UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

PROPUESTA PARA EL ASEGURAMIENTO DE CALIDAD EN EL DESARROLLO DE SOFTWARE

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

MAESTRO EN INGENIERÍA (PLANEACIÓN)

P R E S E N T A

AURELIO ADOLFO MILLÁN NÁJERA

DIRECTOR DE TESIS: M.I. OCTAVIO ESTRADA CASTILLO

CIUDAD UNIVERSITARIA, 1998

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Dedico esta tesis:

a la memoria de mi padre Vicente Millán B.;

a mi esposa Lucía Vázquez R.;

a mi madre Carmen Nájera B.;

a mi abuelita Asunción Barrera M.;

a mis hermanos Alejandro y Dolores;

a mis sobrinos Berenice, Belen, Diego, Isabella y Paola;

a mis demás familiares, amigos y compañeros;

a mis profesores Arturo Fuentes Z., Gabriel Sánchez G., Gonzalo Negroe P. (†), Idalia Flores de la M., Javier Suárez R., Octavio Estrada C., Patricia Aguilar J. y Rubén Téllez S.

Agradecimiento:

Al M. en I. Octavio Estrada Castillo por su guía y dedicación en la dirección de esta tesis.

PRÓLOGO

En la actualidad el desarrollo de programas para computadora (software) se ha incrementado grandemente en México y en el mundo, debido al gran auge que ha tenido la computación. La mayoría de las empresas quieren utilizar a la computadora como una herramienta para realizar los procedimientos que efectúan actualmente en forma manual. Dichos procedimientos están ahora siendo automatizados para ser ejecutados por una computadora, por lo que se requiere elaborar programas de computadoras para ellos, los cuales serán integrados en sistemas de información, con la intención de contar con información verídica y proporcionarla en el momento adecuado para la toma de decisiones y de esta manera ofrecer una mejor atención y servicio a sus clientes, aunque en la realidad los programas para computadora desarrollados presentan diferentes problemas.

Este trabajo va dirigido principalmente a los desarrolladores de software de un área de cómputo, en donde no se cuenta con procedimientos sistematizados y documentados necesarios para poder elaborar programas de computadora. A menudo, los desarrolladores se enfrentan al problema de fabricar un software y éste, antes de liberarlo, presenta diferentes problemas, como pueden ser: que no cumple con la totalidad de los requerimientos del usuario; que no cumple los requerimientos técnicos estipulados; que el software no resuelve todo lo que se quería originalmente; que el software no presenta la adecuada documentación e información para futuros mantenimientos; que su mantenimiento se vuelve muy difícil de realizarse, debido a que no se cuenta con los manuales técnicos respectivos; que no siguieron estándares de programación; que el software no se concluye en el tiempo establecido; que el software presenta tiempos de respuestas no adecuados; etc.

Ahora bien, por otro lado, puede suceder que el software desarrollado cumple con todos los requerimientos del usuario, tanto funcionales como técnicos, pero la forma como se liberó el sistema no fue la más adecuada, por ejemplo no se capacitó a las personas que lo utilizarían, o no se proporcionó toda la información necesaria para utilizarlo, etc., por lo que el sistema no es bien aceptado.

Todo lo anterior se detecta, surge o sale a la vista principalmente cuando se quiere liberar el sistema a producción, es entonces, cuando los desarrolladores se dan cuenta de la necesidad de contar con un mecanismo por medio del cual, se garantice o se asegure la calidad del software que se desarrolla, cumpliendo entre otras cosas con todos los requerimientos del usuario, tanto explícitos como implícitos; así como ir complementando todos aquellos procedimientos con que cuentan para desarrollar un software adecuado a las necesidades; con los estándares establecidos; con los procedimientos definidos; en los tiempos fijados, etc.

Hoy en día, existe una gran cantidad de enfoques para desarrollar software, los cuales son muy diversos, heterogéneos y están englobados todos dentro de la ingeniería de software. Así mismo el enfoque que se le da al aseguramiento de

calidad en el desarrollo de software actualmente no es suficiente, por lo que se requiere que las empresas desarrolladoras de software tengan la confianza y seguridad de que los productos que elaboran satisfacen en primer lugar con todos los requisitos que desea el cliente y además cumplan con los requerimientos de calidad esperados.

Por otro lado, no existe un documento que reúna las actividades, responsabilidades, formatos, procedimientos, estándares, etc., que se requieren llevar a cabo para realizar el aseguramiento mencionado; por esto se pensó que sería de mucho valor para los desarrolladores de software, proporcionarles una propuesta de un documento que les sirviera de base o como complemento para poder lograr de una mejor manera el aseguramiento de la calidad en el desarrollo del software; además considero que dicha propuesta se convierte también de importancia para los directivos, jefes de departamento, clientes, etc., ya que les ofrece una forma de detectar posibles problemas, tomando las acciones necesarias, en el momento adecuado, con lo cual se disminuirán problemas, costos y tiempos.

Por todo lo anteriormente dicho, el objetivo general de la presente tesis es el de elaborar una propuesta para el aseguramiento de calidad en el desarrollo de software.

Para lograrlo se tendrán los siguientes objetivos específicos:

En primer lugar se planteará la problemática sobre el aseguramiento de calidad del software. (Capítulo I).

En segundo lugar se realizará una investigación sobre el tema de aseguramiento de calidad en el desarrollo de software, presentando por un lado, algunos enfoques para el desarrollo de software, y por otro lado, los enfoques sobre el aseguramiento de la calidad del mismo. Se hará un análisis comparativo de los enfoques, presentando los puntos más importantes que toman en consideración, las ventajas y desventajas de cada uno de ellos, se aplicará la técnica TKJ para obtener los puntos substanciales que deben tener, tanto un proceso de desarrollo de software, como el aseguramiento de la calidad del mismo. (Capítulo II).

En tercer lugar se presentará la propuesta para el aseguramiento de calidad en el desarrollo de software, la cual contendrá un plan de inspección para la misma. (Capítulo III).

ÍNDICE

I. Introducción

I.1- Problemática actual

Reflexionemos un poco sobre lo que está ocurriendo en México y en el resto del mundo en relación al desarrollo de software o de programas para computadora en las empresas; dicho desarrollo se ha incrementado grandemente, debido al gran auge que ha tenido la computación, véase la figura 1.

Actualmente tanto el sector público como el sector privado utilizan a la computadora como un auxiliar para llevar el control de la información, así como para ejecutar procesos automatizados que ellos tenían que realizar, por ejemplo: en bancos, en tesorerías, en hospitales, en reservaciones aéreas, en hoteles, en centros de autoservicio, en farmacias, en restaurantes, en compañías telefónicas, en agencias de mantenimiento de autos, en universidades, etc., en todas las anteriores empresas se requieren de programas de computadoras específicos que efectúen los procedimientos requeridos para cada una de ellas.

A continuación se verá la problemática por ejemplo en un banco, en éste se lleva el control de todos sus cuenta habientes a través de la computadora, es decir, tiene almacenada en grandes bases de datos toda aquella información relacionada con ellos, que en un momento dado tiene que consultar para decidir en un cajero automático si se le proporciona o no dinero, dependiendo si éste tiene dinero o no en su cuenta o si cuenta con crédito disponible para otorgárselo, etc.

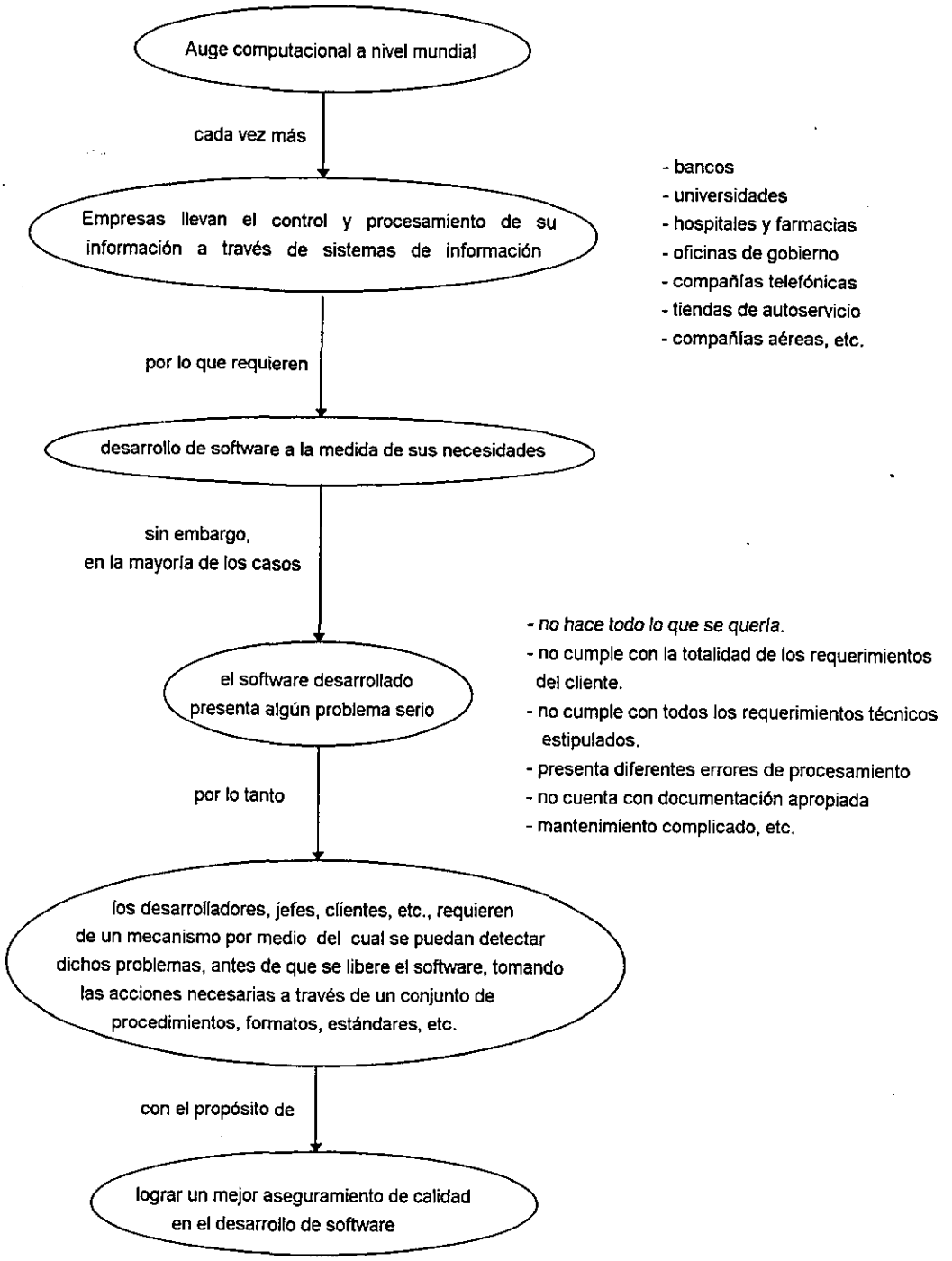

Figura 1. Problemática presentada sobre el aseguramiento de calidad en el desarrollo de software en las empresas.

El banco necesita saber toda esta información para poder tomar una decisión en un instante de tiempo, entonces el banco requiere en primer lugar tener esa

información grabada en medios informáticos, y en segundo lugar debe de contar con un conjunto de programas para computadora adecuados para cubrir todas las necesidades propias del mismo, porque de nada serviría contar con mucha información guardada en algún lugar y no poder utilizarla apropiadamente, debido a que el software desarrollado no cuenta con la calidad requerida en cuanto a tiempo de respuesta, o que no se tuvieron en cuenta algunas opciones, o no proporciona la información correcta, o no hace lo que se quería.

Como se puede observar el banco depende fuertemente de sus sistemas de cómputo, tanto software como hardware (programas y equipos físicos). De ahí la importancia que tiene el tema de aseguramiento de la calidad del software (SQA, por sus siglas en inglés, Software Quality Assurance), ya que la mayoría de las compañías mencionadas anteriormente dependen en un alto grado del manejo de su información a través de sistemas de información.

Para las empresas cada vez es más necesario contar con sistemas de información, que de alguna manera les ayuden a dar un mejor servicio a sus clientes y sobre todo tener la confianza de que la información que están proporcionando los programas de cómputo sea la correcta, así como la integridad de la misma, es decir, que todas las actualizaciones y operaciones que hagan dichos programas sobre la información, queden hechas debidamente.

Las compañías esperan que los sistemas de información les proporcionen información verídica en el instante adecuado, por lo tanto se requiere que los programas desarrollados para sus sistemas de información tengan una alta calidad. Pero, ¿qué se entiende que un programa de computadora desarrollado tenga calidad?, por ejemplo en el caso del banco sería: que el software no tuviera problemas de fallas o interrupciones debidas a variables no tomadas en cuenta; que se cuente siempre con información verídica; que el programa de cómputo realice todas las operaciones correctamente y actualice todas sus bases de datos involucradas; que todos los programa de cómputo cuenten con el soporte de documentación necesarios para que en caso de falla, modificaciones o actualizaciones, se puedan hacer de una manera adecuada y correcta; así mismo se deberá de actualizar la información que se modificó o aumentó, así como la adecuada publicación y distribución necesaria entre todo el personal involucrado para su correcto manejo, etc.

Por otro lado, debido a la situación de crisis económica que prevalece en México y en muchos otros países, el desarrollo de software, hoy en día, ocupa un lugar muy importante en empresas nacionales como internacionales, que tienen la necesidad actual de desarrollar software a la medida de sus necesidades. Además, la importancia de contar con información verídica y oportuna para la toma de decisiones, en estos tiempos hace que el software desarrollado deba de cumplir con todos los requerimientos estipulados por el usuario y con los

requerimientos técnicos establecidos, ya que si no es así, se tendrán serios problemas en las empresas que lo utilizarán.

Actualmente el software ha superado al hardware como la clave del éxito de muchos sistemas basados en computadoras. Lo que diferencia a una compañía con su competidora es la oportuna obtención y veracidad de la información dada por el software que emplean.

Durante las primeras décadas de la informática el principal desafío era el desarrollo del hardware, de manera que se redujera el costo de procesamiento y almacenamiento. Ahora el problema es mejorar la calidad del software.

Hace algunos años bastaba con conseguirse algunos paquetes de software y las compañías se adaptaban a ellos, en muchas ocasiones haciendo pasos intermedios manuales para su ejecución, e intercalando el uso y empleo de varios de ellos; pero ahora parece que todo ha cambiado, sobre todo con el Tratado de Libre Comercio, en el cual empresas de diferentes países deben de competir con sus similares. Por dicha razón las empresas que desarrollan software están preocupadas en que los productos que elaboren, en este caso productos de software, deben producirse con calidad, ya que si no se tiene la calidad necesaria en el bien que se produce, la empresa empezará a perder clientes, los cuales no estarán satisfechos con el producto adquirido o desarrollado, ya que a su vez tendrán serios problemas internos debidos a los resultados que presenta el software desarrollado.

Por tal motivo, hoy en día todas las personas y empresas que elaboran software, deben de cuidar la calidad de sus productos desarrollados, por lo tanto, el tema de aseguramiento de la calidad del software se vuelve muy importante e incluso vital para la supervivencia de las empresas, ya que ahora no solamente se compite con empresas nacionales, sino que han entrado a México muchas compañías que producen software; además, la situación económica que actualmente se vive, aquí y en muchas otras partes, ha obligado a que las mismas empresas desarrollen sus programas de software que necesiten, es decir software a la medida de sus necesidades, que les resuelva su problemática y sobre todo con los recursos humanos y las limitaciones con que cuenta cada una de ellas.

Por todo lo anterior, se tiene la necesidad de lograr que los productos de software que se elaboren tengan calidad. Se sabe que para lograr calidad en un producto se necesitan controlar muchos aspectos como pueden ser: contar con la gente más capaz para desarrollar el software, se requiere que esta gente esté actualizada, capacitada, que sigan metodologías de desarrollo, planes de aseguramiento de calidad y sobre todo que sea profesional y ética.

El aseguramiento de calidad del software actualmente en México se lleva a cabo mediante la ingeniería del software, a través de la aplicación de sus diferentes

paradigmas que se emplean en el desarrollo de sistemas de programación, entre los que se pueden nombrar, entre otros: Ciclo de vida clásico, Construcción de prototipos, Técnicas de cuarta generación, etc. Todos los anteriores paradigmas están dirigidos al desarrollo de software, abarcando el tema de aseguramiento de la calidad de una manera muy general, como una parte del mismo desarrollo, es decir, la parte de los métodos que cubre el aseguramiento, por ejemplo, en la mayoría de ellos sería la correspondiente a las pruebas o evaluaciones que se le realizan al sistema para comprobar que hace lo que se quería y que los resultados son los que esperaban, tratando de descubrir todos aquellos errores que pudiera tener el software desarrollado.

El enfoque que se le da al aseguramiento de calidad del software actualmente no es suficiente, ya que los desarrolladores de programas de computadora de las empresas necesitan tener la seguridad de que los productos que elaboran cumplan en primer lugar con todos los requisitos que desea el cliente y además se debe de tener el cuidado, control y seguimiento, de que los productos que se elaboren cumplan con los requerimientos de calidad esperados, como por ejemplo, que cumpla con los estándares tanto nacionales como internacionales, por lo tanto, se debe de tener un conjunto de actividades de aseguramiento, con el objeto de brindar la confianza apropiada de que el software que se está diseñando cumplirá con todos los requerimientos establecidos por el usuario, así como con los requerimientos establecidos por los estándares durante todas las etapas de desarrollo del mismo y no solamente en la etapa de pruebas antes mencionada.

Algo importante que se puede mencionar al aplicar la etapa de pruebas al software diseñado, es corregir todos aquellos problemas que se vayan presentando en dicha etapa, (la cual es una de las últimas etapas de las metodologías) pero el aseguramiento de calidad va más allá, es decir, se debe prevenir de alguna manera que no se presenten dichos problemas hasta ese punto de la metodología, sino que antes de llegar ahí, sean detectados y corregidos, por lo tanto, entre más rápido se detecte una desviación, más sencillo y barato será corregirla.

El crecimiento en costo e importancia del software debe originar la creación de guías o procedimientos relacionados con el aseguramiento de calidad del software durante el desarrollo del mismo.

Estas guías o planes deberán contener conceptos y prácticas de aseguramiento del software, los cuales identifiquen actividades específicas que caen dentro de la disciplina de aseguramiento del software y proveen información detallada para el desarrollador o gestor, convirtiéndose en una técnica fundamental de aseguramiento de calidad.

El aseguramiento de calidad del software no es solamente detectar y corregir la mayoría de los errores que presenta un sistema de software antes de entrar en producción, si no que además es conseguir la satisfacción del cliente, cumpliendo con todos los requisitos implícitos tanto de usuarios, técnicos, como de estándares, a través de la aplicación de los métodos, herramientas y procedimientos de desarrollo, así como generar procedimientos para el aseguramiento de calidad del software.

El aseguramiento de calidad tiene un fuerte impacto directo sobre el costo y planes de un proyecto, si no es bien planeado y bien implementado, puede ocasionar grandes retrasos en el proyecto y gastos fuertes no considerados.

En pocas palabras la intención de la tesis es realizar una investigación sobre el tema de aseguramiento de calidad del software, presentando algunos enfoques y a partir de ahí, se planteará y desarrollará una propuesta para el aseguramiento de calidad en el desarrollo de software.

I.2- Objetivos

Una vez planteada la problemática sobre el aseguramiento de calidad del software, el objetivo general de la tesis es elaborar una propuesta para el aseguramiento de calidad en el desarrollo de software, la cual sirva de base o de complemento a los desarrolladores de programas de computadora de un departamento de cómputo.

Para alcanzar este objetivo general se tendrán los objetivos específicos siguientes:

- Se realizará una investigación para conocer el estado del arte que guarda el tema del aseguramiento de calidad del software.

- Se presentará en forma breve el marco teórico de algunos enfoques tanto para el desarrollo del software como para el aseguramiento de calidad del mismo, con el propósito de que sirva de base para este trabajo y siguientes investigaciones.

- Se hará un análisis comparativo de los enfoques, presentando los puntos más importantes que toman en consideración, las ventajas, desventajas de cada uno de ellos, se obtendrán los puntos substanciales que deben tener, tanto un proceso de desarrollo de software como el de aseguramiento de calidad del mismo.

- Finalmente, se presentará una propuesta para el aseguramiento de calidad en el desarrollo de software, la cual contendrá un plan de inspección para la misma.

I.3- Metodología a seguir para el desarrollo del presente trabajo

La metodología que se seguirá para el desarrollo del siguiente trabajo de tesis será:

- Primero se realizará una investigación sobre el tema de aseguramiento de calidad del software, para lo cual se hará una recopilación y análisis de la información sobre el tema, en libros, revistas, internet, etc.

- Se presentará el marco teórico de algunos enfoques sobre procesos de desarrollo de software, así como del aseguramiento de calidad del mismo, presentándolos en forma de resumen.

- Se realizará un análisis de los diferentes enfoques, presentándose los cuadros resúmenes respectivos de los puntos más importantes que consideran, así como las ventajas y desventajas que presentan cada uno de ellos. Mediante la utilización de la técnica TKJ se obtendrán, por un lado, las etapas necesarias que debe contener un proceso de desarrollo de software y por otro, los puntos que se deben tomar en consideración para obtener un mejor aseguramiento de calidad del mismo, presentando los resultados obtenidos en forma de cuadros resúmenes.

- Basándose en la investigación y en el análisis anterior, se planteará y desarrollará una propuesta para el aseguramiento de calidad en el desarrollo de software, presentando los resultados en forma de cuadros resúmenes, así como el marco de referencia correspondiente y el desarrollo de cada uno de los puntos que considera la propuesta, concluyendo con un cuadro resumen de un ejemplo de un plan de inspección para dicha propuesta. Cabe hacer mención que en varios puntos del trabajo se hace referencia a anexos, con la idea de que el lector pueda complementar de una manera más detallada, si así lo desea.

- Y por último se presentarán las conclusiones correspondientes.

II.- Marco teórico

II.1. Conceptos generales

Para iniciar se darán algunas definiciones de varios términos que se manejarán a lo largo de este trabajo.

