esencial del producto sino que en la actualidad es un factor estratégico clave del que dependen la mayor parte de las organizaciones, no sólo para obtener su posición en el mercado sino para asegurar su supervivencia.

El Aseguramiento de la Calidad nace como una evolución natural del Control de Calidad que resultaba limitado y poco eficaz para prevenir la aparición de defectos. Para ello, se hizo necesario como forma de vida y que; en todo caso, sirviera para anticipar los errores antes de que estos se produjeran. Un Sistemas de Calidad se centra en garantizar que lo que ofrece una organización cumpla con las especificaciones establecidas previamente por la empresa y el cliente, asegurando una calidad continua a lo largo del tiempo. Las definiciones, según la Norma ISO son:

Aseguramiento de la Calidad:

Conjunto de acciones planificadas y sistemáticas, implementadas en el Sistema de Calidad, que son necesarias para proporcionar la confianza adecuada de que un producto satisfaga los requisitos dados sobre la calidad.

Sistema de Calidad:

Conjunto de la estructura, responsabilidades, actividades, recursos y procedimientos de la organización de una empresa, que ésta establece para llevar a cabo la gestión de su calidad.

De acuerdo con Cadena Gómez E., Miembro destacado de la Sociedad de Servicios en Informática, que dice en su libro "ISO 9000 Una Visión General": que cuando surgieron las normas internacionales de ISO 9000 sobre sistemas de aseguramiento de calidad, el sector de desarrollo de software fue uno de los menos activos en su adopción; pues había una enorme dificultad en la interpretación y aplicación de estas normas, tan genéricas, al desarrollo de software, y los acuerdos alcanzados en dichos estándares de calidad no contenían muchos factores críticos que influyeran en la calidad del desarrollo de software y que; por lo tanto no podrían ser suficientes para delinear un estándar de calidad. Dada esta situación, surgió la necesidad de crear organismos de certificación en calidad para desarrolladores de software. Así surgió el esquema Británico Tick it, de alcance internacional, aplicado a la certificación de sistemas de calidad para la industria del software.

Los objetivos primordiales del esquema Tick it fueron, además de desarrollar un sistema de certificación aceptable en el mercado, estimular a los desarrolladores de software a implementar sistemas de calidad, dando la dirección y guías necesarias para tal efecto. El objetivo del certificado Tick it es demostrar que las practicas necesarias para asegurar la calidad durante el desarrollo de software existen y son verificables.

Se ha evolucionado mucho en los enfoques de la calidad. A finales de los sesenta cada vez que se quería afrontar la mejora del software todo se dirigía a la mejora del código, dejando olvidado todo lo que le rodeaba. Ya en los años ochenta se comenzó a tener en cuenta los aspectos de especificaciones, diseño, evaluación y gestión del software. Pese a estos cambios, persiste la problemática ya que el nivel de complejidad del software ha ido aumentando y no se alcanzan los niveles de calidad deseados.

En la compleja sociedad de hoy en día, una norma de calidad tiene las características de una ley industrial. La fábrica moderna es una comunidad industrial compleja, con leyes propias que indican lo que es correcto y lo que no es. Una de estas leyes industriales es la Norma de Calidad.

El Aseguramiento de la Calidad se ha vuelto en la postrimería del siglo XX un principio y un fin imperativos que rigen el desarrollo científico y tecnológico de la vida moderna; y puesto que estas dos esferas giran actualmente entorno a los sistemas de información, consecuentemente el aseguramiento de la calidad en software es de la más relevada prioridad.

2. OBJETIVO.

Ofrecer a los desarrolladores de software un modelo metodológico que facilite los procesos evaluatorios.

3. EXPECTATIVA.

Mejorar la calidad de la evaluación para que de como resultado una alta calidad de la producción del software.

4. ENFOQUE FILOSOFICO PRO-EXCELENCIA

En la búsqueda del aseguramiento de la calidad, hay un punto de vista en el que convergen todas las opiniones de los especialistas y es en el compromiso que puedan tener todos los implicados en los procesos productivos. Esto significa para una empresa o equipo de trabajo cualquiera, con aspiraciones por la excelencia, la necesidad de crear un sistema organizado que contenga toda la información necesaria de procedimientos, métodos, formatos, etc. que se deben observar en todas y cada una de las áreas que forman la cadena del proceso productivo, para garantizar el bien o servicio de manera oportuna y al costo justo, en forma consistente.

Establecer las normas es algo relativamente fácil, la dificultad estriba en lograr una respuesta entusiasta; como ya lo dijimos, por parte de los implicados para conjuntar el mejor esfuerzo de cada uno de ellos. Para ello no hay recetas o fórmulas concretas; lo que si resulta indispensable es cambiar hábitos y costumbres que nos han mantenido en un nivel de subdesarrollo y estancamiento, crear una nueva cultura de trabajo y comportamiento, donde predomine la dedicación, la eficiencia y la dignidad humana, y esta cultura difundirla a todos los que de alguna manera dependan o se relacionen con nosotros, con el ejemplo.

Hay varias posturas filosóficas al respecto, pero resulta interesante la que asume SOSA Pulido D. en su libro Calidad Total, donde habla de "Los Cuatro Pilares de la Calidad".

PRIMERO: QUE TODOS SEPAMOS LO QUE DEBEMOS LOGRAR EN NUESTRO TRABAJO.

Se debe tener una idea clara de las metas y la razón de ser de nuestro puesto. Además tenemos la necesidad de que reflexionemos, o hagamos reflexionar a los demás, acerca de la razón de ser de nuestros puestos, y de la contribución que aportamos para los resultados de nuestros departamentos.

SEGUNDO: QUE TODOS NOS CAPACITEMOS PARA HACER NUESTRO TRABAJO BIEN HECHO SIEMPRE DESDE LA PRIMERA VEZ.

Este paso se divide en dos:

- **Capacitación**. Para cada uno de los puestos, se debe capacitar al personal, que se hagan especialistas en su trabajo; crear una conciencia ambiciosa en el aprendizaje, ya que probablemente esos conocimientos nos lleven a la excelencia, recordando siempre que: el trabajo no desarrolla gente, da la oportunidad de que la gente se desarrolle. Este punto es de trascendental importancia porque tal esfuerzo le da la oportunidad a la empresa de mantener actualizado a su personal.

  Para bien de nuestra sociedad se observa una intención recurrente a los procesos de capacitación; pero todavía no se tocan los niveles mínimos deseados para una comunidad productiva, que aspira al aseguramiento de la calidad, pues hay muchas empresas que no quieren invertir en la capacitación por considerarlo, un gasto inútil.

- **Estandarización**. En este paso se deben redactar los métodos y procedimientos para tener un documento (estándar de trabajo) por escrito que se pueda consultar; esto nos servirá para nunca más volver a incurrir en las fallas que ya hemos superado.

TERCERO: QUE TENGAMOS LO NECESARIO PARA HACER NUESTRO TRABAJO BIEN HECHO SIEMPRE A LA PRIMERA VEZ.

Se debe borrar la costumbre de la "improvisación" ya que en la mayoría de los casos las improvisaciones sirven a medias o por poco tiempo; por lo tanto, es recomendable "que todo el personal cuente con los medios físicos y técnicos apropiados para hacer su trabajo bien siempre desde el principio".

Los medios físicos no necesariamente tienen que ser de lo más sofisticados, de los más caros, ni de los más modernos, simplemente que sean los convenientes y los más adecuados.

CUARTO: QUE TODOS TENGAMOS EL DESEO DE HACER UN BUEN TRABAJO.

Esta condición o situación puede lograrse sólo si hay un ambiente laboral donde la comunicación sea armoniosa y fluida sin importar los niveles jerárquicos. Aquí la motivación juega un papel relevante facilitando los procesos integradores. Es esencial que todos los miembros se sientan incluidos para el proyecto empresarial del que se trate.

Se puede contar con toda la tecnología de punta para la realización de un proyecto y contar con los suficientes recursos financieros pero no es esto lo que hace grande a una empresa, sino el esfuerzo conjunto de todos los trabajadores; es la gente la que mueve los equipamientos.

Las técnicas motivacionales son muy diversas, lo pertinente es eliminar las que buscan la eficiencia relacionando tiempo y costo; es decir, producir más en el menor tiempo posible. En cambio es inclinarse por aquellas que contemplen el desarrollo humano; esto, con seguridad nos puede llevar hacia una practica laboral más participativa.

El grado de dificultad para dar cumplimiento a estos cuatro principios, esta en razón de las dimensiones de la empresa o proyecto de que se trate. Es evidente que entre más grande sea el proyecto de trabajo, requerirá una planta más numerosa de personal; y entre más recursos humanos se concentren en una empresa, más difícil resulta conjuntar esfuerzos.

Las dimensiones de los proyectos de trabajo, concretamente en el campo del software se especifican de la siguiente manera:

PROYECTO	PERSONAS	DURACIÓN
TRIVIAL	1	1-4 SEMANAS
PEQUEÑOS	1	1-6 MESES
MEDIANOS	2-5	1-2 AÑOS
GRANDES	5-20	2-3 AÑOS
MUY GRANDES	100-1000	4-5 AÑOS
EXTREMADAMENTE GRANDES	1000-5000	5-10 AÑOS

En esta era que se ha dado en llamar "de la cibernética" resulta obvio que los sistemas de información se encuentran íntimamente ligados a todas las actividades productivas de la sociedad humana. En la actualidad el software no sólo se utiliza en los procesos administrativos, sino también en algunos procesos de la producción, como en el caso del software que maneja los robots ensambladores de las empresas automotrices; o en el caso del

software que se utiliza en el tráfico aéreo. Lo que es más, en algunos casos el software va integrado en el producto mismo, como en el caso de los automóviles electrónicos que tienen una computadora.

Como ya se mencionó en un principio el software es un producto y a la vez un medio. Esta doble función denota la preponderancia del software en la vida productiva del hombre moderno, por lo que es plenamente justificable la preocupación por obtener un software con calidad.

5. CALIDAD DEL SOFTWARE.

Desde luego que hay una gran variedad de enfoques al respecto. Para el propósito del presente trabajo se ha optado por considerar la calidad del software al conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad aquí es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usuabilidad, seguridad e integridad.

La calidad del software es medible y varía de un sistema a otro o de un programa a otro. Un software elaborado para el control de naves espaciales debe ser confiable al nivel de "cero fallas"; un software hecho para ejecutarse una sola vez no requiere el mismo nivel de calidad; mientras que un producto de software para ser explotado durante un largo período (10 años o más), necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación, Implementación y auditoría (pruebas) del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

En la búsqueda de la calidad del software, es recomendable definir una política de trabajo que se sustente sobre tres principios básicos: tecnológico, administrativo y ergonómico.

- El principio tecnológico define las técnicas a utilizar en el proceso de desarrollo del software.
- El principio administrativo contempla las funciones de planificación y control del desarrollo del software, así como la organización del ambiente o centro de ingeniería de software.

- El principio ergonómico define la interfaz entre usuario y el ambiente automatizado.

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura. Para el aseguramiento de la calidad es necesario su control y evaluación.

En cuanto al control, es indispensable establecer los parámetros, indicadores o criterios de medición; pues no se puede controlar lo que no se puede medir. Una vez seleccionados los índices de calidad es importante estructurar el proceso de control, que requiere los siguientes pasos:

- Definir el software que va a ser controlado: clasificación por tipo, esfera de aplicación, complejidad, etc., de acuerdo con los estándares establecidos para el desarrollo del software.
- Seleccionar una medida que pueda ser aplicada al objeto de control. Para cada clase de software es necesario definir los indicadores y sus magnitudes.
- Crear o determinar los métodos de valoración de los indicadores: métodos manuales como cuestionarios o encuestas estándares para la medición de criterios periciales y herramientas automatizadas para medir los criterios de cálculo.
- Definir las regulaciones organizativas para realizar el control: quiénes participan en el control de la calidad, cuándo se realiza, qué documentos deben ser revisados y elaborados, etc.

Respecto a la evaluación, se puede afirmar que es el momento más consistente para el aseguramiento de la calidad del software. Y de acuerdo a la expectativa que nos guía resulta crucial poner todos los reflectores ante cada una de las actividades evaluatorias de la producción del software.

El esfuerzo de poner atención total en todos los procedimientos de dicha evaluación, nos pone en la vía propia de la calidad; pues el termino evaluación lleva en si una connotación de perfectibilidad.

De manera general, la evaluación de software comprende el conjunto de pruebas que se le aplican y se refiere al proceso de localizar y enmendar errores; pero no se trata de esperar hasta el momento final cuando el producto ya esta terminado, para poder evaluar. Una evaluación entendida de esta manera nos llevaría a

lamentaciones y pérdidas excesivas. Es conveniente mantener una actitud crítica, con una mirada siempre evaluatoria durante todo el proceso de desarrollo del software, desde el momento de análisis y diseño hasta su explotación, pasando por sus fases intermedias.

En un mundo perfecto, las evaluaciones podrían asegurar que todas las fallas de software fueran removidas antes de la presentación. En el presente, existen muchas razones para esto, pero es una aspiración equivocada. Primero, hay literalmente millones de posibles patrones a través de software abierto. No es viable verificar cada una de las combinaciones posibles de entradas, estados y salidas. Segundo, los cambios siempre serán requeridos por medio de la vida del software. Ninguna prueba será única ni para todos ni para cada uno de los cambios (no importa si es una actualización o una enmienda) y cargará con la posibilidad de que un nuevo y previo comportamiento no probado sea introducido.

Dado que una prueba completa no es viable, la pregunta de ¿qué tantas pruebas deben hacerse y dónde se deben aplicar? es un cuestionamiento importante. Por ejemplo, los proyectos en áreas de seguridad crítica donde la integridad del software es extrema (como lo es en las aplicaciones militares o del espacio, el software médico y en los sistemas de aeroplanos vuelo-por-línea) debe ser probada con mucho mayor rigor que, por decir, un simple paquete de procesador de palabras. Esto implica que no hay un enfoque de la mejor práctica global y la extensión de la prueba depende, en un alto grado, de la aplicación particular.

El presupuesto (tanto en tiempo como en dinero) asignado a las pruebas es a menudo considerado como el sobrante al final de todo el proceso. La cantidad correcta es difícil de establecer con anticipación, así como varía tanto de una organización a otra, como dentro de una organización y de un proyecto a otro. A pesar de esto, las pruebas deben ser planeadas desde un principio y no al final. Es sólo manteniendo registros cuidadosos del tiempo que se utiliza en las pruebas y teniendo efectividad de estos registros como los administradores podrán comenzar a construir el perfil necesario de las necesidades del área de pruebas.

La mayoría de las pruebas es vista como un mal necesario después del estado de implantación de un proyecto de software. A pesar del incremento que se ha producido en los sistemas complejos, parece que existe un pequeño aumento de correspondencia al esfuerzo dedicado a las pruebas o en la conciencia de las técnicas de prueba. Como resultado de la actitud ampliamente sostenida de "codificar luego probar" cualquier gasto excesivo en los primeros estados de un

proyecto tenderá a disminuir el estado de pruebas, con el resultado que menos tiempo y esfuerzo están disponibles para llevar a cabo estas actividades.

Esto se encuentra avalado por investigaciones recientes, las cuales indican que existen pocos practicantes en la industria del Reino Unido que consideran que suficiente tiempo está permitido para las pruebas (por lo general, el 15% del tiempo y esfuerzo es considerado como un límite generoso). Para aprender cómo la industria del software, como en todo, se desarrolla en Estados Unidos, se debería mirar los ejemplos establecidos por compañías líderes en ese país, donde las pruebas se tratan con mucho mayor respeto que en cualquier otra parte de la disciplina de la ingeniería de software y un 40% está permitido para realizar las pruebas.

Aparte de las falta de tiempo y esfuerzo asignado a las pruebas está el hecho de que el nivel de automatización que de manera rutinaria se aplica en esta fase es baja comparada, por ejemplo, con la fase de diseño. Esto es de sorprender, sobre todo si se considera que una gama de herramientas está disponible y que intrínsecamente proporcionan la repetición y exactitud requeridas en la prueba.

En el Reino Unido, un informe elaborado por el Departamento de Comercio sugiere que la aplicación de herramientas para la prueba de software debería de resultar en un ahorro del 9 al 18% para el desarrollo del sistema. El informe también señala que cualquier inversión en herramientas de software proveerá un retorno de dos a tres veces en términos de tiempo y esfuerzo ahorrados.

6. MARCO CONCEPTUAL

6.1. Propósito

El objetivo de esta sección es el de presentar los conceptos alrededor de los cuales gira la evaluación de la calidad de los productos del software.

6.2. Conceptos fundamentales.

Es importante antes de entrar en detalle definir de manera precisa los términos que se van a manejar con mayor frecuencia en este documento.

6.2.1. Proceso del Software

Hablar del proceso del software, es referirnos al conjunto de etapas ó actividades que son aplicables a todos los proyectos del software, indistintamente de su tamaño o complejidad. Cada actividad comprende una colección de tareas que se adaptan a las peculiaridades del proyecto de software y a los requisitos del equipo. Existen varios enfoques del proceso de software. El más común es el que se utiliza en este documento para enmarcar una propuesta de evaluación orientada al aseguramiento de la calidad del software.

6.2.2. Producto de software.

Un producto de software es aquel conjunto de elementos de software (programas, tablas, reportes, documentación, etc.) que tienen un propósito específico y completo desde el punto de vista del usuario, de tal manera que la sustracción de cualquiera de los elementos del conjunto daría como resultado que el propósito no se cumpliera.

La definición anterior nos da la idea de que un programa por si solo, a pesar de que puede tener un propósito específico, no necesariamente es un producto de software, puesto que lo que haga puede que no sea todo lo que un usuario necesita para apoyar su actividad.

Entre las características que un producto de software tiene podemos identificar las siguientes:

- Tiene una fecha de inicio de desarrollo y una fecha esperada de terminación definidas.

- Es lo suficientemente grande como para que apoye alguna función del usuario hacia el cual está dirigido.
- Es lo suficientemente pequeño para que en la mayoría de los casos sólo requiera de una persona para su desarrollo.

6.2.3 Proyecto

Un proyecto está integrado por uno o más proyectos (que vendrían a ser subproyectos de él) o por uno o más productos de software.

Contrastando contra un producto de software, la duración de un proyecto puede ser indefinida, puesto que al proyecto se le pueden agregar nuevos productos de software conforme pasa el tiempo y se descubren nuevas necesidades de los usuarios u oportunidades de mejorar la forma en que realizan sus actividades.

El número de personas que están involucradas en un proyecto puede ser una o muchas y este número puede variar con respecto al tiempo. El propósito de un proyecto es el de proporcionar un conjunto de facilidades que apoyen a un conjunto de actividades del usuario que lógicamente están relacionadas entre sí.

6.2.4. Versión.

Una versión es una fotografía de un producto de software en un momento del tiempo. Una versión se caracteriza por lo siguiente:

- Cumple con una serie de requerimientos previamente definidos por el usuario en el análisis.
- Al momento de generarse la versión queda "congelada" y no se le pueden realizar cambios.

Las versiones de un producto de software se diferencian entre sí por un identificador de versión, el cual es una serie de números separados por puntos cuando la serie consta de más de un número.
Existen tres tipos de versiones, cada una de las cuales tiene su manera muy particular de ser identificada:

- ***Versión (completamente) nueva.***

  La primera versión de un producto de software cae siempre en esta categoría. Cada vez que los requerimientos de una nueva versión impliquen cambios significantes en la esencia o estructura interna del

producto, un cambio de paradigma, o un rediseño, se considerará que la versión del producto corresponde a esta categoría.

Este tipo de versiones puede ser fácilmente reconocida, pues su identificador siempre es un número entero sin decimales significativos (ej. 1 ó 1.0, 2 ó 2.0, etc.). El número de estas versiones se calcula truncando el número de la versión previa a ésta y sumando "1" al resultado. La primer versión siempre es 1.0.

- ***Versión por cambios de funcionalidad.***

  Cuando el producto de software no es objeto de cambios significativos en su estructura o filosofía y sólo se le agrega o modifica un poco la funcionalidad, el identificador de versión es de la forma X.Y, el cual se obtiene sumando "0.1" al número de la versión anterior, truncándose cualquier decimal que sobre (si la versión anterior es la 2.1.2, la siguiente versión debe ser 2.2, pues el último "2" debe ser eliminado para mantener el formato X.Y.

- ***Versión de mantenimiento.***

  Si a un producto de software sólo se le va a efectuar mantenimiento, sin realizar cambios en su estructura o filosofía, ni se van a agregar o modificar funciones, la versión se considera de mantenimiento, y su número de versión debe ser de la forma X.Y.Z, el cual se calcula sumando "0.0.1" al número de versión anterior (ej.: si la versión anterior es 2.3, la nueva debe ser 2.3.1 y si se requiere otra versión de mantenimiento posterior, ésta deberá ser 2.3.2).

6.2.5. Regla.

Por regla entenderemos toda aquella norma que debemos de seguir durante el desarrollo de un producto de software. Dichas normas pueden haber sido impuestas por las características y restricciones del sistema operativo, la máquina o el manejador de base de datos que usamos, o pueden ser la consecuencia de la aplicación de políticas de la organización. La característica más importante de las reglas es que siempre se deben de seguir, por ningún motivo pueden pasarse por alto.

6.2.6. Estándares.

Los estándares son lineamientos que guían la manera de efectuar una actividad, la cual en nuestro caso es el desarrollo de software. Los estándares son auto-impuestos por un consenso de los mismos desarrolladores y se particularizan por su orientación hacia la generalidad: los estándares se crean pensando en los casos posibles más comunes. Por lo anterior, los estándares siempre se deben de seguir, salvo en las raras ocasiones en que el problema a atacar requiera una solución de software que no esté contemplada por los estándares. En dicho caso, nuevos estándares deben de incluirse, especificando detalladamente en que situaciones deben de emplearse.