Una definición común para el software sería la siguiente: el software son programas de computadora (conjunto de instrucciones), los cuales permiten adaptar a la computadora a diferentes fines u objetivos, por ejemplo: el software que permite calcular la nómina de los empleados de una empresa; el software que permite llevar el control de los pagos de predial en el Gobierno; el software que permite resolver una ecuación de segundo orden; el software que permite tener un procesador de palabras, en una computadora, etc.

Según la norma NMX-CC-001[1] y NMX-CC-002/1[2], 1995, el software es una categoría genérica de un producto, es decir, un producto es el resultado de actividades o de procesos, pudiendo ser tangible o intangible o bien una combinación de los dos.

En la norma ISO 8402 (administración y aseguramiento de la calidad), los productos se clasifican en cuatro categorías genéricas, véase figura 2.

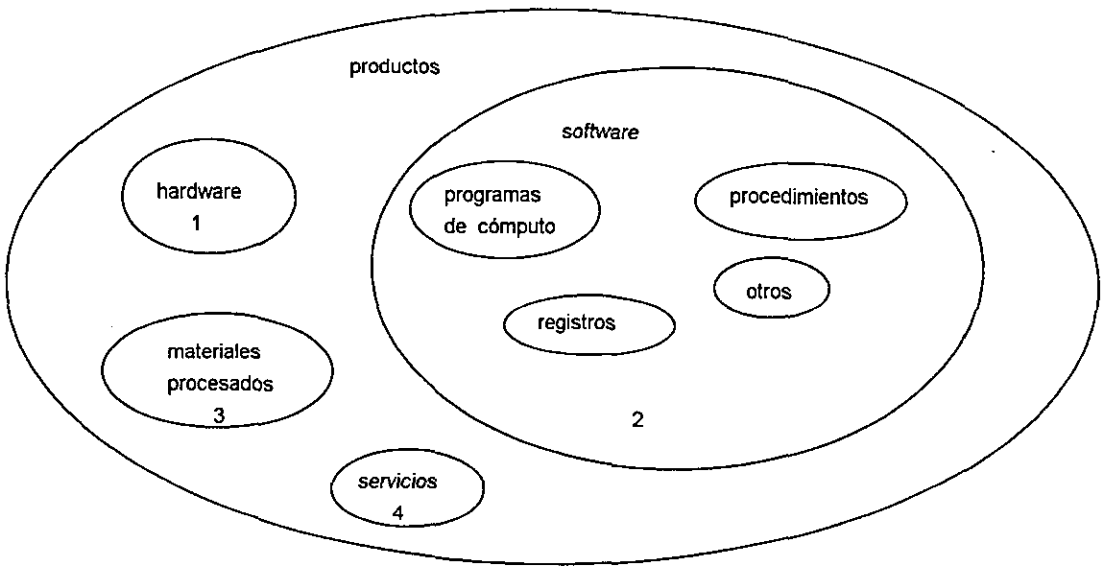

Figura 2. Clasificación de los productos, según la norma ISO 8402.

Como se puede observar en la figura anterior un producto puede ser:

1. Hardware. Un producto tangible con características distintas, por ejemplo, piezas, componentes, ensambles, etc.;

2. Software. Una creación intelectual que consiste en información, expresada a través de medios de soporte, por ejemplo, programas de cómputo, procedimientos, información, datos, registros, etc.;

3. Materiales procesados. Un producto tangible generado por la transformación de materias primas en un estado deseado, por ejemplo, materias primas, líquidos, sólidos, gases, alambres, etc.;

4. Servicios. Es el resultado generado por actividades en la interrelación entre el proveedor y el cliente y por las actividades internas del proveedor para satisfacer las necesidades del cliente, por ejemplo, mantenimiento, garantía, etc.

Por otro lado, los sistemas de información, están compuestos de cinco componentes principales, véase figura 3.

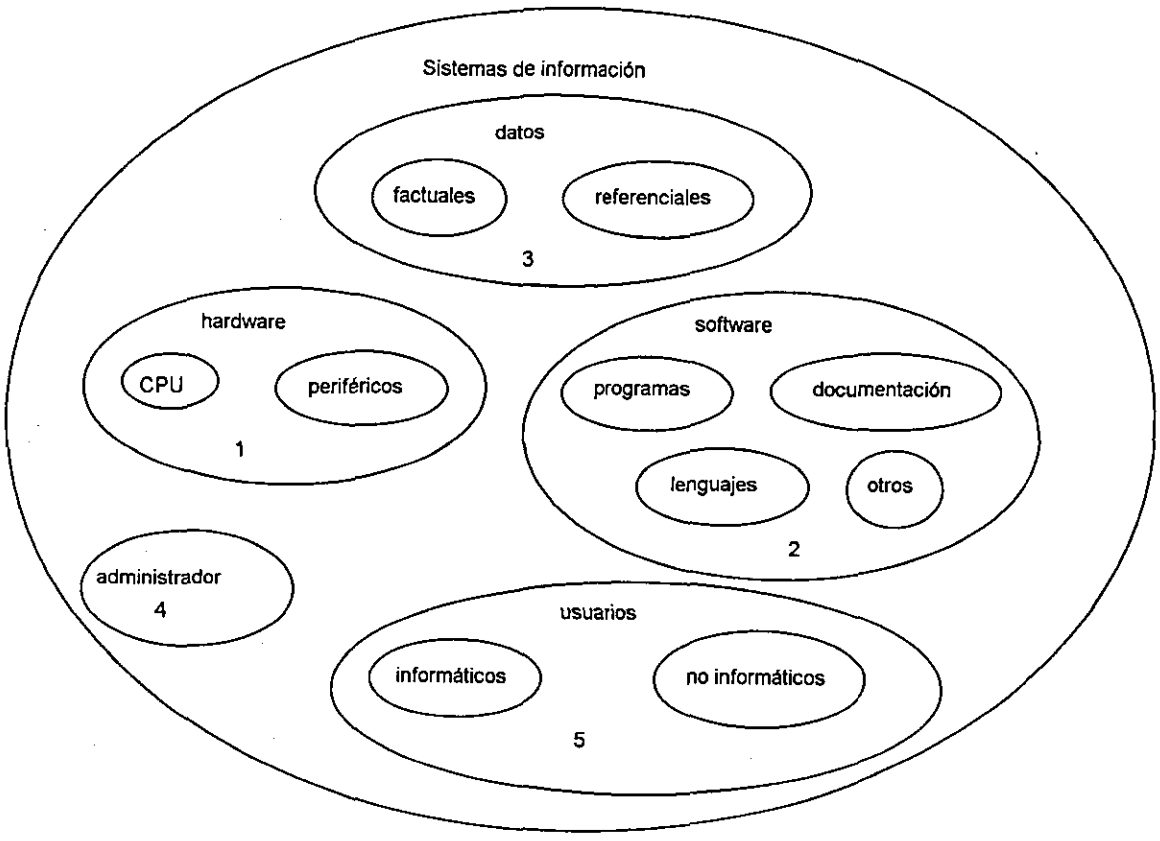

Figura 3. Componentes de un sistema de información.

Un sistema de información está integrado por:

1. Un equipo físico o hardware, el cual a su vez está integrado por una unidad central de procesamiento y periféricos de entrada y salida;

2. Un soporte lógico o software, el cual está integrado por un conjunto de programas, documentación, lenguajes, etc. El software debe realizar una gestión de los datos, (es decir, creación, recuperación y actualización), un manejo de las comunicaciones y tratamientos específicos;

3. Un contenido de datos que puede ser factual o referencial;

4. Un administrador, su misión es asegurar la calidad y permitir el uso correcto y permanente de los datos memorizados y ;

5. Los usuarios, es decir, el grupo de personas que han de acceder al sistema de información, los cuales pueden ser informáticos o no informáticos.

Ahora bien, las definiciones de inspección y calidad, según la Norma Mexicana IMNC NMX-CC-001, son:

La inspección es una actividad tal como la medición, comprobación, prueba, o comparación de una o más características de un elemento y confrontar los resultados con los requisitos especificados, a fin de establecer el logro de la conformidad para cada una de estas características;

La calidad es el conjunto de características de un elemento que le confieren la aptitud para satisfacer necesidades explícitas e implícitas.

Pero, ¿qué se entiende por calidad de software?, según Pressman[3], 1995, establece que la calidad del software se define como:

... "la Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente"... " la anterior definición sirve para hacer hincapié en tres puntos importantes: 1. Los requisitos del software son la base de las medidas de la calidad"... "2. Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería del software"... "3. Existe un conjunto de requisitos implícitos que a menudo no se mencionan"... por lo que concluye que ... "la calidad del software es una mezcla de ciertos factores que varían para las diferentes aplicaciones y los clientes que las solicitan"...

El aseguramiento de calidad del software[3y4], 1995, es una "actividad de protección", que se aplica a lo largo de todo el proceso de Ingeniería de Software, engloba: métodos y herramientas de análisis, diseño, codificación y prueba; revisiones técnicas formales que se aplican durante cada paso de la Ingeniería del Software; una estrategia de prueba multiescalada; el control de la documentación del software y de los cambios realizados; un procedimiento que asegure un ajuste a los estándares de desarrollo del software; y mecanismos de medida y de información.

El aseguramiento de calidad del software es una actividad esencial en cualquier empresa que produce productos que van a ser usados por otros.

Durante los primeros años de la informática (50, 60), la calidad era responsabilidad únicamente del programador.

Durante los años setentas se introdujeron estándares de garantía para el software en los contratos militares de desarrollo de software.

Actualmente, la responsabilidad del aseguramiento de calidad en el desarrollo de software, corresponde a todos los constituyentes de una organización que de una u otra manera participan en el desarrollo del mismo, entre los que podemos mencionar a: ingenieros de computación o de software, gestores del proyecto, clientes, desarrolladores, personas que trabajan dentro del grupo de aseguramiento de la calidad, etc.

El propósito principal es asegurar que el software desarrollado en cualquier lugar, cumpla con todos los requerimientos del usuario y como propósito secundario, definir e implementar reglas específicas con las cuales se asegure que el software que se desarrolle sea un producto de alta calidad.

Un plan de inspección es un mecanismo para detectar problemas, por lo tanto se vuelve indispensable en el aseguramiento de la calidad.

Un plan de inspección es un documento que en forma esquemática, abreviada y haciendo referencia a otros documentos, trata de dar respuesta a las siguientes preguntas:

¿Qué inspeccionar?
¿Quién inspecciona?
¿Dónde se inspecciona?
¿Cómo se inspecciona?
¿Cada cuándo?
¿Dónde se registra la inspección?
¿Cómo comparar con lo deseado?
¿Cómo reaccionar si no se cumple lo deseado?

A continuación se presentan en forma general algunos enfoques sobre procesos de desarrollo de software.

II.2.- Diferentes enfoques para el desarrollo de software

II.2.1.- Enfoque del ciclo de vida clásico para el desarrollo de software, según Pressman, Sommerville[5] y Fairley[6]

- Ingeniería del sistema. Debido a que el software es parte de un sistema mayor, se establecen los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software;

- Análisis de los requisitos del software. Se debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas. Los requisitos, tanto del sistema como del software, se documentan y se revisan con el cliente;

- Diseño. El diseño cubre la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz, los cuales deben estar documentados;

- Codificación. El diseño se traduce a una forma legible para la computadora;

- Prueba. La prueba se centra en la lógica interna del software, asegurando que las entradas producen los resultados que se requieren;

- Mantenimiento. El software suele sufrir cambios una vez que se ha entregado al usuario, éstos debidos a fallas encontradas, a adaptaciones o nuevos requerimientos del usuario.

II.2.2.- Enfoque del ciclo de vida estructurado para el desarrollo de software, según Yourdon[7], 1993

- La encuesta. Esta etapa comienza cuando el usuario solicita que una o más partes de un sistema se automaticen.

  - Identificar a los usuarios responsables y se crea un grupo de trabajo inicial.
  - Identificar las deficiencias actuales en el ambiente del usuario.
  - Establecer metas y objetivos para el nuevo sistema.
  - Determinar si es factible automatizar el sistema y de ser así, sugerir escenarios aceptables.
  - Preparar el esquema que se usará para guiar el resto del proyecto.

- Análisis del sistema. Transformar las políticas del usuario y el esquema del proyecto en una especificación estructurada tomando en cuenta las restricciones administrativas y operacionales, es decir, se modela el ambiente del usuario con diagramas de flujo de datos, diagramas de entidad relación, diagramas de transición de estado, etc. Como producto final, se tendrá una descripción formal de lo que el nuevo sistema debe hacer independientemente de la naturaleza de la tecnología que se utilice para su realización.

- Diseño. Crear una jerarquía apropiada de módulos de programas y de interfaces entre ellas, para implementar la especificación estructurada creada en el análisis. Además se transforman los modelos de datos entidad-relación a un diseño de bases de datos.

- Implementación. Codificar e integrar módulos en un soporte progresivamente más completo del sistema final, por medio de la programación estructurada y la implementación descendente.

- Generación de pruebas de aceptación. Producir un conjunto de casos prueba de aceptación, a partir de la especificación estructurada generada en el análisis, en base al sistema aceptable desde el punto de vista del usuario.

- Garantía de calidad. También conocida como la prueba final o de aceptación, requiere como entrada los datos de la prueba de aceptación y el sistema producido en la implementación, dando como resultado el sistema aceptado. Es importante llevar a cabo esta actividad durante el análisis, diseño y programación para asegurar que se hayan realizado con un nivel apropiado de calidad.

- Descripción de procedimientos. Generar una descripción formal de las partes del sistema que se harán en forma manual, lo mismo que la descripción de cómo interactúan los usuarios con la parte automatizada del nuevo sistema. El resultado final es el manual de usuario.

- Conversión de bases de datos. Esta actividad requiere como entrada la base de datos actual del usuario y la especificación del diseño, generando la base de datos convertida.

- Instalación. La actividad final es la instalación; sus entradas son el manual de usuario, la base de datos convertida y el sistema aceptado.

II.2.3.- Enfoque de la construcción de prototipos para el desarrollo de software, según Pressman, Kendall & Kendall[8] y Fairley

- Recolección y refinamiento de los requisitos. El técnico y el cliente se reúnen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesario una mayor definición;

- Diseño rápido. El diseño rápido se enfoca sobre la representación de los aspectos del software visibles al usuario;

- Construcción del prototipo. Se construye un prototipo;

- Evaluación del prototipo por el cliente. El prototipo es evaluado por el cliente/usuario;

- Refinamiento del prototipo. La anterior etapa permite que el prototipo se refine a los requisitos del software a desarrollar y se pase a la etapa de diseño rápido formando un ciclo interno, así hasta obtener el producto de ingeniería; y

- Producto de ingeniería. Es aquel prototipo que cumple el 100 % de los requerimientos del cliente/usuario, el cual ya no requiere ningún refinamiento.

II.2.4.- Enfoque de las técnicas de cuarta generación para el desarrollo de software, según Pressman y McClure[9]

- Recolección de requisitos. El cliente describe los requisitos;

- Estrategia de diseño. Se desarrolla una estrategia de diseño, utilizando herramientas de cuarta generación;

- Implementación. Se implementa en un lenguaje de cuarta generación y;

- Prueba. Se llevan a cabo las pruebas necesarias.

II.2.5.- Enfoque de un proceso de desarrollo de software, según Pressman

- La fase de definición se centra en el "qué".

En esta etapa se identifican los requisitos clave del sistema y del software, para lo cual se deben de realizar los pasos siguientes:

Análisis del sistema. Se define el papel de cada elemento de un sistema informático;

Planificación del proyecto. Se analizan los riesgos y se asignan los recursos, se estiman los costos, se definen las tareas y se planifica el trabajo y;

Análisis de requisitos. Es necesario disponer de información más detallada.

- La fase de desarrollo se centra en el "cómo".

En esta fase se diseñan las estructuras de datos, así como la estructura del software, se implementa los procedimientos, se traduce el diseño obtenido a un lenguaje de programación y se realizarán las pruebas necesarias, por lo tanto los pasos necesarios de dicha fase serán:

Diseño del software. Se traducen los requisitos del software a un conjunto de representaciones que describen la estructura de los datos, la arquitectura, el procedimiento algorítmico y las características de la interfaz;

Codificación. El diseño debe ser traducido a un lenguaje de programación, dando como resultado las instrucciones ejecutables por la computadora;

Prueba del software. Una vez que ha sido implementado el software, éste debe ser probado para descubrir los errores que pudiera tener, ya sea lógicos, funcionales, etc.

- La fase de mantenimiento se centra en el "cambio" que va asociado a la corrección de errores.

II.2.6.- Enfoque del ciclo de vida del software, según la NASA[10], 1995

- Iniciación y conceptualización del software. Durante esta fase el concepto de software es desarrollado, al igual que la factibilidad del mismo es evaluada, la estrategia de adquisición es desarrollada y se ve si un contrato es necesario para formularlo.

- Definición de requerimientos del software. En esta fase los requerimientos del sistema deben ser analizados y documentados como requerimientos de software. La planeación de pruebas debe empezar con un método para verificar que cada uno de los requerimientos sea identificable e incluirlo en un plan de pruebas preliminar. Métodos, estándares y procedimientos deben ser revisados y detallados. Esta fase termina con la revisión técnica formal de los requerimientos entre el desarrollador y el adquiridor del software.

- Diseño preliminar del software. El objetivo de la fase del diseño preliminar o arquitectónico es desarrollar en forma general el diseño para el software, identificando todos los requerimientos de los componentes del mismo. Se requiere un proceso formal para controlar, manejar y documentar todos los cambios de los requerimientos del software. Esta fase termina con la revisión técnica formal del diseño preliminar. Durante esta fase el solicitante y el desarrollador deben quedar de acuerdo sobre la arquitectura del sistema que se producirá. Los retrabajos y acciones resultantes de esta revisión deben ser revisados, seguir su avance y terminación.

- Diseño detallado del software. Durante la fase del diseño detallado o crítico, el diseño arquitectónico es expandido a nivel de unidad. La documentación de las interfaces de control, debe ser terminada y revisada con planes de pruebas. Las reservas y recursos del sistema objeto deben ser reestimadas y analizadas

y el grupo de gentes de desarrollo y los recursos de pruebas deben ser validados. Esta fase termina con la revisión técnica formal del diseño crítico.

- Programación del software. Durante esta fase el software es codificado y probado modularmente, toda la documentación debe ser realizada en un formato casi de presentación final, incluyendo la documentación del código interno. Al final de esta fase todos los productos deben estar listos para entregarse, debiendo estar sujetos a un período de modificaciones durante la etapa de pruebas.

- Pruebas e integración del software. El objetivo de esta fase es integrar los módulos en un sistema completo, descubriendo y corrigiendo cualquier no conformidad y demostrando que el sistema cumple con las métricas requeridas. Durante esta fase, se ejecuta el plan de pruebas, la documentación debe de estar al día y concluida y los productos deben estar terminados para entregarse. Cuando se aplican las pruebas al sistema, se detectan errores y se inicia una serie de correcciones en el software, en la documentación y en las bases de datos.

- Aceptación y de entrega. Durante la fase de aceptación y entrega, el procedimiento de aceptación formal es llevado a cabo. Como mínimo se debe mostrar que el software desarrollado cumple con los requerimientos solicitados. El proceso también debe incluir pruebas por parte del solicitante, uso de campos y otros arreglos por medio de los cuales se asegure que el software funciona correctamente en el ambiente deseado. Esta fase es muy parecida al final de la fase previa.

- Operación y soporte. Durante esta fase el software es usado para alcanzar los objetivos por los cuales fue desarrollado. Todavía en esta etapa se realizan correcciones y modificaciones para alcanzar la satisfacción total del usuario, así mismo se le da el soporte ingenieril necesario

II.2.7.- Enfoque del modelo en espiral para el desarrollo de software, según Sybase[11], 1995

- Identificación.

Se determinan los requerimientos iniciales y un proyecto cuyo plan se irá ajustando en base a los comentarios del cliente.
Requerimientos del negocio:
Se define el entorno del negocio y los objetivos relacionados con el desarrollo del nuevo sistema. Se plantean los planes de prueba para establecer los criterios de

aceptación del sistema. Por último se revisan los requerimientos definidos para empezar a trabajar con la siguiente etapa.
Requerimientos del sistema:
Se identifican las áreas del negocio involucradas en el sistema. Se establecen los resultados más importantes que deben ser obtenidos por el sistema cuando éste se declare listo tanto por los usuarios como los desarrolladores. Una vez seleccionadas las tareas y los datos que serán implementados en la primera construcción, se revisan los objetivos y el cumplimiento de los estándares que hayan sido fijados.
Requerimientos de los subsistemas:
Los límites entre las funciones relacionadas de los sistemas son establecidos. Además los resultados básicos que cada subsistema debe de obtener son definidos. Se seleccionan los requerimientos que intervendrán en la segunda construcción y finalmente se revisan nuevamente las necesidades.
Requerimientos finales:
Se examinan y validan todos los requerimientos que intervendrán en la segunda construcción y finalmente se revisan nuevamente las necesidades.

- Diseño.

Se empieza a plantear a ciertos niveles las alternativas de solución.
Diseño conceptual:
Las tareas e información son bosquejadas como parte del nuevo sistema. Se presenta el primer intento por representar a detalle la información y por descomponer, transformar y modelar los procesos. Se piensa en un modelo conceptual de interfaz con el usuario y se plantean mecanismos de seguridad. También se debe considerar en forma conceptual la factibilidad de los requerimientos a nivel de servicios, elasticidad, periodos aceptables de desempeño, etc.
Diseño lógico:
El diseño conceptual es llevado a entidades lógicas con atributos y se toman en cuenta los mecanismos para garantizar la integridad de la información. Los procesos son implementados en transacciones que pueda manejar el sistema. La interfaz con el usuario es diseñada en su presentación de transacciones delineadas. Se modifican los modelos de datos y procesos para verificar su factibilidad de los requerimientos junto con las consideraciones pertinentes ante ajustes posteriores.
Diseño físico:
Se diseña la distribución y los tipos de datos de la información. Se detalla el sistema en subsistemas y sus procesos. Para la interfaz con el usuario se establecen las pantallas y reportes junto con detalles derivados de las observaciones del cliente.
Diseño final:
Se completan todos los modelos y representaciones del sistema en detalle suficiente para soportar la construcción final.

- Construcción.

Se implementa en un prototipo desde las ideas iniciales hasta los modelos particulares con todo detalle.
Prueba del concepto:
Se prueba que tan bien el diseño conceptual puede manejar los requerimientos importantes del negocio.
Primera construcción:
Se evalúan los requerimientos y el diseño lógico del sistema propuesto, elaborando el diseño, codificación y prueba de un software basado en requerimientos específicos del negocio, sistema y tecnología.
Segunda construcción:
Se construye y prueba el código para el sistema.
Construcción final:
Se termina la construcción de todas las funciones del sistema y se prueba su adecuado desempeño.

- Evaluación.

Se realiza un análisis del riesgo de llevar a cabo el proyecto y se confronta con el cliente, desde resultados parciales hasta las pruebas finales del sistema completo.
Análisis de riesgo:
Se revisa y evalúa sobretodo las ventajas del desarrollo del sistema planeado en este ciclo. Se obtiene la información para revisarse en las futuras construcciones junto con la recomendación del como proceder.
Primera evaluación:
Se evalúa el progreso del desarrollo del sistema planteado en el aspecto de requerimientos del software, diseño lógico y primera construcción. Con la experiencia obtenida de las evaluaciones de los requerimientos, diseños y software del segundo ciclo se realizan las recomendaciones para el siguiente.
Segunda evaluación:
Se examinan las actividades del tercer ciclo para evaluar el progreso y hacer las recomendaciones pertinentes.
Prueba final:
Usuarios y clientes prueban el sistema para su aceptación total, previa planeación de las pruebas y procedimientos para conducirlas. Se liberan los planes de puesta en marcha del software junto con su documentación.

II.2.8.- Análisis comparativo de los diversos enfoques sobre desarrollo de software

A continuación se presenta un cuadro resumen de los puntos más importantes que toman en cuenta cada uno de los enfoques sobre el desarrollo de software anteriormente vistos.

Enfoques / Puntos importantes que toman en consideración cada uno de ellos, tal y como lo presentan.	CVC	CVE	CP	TCG	PDS	CVS	ME
Ingeniería del sistema	x						
Análisis de requisitos	x				x		
Diseño	x	x			x		
Codificación	x				x		
Pruebas	x			x	x	x	
Mantenimiento	x				x		
La encuesta		x					
Análisis del sistema		x			x		
Implementación		x		x			
Generación de pruebas de aceptación		x					
Garantía de calidad		x					
Descripción de procedimientos		x					
Conversión de datos e instalación		x					
Recolección y refinamiento de los requisitos			x				
Diseño rápido			x				
Construcción del prototipo			x				x
Evaluación del prototipo por el cliente			x				x
Recolección de requisitos				x			
Estrategia de diseño				x			
Planificación del proyecto					x		
Iniciación y conceptualización del software						x	
Definición de requerimientos						x	
Diseño preliminar						x	
Diseño detallado						x	
Programación						x	
Integración						x	
Aceptación y entrega						x	
Operación y soporte						x	
Identificación							x
Diseño conceptual							x
Diseño lógico							x
Diseño físico							x
Diseño final							x

CVC Ciclo de Vida Clásico. TCG Técnicas de Cuarta Generación. ME Modelo en Espiral.
CVE Ciclo de Vida Estructurado. PDS Proceso de Desarrollo del Software.
CP Construcción de Prototipos. CVS Ciclo de Vida del Software.

Cuadro 1. Puntos más importantes que toman en consideración cada uno de los enfoques para el desarrollo de software.

A continuación se presentan algunas ventajas y desventajas de cada uno de los enfoques presentados

Enfoque	Ventajas	Desventajas
Ciclo de Vida Clásico	Es el más antiguo y más ampliamente usado, nos ofrece un enfoque sistemático y secuencial del desarrollo.	No es 100 % aplicable a todas las situaciones. Se presentan algunos problemas cuando se tiene que regresar a las fases anteriores. Se requiere tener al principio todos los requerimientos. El cliente tendrá una versión del software hasta las etapas finales, en donde se pueden descubrir algunos errores no detectados, por lo que puede salir bastante costoso, al no considerar una etapa de aseguramiento de la calidad.
Ciclo de Vida Estructurado	Incluye el análisis y diseño estructurado, con lo cual facilita el proceso de desarrollo, ofreciendo un software seguro y más fácil de mantener. Le da importancia al aseguramiento de la calidad y a la creación de manuales.	No cubre la parte de mantenimiento, ni la puesta en marcha del sistema.
Construcción de Prototipos	Se basa en la idea de que al usuario se le debe presentar, lo antes posible, un prototipo para que él lo evalúe y retroalimente sus comentarios al desarrollador para que sea modificado hasta que el usuario este totalmente satisfecho con el software.	Se requiere que el usuario participe activamente y por largos periodos en las evaluaciones del sistema, el avance depende fuertemente de los comentarios y aportaciones que él haga. El usuario al estarlo evaluando continuamente, por lo regular siempre quiere aumentar algo más, por lo que se debe de definir perfectamente en un principio el alcance del mismo.
Técnicas de Cuarta Generación	El número de pasos a seguir se reduce notablemente con respecto a otros enfoques, además el trabajo a realizar también se reduce y se simplifica de manera considerable. Se cuenta con herramientas de software que automáticamente generan código fuente, reportes, pantallas, etc., partiendo de especificaciones del sistema, tiene la habilidad de especificar software a una máquina a un nivel muy cercano al lenguaje natural, por lo que se vuelve más sencillo y rápido el desarrollo de software en aplicaciones sencillas.	Requiere de herramientas automatizadas, para dar los resultados esperados, por ejemplo de un generador de código, se tiene el problema que el código que se genera no es óptimo y además, no es fácilmente modificable. Se utilizan lenguajes no procedurales por lo que dificulta grandemente su mantenimiento. Da buenos resultados para aplicaciones pequeñas y medianas. Se requiere de herramientas específicas las cuales tienen un costo considerable.
Proceso de Desarrollo del Software	Es un enfoque que presenta tres fases genéricas que todo proceso de desarrollo presenta: definición, desarrollo y mantenimiento.	No toma en cuenta el aseguramiento de la calidad del software, así como ni la puesta en marcha.
Ciclo de Vida del Software	Es uno de los procesos de desarrollo más completo en cuanto a las actividades que se deben desarrollar, además de ser uno de los que más se emplean en la actualidad.	Le hace falta cubrir la parte de aseguramiento de la calidad como tal.
Modelo en Espiral	Presenta un enfoque evolutivo, resultado de la combinación del ciclo de vida clásico y construcción de prototipos, presenta un análisis de riesgo no tomado en cuenta en los demás enfoques. Progresivamente se construyen versiones más completas del software desarrollado. es un enfoque que va de lo general a lo particular.	No cubre la etapa de mantenimiento como tal. Es un enfoque relativamente nuevo, por lo que no se ha usado bastante. Requiere una considerable habilidad para la valoración del riesgo.

Cuadro 2. Algunas ventajas y desventajas que presentan los enfoques para el desarrollo de software.

II.2.9.- Etapas que debe tener un proceso de desarrollo de software.

Ahora bien, analizando y aplicando la técnica TKJ (véase anexo 1) a la información anterior, se obtienen las etapas necesarias o puntos substanciales que debe tener un proceso de desarrollo de software.

Definición de requerimientos.
Análisis.
Diseño.
Implementación.
Pruebas.
Puesta en marcha y mantenimiento.
Aseguramiento de la calidad.

Cuadro 3. Puntos substanciales que debe contener un proceso de desarrollo de software.

Las actividades de la propuesta del proceso de desarrollo de software son:

Definición de requerimientos	Análisis	Diseño	Implementación	Pruebas	Puesta en marcha y mantenimiento
El cliente y el grupo desarrollador deben de definir claramente y sin ambigüedades todos los requerimientos que se quieren del software.	Se debe de entender que es lo que se quiere que haga el sistema y tener claro y preciso cómo se realizan los procedimientos involucrados. Se identifican las restricciones y necesidades de funcionamiento, así como definir las funciones a realizar, obteniéndose las especificaciones del sistema.	Una vez que se tiene claro la forma como se obtiene lo que se quiere, ahora se debe de realizar el diseño conceptual que satisfaga dichas necesidades, después se debe de realizar el diseño detallado para obtener la solución deseada.	Una vez que se realizó el análisis y el diseño respectivo, ahora se debe de desarrollar la codificación respectiva, utilizando para ello un lenguaje de programación adecuado, técnicas de programación y siguiendo los estándares y procedimientos de la empresa.	Una vez que se realizó la programación se deben de llevar a cabo los diferentes tipos de prueba, comprobando que el sistema funciona correctamente y da los resultados esperados, cumpliendo totalmente los requerimientos del cliente.	Una vez que está libre de errores el software, se debe de llevar a cabo la puesta en marcha del mismo, cuidando la instalación adecuada del sistema y la capacitación que se debe de dar a los usuarios para que todo funcione como debe ser, iniciándose la fase de mantenimiento la cual durará durante todo el tiempo que se emplee dicho software.
a s e g u r a m i e n t o d e c a l i d a d					

Cuadro 4. Breve explicación de los puntos substanciales que debe tener un proceso de desarrollo de software.

Como se puede observar la propuesta para un proceso de desarrollo estará integrada por seis actividades sucesivas y una en forma paralela a ellas, la actividad de aseguramiento de calidad.

Al ser una serie de actividades sucesivas, cada fase requiere información específica, la cual será procesada y generará información necesaria para la siguiente fase y así sucesivamente, por lo tanto la actividad de aseguramiento de calidad deberá estar presente entre cada una de las fases, cuidando los detalles antes mencionados.

Ahora bien, esta propuesta no esta limitada a esas actividades solamente, sino al contrario, en cualquier fase se pueden usar pasos o actividades sugeridas por otros enfoques, de tal manera que se complementen y se aproveche mejor lo que hay sobre el desarrollo de software.

El aseguramiento de calidad la mayoría de los autores lo ponen solamente entre las fases de desarrollo, con la intención de revisar que esté completa la fase actual y se pueda iniciar sin ningún problema la fase siguiente, yo coincido con esa idea, pero agrego que el aseguramiento de calidad debe de llevarse a cabo durante todo el proceso de desarrollo de software, desde el inicio hasta el final, tomando en cuenta primeramente el paso de una fase a otra con revisiones técnicas formales, y en segundo lugar, realizar revisiones y auditorías, que les llamaría de rutina para asegurar que se están cumpliendo otros puntos importantes durante todo el proceso de desarrollo, para obtener un mejor aseguramiento, los cuales serán tratados más adelante, por ejemplo, que se esté siguiendo la metodología seleccionada, que se cuenten con los manuales respectivos y que éstos estén completos, que se estén siguiendo los estándares y procedimientos establecidos por la empresa, que se estén efectuando auditorías sobre las revisiones realizadas, etc.

Cabe hacer mención, que el proceso de desarrollo de software puede ser cualquier otro, lo importante es que se tenga un aseguramiento de calidad del software entre y durante todas las fases de dicho procedimiento, véase figura 4.

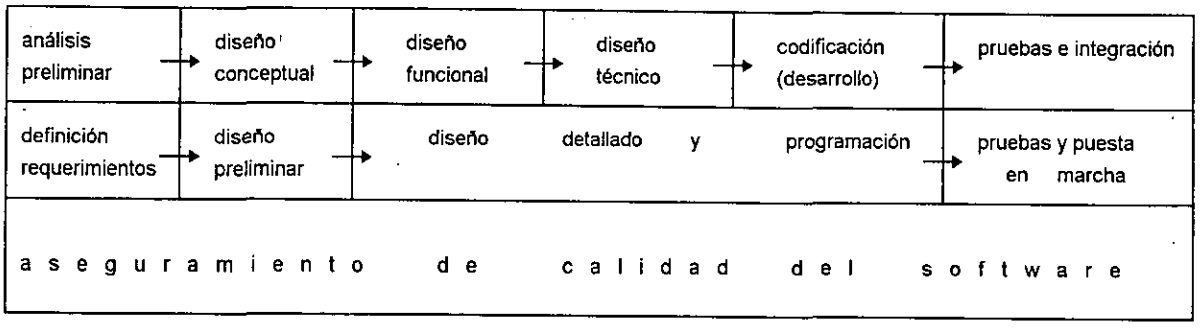

Figura 4. El aseguramiento de calidad se debe de llevar a cabo durante todo el proceso de desarrollo, independientemente del que se emplee.

A continuación se presentan en forma general algunos enfoques sobre el aseguramiento de calidad del software.

II.3.- Diferentes enfoques sobre aseguramiento de calidad del software.

II.3.1.- Enfoque de la IEEE estándar 730-1984 y 983-1986 sobre un plan de aseguramiento de calidad del software[12 y 3]

- Propósito del plan.
- Documentos de referencias.
- Gestión. El plan debe especificar la organización que se tendrá, la repartición de tareas y la asignación de las responsabilidades.
- Documentación. El plan debe especificar los documentos desarrollados, tales como los de requerimientos, los de diseños y los de verificaciones, así como los documentos de gestión, tales como el plan de desarrollo y los planes de procedimientos y estándares.
- Estándares, prácticas y convenciones. El plan debe incluir los estándares para la documentación, las especificaciones para la lógica del programa, para la codificación y para los comentarios.
- Revisiones y auditorías. El plan debe de especificar que clase de revisiones o auditorías deberán ser aplicadas, además debe contener las cuestiones de cómo, cuándo y quién deberá aplicarlas.
- Gestión de la configuración del software. El plan debe especificar cómo se llevará el control de cambios del software que se está desarrollando.
- Reporte de problemas y acciones correctivas. El plan debe de proporcionar ideas sobre el seguimiento que se le debe dar a los problemas y cómo asegurar que las resoluciones son llevadas a cabo.
- Herramientas, técnicas y metodologías. El plan debe de proporcionar ideas sobre las herramientas, técnicas y metodologías que se pueden usar para promover la calidad del software.
- Control del código.
- Control de los medios físicos de almacenamiento del software.
- Control del distribuidor.
- Agrupación, mantenimiento y retención de registros.

II.3.2.- Enfoque de Pressman, sobre el aseguramiento de calidad del software[3], 1995

El aseguramiento de calidad del software, según Pressman, comprende una gran variedad de tareas, asociadas con siete actividades principales: aplicación de métodos técnicos; realización de revisiones técnicas formales; prueba del software; ajuste a los estándares; control de cambios; mediciones; registro y realización de informes.

- El aseguramiento de calidad comienza con un conjunto de herramientas, técnicas y metodologías que ayudan al analista a conseguir una especificación y un diseño de alta calidad. Una vez que se ha creado una especificación (o prototipo) y un diseño, debe ser garantizada su calidad. La actividad central que permite garantizar la calidad, es la revisión técnica formal.

- La revisión técnica formal es una especie de reunión del personal técnico con el único propósito de descubrir problemas de calidad, la cual debe ser llevada a cabo por profesionales de la ingeniería del software.

- La prueba del software combina una estrategia de múltiples pasos con una serie de métodos de diseño de casos de prueba, que ayudan a asegurar una efectiva detección de errores.

- El grado de aplicación de procedimientos y estándares en el proceso de la ingeniería del software varía de empresa a empresa. En muchos casos los estándares vienen dados por los clientes o por mandamientos de regulación, mientras que los procedimientos son puestos por las empresas.

- El proceso de control de cambios contribuye directamente a la calidad del software, al formalizar las peticiones de cambio, evaluar la naturaleza del cambio y controlar el impacto del cambio.

- La medición es una actividad integral para cualquier disciplina. Un objetivo importante de las actividades del aseguramiento de la calidad es seguir la pista a la calidad del software y evaluar el impacto de los cambios de metodología y de procedimiento que intentan mejorar la calidad del software. Para conseguir esto se deben de considerar las métricas del software.

- El registro de información y la generación de informes para el aseguramiento de calidad del software dan procedimientos para la recolección y divulgación de la información del aseguramiento de la calidad. Los resultados de las revisiones, control de cambios, pruebas y otras actividades del aseguramiento de calidad deben convertirse en una parte del registro histórico de un proyecto y deben ser divulgadas a la plantilla de desarrollo para que tengan conocimiento de ellos.

II.3.3.- Enfoque del plan de aseguramiento de calidad del software según Acis[13], 1994

- Introducción (Propósito y alcance).
- Aplicabilidad.
- Documentos aplicables.
- Gestión y planeación del programa.
  (Plan SQA Acis; Organización; Tareas; Formación de software; Personal del SQA y Certificación de desarrolladores de software)
- Programa de monitoreo de recursos.
- Programa de auditorías SQA.
- Registros SQA.
- Reportes de estado.
- Documentación de software.
- Definición de requerimientos.
- Procesos de desarrollo del software.
- Revisiones del proyecto.
- Herramientas y técnicas.
- Gestión de la configuración del software.
- Liberación de procesos.
- Control de cambios.
- Reporte de problemas.
- Pruebas del software.

II.3.4.- Enfoque del plan de aseguramiento de calidad del software según Butler [14], 1995

- Historia de cambios. En este punto se muestra la historia de cambios de versiones que se han tenido, desde la versión original hasta la actual.

- Introducción. El propósito general del SQA es asegurar que el software que empieza a ser desarrollado sea de la más alta calidad posible, garantizando que cumple con los requerimientos explícitos e implícitos puestos por los usuarios y que los desarrolladores cumplen con las especificaciones de estándares y criterios de desarrollo.

- Gestión. Se especifica la organización que tendrá el grupo SQA, como quién será el líder y quienes los miembros del grupo, especificando las tareas y responsabilidades.

- Documentación. El propósito del mantenimiento de la documentación es para mejorar la calidad del software, guardando las secuencias de cambios y métricas, posteriormente éstos pueden ser usados para crear líneas bases de investigación en futuros proyectos.

- Estándares y convenciones. Los estándares y convenciones son usados para asegurar que las tareas tengan calidad y sus correcciones sean simples y fáciles de realizar. Deben de existir estándares para requerimientos, para diseño y para la programación.

- Revisiones y auditorías. Las revisiones y auditorías tienen como objetivo encontrar problemas en el desarrollo del software, antes de que éstos se vuelvan mayores.

- Reportes de problemas y acciones de corrección. Los reportes de problemas y acciones correctivas deben ser manejadas a través de la colección y distribución de formas de reportes y revisiones, así como del seguimiento de los mismos.

- Herramientas, técnicas y metodologías. Durante el proceso de desarrollo se necesitan de diferentes herramientas, técnicas y metodologías, las cuales nos dicen como se debe hacer y los pasos que se tienen que seguir.

- Colección de registros. La configuración del software será la responsable para guardar y mantener todos los documentos producidos durante el proceso de desarrollo de software, incluyendo el control de cambios.

II.3.5.- Enfoque de Dobbins para el aseguramiento de calidad del software y su evaluación[15], 1990

- Organización del aseguramiento de calidad del software (SQA).
  - Independencia.
  - Estructura.
  - Procedimientos.
  - Manual de procedimientos.

- Iniciación de las actividades del SQA.

- Planeación del SQA.

- Proceso de inspección del software.
  - Fases de la inspección.
    - Planeación.
    - Presentación.
    - Preparación.
    - Reunión de inspección.
    - Retrabajo.
    - Seguimiento.
  - Inspecciones de diseño y de código.

- Inspección de documentos.

- Control de la configuración del software.
  - Almacenamiento de la documentación de versiones.
  - Control y distribución de los cambios.
  - Control y registro de las identificaciones de las versiones.
  - Solicitudes de cambios al software.

- Actividades de pruebas SQA.
  - Reporte de errores.
  - Análisis de las tendencias de los reportes de errores.
  - Monitoreo de pruebas.
  - Validación de pruebas.

- Procuramiento del aseguramiento de la calidad de software.
  - Revisión del contrato.
  - Reconocimiento del vendedor.
  - Declaraciones sobre subcontrataciones de trabajo.

- Auditorías de calidad.

- Estándares y especificaciones.

II.3.6.- Enfoque de Stamm sobre un plan de aseguramiento de calidad del software[15], 1981

- Organización. Definición de reglas y responsabilidades de cada grupo en la organización del proyecto.

- Definición de requerimientos. Define la metodología a seguir para asegurar que todos los requerimientos de las especificaciones del nivel superior sean satisfechas por las especificaciones de bajo nivel y se establezca la verificación de todos los requerimientos a través de la definición de planes de pruebas.

- Documentación. La documentación que se generará, deberá estar bajo una política formal de aseguramiento, controlada a través de la comunicación de todos los elementos de la organización; estándares para la generación de la documentación; y las mediciones que se aplicarán para asegurar la conformidad con los estándares.

- Metodología de ingeniería de software. La aplicación de las metodologías de ingeniería de software relacionadas con la calidad del proyecto, deberán estar definidas y proveer monitoreos de chequeos durante su desarrollo.

- Entrenamiento. Los requerimientos para certificar los conocimientos del personal de desarrollo del software, deberán estar definidos y ser aplicados al proyecto

- Revisiones formales. Las revisiones y su metodología a seguir deberán estar bien definidas para asegurar una buena revisión.

- Programa de pruebas. El plan de SQA deberá definir específicamente las medidas que se utilizarán en las revisiones técnicas del plan de pruebas de procedimientos y deberán estar de acuerdo con los estándares establecidos; las reglas que deberán seguir los miembros del grupo de SQA en la conducción de las pruebas y certificaciones de los resultados; los formatos de errores que se utilizarán en las pruebas del sistema; los requerimientos necesarios para probar los caminos lógicos de los procesos; las medidas que se deberán tomar para asegurar el control del hardware que se utilice para probar el software.

- Gestión de configuración. Las consideraciones de aseguramiento de la calidad dan lugar a los requerimientos sobre los sistemas de gestión de configuración los cuales deben incluir una librería de software con su respectivo control en los procedimientos para asegurar la identificación de ambigüedades en los productos y prevenir modificaciones no autorizadas.

II.3.7.- Enfoque de la ISO 9000-3 sobre estándares para el aseguramiento de calidad y la gestión de la misma en el desarrollo, suministro y mantenimiento del software[17], 1991

- Sistema de calidad. Estructura.

  Responsabilidad directiva.
  Sistema de calidad.
  Auditorías internas.
  Acciones correctivas.