6.2.7. Estilo.

El estilo es personal y se define como las diferentes variaciones en la presentación del código fuente y documentación que caracterizan a cada desarrollador. Estas diferencias son sólo a nivel presentación, y son superficiales. En ningún momento se habla de que el estilo sea una desviación con respecto a los estándares y reglas.

Un desarrollador debe de cuidar los siguientes aspectos en su estilo:

- Legibilidad (que pueda ser leído fácilmente por otros).
- Consistencia (que el estilo no varíe de programa a programa o de documento a documento).
- Coherente (que no viole los estándares).

6.2.8. Factor de calidad.

Para asegurar que un producto de software sea de calidad, la cual es muy difícil de definir y medir, se han seleccionado, de la literatura de ingeniería de software, una serie de factores que debe cumplir un producto de software para poder ser considerado de calidad.

6.2.9. Métricas.

Para poder saber si un producto de software cumple con alguno de los factores, es necesario realizar una medición de uno o más valores llamados métricas. Una buena métrica debe ser fácil de medir aplicando una escala o regla y debe de evitar al máximo la subjetividad del que la mide.

7. GUÍA DE EVALUACIÓN PARA EL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE.

7.1. Propósito

En la presente sección se presentan lineamientos para facilitar la evaluación de productos de software. Se proponen las preguntas que deberán contestarse clasificadas en cada una de las etapas del proceso de desarrollo del software y por supuesto, se explica en que consisten dichas etapas.

7.2. Análisis

La etapa de análisis comprende la identificación de requerimientos y del problema que va a ser resuelto por el nuevo sistema de información, el cual puede constar de sistemas manuales y/o productos de software.

7.2.1. Análisis de requerimientos.

En esta etapa se descubren las necesidades de los usuarios en relación a la información que es manejada por un subconjunto de sus actividades. Para verificar que este proceso se realiza con calidad se debe dar respuesta a las siguientes preguntas:

1. *¿Fueron debidamente documentados los requerimientos?* Para evaluar este punto se requiere:

- Revisar en el "*Documento de Propuesta*" las secciones de "Antecedentes", "Propuesta Funcional" y "Propuesta Técnica". La sección de "Antecedentes" debe contener una descripción breve del problema, mientras que las secciones de "propuestas" deben contener los requerimientos técnicos y funcionales del producto de software.

2. *¿Se han identificado en la planeación los requerimientos que serán resueltos en futuras versiones del software?* Para evaluar este punto se requiere revisar lo siguiente:

- Los documentos de "*Descripción de Actividades*", "*Relaciones Estructurales*" y "*Representación de Actividades*" deben de existir para el área de la organización donde actúa el producto de software.

- Si el producto de software que se va a construir para satisfacer las necesidades de las diferentes actividades descritas en los documentos anteriores va a ser implementado en varias versiones, debe de existir un documento de "*Plan de Desarrollo del Producto de Software*", el cual debe contener una breve descripción de los requerimientos que serán satisfechos en cada una de las versiones del producto.

3. ¿La documentación de los requerimientos nos lleva al análisis? Para evaluar este punto se requiere revisar lo siguiente:

- El "*Documento Sociotécnico*" debe mostrar en forma gráfica los requerimientos que las actividades de los usuarios hacen al producto de software.
- Los requerimientos anteriores deben de estar explícitamente descritos en la propuesta.
- El documento de propuesta debe contener los requerimientos técnicos del producto de software.

7.2.2. Análisis del dominio del problema.

Aquí se delimita el área o problema sobre el cual actuará el producto de software.

4. *¿Es completo, consistente y exacto el análisis del dominio de las especificaciones (es decir, se contemplan todas las alternativas, alcances, restricciones, limitaciones)?* Para evaluar este punto se requiere revisar lo siguiente:

- Revisar los diagramas del "*Documento Sociotécnico*" para ver si está bien definido el contexto dentro del cual se moverá el producto de software (ver cómo interactúa con el entorno humano).
- Se deberá revisar que la propuesta contenga apartados sobre alcances, alternativas, restricciones y limitaciones. En muchos casos, sólo se presenta una alternativa de solución del problema.

5. *¿Está bien definida la parte automatizable (porción del problema a resolverse por el producto de software) y la parte manual del problema?* Para evaluar este punto se requiere revisar lo siguiente:

- El documento "*Sociotécnico*" debe de presentar un diagrama en donde se muestre claramente cada proceso que desarrolla el o los usuarios, dividido en sus actividades individuales. Cada actividad que requiera ser automatizada

deberá de estar debidamente marcada en el diagrama, mostrándose qué es lo que necesita de entrada y de salida del sistema computacional.

6. *¿Existe documentación que nos lleve del análisis al diseño?*

- Verificar que el documento de "*Propuesta*" contenga información sobre las características funcionales del producto (propuesta funcional) y una pequeña propuesta técnica basada en el estudio realizado en el análisis y en las necesidades funcionales.

7.3. Diseño.

En esta fase se realizan los "planos" del producto de software para su posterior construcción. El diseño debe de ir de acuerdo a las especificaciones generadas durante el análisis.

7.3.1. Planeación del diseño.

En esta sección se pretende averiguar si el diseño sólo está encaminado a satisfacer los requerimientos inmediatos de los usuarios del producto de software, o si se han tomado en consideración necesidades futuras.

7. *¿Se pueden identificar en el modelo de datos los atributos, tablas o relaciones que se reflejen en la información de futuros requerimientos?* Para evaluar este punto se requiere revisar lo siguiente:

- En el documento "*Sociotécnico*" deben de venir listados los requerimientos generales del área de estudio, producto de la realización del análisis.
- Revisar si los requerimientos de la propuesta y otros más que se encuentren en el documento "*Sociotécnico*" se encuentran representados en el esquema de datos, definido en el "*Documento de Diseño*", en forma de entidades, atributos y relaciones.

8. *¿Las funciones de software se encuentran parametrizadas para poder ser utilizadas para otros requerimientos?* Para evaluar este punto se requiere revisar lo siguiente:

- En el documento "*Sociotécnico*" deben de venir listados los requerimientos generales del área de estudio, producto de la realización del análisis.
- Revisar que la definición de funciones y procedimientos abarque no sólo a los requerimientos de la propuesta, sino a más requerimientos del área.

7.3.2. Modelo de datos.

El modelo de datos es el "plano" de cómo se representa la información en el sistema, desde un punto de vista lógico.

9. *¿Los elementos del modelo de datos corresponden con los elementos implicados en el problema?* Para evaluar este punto se requiere revisar lo siguiente:

- Cada elemento (entidad o regla) del problema debe poder ser identificado dentro del modelo de datos, ya sea como entidad, relación o restricción. Las características de cada entidad deben identificarse como atributos en el modelo. No debe de faltar información, ni debe de haber información que no se relacione con el problema. Esta información se obtiene de "*Conceptos y Políticas Básicas*".

10. *¿Es consistente el modelo de datos con los requerimientos del producto de software?*

- La información directamente relacionada con los requerimientos debe estar completamente representada en el modelo de datos, ya sea en las entidades, atributos o relaciones. Esta información está contenida en los documentos de "*Propuesta*" y "*Sociotécnico*".

7.3.3. Documentación del diseño.

En esta sección se verifica que la documentación del diseño cumpla con las siguientes preguntas:

11. *¿Es posible saber fácilmente para qué sirve cada uno de los módulos que comprenden el producto de software?* Para evaluar este punto se requiere revisar lo siguiente:

- Cada módulo (una o más operaciones que el usuario percibe como una unidad funcional de su trabajo) debe estar descrito en el documento de "*Diseño*". Además de contar con una descripción que permita saber el propósito del mismo, cada módulo y/u operación debe tener un nombre descriptivo apegado a los estándares de nomenclatura.

12. *¿Existe documentación del diseño que permita mapearlo hacia la implementación?* Para evaluar este punto se requiere revisar lo siguiente:

- El documento de "*Diseño*" debe contener la descripción de cada módulo y/o

operación , sus estructuras de datos de entrada y salida, una descripción del algoritmo y una tabla de excepciones.

- El documento de "*Diseño*" debe contener la representación de las formas (documentos, pantallas, reportes, etc.) de entrada y salida y un diagrama de interacción con el usuario.

7.3.4. Funcionalidad.

En esta sección se verifica el grado en que el diseño cumple con la funcionalidad que el usuario espera, de acuerdo a los requerimientos.

13. *¿Se definen las funciones principales del producto de software en forma limitada y sin ambigüedad?* Para evaluar este punto se requiere revisar lo siguiente:

- Cada función debe tener un nombre descriptivo y debe contar con una descripción de su propósito en el documento de "*Diseño*".
- El algoritmo y la descripción de la función deben permitir ver que la función realiza una sola cosa desde el punto de vista lógico.

14. *¿Se reflejan los requerimientos del producto de software en el diseño del mismo (tomando en cuenta restricciones internas y externas)?* Para evaluar este punto se requiere revisar lo siguiente:

Para cada requerimiento debe de existir en el documento de "*Diseño*" un conjunto de funciones encaminadas a satisfacerlo.

15. ¿Realizan los algoritmos la función o funciones deseadas? Para evaluar este punto se requiere revisar lo siguiente:

- Rastrear el algoritmo de cada función para descubrir si éste realiza su propósito.
- Realizar pruebas de escritorio si la complejidad del algoritmo así lo amerita.

7.3.5. Retroalimentación al usuario.

En esta sección se verifica la calidad de la retroalimentación que el producto de software da al usuario.

16. *¿Proporciona el sistema ayuda en línea al usuario?*

- Revisar el documento de "*Diseño*" para descubrir la existencia de funciones que presten ayuda en línea al usuario. Cuando el mismo ambiente de desarrollo

provee facilidades de ayuda, el documento debe mencionar de qué manera se van a utilizar.

17. *¿Se utilizan mecanismos de ayuda o "hints"?*

- El documento de "*Diseño*" debe contener funciones de "hint" o, en el caso de que el ambiente de desarrollo provea dichas facilidades, el documento debe mencionar cómo se hará uso de ellas.

7.3.6. Arquitectura.

En esta sección se revisa la calidad de la arquitectura del producto de software.

18. *¿Están bien definidas las interfases entre los elementos del sistema (Entradas, salidas, formatos y protocolos bien delimitados y especificados)?*

- El documento de "*Diseño*" debe especificar cómo se comunicarán entre sí los diferentes módulos del producto de software, mostrando estructuras de datos de entrada y salida.

19. *¿Tiene el sistema una interfase externa que facilite al usuario el interactuar con él sin problemas (apego a estándares)?*

- El documento de "*Diseño*" debe contener el diseño de las formas de entrada/salida (pantallas, ventanas, documentos, reportes, etc.), las cuales deben de apegarse a los estándares definidos para el ambiente específico donde se implementará el producto de software.

20. *¿Tiene el sistema una interfase externa que posibilite la interacción con otros sistemas, ya sea suministrando salidas o recibiendo entradas?*

- El documento de "*Diseño*" debe especificar cómo se comunica el producto de software con otros productos (a través de archivos de datos, mensajes a través de la red, llamadas a funciones de biblioteca, etc.). En todos los casos se deben definir los formatos con el que un producto debe intercambiar información con otro.