- Sistema de calidad. Actividades del ciclo de vida.

  Revisión del contrato.
  Especificación de requerimientos del cliente.
  Planeación del desarrollo.
  Planeación de la calidad.
  Diseño e implementación.
  Pruebas y validaciones
  Aceptación.
  Replicación, liberación e instalación.
  Mantenimiento.

- Sistema de calidad. Actividades de soporte.

  Gestión de la configuración.
  Control de documentos.
  Registros de calidad.
  Medición.
  Reglas, prácticas y convenciones.
  Herramientas y técnicas.
  Adquisición.
  Inclusión de productos de software.
  Entrenamiento.

II.3.8.- Análisis comparativo de los diversos enfoques sobre aseguramiento de calidad del software.

A continuación se presenta un cuadro resumen de los puntos más importantes que toman en cuenta cada uno de los enfoques sobre aseguramiento de calidad del software anteriormente vistos

Puntos importantes que toman en consideración cada uno de ellos, tal y como lo presentan. / Enfoques	IEEE 983	Pressman	Acis	Butler	Dobbins	Stamm	ISO 90000-3
Propósito del plan	x						
Documentos de referencias	x						
Gestión	x			x			
Documentación	x		x	x		x	
Estándares, prácticas y convenciones	x			x			x
Revisiones y auditorías	x			x			
Gestión de la configuración del software	x		x			x	x
Reportes de problemas y acciones correctivas	x			x			
Herramientas, técnicas y metodologías	x	x		x			
Control del código	x						
Control de los medios físicos de almacenamiento	x						
Control del distribuidor	x						
Agrupación, mantenimiento y retención de registros	x						
Revisión técnica formal		x					
Pruebas del software		x	x				
Aplicación de procedimientos y estándares		x					
Control de cambios		x	x				
Mediciones		x					x
Registro de información y generación de informes		x					
Introducción			x	x			
Aplicabilidad			x				
Documentos aplicables			x				
Gestión y planeación			x				
Programa de monitoreo de recursos			x				
Programa de auditorías SQA			x				
Registros SQA			x				
Reportes de estado			x				
Definición de requerimientos			x			x	
Procesos de desarrollo del software			x				
Revisiones del proyecto			x				
Herramientas y técnicas			x				

(continuación) Enfoques / Puntos importantes que toman en consideración cada uno de ellos, tal y como lo presentan.	IEEE 983	Pressman	Acis	Butler	Dobbins	Stamm	ISO 9000-3
Liberación de procesos			x				
Reporte de problemas			x				
Historia de cambios del plan				x			
Estándares y convenciones				x			
Colección de registros				x			
Organización del SQA					x	x	
Iniciación de actividades del SQA					x		
Planeación del SQA					x		
Proceso de inspección del software					x		
Inspección de documentos					x		
Control de la configuración del software					x		
Actividades de pruebas SQA					x		
Procuramiento del SQA					x		
Auditorías de calidad					x		
Estándares y especificaciones					x		
Metodología de ingeniería de software						x	
Entrenamiento						x	x
Revisiones formales						x	
Programa de pruebas						x	
Responsabilidad directiva							x
Sistema de calidad							x
Auditorías internas							x
Acciones correctivas							x
Revisión del contrato							x
Especificación de requerimientos del cliente							x
Planeación del desarrollo							x
Planeación de la calidad							x
Diseño e implementación							x
Pruebas y validaciones							x
Aceptación							x
Replicación, liberación e instalación							x
Mantenimiento							x
Control de documentos							x
Registros de calidad							x
Herramientas y técnicas							x
Adquisición							x
Inclusión de productos de software							x

Cuadro 5. Puntos más importantes que toman en consideración cada uno de los enfoques sobre el aseguramiento de calidad del software.

A continuación se presentan algunas ventajas y desventajas de cada uno de los enfoques presentados:

Enfoque	Ventajas	Desventajas
IEEE 983	Es un formato estándar normalizado	Los puntos de documentación, gestión de la configuración del software, control del código, control de los medios físicos del almacenamiento, podían ser integrados en uno solo que sería, por ejemplo, configuración del software; Además le hace falta algunos otros, como por ejemplo, pruebas. Además menciona de manera muy general los puntos que conviene abarcar, pero no dice como se pueden implementar y llevarlos a la práctica en alguna organización.
Pressman	Presenta información formal y exhaustiva sobre la Ingeniería del Software (paradigmas).	La información es presentada desde el punto de vista de la Ingeniería del software y no como parte de un aseguramiento de la calidad.
Acis	Hace énfasis en las técnicas, procedimientos y metodologías que se deberán usar para asegurar la entrega a tiempo del software y que cumplan con los requerimientos especificados.	Los puntos de documentación y control de cambios podían estar dentro de la gestión de configuración del software. Además menciona los puntos que se deben tomar en consideración en forma general, pero no dice cómo implementarlos.
Butler	Toma en consideración el punto de la versión actual y de la historia de cambios de las anteriores versiones que se tienen del plan, ninguno de los otros enfoques lo consideran.	No toma en consideración varios puntos importantes como por ejemplo: la gestión de configuración del software y la planeación. Además son tratados de manera muy general
Dobbins	Hace mucho énfasis en que la calidad del software se obtiene planeando y construyendo con calidad y además planeando y realizando evaluaciones de la misma.	No le da importancia a los métodos, herramientas y procedimientos, los cuales son vitales para el desarrollo del software, además los puntos son tratados de manera muy general.
Stamm	Hace mucho énfasis en que el aseguramiento de la calidad del software es parte del trabajo de todos los que participan en la organización y además de que se debe aplicar el aseguramiento de la calidad en cada una de las etapas que integran el ciclo de vida de desarrollo del mismo. Así mismo le da mucha importancia al entrenamiento y capacitación del personal desarrollador para la certificación del mismo, el cual no es considerado en los otros.	No le da importancia a las auditorías. Los puntos son tratados de manera muy general.
ISO 9000-3	Proporciona una serie de lineamientos o estándares muy importantes para el aseguramiento de la calidad y gestión de la misma en el desarrollo, suministro y mantenimiento del software.	No es en sí un plan, solamente estándares.

Cuadro 6. Algunas ventajas y desventajas que presentan los enfoques para el aseguramiento de calidad del software.

II.3.9.- Puntos substantivos que se deben tomar en consideración para un aseguramiento de calidad del software.

Ahora bien, analizando y aplicando la Técnica TKJ a la información del cuadro 5, se obtiene el cuadro 7, el cual muestra los puntos substanciales que debe contener el aseguramiento de calidad del software.

Gestión
Metodologías, técnicas y herramientas
Estándares y procedimientos
Revisiones y auditorías
Gestión de la configuración del software

Cuadro 7. Puntos substanciales que debe contener el aseguramiento de calidad del software.

A continuación se da una breve explicación de cada uno de los puntos substanciales del aseguramiento de calidad del software.

Gestión.	Métodos, técnicas y herramientas.	Estándares y procedimientos.	Revisiones y auditorías.	Configuración del software.
Este punto abarcará todo lo relacionado con la planeación, organización, preparación, ejecución, evaluación y control de cada uno de los demás puntos substanciales del plan propuesto	Este punto abarcará algunos de los paradigmas de la Ingeniería de Software, es decir los métodos, las técnicas y las herramientas que existen para desarrollar sistemas de información.	Este punto abarcará los estándares y manuales de procedimientos que se deberán tener y seguir internamente en la organización para desarrollar el software	Este punto abarcará para cada etapa del desarrollo del software, las revisiones formales y auditorías que se deberán aplicar, conteniendo la planeación de cuándo se deben de llevar a cabo, quienes deberán participar, que se revisará, formatos de revisiones y auditorías, formatos de reporte, firmas de legalización de documentación, etc.	La configuración del software es el arte de identificar, organizar, mantener y controlar las modificaciones que sufre el software que construye un equipo de desarrollo y se aplica a lo largo de todo el proceso de ingeniería del software, por lo tanto, la gestión de la configuración del software abarcará varios puntos como: la identificación de las líneas bases; el control de la documentación (programas fuentes, programas de código, manuales técnicos, de usuarios, información de procedimientos, formatos de reportes, errores, folders de desarrollo, etc.); el control de cambios; la actualización y control de las versiones; la publicación y repartición de las nuevas versiones, auditorías de la configuración, etc.

Cuadro 8. Breve explicación de los puntos substanciales que debe tener el aseguramiento de calidad del software.

Además, el aseguramiento de calidad del software depende en gran medida de lo siguiente:

- La correcta definición de los requerimientos del software.
- Estándares especificados que definen un conjunto de criterios o procedimientos de desarrollo que guíen la forma correcta para la elaboración del software.
- La correcta interpretación de los requerimientos implícitos.
- Del trabajo de todos los que participan en la elaboración del software, incluyendo al usuario.
- Del cuidado que se le dé en cada una de las etapas del proceso de desarrollo del software.

Por lo tanto, el aseguramiento de calidad del software se logrará tomando en consideración lo anterior y realizando una adecuada gestión sobre los puntos substanciales de la propuesta con la intención de verificar y asegurar de que se están aplicando correctamente las metodologías, técnicas, herramientas, estándares, procedimientos, revisiones, auditorías y la configuración del software que se desarrolla, durante cada una de las fases del proceso de desarrollo del software empleado.

III.- Propuesta para el aseguramiento de calidad en el desarrollo de software

III. 1.- Presentación

En la actualidad la creciente automatización informática de la mayoría de los procesos que se llevan a cabo en las empresas públicas y privadas, dan como resultado que se requiera cada vez más el desarrollar software a la medida de las necesidades en cada una de ellas, aunado a las crisis económicas mundiales y a los tratados de libre comercio que se están firmando, es necesario contar con mecanismos para el aseguramiento de calidad del software, por medio de los cuales se tengan una guía de pasos y actividades para poder asegurar de una mejor manera la calidad del software que se desarrolle. Este capítulo muestra una propuesta para el aseguramiento de calidad en el desarrollo de software.

Cabe hacer la aclaración que se trata de una propuesta cuyo objetivo es el de ayudar a tratar de asegurar lo más que se pueda la calidad del software que se desarrolle en una empresa, a través de sugerencias de actividades que se deberán ir realizando durante el desarrollo del mismo, pero que dependerán ampliamente de la organización que tenga la empresa, así como de los procedimientos, técnicas, herramientas, métodos, etc., que se utilicen en la misma, además de la experiencia que tenga la empresa y el grupo de desarrollo respectivo.

A lo largo de la propuesta se mostrarán ciertas organizaciones o procedimientos, los cuales no son únicos, y se utilizan solamente como ejemplos. Cada empresa de acuerdo a sus políticas, normas y procedimientos que empleen, deberán ir formando sus propios procedimientos para el aseguramiento, con la idea de irlos

perfeccionando cada vez mejor con el paso del tiempo y de la experiencia que se vaya adquiriendo, por tal motivo se presentan diferentes ejemplos en anexos, con la idea de que el lector pueda complementar de una manera más detallada, si así lo desea.

El enfoque práctico de la propuesta es que el aseguramiento de calidad del software será trabajo de todas aquellas personas que intervienen en el desarrollo del mismo. Cada elemento del proceso de desarrollo de software envuelve aspectos de aseguramiento de la calidad, con lo cual cada miembro del grupo desarrollador debe identificarlos para poder obtener un producto con una alta calidad.

En el capítulo anterior, se obtuvieron los puntos substanciales que debe contener un proceso de desarrollo, así como los del aseguramiento de calidad del software. (véanse cuadros 3 y 7 respectivamente).

En la figura 5, se presenta el marco de referencia de la propuesta para el aseguramiento de calidad en el desarrollo de software:

La idea de la propuesta presentada es que el aseguramiento de calidad en el desarrollo de software se basa en tres aspectos importantes:

1. El proceso de desarrollo que se utilice, el cual debe de aplicarse rigurosamente de acuerdo a la metodología de desarrollo seleccionada, siguiendo y realizando lo más que se puedan las actividades especificadas en ella.

2. La gestión que se tenga de los otros puntos substanciales del aseguramiento de calidad sobre el proceso de desarrollo que se utilice, es decir, la planeación, la organización, la preparación, la ejecución, la evaluación, el seguimiento y el control que se tenga de: los métodos, herramientas y técnicas; los estándares y procedimientos; las revisiones y auditorías; y la configuración del software, sobre cada una de las fases del proceso de desarrollo, con lo cual se cuidará el aseguramiento de calidad en el desarrollo de software.

3. El aseguramiento de la calidad del software será trabajo de todas aquellas personas (actores del sistema) que estén involucradas en el desarrollo del mismo. Cada elemento tanto del proceso de desarrollo como de la propuesta envuelven aspectos de aseguramiento de la calidad. Por lo tanto, estarán involucrados los directivos, jefes, desarrolladores, clientes, usuarios y demás empleados, que de alguna u otra manera participan en el desarrollo, por consiguiente, estarán igualmente involucrados los diferentes departamentos de la empresa desarrolladora, así como los clientes y usuarios respectivos.

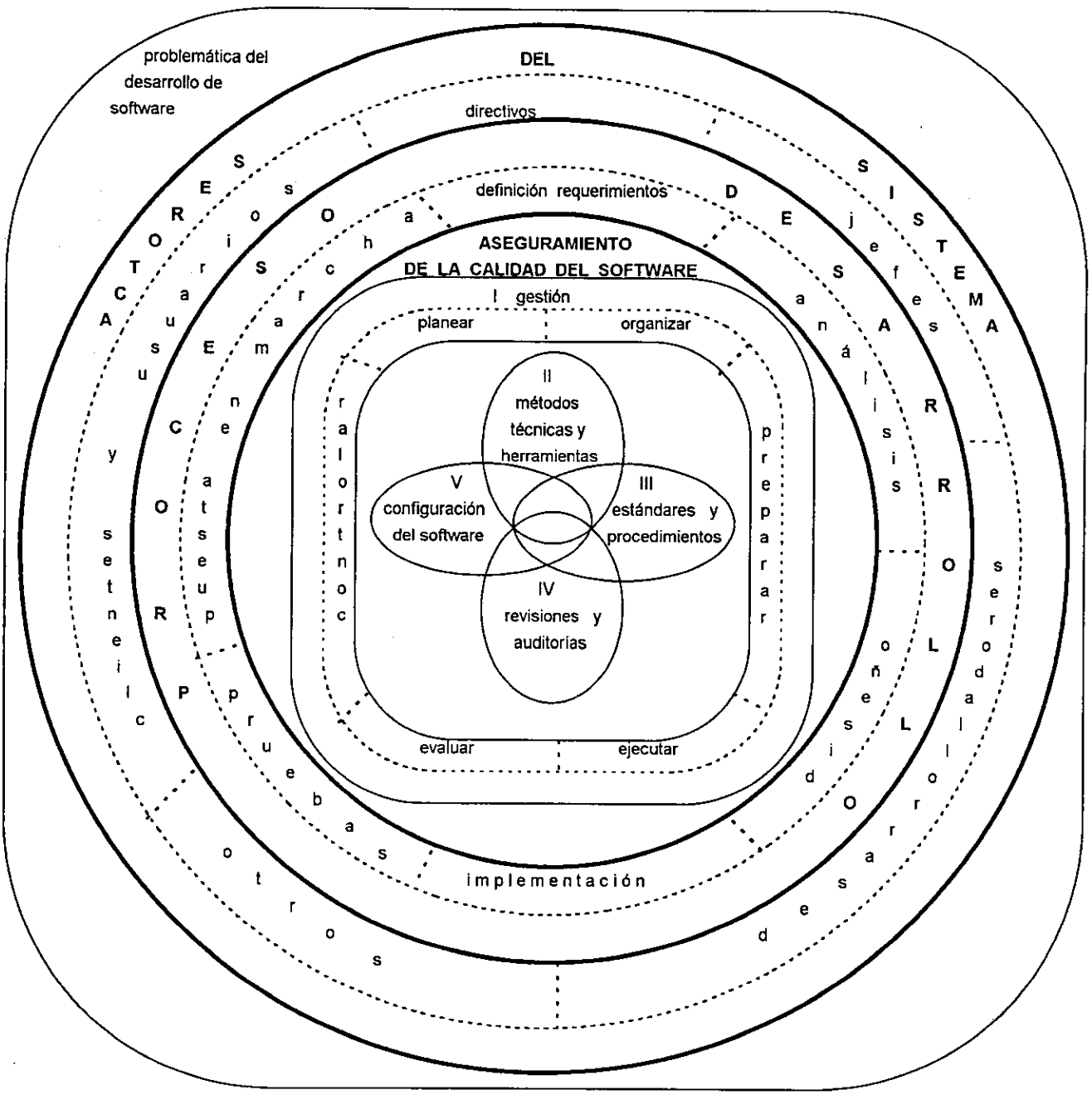

Figura 5. Marco de referencia de la propuesta para el aseguramiento de calidad en el desarrollo de software.

El objetivo primordial de la propuesta es en primer lugar, garantizar que el software cumpla con todos los requerimientos del usuario; y en segundo lugar, el contar con procedimientos escritos, por medio de los cuales se asegure que el

desarrollo de un software marcha bien y se puedan detectar posibles avisos, desviaciones o problemas que se pudieran presentar.

Obsérvese en la figura anterior, en la parte relacionada con el aseguramiento de la calidad del software, ésta toma en consideración cinco puntos elementales indicados como (I,II,III,IV y V), de los cuales la gestión (I) se debe de realizar sobre los otros cuatro puntos restantes. Por otro lado, estos puntos pueden abarcar otros más, que la mayoría de los autores los ponen por separado, por ejemplo: el caso de la documentación y de pruebas, son esenciales para lograr el aseguramiento de la calidad, pero son finalmente subpartes de otros, la documentación está incluida en el punto de la configuración del software, mientras que las pruebas están incluidas en los paradigmas de la ingeniería de software, en específico por un lado dentro de las técnicas para probar el software desarrollado y por otro en la fase de pruebas del proceso de desarrollo.

A continuación se explica cada uno de estos puntos.

III.2.- La gestión

La gestión se debe de llevar a cabo sobre cada uno de los otros puntos que integran la propuesta y además, sobre cada una de las fases del proceso de desarrollo seleccionado, sin olvidar que para cada empresa, ésta deberá tener definidas sus propias políticas, estándares y procedimientos para el desarrollo de software, en donde deberán estar especificados la repartición de tareas y la asignación de las responsabilidades de cada grupo, entre otras cosas.

La gestión o conducción del proyecto de software representa el primer nivel del proceso de ingeniería de software y de la propuesta, debido a que cubre el proceso de desarrollo desde el principio al fin, es decir, empieza antes de que comience el trabajo técnico, continúa a medida que el software evoluciona desde el concepto hasta la realidad y culmina sólo en el momento en que se abandona el software.

En el cuadro 9, se incluyen las actividades que contempla el proceso de gestión:

En relación a la evaluación, la medición permite a los gestores o desarrolladores entender mejor el proceso de ingeniería del software y del software que se produce. Las métricas para la productividad y la calidad se pueden definir mediante medidas directas e indirectas.

En la industria se utilizan tanto las métricas orientadas al tamaño como las orientadas a la función, véase cuadro 10.

Planear: Trazar el plan de acción a seguir, identificar las tareas a realizar, los recursos necesarios que se requerirán, agendas de desarrollo, formación y capacitación de personal, etc., como por ejemplo, qué pasos se deberán seguir, cuándo se debe de llevar a cabo alguna reunión, una inspección, a quién convocar, etc.
Organizar: Es una actividad que abarca todo lo relacionado para poder llevar a cabo lo que se va a hacer, definir quiénes estarán, establecer tareas, responsabilidades, lugares de reunión, etc.;
Preparar: Disponer lo necesario para llevarlo a cabo, por ejemplo: diseñar las pruebas que se realizarán a cierto producto, preparar el orden del día que se verá en una reunión, que cosa se revisará, que documentos o pruebas se deberán tener, el procedimiento que se deberá seguir, etc.
Ejecutar: Llevar a cabo la actividad a realizar, tomando en cuenta aspectos importantes de lo que se debe de ir llevando a cabo, cómo se debe de llevar a cabo, etc., por ejemplo, el llevar a cabo una prueba, seguir el procedimiento para realizarla, etc.
Evaluar: Valorar, cuantificar, medir, interpretar la diferencia entre lo medido y el estándar correspondiente, en algunos casos sería verificar o asegurar que el producto está correcto y consistente con respecto a los requerimientos y a los estándares correspondientes. Es necesario que se lleve un registro de las mediciones realizadas, también que se cuente con la información necesaria para poder realizar la actividad de evaluación como pueden ser: los requerimientos, los estándares, los formatos necesarios correspondientes, así como los procedimientos de cómo hacerlo. Se deberá también tomar nota de los resultados que se obtengan.
Controlar: Esta actividad tiene como propósito el mantener lo que esta en papel (estándares) en el estado deseado, vigilando cómo se están efectuando las actividades planeadas, tomar las medidas necesarias en caso de desviación y dar seguimiento adecuado para que se concluyan adecuadamente dichas actividades.

Cuadro 9. Actividades que contempla el proceso de gestión.

Métricas orientadas al tamaño.	Métricas orientadas a la función.
Las métricas del software orientadas al tamaño son medidas directas del software y del proceso por el cual se desarrolla (líneas de código), son bastantes polémicas y no están aceptadas universalmente como mejor modo de medir el proceso de desarrollo de software.	Las métricas orientadas a la función se basan en medidas cuantificables del dominio de información del software y valoraciones subjetivas de la complejidad del mismo (número de entradas de usuario, número de salidas de usuario, número de peticiones al usuario, número de archivos y número de interfaces externas).

Cuadro 10. Métricas que se utilizan en la industria del software.

Las métricas para la calidad del software son:

Durante el proceso de desarrollo:	Cuando el software ha sido distribuido al cliente o usuario:
La complejidad del programa; la modularidad efectiva y el tamaño del programa.	El número de defectos no descubiertos en las pruebas y la facilidad de mantenimiento del sistema.

Cuadro 11. Algunos ejemplos de métricas para la calidad del software

Las métricas pueden estar en forma de listas de comprobaciones usadas para obtener el grado de los atributos específicos del software.

Los factores que afectan a la calidad del software se pueden clasificar en dos grandes grupos:

- *Factores que pueden ser medidos directamente;*
- Factores que sólo pueden ser medidos indirectamente.

Se debe comparar el software con alguna referencia (estándar) y llegar a una indicación de la calidad.

Cavano y McCall[18], 1978, propusieron una clasificación de los factores que afectan a la calidad del software. Estos factores de calidad del software se centran en tres aspectos importantes de un producto de software: sus características operativas, su capacidad de soportar cambios y su adaptabilidad a nuevos entornos, véase anexo 2.

Por otro lado, de la investigación y análisis realizado se concluye que para el desarrollo de software intervienen principalmente tres cosas:

La primera, sería el proceso de desarrollo de software independientemente del paradigma de ingeniería de software que se empleara.

La segunda, sería la organización interna de la empresa que desarrollará el software, la cual estaría compuesta, por ejemplo, de:

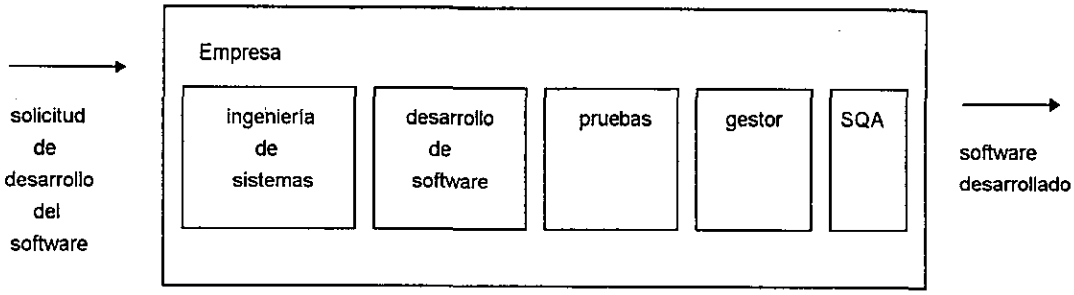

Figura 6. Ejemplo de grupos que intervienen en el desarrollo de software.

En relación a la empresa que desarrollará el software, podemos observar de la anterior figura que tendrá una organización interna, la cual puede estar compuesta de departamentos o grupos, como por ejemplo: el de ingeniería de sistemas, el desarrollador del software, el de pruebas e integración, el gestor del

proyecto y el grupo de aseguramiento de la calidad, entre otros. Cada uno de ellos deben tener sus normativas, así como las actividades que deberán realizar y su responsabilidades, por lo tanto, la empresa requerirá tener los procedimientos y estándares que haya establecido o fijado, así mismo los métodos, técnicas y herramientas que se emplearán en el desarrollo de software y a su vez deberá tener un control y seguimiento de toda la configuración del software.

Ahora bien, por un lado se debe de seguir un proceso de desarrollo de software y por otro los recursos humanos o grupos de trabajo quienes lo van a llevar a cabo, por lo tanto, el aseguramiento de calidad del software dependerá tanto de las personas que intervienen como de que se vayan cumpliendo cada una de las actividades de las fases del proceso de desarrollo.

La tercera, sería para lograr lo anterior, se deberá realizar una gestión o conducción de los demás puntos substanciales del modelo propuesto, es decir, se deberá revisar o inspeccionar que se estén: siguiendo correctamente las metodologías y los procedimientos establecidos; empleando y aplicando correctamente las técnicas y los estándares correspondientes; llevando a cabo las revisiones o auditorías de acuerdo a los procedimientos estipulados; realizando el control y seguimiento de la configuración del software, etc. Con la idea de detectar posibles desviaciones o problemas en el desarrollo de software y darles una solución adecuada.

III.3.- Métodos, técnicas y herramientas

La ingeniería del software surge de la ingeniería de sistemas y de hardware, abarcando un conjunto de tres elementos claves: métodos, técnicas y herramientas, que facilitan al conductor o gestor controlar el proceso de desarrollo de software y suministrar a los que practiquen dicha ingeniería, las bases para construir software de alta calidad de una forma productiva.

- Los métodos nos indican cómo construir técnicamente el software, abarcando tareas como: planificación y estimación de proyectos; análisis de los requisitos del sistema y del software; diseño de estructuras de datos; arquitectura de programas y procedimientos algorítmicos; codificación; pruebas; y mantenimientos.

- Las herramientas suministran un soporte automático o semiautomático para los métodos, por ejemplo, diagramas de flujos de datos, diagramas de transición de estados, cartas estructuradas, diccionarios de datos, estructuras lógicas de control, diagramas de flujo estructurados, etc.

- Las técnicas o procedimientos son el pegamento que junta los métodos y las herramientas y facilita un desarrollo racional y oportuno del software. Los procedimientos definen la secuencia en la que se aplican los métodos, las salidas que se requieren, los controles que ayudan a asegurar la calidad, coordinar los cambios y las directrices que ayuden a los gestores del software a evaluar el progreso.

La ingeniería del software está compuesta por una serie de pasos que abarcan los métodos, herramientas y procedimientos antes mencionados, a los cuales se les denomina paradigmas de la ingeniería del software.

La elección de un paradigma para la ingeniería del software se lleva a cabo de acuerdo con la naturaleza del proyecto, la aplicación, los métodos, las herramientas a usar, los controles y las salidas requeridas.

El aseguramiento de calidad del software, debe contemplar que la empresa tenga por escrito las metodologías, técnicas y herramientas que se puedan emplear en el desarrollo de software. Así mismo deberá checar que el personal este capacitado y que domine los paradigmas, además de verificar que dicha información esté a disposición de los desarrolladores del grupo y que esté completa y actualizada.

Existen varios paradigmas importantes entre los que se pueden nombrar: el ciclo de vida clásico; el ciclo de vida estructurado; la construcción de prototipos; el modelo en espiral, etc.

Ahora bien, los paradigmas pueden y deben combinarse, de tal manera de aprovechar las ventajas que presentan cada uno de ellos.

Pressman y Dobbins mencionan que la calidad del software es obtenida mediante la calidad en la planeación y en la construcción del software y por las evaluaciones de la planeación y rendimientos. Por lo tanto existen dos áreas importantes que están relacionadas, la ingeniería del software y el proceso de desarrollo de software.

El proceso usado para el desarrollo de software, debe ser entendible y manejable. Los productos deben ser evaluados para determinar una calidad inicial e identificar que aspectos del proceso deben ser perfeccionados para mejorar la calidad del producto del software.

El proceso de desarrollo de software, según Pressman, contiene tres fases genéricas, independientes del paradigma de ingeniería elegido, las cuales son: definición, desarrollo y mantenimiento, estas etapas aparecen en todos los desarrollos de software, independientemente del área de aplicación, de la complejidad o tamaño del proyecto.

Ahora bien una actividad que aparece nombrada en casi todos los paradigmas de la ingeniería del software es la prueba del software, la cual es considerada como un elemento crítico para el aseguramiento de calidad del software por tal motivo se presentará en una forma más detallada.

La prueba del software.

La prueba del software es una actividad considerada muy importante y representa una revisión final de las especificaciones del diseño y de la codificación de un producto de software. Debido a que el hombre no es perfecto en lo que hace, existe falta de comunicación, no se siguen técnicas de análisis, diseño, codificación, etc., por lo tanto, hay grandes posibilidades de que un producto tenga errores.

Esta actividad tiene como fin, encontrar errores que no han sido identificados aún, muchos autores mencionan que una prueba del software tiene éxito si se encontraron errores, ya que es el propósito de la misma. Por consiguiente los objetivos de la prueba del software, según Pressman, son:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Por lo tanto, el objetivo es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo. La prueba no puede asegurar la ausencia de defectos; solamente puede demostrar que existen defectos en el software.

Existen una gran variedad de métodos de diseño de casos prueba para el software, proporcionando un mecanismo de ayuda para asegurar la completitud de las pruebas y conseguir la mayor probabilidad de descubrimiento de errores.

Cualquier producto de ingeniería, puede ser probado de una de dos formas:

- Conociendo la función específica para la que fue diseñado el producto, llamadas pruebas de la caja negra.

- Conociendo el funcionamiento del producto, llamadas pruebas de la caja blanca.

Las pruebas de la caja negra son diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa. Las técnicas de prueba de la caja negra se centran en el ámbito de la información de un programa de forma que se proporcione una cobertura completa de prueba.

Las pruebas de la caja negra intentan encontrar errores de: funciones incorrectas o ausentes; errores de interfaz; errores en estructuras de datos o en accesos a bases de datos; errores de rendimiento; y errores de inicialización y de terminación.

Las pruebas de la caja blanca se centran en la estructura de control del programa. Se derivan casos de prueba que aseguren que durante la prueba se han ejecutado por lo menos una vez todas las sentencias del programa, o caminos independientes de cada módulo y que se ejercitaron todas las condiciones lógicas de verdadero y falso.

Una estrategia de prueba del software proporciona una guía para el desarrollador del software, el control de calidad para la organización y para el cliente un plan que describe los pasos que se llevarán a cabo como parte de la prueba (cuándo se deben planificar y realizar, cuánto esfuerzo, tiempo y recursos se van a requerir, etc.). Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de pruebas y la agrupación y evaluación de los datos resultantes.

Una estrategia de prueba de software debe ser suficientemente flexible para promover la creatividad y la adaptabilidad necesarias para adecuar la prueba a todos los grandes sistemas basados en software. Al mismo tiempo, la estrategia debe ser suficientemente rígida para promover un seguimiento razonable de la planificación y la gestión, a medida que progresa el proyecto.

El proceso de ingeniería del software se puede ver como una espiral, al movernos de afuera hacia adentro, encontramos inicialmente a la ingeniería del sistema, la cual define el papel del software y conduce al análisis de los requisitos del software, donde se establece el campo de información, la función, el comportamiento, el rendimiento, las restricciones y los criterios de validación del software. Al movernos más hacia el interior de la espiral, llegamos al diseño y por último, a la codificación.

Para desarrollar software, damos vueltas en espiral a través de una serie de flujos o líneas que disminuyen el nivel de abstracción en cada vuelta, por lo tanto, la

parte más importante del desarrollo del software es el análisis, ya que sobre de él se hará el diseño y luego la codificación, véase figura 7. Si el análisis está incorrecto, se acarrea ese error a las fases posteriores, por lo tanto, se necesita asegurar que el análisis este bien hecho y completo, para evitarnos errores futuros y así sucesivamente, el diseño deberá cumplir con todo lo del análisis y a su vez la codificación con el diseño, etc.

Una estrategia de la prueba del software se puede tener, si nos movemos hacia afuera de la espiral, véase figura 7 y cuadro 12:

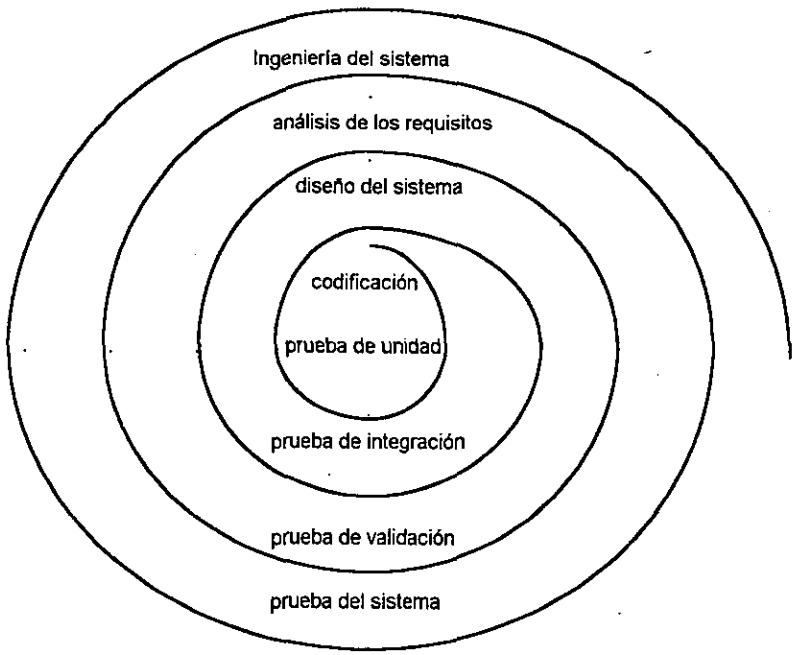

Figura 7. Ejemplo de una estrategia de prueba, según Pressman.

La prueba de unidad.	La prueba de integración.	La prueba de validación.	La prueba del sistema.
Se centra en cada unidad del software, tal como está implementada en código fuente.	El foco de atención es el diseño y la construcción de la arquitectura del software.	Es donde se validan los requisitos establecidos como parte del análisis de requisitos del software, comparándolos con el sistema que ha sido construido.	Es donde se prueba como un todo el software y otros elementos del sistema.

Cuadro 12. Ejemplo de una estrategia de prueba, según Pressman

Las actividades de la estrategia de prueba se muestran en el anexo 3.

Ahora bien, Sommerville[5], 1988, menciona que la prueba es un proceso para establecer la existencia de errores en un programa, además la prueba del software solamente puede demostrar la presencia de errores en el mismo, no su ausencia, es decir, que después de haber aplicado una prueba muy completa, siempre es posible que aún existan errores, lo cual coincide con Pressman.

El proceso de pruebas, según Sommerville, está integrada por las etapas siguientes, véase cuadro 13.

Las pruebas de funciones.	Las pruebas de módulos.	Las pruebas de subsistemas.	Las pruebas del sistema.	Las pruebas de aceptación.
Son aquellas donde se prueban las funciones que componen un módulo para garantizar que opere de manera correcta.	Una vez que se ha probado en forma individual cada función de un módulo, se procede a la prueba de la conjunción de las mismas, pero vistas como un módulo.	Son las pruebas que se realizan a cada subsistema, los cuales están formados por varios módulos.	Son aquellas pruebas en las cuales se integran los subsistemas que conforman el sistema completo.	Son aquellas que realiza el propio usuario, utilizando para ello, datos reales.

Cuadro 13. Etapas del proceso de pruebas, según Sommerville.

Las pruebas de funciones y de módulos las realiza el propio programador, mientras que las demás deben ser realizadas por un grupo de comprobadores

Para las pruebas de funciones no se requiere una especificación formal de prueba, las de módulos y de subsistemas se deben planear a medida que se formula el diseño del subsistema, mientras que para las del sistema y las de aceptación se deben preparar en la etapa de diseño o durante la aplicación.

Cualquier técnica que se emplee para probar el software desarrollado, su objetivo primordial es encontrar errores que no se han detectado hasta ese momento y que éstos sean corregidos, con lo cual se requiere de una planeación y seguimiento de las acciones correctivas que se tengan que dar, con el fin de asegurar la eliminación de los errores detectados (véase gestión de la configuración del software).

III.4.- Estándares y procedimientos

III.4.1 Estándares

El grado de aplicación de estándares en el proceso de la ingeniería del software varía de empresa a empresa. En muchos casos los estándares vienen dados por los clientes o por mandamientos de regulación.

La norma ISO 9000-3, proporciona una serie de estándares para el aseguramiento de calidad y la gestión de la misma, independientemente del ciclo de vida que se utilice, véase anexo 4.

El aseguramiento de calidad en los estándares puede ser llevada a cabo por los encargados del desarrollo de software como parte de una revisión técnica formal o mediante su propia auditoría. Por lo tanto la organización debe de establecer un plan y debe adquirir o desarrollar sus estándares.

Los estándares de software son definidos como los criterios establecidos por medio de los cuales los productos de software pueden ser comparados, por lo tanto deben existir estándares de documentación, estándares de diseño y estándares de codificación, los cuales son necesarios en una empresa para que todos los participantes en el desarrollo de software tengan una guía de que tomar en cuenta y cuales son las políticas o reglas para hacer el diseño, la programación y la documentación.

Los estándares no es más que decir las reglas de cómo se debe hacer o qué se debe tomar en consideración, por ejemplo, al crear la documentación de un sistema. Qué es lo que se permite, cómo se debe de hacer, reglas a seguir, se va a escribir con mayúsculas solamente, un comentario debe ir entre paréntesis, la numeración de puntos será arábiga, que tantos subincisos se permitirán, etc.

Estándares de documentación.

En relación a los estándares de documentación, la programación incluye el código fuente y todos los documentos de apoyo generados durante su análisis, diseño, desarrollo, codificación, pruebas, instalación y mantenimiento de un sistema.

La documentación interna debe incluir estándares para la documentación de presentación, así como del asunto que se trate, nombre del grupo de trabajo, nombre del responsable, del programador, etc.

Deben de existir formatos específicos para los documentos de la definición de requerimientos, del diseño conceptual, diseño detallado, plan de pruebas, pruebas, reportes, revisiones, auditorías, listas de sucesos, propuesta de un cambio, orden de cambio, etc.

Estos documentos deben ser diseñados, comentados y revisados por personal de los diferentes grupos que intervienen en el desarrollo y aprobados por las autoridades pertinentes, las cuales deben de distribuir, mantener y supervisar que sean bien utilizadas, que estén actualizadas, etc.

En las figuras 8 Y 9 se muestran dos ejemplos de formatos, uno de la propuesta de un cambio y otro referente a un reporte de una auditoría

En el anexo 5, se dan otros ejemplos de formatos que se deben utilizar:

Estándares de diseño.

Los estándares son usados para hacer que las tareas aseguren calidad y que sus correcciones sean simples y fáciles de realizar. El aseguramiento de calidad no especifica los métodos que el grupo desarrollador debe seguir, pero todos los métodos deberán sostenerse en los siguientes estándares:

La primera fase es especificar que debe de hacer el sistema (requerimientos), la segunda es determinar como el sistema lo tiene que hacer (diseño) y la tercera es la implementación (codificación).

Las especificaciones de requerimientos que estén incompletas, inconsistentes y engañosas pueden guiar a un diseño y a un código incorrecto.

Las especificaciones de requerimientos deben: expresar que necesidades se tienen y no como hacerlas; abarcar las entradas del sistema de la cual el software es una parte; abarcar el medio ambiente en el cual el sistema operará; ser un modelo cognoscitivo, es decir, describirán al sistema como es percibido por sus usuarios; ser bastantes completas como para determinar si la implementación propuesta satisface las especificaciones para un caso de prueba arbitrario; ser tolerantes a aumentos y faltantes.

FPC _____

FORMATO DE PROPUESTA DE CAMBIO

Nombre y dirección de quien propone el cambio: _____

_____

INFORMACIÓN DE LAS AFECTACIONES DEL CAMBIO:

Afectación a: _____

Documento: _____

Elemento: _____

Plan de prueba: _____

_____

Líneas bases afectadas: _____

Otros elementos que afecta: _____

DESCRIPCIÓN DEL CAMBIO: _____

_____

_____

_____

NECESIDAD DEL CAMBIO: _____

_____

_____

_____

IDENTIFICACIÓN DE LOS EFECTOS EN LAS CONFIGURACIONES INVOLUCRADAS:

Efectos sobre el plan original de desarrollo: _____

_____

Efectos sobre elementos de configuraciones: _____

_____

Requerimientos de cambio: _____

_____

Soluciones alternativas: _____

_____

Fecha que se requiere que se apruebe: _____

INFORMACIÓN SOBRE EL RESULTADO DE LA PROPUESTA:

Aprobada: No aprobada:

Nombres y firmas: _____

_____

_____

Fecha: _____

Figura 8. Formato de una propuesta de cambio.

FRA _____

FORMATO DE REPORTE DE UNA AUDITORÍA

IDENTIFICACIÓN DEL REPORTE:

Identificación de la auditoría: _____

Fecha de la realización de la auditoría: _____

Nombres y firmas de los miembros del grupo auditor: _____

RESULTADOS OBTENIDOS DE LA AUDITORÍA:

Número de versión del producto auditado: _____

Anomalías encontradas en la auditoría: _____

Recomendaciones para cada anomalía encontrada: _____

RESUMEN DE LA AUDITORÍA:

Resumen de la auditoría practicada: _____

Estado de la auditoría: _____

Fecha de la siguiente auditoría a realizar: _____

Figura 9. Formato de un reporte de una auditoría.

El diseño del sistema será seguido por la codificación del software a través de un proceso casi mecánico, por lo cual se requieren seguir los estándares siguientes: el diseño deberá exhibir una organización jerárquica que permita un control inteligente entre sus componentes; el diseño deberá ser modular, particionando los componentes para que realicen funciones específicas; el diseño deberá contener las representaciones de datos y procedimientos por separado; el diseño conducirá a las interfaces que reducirán la complejidad entre los módulos y el medio ambiente externo; el diseño se derivará del uso de un método y será guiado por la información obtenida durante el análisis de los requerimientos del software.

El código del software debe estar autodocumentado, debe ser sencillo, debe ser fácil de leer y de modificar. Un código que está escrito en secreto o pobremente, será dificultoso probarlo y mantenerlo.

Un software bien diseñado es fácil de mantener y comprender, proporcionando un alto grado de confiabilidad.

Dada una definición de requisitos, el ingeniero en computación debe desarrollar el diseño de un sistema de programación que satisfaga esos requisitos, mediante las siguientes etapas:

- Deben establecerse los subsistemas que componen el sistema original.

- Cada subsistema debe dividirse en componentes individuales y ha de establecerse la especificación de los subsistemas definiendo la operación de esos componentes.

- Después, cada programa se puede diseñar a base de subcomponentes que actúen recíprocamente.

- Después, hay que refinar cada componente. Esto suele implicar la especificación de cada componente como una jerarquía de subcomponentes. En este proceso de refinamiento hay que especificar con detalle los algoritmos utilizados en cada componente.

Existen gran cantidad de metodologías de diseño, pero todas caen en alguna de estas tres áreas:

- Diseño funcional descendente. El sistema se diseña desde un punto de vista funcional, empezando con una visión de alto nivel y refinándola de manera progresiva hasta llegar a un diseño más detallado.

- Diseño orientado al objeto. El sistema se ve más como una colección de objetos que como funciones que pasan mensajes de un objeto a otro. Cada objeto tiene su propio conjunto de operaciones asociadas.

- Diseño controlado por los datos. Plantea que la estructura de un sistema de software debe reflejar la estructura de los datos que éste procesa. Por lo tanto, el diseño del software se obtiene de un análisis de los datos del sistema de entrada y salida.