21. *¿La interfase con la base de datos utiliza los recursos de acceso adecuados a la situación y a las características de la aplicación?*

- Verificar en el documento de "*Diseño*" la manera en que los algoritmos hacen acceso a los datos contenidos en la base de datos. Estos deben de realizarse de acuerdo a los estándares definidos para el acceso de datos de diferentes

dominios de acuerdo a las facilidades que proporcione el ambiente donde se implementará el diseño. Si se realizan accesos a información fuera del dominio de la aplicación, el documento debe especificarlo así, y se deberá mencionar la estrategia o herramienta que se usará para tener dicho acceso.

22. *¿Existe un estándar y estilo en todos los elementos que constituyen una misma aplicación?* (Consistencia entre los elementos del sistema).

- Verificar que los formatos de comunicación entre elementos del sistema estén estandarizados. Esto se revisa en el documento de "*Diseño*".
- Las interfases de entrada y salida con el usuario deben de ser estándares: todas las interfases del mismo tipo deberán tener la misma funcionalidad, presentación y filosofía de uso. Esta información debe estar contenida en el documento de "*Diseño*".

23. *¿Son funcionalmente independientes los módulos?*

- Revisar en el documento de "*Diseño*" los algoritmos de los diferentes módulos para descubrir efectos secundarios del funcionamiento de un módulo sobre otro. Un efecto secundario de funcionalidad es cuando el funcionamiento de un módulo se ve afectado por el funcionamiento de otro de una manera no deseable, debido generalmente a la mala compartición de estructuras de datos globales.

7.3.7. Manipulación de datos.

En esta sección se revisa el uso que se hace de las estructuras para el manejo de los datos.

24. *¿La estructura de datos local soporta con precisión los modelos de datos diseñados para la aplicación?* Ej. : Manejo adecuado de registros.

- Verificar que las estructuras de datos definidas en el documento de "*Diseño*" sean de los tipos adecuados para realizar la interfase con la base de datos o con las interfases internas y externas.
- En caso de modificaciones a una estructura de datos, debe revisarse que el impacto de aquél no trascienda las fronteras de los propósitos para los que fue creada. (Si una estructura de datos se modifica, puede que impacte en dos módulos, cuando en realidad sólo es uno el que la necesita y el otro la utiliza sólo porque necesita un campo o dos de ella, en cuyo caso podría definirse una estructura de datos para el segundo módulo).

25. *¿Los tipos lógicos, tipos físicos y datos representan con precisión las características de los atributos de las entidades?* (Declaraciones apropiadas).

- Contrastar las definiciones del documento de "*Conceptos y Políticas Básicas*", los requerimientos del producto de software y el documento de "*Diseño*" para descubrir si las definiciones de tipos y datos corresponden con el tipo de operaciones, formatos que se utilizan con ellos, y revisar si, en general, son compatibles con la realidad (Ej. : El campo "años de experiencia" debe ser numérico, ya que puede requerirse en operaciones y comparaciones matemáticas y su formato de presentación es numérico).

26. *¿Las cantidades numéricas corresponden lo más cercanamente posible a la realidad?* (Constantes correctas).

- Comparar las definiciones de las constantes contenidas en el documento de "*Conceptos y Políticas Básicas*" con las definiciones del documento de "*Diseño*" para descubrir posibles incompatibilidades.

7.3.8. Algoritmo.

A continuación se revisa el diseño del o los algoritmos desde el punto de vista técnico.

27. *¿El algoritmo puede ser comparado con los requerimientos funcionales solicitados? (*Algoritmo lógicamente correcto).

- Comparar los requerimientos funcionales del producto de software con las definiciones de las funciones contenidas en el documento de "*Diseño*".
- Revisar el algoritmo de cada función para descubrir si éstos realizan las funciones que satisfacen la propuesta funcional. Se recomienda realizar pruebas de escritorio.

28. *¿El manejo de márgenes de error, de truncamiento y de cantidades son adecuados a las características de la información que maneja el producto de software?*

- Revisar los algoritmos en busca segmentos de algoritmo que realicen operaciones de truncamiento, redondeo o formateo de datos, para descubrir posibles inconsistencias entre la forma en que los datos se manejan en el producto y los resultados que se esperan en la realidad (Ej. : Si se espera que las operaciones con dinero se presenten con 2 dígitos de centavos, revisar que los redondeos o truncamientos a 2 dígitos se realicen en los pasos correctos del algoritmo).

29. *¿Es razonable la complejidad lógica del algoritmo con respecto a la complejidad del problema?*

- Dado un problema, ¿puede éste ser resuelto de una manera más sencilla?
- En caso de que sea posible, ¿existe alguna restricción técnica o lógica que lo impida, y si es así, está documentada en el diseño?

7.3.9. Reutilización.

En esta sección se verifica el reuso de los elementos que se diseñan.

30. *¿Se ha empleado el reuso de otros elementos de software?*

- Se requiere un profundo análisis de la funcionalidad y algoritmos contenidos en el documento de "*Diseño*" para descubrir puntos en donde podrían reutilizarse elementos de otros sistemas, bibliotecas o elementos del mismo producto de software.
- ¿Existen justificaciones (restricciones o limitantes del lenguaje a usar o inherentes al problema) para no reutilizar elementos del diseño donde podrían aparentemente ser utilizados? ¿Están documentadas?

31. *¿Se ha evitado el duplicado de funciones similares?*

- Aplicar el punto 2 de la pregunta anterior.

7.3.10. Manejo de errores.

En esta sección se revisa la manera en que el producto de software responde a situaciones de error y cómo las maneja.

32. *¿Se ha definido con detalle qué pasará con el funcionamiento del software en caso de que se incurra en un error a la hora de alimentar datos, o bien, en caso de un mal funcionamiento interno?* (Tratamiento y tolerancia de errores)

- Verificar que el documento de "*Diseño*" contenga tabla de excepciones y que se especifique claramente cómo actuará el producto de software en el caso de que uno de estos errores se presente.
- Revisar que se especifique qué funciones y/o facilidades del lenguaje de implementación se utilizarán. Si se requiere desarrollar funciones especiales para ello, ellas deberán estar contenidas en el documento.

33. *¿Se han implementado procedimientos de recuperación de errores?*

- El documento de "*Diseño*" debe contener una descripción de procedimientos genéricos y/o específicos de qué es lo que se debe hacer para que el producto de software no falle. Deben mencionarse las facilidades del lenguaje que se utilizarán, en caso de que ese las provea.

34. *¿Son claros los mensajes de error?*

- Comparar el propósito de los mensajes de error, el tipo de excepción que manejan y la redacción para ver si son compatibles.

35. *¿Ayudan los mensajes de error a evitar reincidir en los errores?*

- Para cada mensaje de error relacionado con una acción del usuario, revisar si su redacción transmite de manera no ambigua el origen del error.

36. *¿Han sido especificados los mensajes de error conforme a los estándares?*

- Comparar los mnemónicos y la redacción de cada mensaje de error contra los estándares.

37. *¿Se proporciona ayuda adecuada para prevenir errores derivados de la captura?*

- Comparar las especificaciones de la excepción contra la redacción del mensaje de error.
- Revisar que el producto de software proporcione información al usuario sobre tipos de datos y opciones válidas para cada campo o recurso de captura, e indicándole las acciones que puede realizar.

7.3.11. Interoperabilidad.

Revisión de la facilidad que tiene el producto de software para relacionarse con otros.

38. *¿El producto de software o alguno de sus elementos puede ser llamado desde otro producto de software y seguir proporcionando su misma funcionalidad?*

- Revisar que el producto de software tenga elementos que puedan ser aprovechados por otro producto de software (rutinas genéricas, interfases de captura o presentación de datos que puedan compartirse).

39. *¿Se han definido las funciones que el producto de software ofrece a otros?*

- Revisar la existencia de este punto en el documento de "*Diseño*".

40. *¿Se tiene una especificación clara y sencilla de cómo hacer uso de los módulos y elementos externos del sistema desde otros módulos y sistemas?*

- Revisar en el documento de "*Diseño*" la existencia de especificaciones de comunicación con otros productos de software, ya sea que el producto de software actúe como cliente o proveedor de servicios de los otros productos de software.

7.3.12. Independencia de plataforma.

En esta sección se revisa si el diseño es independiente de la plataforma de implementación. En el documento de "*Diseño*", sin embargo, se puede hacer referencias a características propias de la plataforma de hardware o software seleccionada para la implementación, pero el diseño en general debe ser independiente de éstas.

41. *¿Es independiente el diseño del producto de software de características del hardware?*

- ¿Se tendría que volver a diseñar el producto si cambio de plataforma de hardware?

42. *¿Es independiente la funcionalidad del producto de software de factores relacionados con el software que utiliza?* (Lenguaje, DBMS, sistema operativo).

- ¿Si cambio el lenguaje, el S.O. o el DBMS, funcionará mi producto de software?

43. *¿Es independiente el diseño del lenguaje de implementación?*

- ¿Necesito rediseñar si cambio el lenguaje de implementación?

44. *¿Se ha evitado utilizar características dependientes del sistema operativo del lenguaje?*

- ¿Existe alguna justificación en caso de que se hayan tenido que utilizar dichas características?
- ¿Existe documentación de dicha justificación?

NOTA: Básicamente esta sección requiere de un amplio conocimiento de las características, ventajas y restricciones del software de implementación.

En caso de que no se pueda evitar utilizar dichas características en el diseño, el documento de "*Diseño*" debe de justificar el uso de las mismas.

7.4. Implementación.

Este apartado trata la manera en que el producto de software fue implementado.

7.4.1. Consistencia con el diseño.

En esta sección se revisa que tan consistente es la implementación del producto de software con el diseño.

45. *¿Las interfases cumplen con los detalles necesarios que fueron especificados durante la etapa de diseño?*

- Revisar que las interfases entre las funciones y procedimientos sean consistentes con lo especificado en el documento de "*Diseño*".
- Hacer lo mismo con las interfases con otros productos de software, ya sea que el producto de software a evaluar sea cliente o proveedor de los otros productos.
- Verificar que la apariencia y funcionalidad de las interfases sea la que se especificó en el documento de "*Diseño*".

46. *¿Se ha traducido adecuadamente el diseño al código?*

- Verificar que el código fuente se apegue al algoritmo descrito en el documento de "*Diseño*". Es necesario, como ayuda para la evaluación, que el código fuente cuente con comentarios.
- Verificar que la funcionalidad presentada en la propuesta y en el documento de "*Diseño*" sea satisfecha.

7.4.2. Apego a estándares.

En esta sección se presenta la forma de verificar que el producto de software se apegue a los estándares de implementación.

47. *¿Resultan fáciles de entender los algoritmos que se han utilizado para la implementación del producto de software (instrucciones utilizadas, comentarios, identación)?*

- Verificar que los nombres de variables y procedimientos sean representativos.
- Revisar que no se utilice una variable para un uso distinto al que su nombre sugiere (Ej. : utilizar la variable contador para almacenar un total).
- Verificar que se haya utilizado identación en ciclos, condicionales y que los bloques estén debidamente delimitados.
- Verificar el uso de comentarios, ¿se entienden? ¿Ayudan a entender mejor el código?
- Verificar que no se utilicen instrucciones muy rebuscadas sin comentarios explicativos. ¿Se entiende el uso de estas instrucciones con los comentarios que se anexan a ellas?

48. *¿Se ha hecho un uso adecuado de las convenciones del lenguaje?*

- Revisar el código del producto de software para descubrir si se siguen las sugerencias del manual de programación del lenguaje elegido para la implementación (en caso de que existan dichas sugerencias y que se hayan aceptado como parte de los estándares). Estas sugerencias, por lo general se refieren: uso de minúsculas, mayúsculas y "" en nombres de variables y procedimientos, secuencias de instrucciones convenientes para realizar algunas operaciones, identación sugerida, etc.

49. *¿Se han seguido los estándares para los estilos del lenguaje, los comentarios y los prólogos de los módulos?*

- Revisar que el código fuente cumpla con los estándares propios del lenguaje, de acuerdo al ambiente en donde se haya desarrollado el producto.
  - Nomenclatura de elementos de software.
  - Utilización de secuencias de instrucciones sugeridas para las diferentes operaciones (en el grado que esto sea posible).
  - Buen uso de comentarios.
  - Documentación del código (datos generales: autor, fecha, propósito, versión, etc.)
  - Formas de acceso a elementos de software que no forman parte del producto de software (Ej. : tablas de otros dominios o modelos).

50. *¿Se ha buscado el no utilizar comentarios incorrectos o ambiguos?*

- Revisar que los comentarios expresen de manera clara y sin ambigüedad lo que necesitan decir.

51. *¿Se encuentra el código debidamente documentado?*

- Revisar que no falten comentarios de documentación de datos generales (autor, versión, fecha, etc.).
- ¿Tomando en cuenta la documentación del diseño y los comentarios en el código fuente se puede entender sin problema el programa?

52. *¿Siguen los comentarios los estándares?*

- Revisar que la estructura y presentación de los comentarios vaya de acuerdo a los estándares para ello.

7.4.3. Optimización de recursos.

En esta sección se revisa el uso que hace el producto de software de los recursos de cómputo.

53. Tomando en cuenta aspectos como la complejidad lógica del problema que se intenta resolver, las facilidades del lenguaje y el reuso de elementos ya hechos, *¿el producto de software es razonablemente compacto en términos de líneas de código?*

- La evaluación de este punto puede ser un tanto difícil, pero hay que enfatizar que es durante el diseño donde se debe de verificar que el o los algoritmos son correctos. Con esta suposición, en este punto sólo se verificará que la implementación del algoritmo a partir del diseño sea lo más compacto posible.
- Para cada bloque o segmento de código, ¿existe alguna forma alternativa de implementarlo que sea más eficiente en términos de líneas de código? Si no es así, ese bloque o segmento de código puede considerarse "optimizado" con respecto al número de líneas de código.
- Verificar que no se dupliquen líneas de código similares. En caso de que esto suceda, es probable que puedan agruparse en un procedimiento o función.

54. *¿Se ha evitado el duplicar líneas de código?*

- Aplicar el punto 2 de la pregunta anterior.

55. *¿El tiempo de respuesta es adecuado a las necesidades del usuario?*

- Realizar pruebas del producto de software para determinar el mejor, el peor y el promedio de tiempo de respuesta.
- Verificar que los tiempos de respuesta estén dentro de los rangos aceptables.

56. *¿Consume muchos recursos (cpu, memoria, red) al estarse ejecutando el producto de software?*

- Realizar monitoreos mientras el producto de software está siendo probado. Los monitoreos pueden realizarse con herramientas de UNIX, o herramientas propias de cada DBMS.
- Compare los resultados obtenidos con los rangos de valores aceptables para el consumo de cada recurso.

NOTA: Las pruebas para las dos preguntas anteriores deben de hacerse en condiciones controladas, más o menos similares a las condiciones reales bajo las cuales va a operar el producto de software y deben de realizarse pruebas en condiciones lo más ideal posibles (que sólo ese producto de software esté corriendo).

7.4.4. Independencia de plataforma.

En esta sección se evalúa el grado de independencia de la plataforma de hardware y software que tiene la implementación.

57. *¿Es independiente la implementación de características del ambiente (hardware y software)?*

- Revisar el código fuente para buscar secciones del mismo que hagan uso de características especiales o particulares del hardware y del software utilizados en la implementación. Búsquense características dependientes de: hardware, DBMS, S. O., lenguaje de programación.
- Lo encontrado en el punto anterior, ¿está documentado y justificado en el documento de "*Diseño*"? ¿Es imprescindible, dadas las restricciones y características del producto que se mantenga esa dependencia de plataforma?

7.4.5. Interfase con el usuario.

En esta sección se revisa la calidad de la interfase del usuario en cuanto a cumplimiento con estándares y funcionalidad.

58. *¿La forma de navegación hace uso de la filosofía de manejo de pantallas y/o ventanas del ambiente sobre el cual está implementado el producto de software?*

- Emule al usuario en su actividad diaria y tome en cuenta el volumen de información que maneja con esa interfase y la frecuencia con que lo hace, además del tiempo de respuesta del cliente que espera al usuario (en caso de que lo haya).
- Tomando en cuenta todo lo anterior: ¿La navegación entre formas y campos de captura entorpece la tarea rutinaria del usuario? ¿Los recursos para efectuar la navegación son los más adecuados (ratón, teclado, etc.)?
- Pruebe el producto de software en situaciones de error. ¿La navegación se ve entorpecida al haber errores? (Ej. : El producto marca error en formato en un campo, pero no posiciona el cursor en el campo que debe corregirse).

59. *¿Está la interfase orientada al tipo de actividad y usuario final para el que se planeó?*

- Aplique el punto 1 de la pregunta anterior.
- ¿La interfase, ayuda o entorpece la realización de la actividad con respecto a lo que la misma tomaría haciéndose en forma manual?
- ¿Puede manejar el usuario la información que necesita con esa interfase?
- Revisar que la interfase no presente información en forma ambigua para el usuario.
- Revisar que no se presente información ni elementos superfluos para la actividad de la interfase. Ej. : Una interfase para una presentación de ejecutivos puede contener gráficas que serían innecesarias en una interfase de captura de datos).

60. *¿Cumple los estándares de interfase?*

- Comparar la apariencia y funcionalidad de la interfase con los estándares establecidos para interfases del ambiente en que se desarrolló el producto de software.

7.4.6. Seguridad.

En esta sección se evalúa la calidad de la seguridad del producto de software.

61. *¿Existen mecanismos de control y de protección para los productos de software y los datos?*

- Revisar que el producto de software esté registrado en las herramientas de seguridad pertinentes, de acuerdo al ambiente en que éste fue desarrollado.

- Revisar permisos de acceso a elementos de la base de datos. ¿Tienen acceso sólo las personas o programas que así lo requieren?

62. *¿Se cuenta con permisos específicos de uso para los productos de software?*

- Revise los permisos de acceso para los usuarios del producto de software de acuerdo al ambiente.

63. *¿Está registrado el producto de software en el administrador de documentos?*

- Verificar con el desarrollador si el producto está debidamente registrado en el ADDOC y revisar que esté en el proyecto correcto.

7.5. Auditoría.

Esta sección es sólo una revisión general para verificar que el producto de software tiene como mínimo las siguientes características: documentación, programas fuentes y códigos almacenados en lugar accesible y transmisión del conocimiento del producto. Esta revisión es parecida a la que realizan los auditores de informática y se debe realizar al final del proyecto a manera de revisión general.

7.5.1. Documentación.

En esta sección se realiza la revisión de la existencia de la documentación.

64. *¿Existen manuales técnicos y operativos del producto?*

- Revisar documento de "*Diseño*".

65. *¿Existe documentación que explique el sistema al usuario y lo que éste debe de hacer?*

- ¿Existe "Manual de Usuario"?

7.5.2. Almacenamiento de documentación.

En esta sección se revisa si el producto de software y su documentación están almacenados en el administrador de documentos.

66. *¿Se encuentra el código fuente del producto de software en el administrador de documentos?*

- En este punto es necesario verificar también que esté en el directorio correcto dentro del ADDOC, según el proyecto al que pertenezca.

67. *¿Se encuentra la documentación almacenada en el administrador de documentos?*

Aplicar el punto 1 de la pregunta anterior.

7.5.3. Respaldo de información.

Aquí se verifica que el producto esté debidamente respaldado.

68. *¿Existen respaldos del producto de software?*

- Verificar qué cantidad de respaldos existen y los tipos de medio en los que se respaldó, además de la accesibilidad de estos respaldos para las demás personas que pudieran requerirlos.

7.5.4. Herramientas adicionales.

Aquí se revisa si existen herramientas o procedimientos para verificar los resultados del producto de software.

69. *¿Se encuentra documentado el proceso por el que pasa la información dentro del producto de software, con el fin de producir los resultados deseados para los datos de entrada, y existe forma de verificarlo?*

- ¿Existe un procedimiento manual o una herramienta de software que me permita verificar que los resultados que arroja el producto de software son correctos?
- ¿Está lo anterior documentado en el diseño?

7.5.5. Propagación del conocimiento.

En esta sección se verifica que más de una persona tenga al menos un conocimiento mínimo necesario sobre el producto de software.

70. *¿Se ha instruido a otras personas sobre el funcionamiento interno del producto?*

- ¿Se les ha dado información suficiente a estas personas como para realizar mantenimientos al producto de software y contesten dudas sobre el producto?
- ¿Tienen acceso completo estas personas a los fuentes y la documentación?
- ¿Tienen los passwords necesarios para tener el acceso a los fuentes y a la documentación?

8. - CUESTIONARIO DE EVALUACIÓN PARA EL ASEGURAMIENTO DE LA CALIDAD EN EL SOFTWARE.

Proyecto : _____ Versión: _____

Fecha de Evaluación: ____/____/____

Responsable : _____

Evaluador : _____

INSTRUCCIONES: Marque con 1 las preguntas que se contesten afirmativamente al ser evaluadas.