Algunas herramientas que se utilizan como estándares de diseño son por ejemplo:

- Diagramas de flujos de datos. Son diagramas que se utilizan para describir un diseño, muestran cómo se transforman los datos al pasar de un componente del sistema a otro.

- Diagramas o cartas de estructura. Son gráficas jerárquicas que muestran la relación estructural de los componentes de un sistema, a través de una representación de árbol.

- Un lenguaje mnemónico para la descripción del diseño. Es una notación con algunos atributos de los lenguajes de programación, lo cual hace que sea muy sencillo pasar del diseño a la programación.

Estándares de codificación.

La escritura de un programa de cómputo, se reduce a la escritura de una secuencia de sentencias en un lenguaje disponible. De cómo se exprese cada una de esas sentencias es lo que determina en gran medida que sea claro y que se entienda.

La claridad y entendimiento del código fuente se mejora mediante la utilización de técnicas de codificación estructurada, buen estilo de codificación, documentos adecuados de apoyo, buenos comentarios internos, entre otros.

El objetivo de la codificación o programación estructurada es linealizar el flujo de control a través de un programa de computadora, de modo que la secuencia de ejecución siga a la secuencia en que está escrito el código.

Cualquier segmento de programa debe tener una sola entrada y una sola salida, además debe poder especificarse usando solamente proposiciones de secuenciación, selección e iteración.

La secuenciación sería por ejemplo: S1; S2; S3;

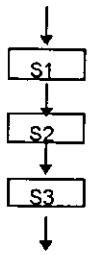

Figura 10. Secuenciación.

La selección sería por ejemplo: If A then S1 else S2

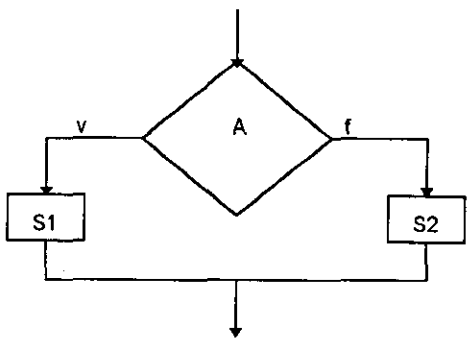

Figura 11. If A than S1 else S2

La iteración sería por ejemplo: while A do S1;

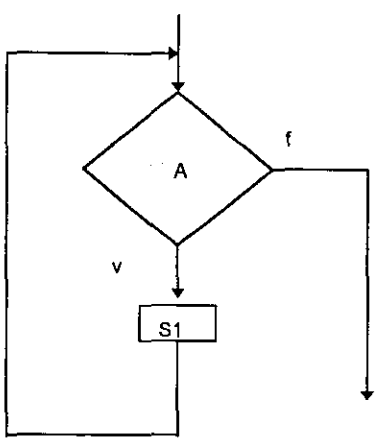

Figura 12. Iteración

Existen otras diferentes construcciones de flujos de control como pueden ser: el case para la decisión; el repeat - until para la iteración; etc., todas ellas con el propósito de hacer más claro y entendible lo que se está programando, aunado con los comentarios que debe de llevar el código permite que quede claro lo que

se esta haciendo, esto trae como consecuencia que el mantenimiento sea más fácil y sencillo de realizar, con todas sus respectivas ventajas como puede ser el costo, el tiempo que se empleará, la facilidad de mantenimiento, etc.

La programación debe ser modular, es decir, un módulo se debe de dedicar a hacer algo muy específico, las partes internas de cada módulo deben de tener una gran interrelación o en otras palabras tener con una cohesión alta, todas sus partes necesarias para obtener el propósito o función del módulo.

Por otro lado, entre módulos el acoplamiento debe ser lo más bajo posible, es decir, cada módulo debe de hacer cosas independiente y la comunicación entre ellos será solamente el paso de los datos, los cuales deben de estar integrados, por ejemplo en estructuras de datos, para su mejor y más fácil manejo, así como su entendimiento, etc.

En cuanto al estilo de la codificación, éste se manifiesta en la forma como el programador expresa una acción o un resultado deseado, el definir y utilizar un buen estilo de programación da como resultado que se obtengan varias ventajas, como son: que el código programado sea fácil de comprender, que sea sencillo y sobre todo fácil de mantener.

Algunos principios generales para un buen estilo de programación, son por ejemplo: utilizar sangrías, paréntesis, espacios, líneas en blanco o márgenes alrededor de los bloques de comentarios para mejorar la legibilidad; utilizar las sangrías adecuadas para realizar una codificación apropiada, donde se distinga fácilmente las estructuras de control utilizadas y se aprecie donde inicia y donde acaba una proposición; utilizar los comentarios necesarios para indicar si se trata de un subprograma, programa, o rutina; utilizar los estándares definidos por la empresa para nombrar programas, subprogramas, rutinas, funciones, procedimientos, variables, etc.

Algunos principios o estándares generales para la programación, son entre otros: debe de evitarse en circunstancias normales el uso de "go to"; la profundidad de anidamiento de las construcciones de un programa debe ser menor a una cierta cantidad especificada; la longitud en líneas en una subrutina no deberá de exceder una cierta cantidad especificada, estas cantidades son definidas por la empresa, es decir, sus estándares propios.

En relación a los comentarios se puede decir que éstos deben ser claros y precisos, usando la terminología adecuada de acuerdo al problema; se deben usar nombres descriptivos para los tipos de datos definidos por el usuario, variables, subprogramas, archivos, etc.; se deben de realzar los comentarios utilizando para ello líneas en blanco o delimitadores; los comentarios para documentar cambios y revisiones deben colocarse en la parte derecha; siempre

se debe de tratar que los comentarios y el código correspondan uno con el otro, así como con los requisitos y especificaciones de diseño

Al principio de cada módulo debe de haber un comentario de prólogo, el cual debe contener:

- El nombre del módulo;
- Una descripción del propósito del mismo;
- Una descripción de la interfaz que incluya: un ejemplo de la secuencia de llamada, una descripción de todos los argumentos y una lista de todos los módulos subordinados;
- Una explicación de los datos necesarios, tales como las variables importantes y su uso, de las restricciones y limitaciones y de otra información importante; y
- Una historia del desarrollo del módulo que incluya: el nombre del diseñador del módulo, el nombre del revisor o auditor y la fecha de la misma, y fechas de modificación y de descripción.

Ahora bien, los comentarios que se incluyen en el cuerpo del código fuente, se usan para describir las funciones de procesamiento y se les conoce con el nombre de comentarios descriptivos, éstos comentarios deben de: proporcionar algo extra; describir los bloques de código en lugar de comentar cada línea; usar líneas en blanco o tabulaciones de forma que sean fácilmente distinguibles del código; y sobre todo que estos comentarios sean correctos y que ayuden realmente a comprender más fácilmente el código escrito

La organización de las estructuras de datos se definen durante el paso de diseño. El estilo en la declaración de datos se establece cuando se genera el código, la declaración de datos debe de llevar un orden para que sea clara, comprensible y más fácil de mantener, por ejemplo, el orden de declaración en fortran sería: todas las declaraciones de las variables; todos los bloques de datos globales; todos los arreglos globales; todas las declaraciones de archivos, etc.

La construcción del flujo lógico del software se establece durante el diseño. La construcción de sentencias individuales es parte del paso de codificación. La construcción de sentencias se debe basar en una regla general: Cada sentencia debe ser simple y directa, utilizando el sangrado respectivo o espacios para incrementar la legibilidad del contenido de la misma, se deben de usar en su caso paréntesis para clarificar las expresiones aritméticas o lógicas, eliminar las comparaciones con condiciones negativas, etc.

III.4.2. Procedimientos

Los procedimientos están definidos como los criterios establecidos por medio de los cuales los procesos de desarrollo y control son comparados, por lo tanto los procedimientos son las secuencias de pasos que deben seguirse para completar algún desarrollo o proceso de control.

Los procedimientos deben estar integrados en un manual de procedimientos. Este manual es importante, debido a tres razones:

Los conjuntos de procedimientos muestran el orden y la secuencia de pasos y/o actividades que se deben seguir para alcanzar un fin, estos procedimientos deben ser hechos de acuerdo a las políticas de la empresa, debiendo estar aprobados por la misma y difundidos ampliamente entre todo el personal.

El departamento interno día a día es guiado por alguno de los procedimientos, los nuevos elementos los pueden consultar para enterarse de como son los procedimientos.

Los departamentos pueden tener lecturas rápidas de los procedimientos para entender qué es lo que se hace exactamente y cómo se debe de llevar a cabo.

El manual de procedimiento debe ser comprensible y aplicable a todos los proyectos, pudiendo depender en cierta medida de la estructura organizacional de la compañía, de los procesos de desarrollo del software y del control de los sistemas empleados.

El manual de procedimiento debe ser revisado, modificado, actualizado, autorizado y difundido de acuerdo a las necesidades de cada empresa.

A continuación se muestra un ejemplo de un procedimiento para realizar un cambio en el software que se desarrolla, en el anexo 6, se muestran otros ejemplos de procedimientos. Implícitamente la propuesta misma es una serie de procedimientos, por ejemplo: los diferentes enfoques para el desarrollo y el aseguramiento de la calidad del software; las estrategias y procesos de pruebas; las revisiones y auditorías; la configuración del software; etc.

- Reconocer la necesidad de cambio.
- El usuario solicita por escrito la solicitud de cambio, a través del formato de propuesta de cambio.
- Se asigna a un individuo o grupo para que sea investigado el cambio, se evalúe y se genere la propuesta de cambio ingenieril.
- La autoridad de control de cambios decide si procede o no la propuesta de cambio. (En caso de que no proceda, se le informa al usuario por escrito).
- En el caso de que si procede, la propuesta de cambio ingenieril se le asigna una prioridad y se genera la orden de cambio del software.
- Personal es asignado a los elementos de las configuraciones que serán afectadas.
- Llevar el control de cambios de los elementos involucrados.
- Efectuar el cambio.
- Revisar el cambio.
- Proceder a realizar el plan de pruebas a las bases líneas generadas.
- Realizar las actividades de pruebas correspondientes.
- Llevar a cabo las revisiones formales de las pruebas realizadas.
- Aprobar las nuevas configuraciones de software, informar a todos los grupos involucrados los cambios efectuados y distribuir las nuevas versiones, documentaciones, etc.

Cuadro 14. Ejemplo de un procedimiento para realizar un cambio en el software que se desarrolla.

III.5.- Revisiones y auditorías

III.5.1 Revisiones

Las revisiones del software son un filtro para el proceso de ingeniería del software. Las revisiones se aplican en varios momentos del desarrollo del software y sirven para detectar defectos que puedan así ser eliminados.

Las revisiones también son conocidas como revisiones formales, revisiones técnicas o revisiones técnicas formales.

Cualquier revisión es una forma de aprovechar la diversidad de un grupo de personas para:

- Señalar la necesidad de mejoras en el producto de una sola persona o un equipo;

- Confirmar las partes de un producto en las que no es necesaria o no es deseable una mejora;

- Conseguir un trabajo técnico de una calidad más uniforme, o al menos más predecible, que la que puede ser conseguida sin revisiones, con el fin de hacer más manejable el trabajo técnico.

El beneficio más obvio de las revisiones es el pronto descubrimiento de los defectos del software, de forma que cada defecto pueda ser corregido antes de llegar al siguiente paso del proceso de Ingeniería del Software y no acarree problemas posteriores.

Los objetivos de la revisión son:

- Descubrir errores en la función, la lógica o la implementación de cualquier módulo del software;

- Verificar que el software bajo revisión alcanza sus requisitos;

- Garantizar que el software ha sido representado de acuerdo con ciertos estándares predefinidos;

- Conseguir un software desarrollado de forma uniforme y hacer que los proyectos sean más manejables.

Las revisiones sirven también como campo de entrenamiento, permitiendo que los ingenieros más jóvenes puedan observar los diferentes enfoques al análisis, diseño e implementación del software. Así mismo sirven para promover la seguridad y la continuidad, ya que varias personas se familiarizan con partes del software que, de otro modo, hubiera sido difícil que lo vieran.

La reunión de revisión debe acogerse a las siguientes restricciones:

Deben convocarse a la revisión un número pequeño de personas (entre 3 y 5);

Se debe preparar por adelantado, pero sin que requiera más de un par de horas de trabajo a cada persona y la duración de la reunión de revisión debe ser menor de un par de horas.

Cada revisión se debe centrar en una parte específica y pequeña del software total.

Un ejemplo de un posible procedimiento para llevar a cabo una reunión se muestra en la parte relacionada con los procedimientos (véase anexo 6).

La lista de sucesos de revisión sirve para dos propósitos: identificar áreas problemáticas dentro del producto y servir como lista de puntos de acción que guíen al productor para hacer las correcciones. Debe existir un procedimiento de seguimiento que asegure que los puntos de la lista de sucesos son corregidos adecuadamente.

Algunas directrices para la revisión son:

- Revisar el producto, no al productor;
- Fijar una agenda y mantenerla;
- Limitar el debate y las impugnaciones;
- Enunciar áreas de problemas, pero no intentar resolver cualquier problema que se ponga de manifiesto;
- Tomar notas escritas;
- Limitar el número de participantes e insistir en la preparación anticipada;
- Desarrollar una lista de preguntas de comprobación para cada producto que haya de ser revisado;
- Disponer recursos y una agenda para las revisiones técnicas formales;
- Llevar a cabo un buen entrenamiento de todos los revisores;
- Repasar las revisiones anteriores.

Para garantizar los productos que se despachan como parte de un desarrollo de software, deberán tener listas de preguntas de comprobación para cada una de las etapas principales, en el anexo 7 se presentan algunos ejemplos de este tipo de listas de comprobación.

III.5.2 Auditorías

Ahora bien, cómo podemos saber si concuerdan nuestros requisitos con los estándares, para ello utilizamos el concepto de auditorías, las cuales serán un elemento muy importante en el aseguramiento de calidad del software.

Las auditorías, según el Gobierno de la NASA[19], se refieren a una parte de la técnica del aseguramiento de la calidad, la cual es usada para examinar la concordancia de los procesos desarrollados con los procedimientos y la concordancia de los productos con los estándares. Una auditoría también puede examinar la concordancia del estado actual de la actividad desarrollada con el estado reportado.

El término auditoría es usado para describir: un cierto número de estados; un cierto número de actividades de desarrollo de software adicionales.

El objetivo de una auditoría es determinar que tanto se apegan los procedimientos con los estándares establecidos.

Las auditorías también pueden comparar el estado actual de un producto con el estado reportado.

Una auditoría puede ser interna o externa, dependiendo del origen del auditor en la organización.

Una auditoría interna es conducida por un auditor interno, perteneciente al grupo de desarrolladores de software y la intensión es detectar problemas, antes de que éstos se conviertan en problemas mayores.

Una auditoría externa es conducida por un auditor externo a la compañía, con la idea de obtener una opinión independiente sobre el progreso de un trabajo. Una ventaja es que la auditoría es más objetiva, más real, al realizarse con un auditor externo.

Una auditoría en general sobre el aseguramiento de calidad del software, debe ser planeada cuidadosamente para examinar todo lo relacionado con la Ingeniería de Software, la gestión y los procesos de aseguramiento, así como todo lo relacionado con sus productos.

A continuación se muestra en el cuadro 15 y 16, los posibles tipos de auditorías y las auditorías que se pueden seguir para la propuesta. En el anexo 8 se muestran las fases de una auditoría, las auditorías de aseguramiento de calidad del software durante el ciclo de vida del software y ejemplos de listas de preguntas de comprobación para una auditoría.

Auditorías internas.	Auditorías externas	Auditorías de aseguramiento de calidad como respuesta a señales de aviso	Auditorías programadas.
Las auditorías internas se usan para detectar problemas potenciales con la idea de que no se presenten sorpresas desagradables en el transcurso del desarrollo del software. Debe haber actividades rutinarias durante el ciclo de vida, en particular alrededor de las fases de transición del mismo. Frecuentes auditorías combinadas con otros programas de monitoreo SQA, permiten asegurar que el estado actual del desarrollo del software es conocido y concuerda con los programas, estándares y procedimientos establecidos.	Las auditorías externas requieren de planeaciones e intervalos de tiempo mayores, pero son programadas con menor frecuencia. Un primer momento para hacer una auditoría externa, es al inicio de la fase de implementación o desarrollo del software. Esta auditoría asegura que los programas que se desarrollen hayan sido implementados de una manera adecuada siguiendo los estándares y procedimientos de la empresa y se haya tenido un adecuado seguimiento. Un segundo momento para hacer una auditoría externa en los proyectos con ciclo de vida clásico, es al iniciar la integración del sistema, ayuda a asegurar que el software está listo para la integración y que los planes y procedimientos están en su sitio y que los procedimientos para el control de software, no hayan presentado problemas. Otro factor a considerar aparte de las auditorías internas y externas, son los resultados de auditorías previas. Si existen varios problemas y acciones en una auditoría, éstas deben ser programadas con mayor frecuencia. Los desarrollos que dan seguimiento a sus procedimientos, que cumplen con sus estándares, que lleven adecuadamente su configuración de software y que cubren sus programas de reportes y estados necesitan menos frecuencia de auditorías.	Algunos proyectos dan muestras de problemas en el desarrollo del mismo. Cuando se presentan señales de avisos, es el momento más adecuado para que el que mando hacer el sistema realice una auditoría externa. Los mismos signos de aviso pueden ser usados para que los desarrolladores del software realicen una auditoría interna, de tal manera que se detecten los problemas que existen y se tomen las medidas necesarias para corregir el rumbo del proyecto a tiempo. Los programas de auditorías deben ser intensificados, si se presenta alguna de las siguientes señales de aviso: - Cambios o tropiezos frecuentes en los programas. - Inconsistencia en la estructura organizacional de los desarrolladores con los planes originales o aparente inconsistencia con la estructura o funcionalidad de los productos a ser producidos. - Cambios inexplicables en la gente del grupo desarrollador. - Incrementos en el número de seguimientos y acciones para resolver problemas, sin un adecuado progreso en sus soluciones. - Retrasos continuos en los programas de software para liberar versiones. - Número excesivo de solicitudes de cambios, etc.	Una adecuada programación de las auditorías debe ser hecha a los desarrolladores por un número de razones. Una auditoría sorpresa anunciada con pocos días, es considerada como destructiva y desmoralizante entre la gente del grupo desarrollador, por lo tanto se debe de evitar. El intento de un programa de auditorías, debe ser el de ayudar a lograr la conformancia con los estándares y procedimientos y los reportes de estados y no agarrarlos en el momento y culparlos de violaciones. Las auditorías programadas permiten una mejor realización de las mismas ya que permiten por ejemplo tener la documentación requerida disponible y preparada para responder las posibles preguntas del auditor.

Cuadro 15. Tipos de auditorías para el aseguramiento de calidad del software

Definición de requerimientos	Análisis	Diseño	Implementación	Pruebas	Puesta en marcha y mantenimiento.
Las auditorías deben dirigirse a la verificación de la documentación relacionada con la revisión técnica formal de la especificación de la definición de los requerimientos, en la cual todo haya quedado claro y sin ambigüedades y que estén firmados de conformidad por el cliente y el desarrollador	Las auditorías se deben de dirigir a la verificación de que la descripción formal del sistema cumpla con todos los requerimientos especificados por el cliente, así como a la verificación de las revisiones técnicas formales de las especificaciones del análisis	Las auditorías en esta fase se deben de dirigir a la verificación de los estándares de diseño, así como a la verificación de las revisiones técnicas formales del diseño conceptual y del diseño detallado.	Las auditorías deben dirigirse a la verificación de que se están siguiendo los estándares de documentación y de codificación, así como los procedimientos de codificación en los programas elaborados,	Las auditorías deben dirigirse a la verificación y cumplimiento de las pruebas programadas, al seguimiento y conclusión de los errores detectados por las pruebas realizadas, así como la verificación de que se han realizado las pruebas correspondientes, de acuerdo al plan de pruebas y con los datos de prueba especificados, obteniéndose los resultados esperados.	Se debe de verificar que se hayan realizado correctamente todas las actividades contempladas en la puesta en marcha, sobre todo lo relacionado con la instalación del sistema, así como la capacitación del personal que utilizará el sistema y que se estén cumpliendo adecuadamente con los programas de mantenimiento programados.

Cuadro 16. Ejemplos de auditorías sugeridas para la propuesta.

III.6.- Gestión de la configuración del software

Al conjunto de información formado por: Programas de computadoras (tanto en código fuente como ejecutable); Documentos relacionados con el software (como pueden ser los manuales técnicos o de usuario, reportes de errores, solicitudes de cambios, instrucciones de instalación, etc.); y las estructuras de datos utilizadas en los programas, se les conoce con el nombre de configuración del software.

La gestión de la configuración del software[10], 1995, es el proceso cuyo objetivo es la identificación de las configuraciones de software en puntos estratégicos de tiempo y el control sistemático de cambios para la identificación de configuraciones con el propósito de mantener la integridad del software y su seguimiento a través del ciclo de vida del software.

Hay cuatro funciones que debe realizar la gestión de configuración del software:

- Identificación de los componentes que integran el sistema de software y que definen sus características funcionales.

- Control de cambios de los componentes.

- Reportes de estado de las solicitudes de cambios.

- Validación de que los elementos controlados cumplen con los requerimientos y que están listos para ser liberados.

Por lo tanto, y en pocas palabras, la gestión de la configuración es el arte de identificar, organizar y controlar las modificaciones que sufre el software que construye un equipo de desarrollo de software.

La gestión de la configuración del software es un conjunto de actividades de seguimiento y control, que inician cuando comienza el proyecto de desarrollo de software y terminan solamente cuando el software queda fuera de circulación, desarrolladas para dar seguimiento y control a los cambios del mismo a lo largo del ciclo de vida.

La gestión de la configuración del software es una actividad de aseguramiento de la calidad del software que se aplica en todas las fases del proceso de ingeniería del software.

La aplicación propia de la gestión de configuración del software es una componente clave en el desarrollo de software con calidad. Los cambios del software sobre el desarrollo son usualmente significativos para el plan del proyecto, ahora bien un cambio no gestionado es muy posible que sea un motivo para no liberar a tiempo el sistema, en el desarrollo de software siempre existirán cambios que se tengan que realizar, cambios que vienen de adentro o de afuera, es decir, los cambios externos son originados por los usuarios, por la evolución del medio operacional y por los improvistos en la tecnología, mientras que los cambios internos, son originados por cuestiones de diseños y métodos, por el avance del mismo desarrollo y por la corrección de errores, por lo que el proceso de la configuración del software fue desarrollado para controlar y gestionar dichos cambios.