ANÁLISIS

Análisis de Requerimientos

1. ¿Fueron debidamente documentados los requerimientos? ____
2. ¿Se han identificado en la planeación los requerimientos que serán resueltos en futuras versiones de software? ____
3. ¿La documentación de los requerimientos nos lleva al análisis? ____

Análisis de dominio del problema

4. ¿Es completo, consistente y exacto el análisis del dominio de las especificaciones? Esta pregunta se puede dividir en las siguientes dos partes: ____
   - ¿Se tiene una visión amplia del entorno donde se encontrará la aplicación?
   - ¿Se han tomado en cuenta todas las posibles áreas de oportunidad que presenta dicho entorno? (es decir alcances, alternativas, restricciones, limitaciones)
5. ¿Esta bien definida la parte automatizable (porción del problema a resolverse por el producto de software) y la parte manual del problema? ____
6. ¿Existe documentación que nos lleve del análisis al diseño? ____

DISEÑO

Planeación del diseño

7. ¿Se pueden identificar en el modelo de datos los atributos, tablas o relaciones que se reflejen en la información de futuros requerimientos? ____
8. ¿Las funciones de software se encuentran parametrizadas para poder ser utilizadas para otros requerimientos? ____

Modelo de datos

9. ¿Los elementos del modelo de datos corresponden con los elementos involucrados en el problema? ____
10. ¿Es consistente el modelo de datos con los requerimientos del producto de software? ____

Documentación del diseño

11. ¿Es posible saber fácilmente para qué sirve cada uno de los módulos que comprende el producto de software? ____
    - El módulo tiene un nombre representativo de lo que hace.
    - Su código está comentariado y existe documentación del mismo.
12. ¿Existe documentación del diseño que permita mapearlo hacia la implementación? ____

Funcionalidad

13. ¿Se definen las funciones principales del producto de software en forma limitada y sin ambigüedad? ____
14. ¿Se reflejan los requerimientos del producto de software en el diseño del mismo (tomando en cuenta restricciones internas y externas)? ____
15. ¿Los algoritmos realizan la función o funciones deseadas? ____

Retroalimentación al usuario

16. ¿Proporciona el sistema ayuda en línea al usuario? ____
17. ¿Se utilizan mecanismos de ayuda o "hints"? ____

Arquitectura

18. ¿Están bien definidas las interfases entre los elementos del sistema? (Entradas, salidas, formatos y protocolos bien delimitados y especificados). ____

19. ¿Tiene el sistema una interfase externa que facilite al usuario el interactuar con él sin problemas (apego a estándares)? ____

20. ¿ Tiene el sistema una interfase externa que posibilite la interacción con otros sistemas, ya sea suministrando salidas o recibiendo entradas? ____

21. ¿La interfase con la base de datos utiliza los recursos de acceso adecuados a la situación y a las características de la aplicación?. ____

22. ¿Existe un estándar y estilo en todos los elementos que constituyen una misma aplicación?. (Consistencia entre todos los elementos del sistema) ____

23. ¿Son los módulos funcionalmente independientes? ____

Manipulación de datos

24. ¿La estructura de datos local soporta con precisión los modelos de datos diseñados para la aplicación?. Ej. manejo adecuado de registros. ____

25. ¿Los tipos lógicos, tipos físicos y datos representan con precisión las características de los atributos de las entidades? (Declaraciones apropiadas).
Ej. Declarar nombres como tipo caracter. ____

26. ¿Las cantidades numéricas corresponden lo más cercanamente posible a la realidad?. (Constantes correctas). ____

Algoritmo

27. ¿El algoritmo puede ser comparado con los requerimientos funcionales solicitados?. (Algoritmo lógicamente correcto). ____

28. ¿El manejo de márgenes de error, de truncamiento y de cantidades son adecuados a las características de la información que maneja el producto de *software*? ____

29. ¿Es razonable la complejidad lógica del algoritmo con respecto a la complejidad del problema? ____

Reutilización

30. ¿Se ha empleado el reuso de otros elementos de software? ____

31. ¿Se ha evitado el duplicado de funciones similares? ____

Manejo de errores

32. ¿Se ha definido con detalle que pasará con el funcionamiento del software en caso de que se incurra en un error a la hora de alimentar datos, o bien, en caso de un mal funcionamiento interno?. (Tratamiento y tolerancia de errores) ____

33. ¿Se han implementado procedimientos de recuperación de errores? ____

34. ¿Son claros los mensajes de error? ____

35. ¿Ayudan los mensajes de error a evitar reincidir en los errores? ____

36. ¿Han sido especificados los mensajes de error conforme a los estándares establecidos? ____

37. ¿Se proporciona ayuda adecuada para prevenir errores derivados de la captura?. ____

Interoperabilidad

38. ¿El producto de software o alguno de sus elementos puede ser llamado desde otro producto de software y seguir proporcionando su misma funcionalidad? (Ejemplo: una forma de captura de direcciones que puede ser llamada desde diferentes módulos como personas, escuelas, organizaciones, etc.). ____

39. ¿Se han definido las funciones que el producto de software ofrece a otros? ____

40. ¿Se tiene una especificación clara y sencilla de cómo hacer uso de los módulos y elementos externos del sistema desde otros módulos y sistemas?. ____

Independencia de plataforma

41. ¿Es independiente el diseño del producto de software de características del hardware? ____

42. ¿Es independiente la funcionalidad del producto de software de factores relacionados con el software que utiliza? (lenguaje, DBMS, Sistema Operativo) ____

43. ¿Es independiente el diseño del lenguaje de implementación? ____

44. ¿Se ha evitado utilizar características dependientes del sistema operativo o del lenguaje? ____

IMPLEMENTACIÓN

Consistencia con el diseño

45. ¿Las interfases cumplen con los detalles necesarios que fueron especificados durante la etapa de diseño?. ____

46. ¿Se ha traducido adecuadamente el diseño al código? ____

Apego a estándares

47. ¿Resulta fácil de entender el (los) algoritmo(s) que se ha(n) utilizado para la implementación del producto de software (instrucciones utilizadas, comentarios, identación)? ____

48. ¿Se ha hecho un uso adecuado de las convenciones del lenguaje? ____

49. ¿Se han seguido los estándares de codificación para el estilo del lenguaje, los comentarios y los prólogos de los módulos? ____

50. ¿Se ha buscado el no utilizar comentarios incorrectos o ambiguos? ____
51. ¿Se encuentra el código debidamente documentado? ____
52. ¿Siguen los comentarios los estándares? ____

Optimización de recursos

53. Tomando en cuenta aspectos como la complejidad lógica del problema que se intenta resolver, las facilidades del lenguaje y el reuso de elementos ya hechos. ¿El producto de software es razonablemente compacto en términos de líneas de código? ____
54. ¿Se ha evitado el duplicar líneas de código? ____
55. ¿El tiempo de respuesta es adecuado a las necesidades del usuario? (REFERENCIA: "*Documento de optimización de recursos de cómputo*"). ____
56. ¿Consume muchos recursos al estarse ejecutando el producto de software?

56a. – cpu ____

56b. – memoria ____

56c. – red ____

Independencia de plataforma

57. ¿Es independiente la implementación del producto de software de características del ambiente (software y hardware)? ____

Interfase con el usuario

58. ¿La forma de navegación hace uso de la filosofía de manejo de pantallas y/o ventanas del ambiente sobre el cual está implementado el producto de software? ____
59. ¿Está la interfase orientada al tipo de actividad y usuario final para el que se planeó? ____
60. ¿Cumple los estándares de interfase? ____

Seguridad

61. ¿Existen mecanismos de control y protección para los productos de software y los datos? ____
62. ¿Se cuenta con permisos específicos de uso para los productos de software? ____
63. ¿Está registrado el producto de software en el administrador de documentos (aplica sólo si ya esta liberado el producto de software)? ____

AUDITORÍA

Documentación

64. - ¿Existen manuales técnicos y operativos del producto? ____

65. ¿Existe documentación que explique el sistema al usuario y lo que este debe de hacer? ____

Almacenamiento de documentación y código

66. ¿Se encuentra almacenado el código fuente en el administrador de documentos? ____

67. ¿ Se encuentra almacenada la documentación en el administrador de documentos? ____

Respaldo de información

68. ¿Existen respaldos del producto de software? ____

Herramientas adicionales

69. ¿Se encuentra documentado el proceso por el que pasa la información dentro del producto de software, con el fin de producir los resultados deseados para los datos de entrada y existe forma de verificarlo? ____

Propagación del conocimiento

70. ¿Se ha instruido a otras personas sobre el funcionamiento interno del producto? ____

9. FACTORES DE CALIDAD Y MÉTRICAS

9.1. Propósito.

Definir los factores de calidad y las métricas utilizadas para evaluarlos, así como mostrar las preguntas utilizadas en el cuestionario de evaluación organizadas por métrica en lugar de hacerlo por fase de desarrollo, esto con la finalidad de establecer la manera en que una métrica es evaluada y por lo tanto los factores de calidad.

9.2. Factores de calidad

9.2.1. Corrección

El grado en que un producto de software satisface sus especificaciones y consigue los objetivos de la misión encomendada por el usuario.

9.2.2. Confiabilidad

El grado en que se puede esperar que un producto de software lleve a cabo sus funciones esperadas con la precisión requerida. La probabilidad de operación libre de fallos de un producto de software en un entorno determinado y durante un periodo de tiempo específico.

9.2.3. Eficiencia

La cantidad de recursos computacionales y de código requeridos por un producto de software para llevara a cabo sus funciones.

9.2.4. Integridad

El grado en que puede controlarse el uso y el acceso al software o a los datos por personal no autorizado.

9.2.5. Facilidad de uso

El esfuerzo requerido para aprender, trabajar, preparar la entrada e interpretar la salida de un producto de software.

9.2.6. Facilidad de mantenimiento

El esfuerzo requerido para localizar y arreglar un error en un producto de software.

9.2.7. Flexibilidad

El esfuerzo requerido para modificar un producto de software una vez que se encuentra ya liberado o en producción.

9.2.8. Facilidad de prueba

El esfuerzo requerido para probar un producto de software de forma que se asegure que realiza las funciones requeridas.

9.2.9. Portabilidad

El esfuerzo requerido para transferir un producto de software de una plataforma (entorno de hardware y software) a otra.

9.2.10. Reusabilidad

El grado en que un producto de software (o alguna de sus partes) se puede reusar en otras aplicaciones.

9.2.11. Facilidad de interoperación

El esfuerzo requerido para enlazar un sistema a otro.

9.3. Métricas

En esta sección se listan y definen las métricas tomadas en cuenta para evaluar los factores de calidad; después de cada métrica se muestran las preguntas utilizadas para evaluarla. Los números asociados a las preguntas corresponden a los asignados en el cuestionario de evaluación.

9.3.1. Facilidad de Auditoría

La facilidad con que se puede comprobar la conformidad con los estándares, las especificaciones del producto y la veracidad de la información.

64. *¿Existen manuales técnicos y operativos del producto?*

66. *¿Se encuentra almacenado el código fuente en el administrador de documentos?*

67. *¿Se encuentra almacenada la documentación en el administrador de documentos?*

69. *¿Se encuentra documentado el proceso por el que pasa la información dentro del producto de software, con el fin de producir los resultados deseados para los datos de entrada y existe forma de verificarlo?.*

70. *¿Se ha instruido a otras personas sobre el funcionamiento interno del producto?*

9.3.2. Exactitud

La precisión de los cálculos y el control.

27. *¿El algoritmo puede ser comparado con los requerimientos funcionales solicitados?.* (Algoritmo lógicamente correcto).

28. *¿El manejo de márgenes de error, de truncamiento y de cantidades son adecuados a las características de la información que maneja el producto de software?*

9.3.3. Normalización de la comunicación lógica

El grado en que se usan los protocolos y las interfases estándar.