Las actividades de la gestión de la configuración del software sirven para:

- Identificar el cambio;
- Controlar el cambio;
- Garantizar que el cambio se realice adecuadamente;
- Informar del cambio a todos aquellos a los que les afecte.

Ahora bien, pero, ¿a partir de dónde se debe de considerar un cambio?, para lo cual se definirá lo que es una línea base, se define una línea base como un punto de referencia en el desarrollo de software que queda marcado por el envío de un elemento de configuración de software y la aprobación de dicho elemento, obtenido mediante una revisión técnica formal, por ejemplo, los documentos de la especificación de diseño se documentan y se revisan. Se encuentran errores y se corrigen. Cuando todas las partes de la especificación se han revisado, corregido y aprobado, la especificación de diseño se convierte en una línea base.

Sólo se pueden realizar cambios futuros en la estructura del software contenidos en la especificación de diseño, tras haber sido evaluados y aprobados, convirtiéndose en una nueva versión del objeto.

De esta manera las líneas bases más comunes para el proceso de desarrollo propuesto, serían por ejemplo:

- Al aprobarse la definición de requerimientos, sería la especificación de requerimientos del sistema.
- Al aprobarse el análisis del software, sería la especificación del análisis.
- Al aprobarse el diseño del software, sería la especificación de diseño.
- Al aprobarse la implementación o codificación, sería el código fuente.
- Al aprobarse las pruebas, serían los planes, procedimientos y datos prueba.
- Al liberarse el software, sería el sistema en funcionamiento con toda su configuración.

Ahora bien, el software bajo control está usualmente dividido en elementos de configuración. Un elemento de configuración es el término usado para cada uno de los componentes lógicos relacionados que hacen del software un elemento discreto, por ejemplo, si un sistema contiene varios programas, cada programa, su documentación respectiva y los datos que requiere serán llamados un elemento de configuración. El número de elementos de configuración en un sistema de software es una decisión de diseño. Cada elemento de la configuración en el desarrollo del proceso va generando más y más componentes hasta formar una configuración de software disponible para su uso.

Cada línea base deberá estar sujeta bajo el control de la configuración, debiendo estar actualizada para reflejar apropiadamente los cambios a los elementos de configuración para los siguientes estados de desarrollo.

La gestión de configuración del software consiste de cuatro subprocesos: (véase cuadro 17)

- Identificación de la configuración
- Control de la configuración
- Reportes de estado de la configuración
- Validación de la configuración.

Identificación de la configuración	Control de la configuración	Reporte de estado de la configuración	Validación de la configuración
La identificación de la configuración es el proceso de definición de cada línea base la cual es establecida durante el desarrollo del ciclo de vida y describe los elementos de la configuración del software y su documentación que se crea en cada una de las líneas bases. El proceso de identificación involucra la selección, el nombramiento y la descripción de los elementos de la configuración del software. La selección involucra el agrupamiento del software dentro de elementos de configuración que estén sujetas a la gestión de la configuración. Un sistema de software está generalmente dividido en un número de elementos de configuración de software, los cuales son independientemente desarrollados, probados y los cuales son finalmente puestos juntos en un sistema durante el nivel de integración del mismo. Una regla general es que un elemento de configuración de software es establecido como una pieza del sistema del software que pueda ser diseñada, implementada y probada independientemente. El nombramiento es un plan de desarrollo de asignación de nombres y/o numeración que correlacione los componentes del software con sus	El control de la configuración es el proceso de evaluación, coordinación y decisión sobre la disposición de los cambios propuestos a los elementos de configuración y del seguimiento de la realización de los cambios aprobados a las líneas bases del software y a sus documentaciones y datos asociados. El proceso de control de cambios asegura que los cambios sean: solicitados, clasificados y evaluados, aprobados o desaprobados, documentados, implementados e incorporados en una nueva línea base. Un proceso de cambio ordenado es necesario para asegurar que solamente cambios aprobados, se estén realizando en cualquier documento o software de líneas bases. El proceso de control de cambios está integrado por: Solicitud del cambio. Una solicitud de cambio al software y/o a documentación puede ser emitida por un usuario, por el comprador, por el revisor o por un miembro del propio grupo de desarrollo. La oficina de gestión de configuración recibe la solicitud de cambio y revisa que este bien llenada y completa. Si la oficina determinada que no esta bien llenada, ésta es retornada al solicitante para su corrección. Cuando está completa, la oficina asigna una identificación única a dicha solicitud para propósitos de seguimiento y registro en las bases de datos o archivos respectivos; Clasificación. Los cambios deben estar clasificados de acuerdo al impacto del cambio y a la necesidad de aprobación, por lo regular las clases de cambios son identificados por un número romano; Evaluación del cambio. Se debe de realizar un adecuado análisis del cambio propuesto en términos del impacto de la funcionalidad del sistema, interfaces, utilidad, costos, planes y requerimientos contractuales. El análisis da como resultado documentación, la cual describe los cambios que se tendrán que hacer en los elementos de configuración y en la documentación y los recursos que se necesitarán para hacer el cambio; Resultado de la solicitud del cambio. Los elementos disponibles son enviados a la oficina para tomar una acción, aprobándose, desaprobándose o difiriéndose una solicitud, todo lo anterior requiriendo de información y análisis. Los elementos rechazados son enviados a quien los envió con su respectivo informe. Los cambios aprobados son enviados al departamento de desarrollo para su realización, se deben de preparar y distribuir las minutas de las reuniones y registrar los estados de las solicitudes de los cambios; Realización del cambio. Los cambios aprobados son autorizados para llevar a cabo	El reporte de estado de la configuración es el proceso usado para dar el seguimiento a los cambios de software. Este proceso asegura que el estado de las líneas bases sea registrado, monitoreado y reportado en acciones como pendiente o completa. También define el estado del código y la documentación asociada. Tiene como objetivo el registro y reporte de estado de la evolución del software durante el ciclo de vida. Provee el seguimiento de los cambios a los registros de líneas bases, códigos, componentes de datos y documentación asociada. Estos documentos que están en cada versión de software y los cambios que se llevaron a cabo sobre esa versión, deben estar incorporados todos en un archivo llamado archivo del control de cambios al igual que los reportes de no conformidad y sus acciones correctivas, con lo cual se tendría guardado los cambios y los contenidos de versiones y liberaciones. El registro empieza desde la primera especificación, por ejemplo la especificación de requerimientos del software, convirtiéndose en una línea base y continua durante todo el ciclo de vida. El registro de información es un elemento clave usado durante las auditorías de configuración tanto de funcionales como físicas, las cuales son hechas durante el proceso de validación. Los reportes de estado proveen una lista de contenidos	La validación de la configuración es el proceso que asegura si una línea base de software contiene todos sus elementos que son necesarios para ser liberado y que los mismos han sido verificados de que cumplen todos sus requerimientos. Como el principal significado de la validación de configuraciones es el asegurar que cumplen, entonces se programan auditorías antes de cada liberación y repartición del sistema de software. La función de validación usualmente consiste de dos auditorías. Una auditoría de configuración funcional y otra física. La auditoría configuración funcional valida que el software ha sido probado para asegurar que su rendimiento está en concordancia con los requerimientos en la documentación de las líneas bases. La auditoría

respectivas documentaciones. Cada componente de software debe estar identificado individualmente, por ejemplo, un sistema esta integrado por varios programas y cada programa por varias subrutinas, de tal manera que se tenga perfectamente identificado de que componente se trata. La descripción consiste en que cada componente debe de tener su documentación de especificaciones tanto funcionales, de rendimiento y de características físicas, la cual se vuelve cada vez más detallada conforme avanza el proceso de diseño y de desarrollo.	las modificaciones tanto en el software como en la documentación respectiva. El departamento de desarrollo requiere de recursos para llevarlos a cabo. Requiere de la documentación y del software de los componentes de las líneas bases a modificar. Para cambios al código, se tiene que desarrollar el diseño, el código tiene que ser escrito y probado, así como su documentación reflejar el cambio solicitado. Cada cambio debe ser hecho y probado localmente, los componentes y la documentación deben ser revisados y entregados al control del programa de librerías para su verificación. Hecho lo anterior la nueva versión toma el lugar en las respectivas líneas bases; Verificación del cambio. Al realizar cambios, éstos deben ser probados primeramente a nivel unitario y después a nivel de elementos de configuraciones de software. Esto puede implicar la realización de pruebas específicas en un plan de pruebas o el desarrollo de una adición en el mismo. Usualmente las pruebas de regresión deben estar incluidas para asegurar que no se han introducido errores por la existencia de nuevas funciones por el cambio. Cada una de las verificaciones debe ser totalmente concluida, el comité del Departamento de desarrollo debe de dejar evidencia de esto en el programa de librería que se lleve, las verificaciones deben ser aceptadas y la gestión debe controlar los cambios para hacer una nueva versión de una línea base; Control de cambios de líneas bases. Los cambios al software no están completos hasta que los cambios al código y a los datos han sido realizados y probados y los cambios a la documentación han sido hechos y todos hayan sido verificados. Para minimizar el número de versiones, los cambios al software son usualmente agrupados dentro de liberaciones. Cada liberación contiene software y documentación que ha sido probada y controlada como un sistema total de software.	para cada entrega del software y documentación asociada. Los reportes contienen la identificación del software inicial y los documentos asociados y sus estados de evolución de las líneas bases, el estado propuesto y cambios aprobados y el estado de realización de aprobación de cambios. Reportes de documentos de información contenidos en los reportes y el significado para la divulgación de la información.	de configuración física verifica que los componentes de software a ser liberados existen actualmente y que todos ellos contienen los elementos requeridos tales como las versiones propias de códigos fuentes y objetos, la documentación, instrucciones de instalación, etc.

Cuadro 17. Proceso de la gestión de la configuración del software, según la NASA.

III.7 Ejemplo de un plan de inspección para el aseguramiento de calidad en el desarrollo de software.

Como se había mencionado en el capítulo dos, un plan de inspección es un mecanismo para detectar problemas, por lo tanto, existirán una infinidad de planes de acuerdo a las necesidades y conveniencias propias de cada empresa.

Un plan de inspección es un documento que en forma esquemática, abreviada y haciendo referencia a otros documentos, trata de dar respuesta a las siguientes preguntas:

¿Qué inspeccionar?
¿Quién inspecciona?
¿Dónde se inspecciona?
¿Cómo se inspecciona?
¿Cada cuándo?
¿Dónde se registra la inspección?
¿Cómo comparar con lo deseado?
¿Cómo reaccionar si no se cumple lo deseado?

Un ejemplo de un posible plan de inspección para el aseguramiento de calidad en el desarrollo de software se presenta en el cuadro 18.

Qué inspeccionar	Quién inspecciona	Dónde inspeccionar	Cómo inspeccionar	Cuándo inspeccionar	Dónde registrar	Contra qué comparar	Cómo reaccionar si no se cumple lo deseado
Definición de requerimientos claros, completos y sin ambigüedades	Grupo SQA o grupo gestor	En la revisión técnica formal de la definición de requerimientos	Inspeccionar que esté debidamente llenada la revisión técnica y que tanto el cliente como el desarrollador hayan firmado de estar de acuerdo, revisar que se hayan definido claramente los requerimientos y que no haya dudas, ni pendientes, ni ambigüedades, etc. Por ejemplo ver lo relacionado con las auditorías sobre definición de requerimientos, los estándares para la especificación de requerimientos, la lista de preguntas de comprobación correspondiente a la etapa de ingeniería de sistemas, análisis de los requisitos del software, así mismo las fases de una auditoría, véase páginas 61 a la 63, 82 y de la 100 a la 105.	Al comenzar un nuevo proyecto de desarrollo, cuando se encuentra el grupo de desarrollo en la etapa de diseño conceptual.	En el formato de reporte de una auditoría, véase página 51.	Especificaciones de los requerimientos del cliente. Los estándares de la compañía desarrolladora en relación a la definición de requerimientos.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que sea corregida y tomada en cuenta para subsiguientes etapas y futuros desarrollos. Reunir a los representantes del cliente y del desarrollador para llegar a un acuerdo en las anomalías encontradas y verificar su cumplimiento y seguimiento.
Utilización y seguimiento de metodologías de desarrollo	Grupo SQA o grupo gestor	En la documentación relacionada con la definición de requerimientos, el análisis, diseño, codificación, pruebas, en la documentación de la configuración del software, etc.	Inspeccionar que se esté siguiendo la metodología seleccionada, por ejemplo en alguno de los paradigmas de la ingeniería de programación, que la metodología esté siendo correctamente aplicada, revisando que esté completa la carpeta de desarrollo, que los folders de informe de desarrollo estén al corriente, etc. Tomar como ejemplo las actividades de los diferentes paradigmas, los tipos de auditoría, las auditorías durante el ciclo de vida del software, las fases para realizar una auditoría, así como la lista de preguntas de comprobación para las fases del ciclo de vida clásico. Véase páginas 12 a la 19 y de la 61 a la 63 y de la 102 a la 105..	Durante el transcurso de cualquiera de las fases o cuando se pase de una fase a otra en el proceso de desarrollo de software que se este empleando.	En el formato de reporte de una auditoría, véase página 51.	Documentación de las metodologías de desarrollo, procedimientos y estándares de la empresa, etc.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunir al grupo gestor con el grupo desarrollador y aclarar los desvíos detectados y fijar revisiones para que se adecuen a las metodologías de desarrollo seleccionadas por la empresa. Verificar que se cumpla lo acordado. Si se detecta que no hay seguimiento, incrementar las inspecciones en cada fase de desarrollo.
Documentos de control de información entre fases.	Grupo SQA o grupo gestor.	En el documento estándar de programación.	Que estén bien llenados y completos los documentos de control, así como las autorizaciones respectivas necesarias, de acuerdo con el control de la configuración y con los procedimientos y estándares relacionados con la planeación del desarrollo de la empresa, véase páginas de la 63 a la 67, 83 y 84.	Cuando se pase de una fase a otra en el proceso de desarrollo del software.	En el formato de reporte de una auditoría, véase página 51.	Estándares y procedimientos de la empresa relacionados con documentos de control.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunir a los grupos involucrados para que cumplan debidamente con la documentación de control.

Qué inspeccionar	Quién inspecciona	Dónde inspeccionar	Cómo inspeccionar	Cuándo inspeccionar	Dónde registrar	Contra qué comparar	Cómo reaccionar si no se cumple lo deseado
Aplicación de estándares en el análisis, diseño, codificación y pruebas.	Grupo SQA o grupo gestor	En la documentación correspondiente al diseño, programación y pruebas y validaciones.	Verificar que se han aplicado correctamente los estándares de la compañía en relación a reglas de análisis, diseño, programación y pruebas, como por ejemplo: metodología de diseño, que se hayan utilizado diagramas de flujos de datos, cartas de estructura, utilización de un lenguaje mnemónico, la claridad de escritura en el programa, que se haya utilizado programación modular, utilización de técnicas de codificación estructurada, que el código esté bien documentado, etc., véase páginas 48 a la 56 y de la 83 a la 96.	En la parte final de las etapas de análisis, diseño, codificación y pruebas.	En el formato de reporte de una auditoría, véase página 51.	Con los estándares de la compañía en relación a diseño, codificación y pruebas.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunir a los grupos involucrados para que cumplan debidamente con los estándares de la compañía.
Aplicación de procedimientos durante el proceso de desarrollo	Grupo SQA o Grupo gestor	En la carpeta que contiene la documentación del proceso de desarrollo del software	Verificar si se llevaron a cabo todos los pasos que marca el manual de procedimientos de la empresa, si existe la documentación respectiva, si está completa, si esta autorizada, etc., véase páginas 57,58 y de la 97 a la 99.	Durante las diferentes etapas del proceso de desarrollo	En el formato de reporte de una auditoría, véase página 51.	Con el manual de procedimientos de la empresa.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunir a los grupos involucrados para que cumplan debidamente con los procedimientos de la empresa.
Configuración del software	Grupo SQA o grupo gestor	En la documentación relacionada con la configuración del software	Que se esté llevando la gestión de la configuración del software adecuadamente, ver si se está llevando a cabo el seguimiento de un cambio, si se cumplió con los procedimientos especificados, si se cumple con las autorizaciones respectivas para un cambio, si se lleva el seguimiento de los mismos, etc. Verificar que se cumplió el proceso de gestión de la configuración del software y aplicar por ejemplo la lista de comprobaciones para la configuración del software, véase páginas de la 63 a la 67 y 104.	Se puede inspeccionar que se cumple en cada una de las fases de desarrollo	En el formato de reporte de una auditoría, véase página 51.	Procedimientos y estándares sobre la configuración de software y desarrollo de la documentación.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunir al grupo de desarrollo para solicitarle que cumpla con la gestión de la configuración de software, dando sus avisos, actualizaciones y seguimientos.

Qué inspeccionar	Quién inspecciona	Dónde inspeccionar	Cómo inspeccionar	Cuándo inspeccionar	Dónde registrar	Contra qué comparar	Cómo reaccionar si no se cumple lo deseado
Que los requerimientos sean cubiertos por el diseño desarrollado.	Grupo SQA o grupo gestor	En la revisión técnica formal de diseño.	Se deben de llevar inspecciones para verificar que los requerimientos del clientes fueron alcanzados con el diseño preliminar y a su vez con el diseño detallado desarrollado, así mismo aplicar por ejemplo las listas de preguntas de comprobaciones para la fase de diseño correspondiente. Véase páginas 12 a la 19, de la 42 a la 47, 85, 100 y 101.	En la etapa final de la fase de diseño e inicio de la fase de implementación o codificación	En el formato de reporte de una auditoría, véase página 51.	La definición de los requerimientos del cliente y la documentación final de las fases de diseño preliminar y de diseño detallado.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con los grupos correspondientes para identificar el motivo por el cual no se está cumpliendo con los requerimientos, dándole el seguimiento necesario para su conclusión.
Que los requerimientos del cliente fueron cubiertos por la programación	Grupo SQA o grupo gestor	En la revisión técnica formal de programación.	Se deben de realizar inspecciones para asegurar que los requerimientos del clientes fueron totalmente cubiertos con la codificación realizada y que cumple con lo estipulado en la fase de diseño anterior. Se puede tomar como ejemplo la lista de preguntas de comprobaciones para la fase de codificación, véase páginas 100 y 101.	En la parte final de la fase de programación e inicio de la fase de pruebas.	En el formato de reporte de una auditoría, véase página 51.	Los requerimientos del cliente y de acuerdo también con los requerimientos de la fase anterior de diseño.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con el grupo desarrollador para corregir el problema detectado.
Que el documento estándar de programaoión este completo	Grupo SQA o grupo gestor	En la carpeta que contiene la documentación del desarrollo de software.	El documento estándar de programación debe de estar completo, es decir cada módulo del sistema a desarrollar debe estar perfectamente definido de forma precisa y detallada, de lo que se tiene que hacer, así de como obtenerlo y el medio ambiente en el cual operará, así mismo sus interfaces externas, etc. Véase páginas de la 100 a la 105.	Durante las fases de diseño, codificación y pruebas.	En el formato de reporte de una auditoría, véase página 51.	Los estándares y procedimientos de diseño, codificación y pruebas, el diseño de datos, la carta de estructura, etc.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con el grupo desarrollador para corregir el problema detectado.
Mantenimiento a los folders de informe del desarrollo de software	Grupo SQA o grupo gestor.	Folders de informe del desarrollo	Los documentos de informes deben mostrar la historia de cada uno de los módulos desde su requerimiento hasta su liberación por el grupo de pruebas, véase páginas 100, 101, 104 y 105.	Durante las fases de diseño, codificación y pruebas.	En el formato de reporte de una auditoría, véase página 51.	Los estándares y procedimientos de la empresa, así como de la carta de estructura del sistema y del plan de desarrollo del software	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con el grupo desarrollador para corregir el problema detectado.

Qué inspeccionar	Quién inspecciona	Dónde inspeccionar	Cómo inspeccionar	Cuándo inspeccionar	Dónde registrar	Contra qué comparar	Cómo reaccionar si no se cumple lo deseado
Que los requerimientos estén cubiertos en las pruebas y la integración	Grupo SQA o grupo gestor	En la revisión técnica formal de pruebas e integración	Se deben de llevar inspecciones para asegurar que los resultados obtenidos fueron los que el cliente y el grupo desarrollador esperaban y que se cumplen con los requisitos implícitos especificados originalmente. Se puede tomar en consideración la estrategia de prueba, los estándares de pruebas y validaciones, así como la lista de preguntas de comprobaciones para la fase de pruebas del software, véase páginas de la 44 a la 47, 85, 86, 100,101 y 105.	En la parte final de la fase de pruebas e integración.	En el formato de reporte de una auditoría, véase página 51.	Requerimientos del cliente (implícitos y explícitos) y los requerimientos especificados en las fases de diseño y codificación.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con el grupo desarrollador para corregir el problema detectado.
Procedimientos de prueba	Grupo SQA o grupo gestor	En el documento estándar de programación	Verificar que los procedimientos para las pruebas de: unidad, integración, validación y del sistema; o de funciones, modulares, subsistemas, del sistema y de aceptación; estén completos y firmados de conformidad por ambas partes (cliente y desarrollador), Considerar las estrategias de pruebas, los estándares para las pruebas y validaciones, así como las listas de comprobaciones relacionadas con la verificación y validación de las pruebas del software, véase páginas 44 a la 47, 85, 86, 100, 101 y 105.	Durante las fases de desarrollo y pruebas	En el formato de reporte de una auditoría, véase página 51.	Procedimientos y estándares de pruebas de la compañía y con el documento estándar de programación	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con el grupo desarrollador y de pruebas para corregir el problema detectado
Resultados de pruebas y reportes	Grupo SQA	Formas de resultados o de reportes de pruebas.	Verificar que los resultados estén correctos de acuerdo a los requerimientos del cliente y que éste este conforme. Se debe de verificar que se han seguido los procedimientos para recabar y reportar los resultados de pruebas y que se hayan realizado los pasos especificados para realizar las pruebas. Véase páginas 44 a la 47, 84, 85, 100, 101 y 105.	Etapa final del desarrollo y durante la fase de pruebas.	En el formato de reporte de una auditoría, véase página 51.	Resultados calculados o proporcionados por el cliente, grupo desarrollador y de pruebas, todos ellos basados en los requerimientos originales.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con el grupo desarrollador, de pruebas y con el cliente para corregir el problema detectado.