18. *¿Están bien definidas las interfases entre los elementos del sistema? (Entradas, salidas, formatos y protocolos bien delimitados y especificados).*

19. *¿Tiene el sistema una interfase externa que facilite al usuario el interactuar con él sin problemas (apego a estándares)?*

20. *¿Tiene el sistema una interfase externa que posibilite la interacción con otros sistemas, ya sea suministrando salidas o recibiendo entradas?*

21. *¿La interfase con la base de datos utiliza los recursos de acceso adecuados a la situación y a las características de la aplicación?.*

45. *¿Las interfases cumplen con los detalles necesarios que fueron especificados durante la etapa de diseño?*

9.3.4. Completitud

El grado en que se ha conseguido la total implementación de los *requerimientos organizacionales así como los de diseño.*

4. *¿Es completo, consistente y exacto el análisis del dominio de las especificaciones? Esta pregunta se puede dividir en las siguientes dos partes:*

   - *¿Se tiene una visión amplia del entorno donde se encontrará la aplicación?*
   - *¿Se han tomado en cuenta todas las posibles áreas de oportunidad que*

presenta dicho entorno? (es decir alcances, alternativas, restricciones, limitaciones)

5. *¿Esta bien definida la parte automatizable (porción del problema a resolverse por el producto de software) y la parte manual del problema?*

13. *¿Se definen las funciones principales del producto de software en forma limitada y sin ambigüedad?*

14. *¿Se reflejan los requerimientos del producto de software en el diseño del mismo (tomando en cuenta restricciones internas y externas)?*

15. *¿Realizan los algoritmos la función o funciones deseadas?*

45. *¿Las interfases cumplen con los detalles necesarios que fueron especificados durante la etapa de diseño?*

46. *¿Se ha traducido adecuadamente el diseño al código?*

9.3.5. Simplicidad

El grado en que un producto de software puede ser entendido sin dificultad.

29. *¿Es razonable la complejidad lógica del algoritmo con respecto a la complejidad del problema?*

47. *¿Resulta fácil de entender el (los) algoritmo(s) que se ha(n) utilizado para la implementación del producto de software (instrucciones utilizadas, comentarios, identación)?*

9.3.6. Concisión

Lo compacto que es el producto de software en términos de líneas de código.

30. *¿Se ha empleado el reuso de otros elementos de software?*

53. Tomando en cuenta aspectos como la complejidad lógica del problema que se intenta resolver, las facilidades del lenguaje y el reuso de elementos ya hechos. *¿El producto de software es razonablemente compacto en términos de líneas de código?*

54. *¿Se ha evitado el duplicar líneas de código?*

9.3.7. Consistencia

El uso de un diseño y estilo uniforme, además de técnicas de documentación a lo largo del proyecto de desarrollo de software.

22. *¿Existe un estándar y estilo en todos los elementos que constituyen una misma aplicación?. (Consistencia entre todos los elementos del sistema)*

48. *¿Se ha hecho un uso adecuado de las convenciones del lenguaje?*

49. *¿Se han seguido los estándares de codificación para el estilo del lenguaje, los comentarios y los prólogos de los módulos?*

50. *¿Se ha buscado el no utilizar comentarios incorrectos o ambiguos?*

9.3.8. Estandarización/Uniformidad

El uso de estándares para el cumplimiento de las normas de diseño y codificación.

9. *¿Los elementos del modelo de datos corresponden con los elementos involucrados en el problema?*

10. *¿Es consistente el modelo de datos con los requerimientos del producto de software?*

24. *¿La estructura de datos local soporta con precisión los modelos de datos diseñados para la aplicación?*. Ej. manejo adecuado de registros.

25. *¿Los tipos lógicos, tipos físicos y datos representan con precisión las características de los atributos de las entidades?* (Declaraciones apropiadas). Ej. declarar nombres como tipo caracter.

26. *¿Las cantidades numéricas corresponden lo más cercanamente posible a la realidad?.* (Constantes correctas).

48. *¿Se ha hecho un uso adecuado de las convenciones del lenguaje?*

49. *¿Se han seguido los estándares de codificación para el estilo del lenguaje, los comentarios y los prólogos de los módulos?*

50. *¿Se ha buscado el no utilizar comentarios incorrectos o ambiguos?*

9.3.9. Manejo de errores

El grado de retroalimentación al usuario y de recuperación del producto de software en caso de errores.

32. *¿Se ha definido con detalle que pasará con el funcionamiento del software en caso de que se incurra en un error a la hora de alimentar datos, o bien, en caso de un mal funcionamiento interno?.* (Tratamiento y tolerancia de errores)

33. *¿Se han implementado procedimientos de recuperación de errores?*

34. *¿Son claros los mensajes de error?*

35. *¿Ayudan los mensajes de error a evitar reincidir en los errores?*

36. *¿Han sido especificados los mensajes de error conforme a los estándares establecidos?*

37. *¿Se proporciona ayuda adecuada para prevenir errores derivados de la captura?.*

9.3.10. Eficiencia en la ejecución

El rendimiento en tiempo y recursos de ejecución de un producto de software.

55. *¿El tiempo de respuesta es adecuado a las necesidades del usuario?*

56. *¿Consume muchos recursos al estarse ejecutando el producto de software?*

56a. *cpu*

56b. *memoria*

56c. *red*

9.3.11. Facilidad de expansión

El grado en que se puede ampliar el diseño arquitectónico, de datos o procedural.

2. *¿Se han identificado en la planeación los requerimientos que serán resueltos en futuras versiones de software?*

7. *¿Se pueden identificar en el modelo de datos los atributos, tablas o relaciones que se reflejen en la información de futuros requerimientos?*

8. *¿Las funciones de software se encuentran parametrizadas para poder ser utilizadas para otros requerimientos?*

9.3.12. Generalidad

La amplitud de aplicación potencial de los componentes del producto de software.

31. *¿Se ha evitado el duplicado de funciones similares?*

38. *¿El producto de software o alguno de sus elementos puede ser llamado desde otro producto de software y seguir proporcionando su misma funcionalidad?* (Ejemplo: una forma de captura de direcciones que puede ser llamada desde diferentes módulos como personas, escuelas, organizaciones, etc.).

9.3.13. Independencia del hardware

El grado en que el software es independiente del hardware sobre el que opera.

41. *¿Es independiente el diseño del producto de software de características del hardware?*

57. *¿Es independiente la implementación del producto de software de características del ambiente (software y hardware)?*

9.3.14. Retroalimentación

El grado en que el producto de software te permite saber lo que está pasando.

16. *¿Proporciona el sistema ayuda en línea al usuario?*

17. *¿Se utilizan mecanismos de ayuda o "hints"?*

65. *¿Existe documentación que explique el sistema al usuario y lo que este debe de hacer?*

9.3.15. Modularidad

La independencia funcional de los componentes del producto de software.

11. *¿Es posible saber fácilmente para que sirve cada uno de los módulos que comprende el producto de software?*
    - El módulo tiene un nombre representativo de lo que hace.
    - Su código esta comentariado y existe documentación del mismo.

13. *¿Se definen las funciones principales del producto de software en forma limitada y sin ambigüedad?*

39. *¿Se han definido las funciones que el producto de software ofrece a otros?*

23. *¿Son los módulos funcionalmente independientes?*

9.3.16. Facilidad de operación

La simplicidad con que un producto de software puede ser utilizado por los usuarios finales y otros desarrolladores.

19. *¿Tiene el sistema una interfase externa que facilite al usuario el interactuar con él sin problemas (apego a estándares)?*

20. *¿Tiene el sistema una interfase externa que posibilita la interacción con otros sistemas, ya sea suministrando salidas o recibiendo entradas?.*

40. *¿Se tiene una especificación clara y sencilla de cómo hacer uso de los módulos y elementos externos del sistema desde otros módulos y sistemas?*

58. *¿La forma de navegación hace uso de la filosofía de manejo de pantallas y/o ventanas del ambiente sobre el cual esta implementado el producto de software?*

59. *¿Está la interfase orientada al tipo de actividad y usuario final para el que se planeó?*

9.3.17. Seguridad

La disponibilidad de mecanismos que controlen o protejan el acceso a los producto de software y a los datos.

61. *¿Existen mecanismos de control y protección para los productos de software y los datos?*

62. *¿Se cuenta permisos específicos de uso para los producto de software?*

63. *¿Está registrado el producto de software en el administrador de documentos (aplica sólo si ya esta liberado el producto de software)?*

68. *¿Existen respaldos del producto de software?*

9.3.18. Autodocumentación

El grado en que el código fuente proporciona documentación significativa.

50. *¿Se ha buscado el no utilizar comentarios incorrectos o ambiguos?*

51. *¿Se encuentra el código debidamente documentado?*

52. *¿Siguen los comentarios los estándares?*

9.3.19. Independencia del sistema de software

El grado en que el producto de software es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones ambientales.

42. *¿Es independiente la funcionalidad del producto de software de factores relacionados con el software que utiliza?* (lenguaje, DBMS, Sistema Operativo)

43. *¿Es independiente el diseño del lenguaje de implementación?*

44. *¿Se ha evitado utilizar características dependientes del sistema operativo o del lenguaje?*

9.3.20. Facilidad de rastreo

La posibilidad de seguir la pista de la representación del diseño o de los componentes reales del producto de software hacia atrás, hacia los requerimientos.

1. *¿Fueron debidamente documentados los requerimientos?*

3. *¿La documentación de los requerimientos nos lleva al análisis?*

6. *¿Existe documentación que nos lleve del análisis al diseño?*

12. *¿Existe documentación del diseño que permita mapearlo hacia la implementación?.*

51. *¿Se encuentra el código debidamente documentado?*

9.3.21. Inducción

El grado en que el software ayuda para permitir que nuevos usuarios utilicen el sistema.

16. *¿Proporciona el sistema ayuda en línea al usuario?*

34. *¿Son claros los mensajes de error?*

60. *¿Cumple los estándares de interfase?*

9.4. Relación entre factores de calidad y métricas

En la tabla 1 se muestra la relación entre los factores de calidad y las métricas, es decir se indican las métricas implicadas en la evaluación final de cada uno de los factores. De manera horizontal se encuentran los factores de calidad y de manera vertical están las métricas. Por ejemplo, en el caso del factor de Corrección éste está evaluado en función de las métricas: Completitud, Consistencia y Facilidad de rastreo.

Met. / fact.	Corre-Ción	Confia-bilidad	Eficien-cia	Integri-dad	Facil. de Mtto.	Flexibi-lidad	Facil. de Prueba	Portabi-lidad	Reusabi-lidad	Inter-operativ.	Facil. de Uso
Facilidad de Auditoría				X			X				
Exactitud		X									
Normaliza-ción										X	
Completitud	X										
Simplicidad		X				X	X				
Concisión			X		X	X					
Consistencia	X	X			X	X					
Estandariza-ción										X	
Manejo de Errores		X									
Eficiencia de Ejecución			X								
Facilidad de Expansión						X					
Generalidad						X		X	X	X	
Independen-cia del HW								X	X		
Retroalimen -tación				X	X		X				
Modularidad		X			X	X	X	X	X	X	
Facilidad de Operación			X								X
Seguridad				X							
Autodocu-mentación					X	X	X	X	X		
Independen-cia del SW								X	X		
Facilidad de Rastreo	X										
Inducción											X

Tabla 1.

9.5. Procedimiento de evaluación.

Para la obtención automática del resultado global, por métrica y por factor se utilizará un formato en Excel el cual incluye las fórmulas para realizar los cálculos que a continuación se explican.

La manera en como se evalúa un producto de software consiste en los siguientes pasos.

1. ***Contestar las preguntas*** del "Cuestionario de Evaluación para el Aseguramiento de la Calidad en los Productos de software" el cual se encuentra en el capítulo anteriormente expuesto. En dicha forma las preguntas están ordenadas por cada una de las fases del desarrollo. Todas las preguntas se contestan con un 1(*sí*) o con un 0 (*no*).
2. ***Evaluar cada una de las métricas.*** La evaluación de cada métrica se obtiene dividiendo el número de afirmaciones entre el numero total de preguntas de la métrica, multiplicando el resultado por cien. *Sólo* se deben de tomar en cuenta las preguntas de la métrica correspondiente. Esto está explicado en el punto 5.3. de esta sección.
3. ***Establecer una ponderación para cada uno de los factores de calidad.*** Se debe de hacer un análisis objetivo por parte del evaluador de cuáles factores son los más relevantes para la aplicación. La ponderación se lleva a cabo asignando un valor entre 1 y cinco, donde 5 es el valor que se le debe de dar a los factores de calidad de suma relevancia. Ejemplo: si se esta evaluando una forma de Oracle, el factor de portabilidad no tiene mucha relevancia, ya que de antemano se sabe que dichas formas solo trabajan bajo una sola plataforma, así que se le asigna un 1. Es posible que exista más de un factor con relevancia de un mismo orden.
4. ***Obtener porcentajes para cada uno de los factores de calidad*** tomando las ponderaciones definidas. Ejemplo: se asignó un 5 como ponderación al factor de calidad corrección, se divide 5 entre la suma de las ponderaciones de todos los factores y se obtiene el porcentaje buscado.
5. ***Evaluar los factores de calidad.*** Para hacer esto, se debe consultar la tabla No.1 y observar que métricas componen a cada uno de los factores de calidad. Ejemplo: el factor de calidad se ve afectado por las métricas de completitud, consistencia y facilidad; suponiendo que las evaluaciones de éstas tres métricas son 50, 75 y 100 respectivamente, la evaluación del factor corrección sería 75, ((50+75+100)/3)

6. ***Obtener la evaluación final de calidad del producto de software.*** Para hacer esto se aplica la siguiente relación:

$$C_{ps} = P_1XF_1+P_2XF_2+...P_nXF_n$$

Donde:

C_{ps} = Calidad del producto de software.
P_1 = Porcentaje del factor de calidad.
F_1 = Evaluación del factor de calidad.

La evaluación ideal es de 100 puntos, suponiendo que todos los factores de calidad obtuvieron una evaluación de 100 puntos. Para efectos de una aplicación real, un producto de software con una evaluación de 75 puntos se considera como de buena calidad.

10. REPORTE DE LOS RESULTADOS DEL PROCESO DE ASEGURAMIENTO DE LA CALIDAD EN LOS PRODUCTOS DE SOFTWARE

10.1. Propósito

Describir los puntos importantes a considerar al momento de elaborar el reporte los resultados del proceso de aseguramiento de la calidad realizado a través de la evaluación de los productos de software.

10.2. Para quién y cuándo se requiere.

Es recomendable conocer el estilo de la persona a la cual va dirigido el reporte de ACS, es decir, la terminología, el grado de tecnificación y el contenido del mismo, deben estar en función del lector al cual vaya dirigido principalmente. Generalmente el reporte va dirigido al responsable del producto evaluado y en ocasiones puede enviarse al jefe de éste.

Por otro lado, es muy importante que se conozca la fecha para la cual el reporte es requerido. En otras palabras, resulta ser más importante un reporte poco pulido pero a tiempo, que uno vistoso pero entregado tarde.

10.3. Objetivos del reporte.

Se recomienda que los objetivos de un reporte sean consistentes con los de la persona que recibirá dicho reporte, con el fin de hacerlos más efectivos.

Los objetivos cubiertos por un reporte deberán de ser consistentes con aquellos fijados durante las revisiones. Se recomienda que los objetivos de un reporte sean positivos.

10.4. Recomendaciones para el responsable.

Las recomendaciones que se incluyen en un reporte de ACS, dirigidas al responsable del producto evaluado, se sugieren que se presenten desde una perspectiva clara y fácil de entender. Un error común de los grupos de AC es el de

escribir reportes altamente técnicos que resultan ser poco atractivos o difíciles de entender.

En términos generales se sugiere que:

- Las recomendaciones sean escritas con un enfoque positivo.
- Se deben incluir datos suficientes para sustentar las conclusiones y recomendaciones que se manejen en un reporte.
- Aquellos factores significativos o eventos que hayan ocurrido después de una revisión de AC deben incluirse en el reporte si es que estos afectan a las recomendaciones y conclusiones que en éste se manejan. Sin embargo, se debe establecer claramente que tales eventos ocurrieron después de la revisión y que por lo tanto, no fueron objeto del mismo escrutinio que aquellos que si existían antes y durante la revisión.

10.5. Formato del reporte.

Los reportes de ACS deben organizarse físicamente de forma que faciliten al lector la obtención de aquellos puntos más importantes de un forma rápida. El tamaño y el contenido variará dependiendo de la importancia del sistema, de lo extenso de la revisión, y de la frecuencia de las revisiones. Sin embargo, la consistencia en el formato permite al lector obtener la información esencial rápidamente del reporte.

10.6. Incluir referencias a los papeles de trabajo.

Los papeles de trabajo empleados durante las revisiones, deben de organizarse para hacer referencia a la información desde el detalle a lo general y viceversa. Lo anterior permitirá al revisor contestar preguntas acerca de los reportes o de la revisión en forma rápida.

10.7. Revisión previa del reporte.

Como parte de la revisión de un reporte antes de su entrega, el que creador de éste deberá hacerse los siguientes cuestionamientos.

- ¿Se han alcanzado los objetivos fijados con el reporte?
- ¿Resulta entendible, fácil de leer el reporte?

- ¿Está escrito el reporte en un estilo acorde al del lector del mismo y a sus objetivos?

- ¿Es el reporte un reporte profesional?

- ¿Si yo recibiera este reporte, me instaría a tomar acciones derivadas de las recomendaciones?

10.8. Revisión externa del reporte.

De ser posible, se recomienda que una o más personas distintas al creador del reporte final efectúen la revisión del mismo. Los comentarios y recomendaciones que se reciban deben evaluarse, y se deben incorporar aquellas ideas que sirvan a el mejoramiento del reporte.

10.9. Preparación del reporte final.

A continuación se presentan algunas recomendaciones de lo que un reporte final debe incluir además de considerar los incisos anteriormente descritos:

- El reporte debe de ser limpio, titulado y debe de incluir fecha e identificación.

- El realizador del reporte debe incluirse en el reporte.

- El reporte no debe contener ningún error gramatical o de deletreo.

11. CONSIDERACIÓN FINAL

En los capítulos que preceden se abordó el tema de la evolución del software; tocando de manera muy ligera las etapas primigenia e intermedia y se trata con énfasis la época actual, donde la preocupación central se define en la búsqueda de la excelencia. En esta sinuosa carrera se manifiestan diversas tendencias; pero es posible que la más relevante sea la de lograr una operatividad simple, confiable y eficiente.

En virtud de lo anterior, el software en general y los sistemas de información en particular, como un medio indispensable para la administración de las organizaciones de hoy en día, que requieren de información confiable y oportuna para una eficaz toma de decisiones, estos no se escapan de la exigencia del aseguramiento de la calidad en todas sus fases de desarrollo y principalmente en la etapa operacional en la cual resaltan todas sus bondades o deficiencias para el procesamiento de la información, misma que forma parte de los activos de toda organización.

Considero que este trabajo contiene sin tratar de ser una metodología única, los aspectos más relevantes en cuanto a una evaluación objetiva de cada una de las fases de desarrollo del Software, tratando de detectar errores y resolverlos oportunamente para procurar una aplicación confiable y efectiva.

El presente trabajo es una propuesta orientada para cuidar de la calidad en el desarrollo del software. En un mundo de cambios impredecibles en todos los campos de la actividad humana, resulta obvio que no se ha pretendido delinear un modelo metodológico paradigmático. Se trata simplemente de un esfuerzo por dignificar la profesión, con la certeza de que hay un extenso espacio por explorar; también es cierto que falta mucho por hacer y descubrir; pero éste, es un intento de crecer; y es indudable que para expanderse en la comunidad encargada de desarrollar los sistemas computarizados, se requiere confrontar los lineamientos propositivos que nos han ocupado con la praxis. Si la propuesta aquí delineada resiste el cotejo con la realidad, el modelo metodológico, en general resultará confiable.

BIBLIOGRAFÍA

1. *ARRANZ Ramonet Antonio.* Administración de Datos y Archivos por Computadora. Primera Edición. Editorial Limusa.

2. *CADENA Eduardo.* ISO 9000 Una Visión General, Soluciones Avanzadas. Abril 1996.

3. *HAMMER Michael y CHAMPY James.* Reingeniería. Duodécima Reimpresión. Enero 1998. Editorial Norma. Colombia.

4. *MANGANELLI Raymond L. y KLEIN Mark M.* Cómo Hacer Reingeniería. Sexta Reimpresión. Mayo 1997. Editorial Norma. Colombia.

5. *NORRIS Mark y RIGBY Peter.* Ingeniería de Software Explicada. Primera Edición. Editorial Limusa. México D.F.

6. *PA Computers and Telecommunications (PACTEL),* para el departamento e Industria (1985). Benefits of Software Engineering Methods and Tools.

7. *PC COMPUTING en Español.* Revista Septiembre 1998.

8. *PC MAGAZINE en Español.* Revista volumen 9 número 90.

9. *PÉREZ Ávila Noé.* Como Hacer una Investigación. Primera Edición. Editorial EDIPSA. México 1991.

10. *PRESSMAN Roger.* Ingeniería de Software. Un enfoque practico. Cuarta edición. Editorial Mc Graw Hill. España.

11. *SOSA Pulido Demetrio.* Calidad Total. Primera Edición. Editorial Limusa. México D. F.

12. *SPENDOLINI Michael J.* Benchmarking. Segunda Reimpresión. Enero 1995. Editorial Norma. Colombia.

13. *STEBBING Lionel.* Aseguramiento de la Calidad. Tercera Reimpresión. Editorial CECSA. México 1996.

14. *TICK IT.* A Guide to Software Quality Management System Construction and Certification. Febrero 1992.

15. http://www.fciencias.unam.mx/revista/soluciones/30s/No37/tick-it.html

16. http://www.kyoncorp.com/mcg/gestion

17. http://infonew.sld.cu/revista/aci/aci05395.htm

18. http://www.ati.es/PUBLICACIONES/novatica/1997/128/nv128.pre.html

19. http://www.ati.es/NOTICIAS/doc/19980530a.html.