Qué inspeccionar	Quién inspecciona	Dónde inspeccionar	Cómo inspeccionar	Cuándo inspeccionar	Dónde registrar	Contra qué comparar	Cómo reaccionar si no se cumple lo deseado
Revisiones y auditorías se hayan realizado de manera adecuada	Grupo SQA	Documentación relacionada con revisiones e inspecciones	Verificar que las revisiones y auditorías se efectuaron de una manera adecuada y correcta de acuerdo a las fases de una auditoría, programas de auditorías, listas de comprobación, etc., así mismo que los resultados fueron realmente los indicados, la forma como se llevaron a cabo y verificando el seguimiento de las inconformidades hasta su conclusión, etc., Véase páginas de la 58 a la 63 y de la 100 a la 105.	Durante las diferentes etapas del proceso de desarrollo y según el programa de auditorías de aseguramiento de calidad establecido.	En el formato de reporte de una auditoría, véase página 51.	Estándares y procedimientos relacionados con revisiones y auditorías, listas de comprobaciones, etc.	Enviar una copia del formato de reporte de la auditoría a los responsables que cometieron la anomalía, para que ésta sea corregida y tomada en cuenta para futuros desarrollos. Reunirse con el grupo gestor y desarrollador para corregir el problema detectado.

Cuadro 18. Ejemplo de un posible plan de inspección para el aseguramiento de calidad en el desarrollo de software.

IV.- Conclusiones

Se realizó un análisis de diferentes enfoques que se utilizan para el desarrollo de un software. Se obtuvieron las fases primordiales que debe abarcar un proceso de desarrollo, las cuales son: Definición de requerimientos, análisis, diseño, implementación, pruebas, puesta en marcha y mantenimiento, y el aseguramiento de la calidad. Esta última fase, no es considerada como tal por la mayoría de los enfoques, pero se volvió de vital importancia para nuestro caso de estudio.

Por tal motivo se procedió a realizar un análisis de diferentes enfoques que existen para el aseguramiento de la calidad del software. Al igual que en el caso anterior se obtuvieron los puntos substanciales que debía contener: Gestión; Métodos, técnicas y herramientas; Estándares y procedimientos; Revisiones y auditorías; y la Configuración del software, por medio de los cuales se cubren los aspectos más importantes para asegurar la calidad del software que se desarrolle.

Una vez realizado lo anterior se presentó la propuesta para el aseguramiento de calidad en el desarrollo de software, la cual se basó en los puntos anteriores, cubriendo las partes substanciales de los diferentes enfoques analizados, es decir, tanto del proceso de desarrollo como del aseguramiento de calidad del software, con la idea de que sirva de base a las personas y empresas que se dedican a desarrollar software y les ayude a ir formando o mejorando sus propios procedimientos, con el fin de lograr una mejora continua de la calidad.

El propósito primario de la propuesta para el aseguramiento de calidad del software es garantizar que el software desarrollado cumple con todos los requerimientos del usuario y como propósito secundario, pero muy importante el definir e implementar reglas específicas (estándares y procedimientos) que aseguren que el software que se desarrolle sea de una alta calidad.

Se proporciona un documento que reúne un conjunto de actividades, responsabilidades, formatos, procedimientos, estándares, etc., los cuales considero que son de gran valor para los desarrolladores de software, ya que se les servirá de base o como complemento para poder lograr de una mejor manera el aseguramiento de la calidad en el desarrollo de software, así mismo también se convierte de importancia para los directivos, jefes de departamento, clientes, etc., ya que les ofrece una forma de poder detectar posibles problemas, así como de tomar las acciones necesarias para su corrección y su seguimiento.

En el desarrollo de un software influyen fuertemente el proceso de desarrollo y los grupos que intervienen en el mismo, por lo tanto el aseguramiento de la calidad depende en gran medida de todas las personas que intervienen en el desarrollo del mismo, es decir, si cada persona es responsable y realiza profesionalmente su trabajo y sus actividades de una manera adecuada, cumpliendo con los estándares y procedimientos, así como el seguimiento de las metodologías seleccionadas, cada una de ellas irá aportando su granito de arena, desde el cliente, los desarrolladores, los jefes, los directivos, etc., los cuales deberán estar capacitados y realizar la conducción necesaria para llevar a buen término el desarrollo del software.

El aseguramiento de calidad del software depende mucho de la participación y tiempo que le dedique el usuario a la definición de requerimientos y a las revisiones y pruebas en donde tenga que participar. Así mismo de la participación, profesionalismo y preparación de todas aquellas personas que trabajan en el desarrollo de un software. Como ya se mencionó es parte del trabajo de todos aquellos que participan en la organización, cuidando el aseguramiento de la calidad en cada una de las partes que integran el ciclo de desarrollo del sistema.

El aseguramiento de calidad en el desarrollo de software se mejorará realizando una adecuada gestión o conducción (planeación, organización, preparación, ejecución, evaluación, control y seguimiento) sobre los puntos substanciales de la propuesta, es decir, con la intención de verificar y asegurar de que se están aplicando correctamente las metodologías, técnicas, herramientas, estándares, procedimientos, revisiones, auditorías y la configuración del software que se

desarrolla, durante cada una de las fases del proceso de desarrollo del software empleado.

Cada organización o departamento de desarrollo debe de preocuparse por tener un procedimiento de aseguramiento, el cual debe ir poco a poco mejorándose y complementándose, con la ayuda de todos sus integrantes y con la experiencia en cada uno de los sistemas de software que se vayan desarrollando, con la idea de ir dando las bases necesarias para tener reglas, estándares, normalizaciones y procedimientos más completos en la empresa respectiva, los cuales ayuden a garantizar la calidad del software que se desarrolle en ella.

El aseguramiento de calidad del software es una estrategia para salvaguardar el diseño, la producción y el soporte del software. Se debe de hacer un esfuerzo total en todas las actividades relacionadas con el desarrollo para asegurar que el software sea formal, que esté totalmente documentado y probado, que sea fácil de mantener y que satisface totalmente los requerimientos del cliente.

La mayor parte de los costos de los sistemas de software no resultan de corregir errores en el mismo, si no más bien de cambios en las necesidades del usuario. Por lo tanto se debe de tener bien definidos por el usuario los requerimientos del sistema y si no es así, se requiere que se le ayude a definirlos perfectamente, antes de iniciar en forma la etapa de análisis del mismo.

Las organizaciones suministran productos que intentan satisfacer las necesidades y/o requisitos de los clientes. Para ser competitivos y mantener un buen desempeño económico, las organizaciones y los desarrolladores necesitan emplear sistemas cada vez más efectivos y eficientes. La familia de normas NMX-CC (001 a la 006) proporciona un conjunto de normas de sistemas de calidad, así mismo la norma ISO 9000-3 proporciona una serie de estándares para el aseguramiento de la calidad y la gestión de la misma, independientemente del ciclo de vida que se utilice en el desarrollo del software.

La propuesta presentada es una primera versión para que sirva de base o complemento a los desarrolladores, la cual como todo, puede ser mejorada y complementada cada vez más de acuerdo a la experiencia que se vaya teniendo en el desarrollo de software, logrando con el tiempo obtener un documento cada vez más completo sobre el tema, el cual esta actualmente tomando la importancia que merece.

Finalmente, puedo concluir que se cumplieron los objetivos planteados al inicio de este trabajo, ya que se investigó el estado del arte que guarda el tema del aseguramiento de calidad del software, se analizaron diferentes enfoques tanto para el desarrollo como para el aseguramiento de calidad del software, obteniéndose sus ventajas y desventajas. A través de la utilización de la técnica TKJ se obtuvieron las etapas que debe tener un proceso de desarrollo de software, de la misma manera se obtuvieron los puntos que debe contener el aseguramiento del software, con base en lo anterior, se elaboró una propuesta para el aseguramiento de calidad en el desarrollo de software, la cual considero que se convierte en un documento que será de gran utilidad tanto para los desarrolladores de software, como para los jefes, directivos y clientes, por medio del cual contarán con un mecanismo para detectar posibles problemas en el desarrollo del mismo y tendrán un documento que no solamente les diga los puntos que se deben considerar, sino que les proporcione una serie de procedimientos, actividades, estándares, formatos, incluyendo un plan de inspección, sirviéndoles como anteriormente mencioné, de base o de complemento para detectar problemas, tomar acciones, controlar y dar seguimiento a los problemas detectados, con la idea de obtener un mejor aseguramiento de calidad en el desarrollo de software.

ANEXO 1.

La técnica TKJ

Se aplicó la técnica TKJ, en primer lugar para obtener las etapas que debe tener un proceso de desarrollo de software y en segundo lugar para obtener los puntos necesarios que debe contener el aseguramiento de calidad del software, de la siguiente manera:

- Cada uno de los enfoques se tomó como una lluvia de ideas substantivas.
- Cada idea fue escrita en una tarjeta.
- Se eliminaron las repeticiones.
- Se agruparon en clases.
- Se le dio nombre a las clases.
- Se eliminaron las repeticiones.
- Obteniéndose entonces las clases substantivas necesarias para cada uno.

ANEXO 2

Clasificación de factores que afectan a la calidad del software, según Cavano y McCall

Características operativas. Operación del producto (su uso):	Capacidades de soportar cambios. Revisión del producto (su modificación):	Adaptabilidad a nuevos entornos. Transición del producto (modificarlo para trabajar en un entorno diferente "transportarlo"):
Adecuación. El grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el cliente. Confiabilidad. El grado en el que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida. Eficiencia. La cantidad de recursos de computadora y de código requeridos por un programa para llevar a cabo sus funciones. Integridad. El grado en el que puede controlarse el acceso al software o a los datos, por personal no autorizado. Amigabilidad. El esfuerzo requerido para aprender un programa, trabajar con él, preparar su entrada e interpretar su salida.	Mantenibilidad. El esfuerzo requerido para localizar y arreglar un error en un programa. Flexibilidad. El esfuerzo requerido para modificar un programa operativo. Facilidad de prueba. El esfuerzo requerido para probar un programa de forma que se asegure que realiza su función requerida.	Portabilidad. El esfuerzo requerido para transferir el programa desde un hardware y/o un entorno de sistemas de software a otro. Modularidad. El grado en que un programa se puede reusar en otras aplicaciones. Facilidad de interoperabilidad. El esfuerzo requerido para acoplar un sistema a otro.

El esquema de graduación propuesto por Cavano y McCall usa las siguientes métricas:

Facilidad de auditoría. La facilidad con que se puede comprobar la conformidad de los estándares.
Exactitud. La precisión de los cálculos y del control.
Normalización de las comunicaciones. El grado en que se usan, el ancho de banda, los protocolos y las interfaces estándar.
Completitud. El grado en que se ha conseguido la total implementación de las funciones requeridas.
Conciso. Lo compacto que es el programa en términos de líneas de código.
Consistencia. El uso de un diseño uniforme y de técnicas de documentación a lo largo del proyecto de desarrollo del software.
Estandarización en los datos. El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.
Tolerancia de errores. El daño que se produce cuando el programa encuentra un error.

Eficiencia en la ejecución. El rendimiento en tiempo de ejecución de un programa.
Facilidad de expansión. El grado en que se puede ampliar el diseño arquitectónico de datos o procedimental.
Generalidad. La amplitud de aplicación potencial de los componentes del programa.
Interdependencia del hardware. El grado en que el software es independiente del hardware sobre el que opera.
Instrumentación. El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen
Modularidad. La independencia funcional de los componentes del programa.
Facilidad de operación. La facilidad de operación de un programa.
Seguridad. La disponibilidad de mecanismos que controlen o protejan los programas o los datos.
Autodocumentación. El grado en que el código fuente proporciona documentación significativa.
Simplicidad. El grado en que un programa puede ser entendido sin dificultad.
Interdependencia del sistema de software. El grado en que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones del entorno.
Facilidad de traza. La posibilidad de seguir la pista a la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requisitos.
Formación. El grado en que el software ayuda para permitir que nuevos usuarios apliquen el sistema.

Hewlett-Packard (H.P.) desarrollo un conjunto de factores de calidad del software, cuya abreviatura es FURPS (por sus siglas en inglés de: funcionalidad, facilidad de uso, fiabilidad, rendimiento y capacidad de soporte), el cual se muestra a continuación.

Funcionalidad.	Facilidad de uso.	Fiabilidad.	Rendimiento.	Capacidad de soporte.
Se obtiene mediante la evaluación del conjunto de características y de posibilidades del programa, la generalidad de la funciones que se entregan y la seguridad de todo el sistema.	Se calcula considerando los factores humanos, la estética global, la consistencia y la documentación.	Se calcula midiendo la frecuencia de fallos y su importancia, la eficacia de los resultados de salida, el tiempo medio entre fallas, la posibilidad de recuperarse a los fallos y la previsibilidad del programa.	Se mide mediante la evaluación de la velocidad de proceso, el tiempo de respuesta, el consumo de recursos, el rendimiento total de procesamiento y la eficiencia.	Combina la posibilidad de ampliar el programa, la adaptabilidad y la utilidad, además de la facilidad de prueba, la compatibilidad, la posibilidad de configuración, la facilidad con la que se puede instalar un sistema y la facilidad con la que se pueden localizar los problemas.

ANEXO 3

Ejemplo de actividades en una estrategia de pruebas al software, según Pressman.

Prueba de unidad.	Prueba de integración.	Prueba de validación.	Prueba del sistema.
Primeramente la prueba se centra en cada módulo individual, asegurando que funcionan adecuadamente como una unidad. La prueba de unidad hace un uso intensivo de las técnicas de prueba de la caja negra, ejercitando caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores. Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad del programa que está siendo probada. Se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente conservan su integridad durante todos los pasos de ejecución del algoritmo. Se prueban las condiciones límite para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento. Se ejercitan todos los caminos independientes de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Finalmente, se prueban todos los caminos de manejo de errores.	Después se deben ensamblar o integrar los módulos para formar el paquete de software completo. La prueba de integración se dirige a todos los aspectos asociados con el doble problema de verificación y de construcción del programa. Durante la integración, las técnicas que más prevalecen son las de diseño de casos de prueba de la caja negra, aunque se pueden llevar a cabo unas pocas pruebas de la caja blanca con el fin de asegurar que se cubren los principales caminos de control. La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados en unidad y construir una estructura de programa que este de acuerdo con lo que dicta el diseño. Integración descendente: Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando con el módulo de control principal. Los módulos subordinados al módulo de control principal se van incorporando en la estructura, ya sea en profundidad o en anchura. Integración ascendente: Empieza la construcción y la prueba con los módulos atómicos o módulos de los niveles más bajos, después se continua con el nivel inmediato superior y así sucesivamente.	Tras la culminación de la prueba de integración, el software está completamente ensamblado como un paquete, se han encontrado y corregido los errores de interfaz y puede comenzar la prueba de validación. Por lo tanto, una vez que el software se ha integrado, se dirigen un conjunto de pruebas de alto nivel. Se deben comprobar los criterios de validación establecidos durante el análisis de requisitos. La prueba de validación proporciona una seguridad final de que el software satisface todos los requisitos funcionales, de comportamiento y de rendimiento. Durante la prueba de validación se usan exclusivamente técnicas de prueba de la caja negra. La validación se logra cuando el software funciona de acuerdo con las expectativas razonables del cliente, por lo tanto se consigue mediante una serie de pruebas de la caja negra que demuestran la conformidad con los requisitos.	Y finalmente, la prueba del sistema verifica que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total. La prueba del sistema está constituida por una serie de pruebas diferentes, cuyo propósito primordial es ejercitar profundamente el sistema de software. Aunque cada prueba tiene un propósito distinto, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas. La primera es la prueba de recuperación, la cual fuerza el fallo del software de muchas formas y verifica que la recuperación se lleve a cabo adecuadamente. La segunda es la prueba de seguridad, la cual intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de posibles accesos no permitidos. La tercera es la prueba de resistencia, la cual esta diseñada para enfrentar a los programas con situaciones anormales. La cuarta y última es la prueba de rendimiento, la cual está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

ANEXO 4.

ISO 9000-3. Estándares para la gestión y el aseguramiento de calidad en el desarrollo, suministro y mantenimiento del software.

Estándares para el aseguramiento de calidad relacionados con la revisión del contrato, según la norma ISO 9000-3

Revisión del contrato.
Cada contrato debe ser revisado por el desarrollador para asegurar que:
El alcance del contrato y sus requerimientos estén definidos y documentados;
Posibles contingencias o riesgos estén identificados;
La información del propietario esté adecuadamente protegida;
Cualquier requerimiento que difiera de los ofrecidos sea resuelto;
El desarrollador tiene la capacidad para cumplir con los requerimientos contractuales;
La responsabilidad del usuario en relación a trabajo subcontratado esté definida;
La terminología sea aceptada por ambas partes;
Y que el cliente tenga la capacidad para cumplir con las obligaciones contractuales.
Los registros de tales revisiones contractuales deben ser guardados.
Artículos sobre calidad que debe contener el contrato:
Criterios de aceptación;
Manejo de los cambios en los requerimientos del cliente durante el desarrollo;
Manejo de los problemas detectados después de la aceptación, incluyendo quejas relacionadas con la calidad;
Actividades a cargo del cliente, especialmente en la especificación de requerimientos, instalación y aceptación; Facilidades, herramientas y artículos de software que serán proveídos por el cliente;
Estándares y procedimientos a ser usados;
Requerimientos de copias.

Estándares para el aseguramiento de calidad relacionados con la especificación de requerimientos del cliente, según la norma ISO 9000-3

Especificación de requerimientos del cliente.
El desarrollador deberá tener un conjunto de requerimientos funcionales completo y sin ambigüedades, debiendo incluir todos los aspectos necesarios para satisfacer las necesidades del cliente (rendimiento, seguridad, confiabilidad, etc.). Se deberán especificarse completamente todas las interfaces entre el producto de software y otros productos de software o hardware.
Durante la especificación de requerimientos del cliente se recomiendan poner atención en los siguientes aspectos:
Asignación de personal de ambos lados responsables de establecer las especificaciones de los requerimientos del cliente; Métodos para acordar los requerimientos y aprobación de cambios; Esfuerzo para prevenir malos entendidos; Registro y revisión de los resultados de reuniones de ambos lados.

Estándares para el aseguramiento de calidad relacionados con el plan de desarrollo, según la norma ISO 9000-3

Planeación del desarrollo:
El plan de desarrollo deberá cubrir lo siguiente:
La definición del proyecto, incluyendo una declaración de objetivos; La organización de los recursos del proyecto, incluyendo la estructura del equipo, responsabilidades, uso de subcontratistas y recursos materiales a ser usados.; Las fases de desarrollo; Identificar en el plan del proyecto las tareas que serán realizadas, los recursos y el tiempo requerido cada una y cualquier interrelación entre tareas; Identificación de planes relacionados, tales como: plan de calidad, plan de gestión de la configuración, plan de integración, plan de pruebas.
El plan de desarrollo deberá ser actualizado conforme avanza el proyecto y cada fase deberá estar definida antes de iniciarla, además deberá estar revisada y aprobada antes de ejecutarla

Plan de desarrollo;
El plan de desarrollo deberá definir una metodología para transformar la especificación de los requerimientos especificados del cliente en un producto de software, esto puede envolver dividir el trabajo en fases y la identificación de: Fases de desarrollo que serán generadas; entradas y salidas requeridas para cada fase; procedimientos de verificación que deberán ser generados para cada fase; Análisis de problemas potenciales asociados con las fases de desarrollo y con la realización de requerimientos específicos.
El plan de desarrollo deberá definir como será gestionado, incluyendo la identificación de:
Plan de desarrollo, implementación y entregas asociadas; Control del avance; responsabilidades organizacionales, recursos y asignación de trabajos; Interfaces técnicas y organizacionales entre los diferentes grupos.
Métodos de desarrollo y herramientas;
El plan de desarrollo deberá identificar los métodos para asegurar que todas las actividades son generadas correctamente, esto puede incluir: Reglas, prácticas y convenciones para el desarrollo; herramientas y técnicas de desarrollo; gestión de la configuración.
Control del avance;
Las revisiones del avance deberán ser planeadas, mantenidas y documentadas para asegurar que los aspectos sobresalientes de recursos son resueltos y asegurar una ejecución efectiva de los planes de desarrollo
Entradas a las fases de desarrollo;
Las entradas de cada fase de desarrollo deberán ser definidas y documentadas. Cada requerimiento deberá ser definido de tal forma que su alcance pueda ser verificado. Los requerimientos incompletos, ambiguos o conflictivos deberán ser resueltos con los responsables respectivos.
Salidas de las fases de desarrollo;
La salida requerida en cada fase de desarrollo deberá ser definida y documentada. La salida de cada fase de desarrollo deberá ser verificada y deberá: Cumplir con los requerimientos relevantes; contener o hacer referencia a criterios de aceptación para avanzar a las fases subsecuentes; Estar conforme a las convenciones y prácticas de desarrollo apropiadas, sea que éstas hayan o no sido establecidas en la información de entrada; Identificar aquellas características del producto que son cruciales a la seguridad y funcionamiento propio; Estar conforme a los requerimientos regulatorios aplicables
Verificación de cada fase;
El desarrollador deberá redactar un plan de verificación de todas las salidas de las fases de desarrollo al final de cada una de ellas. La verificación del desarrollo deberá establecer que las salidas de la fase de desarrollo cumplen con los requerimientos de entrada correspondientes por medio del desarrollo de medidas de control como:
Mantener revisiones del desarrollo en puntos apropiados en las fases de desarrollo; Comparar un nuevo diseño con un diseño similar probado, si éste está disponible; Efectuar pruebas y demostraciones. Los resultados de la verificación y cualesquier otras acciones requeridas para asegurar que los requerimientos especificados son alcanzados deberán ser registrados y checados cuando las acciones sean terminadas. Solamente deberán ser enviadas a la gestión de la configuración las salidas verificadas y aceptadas para un uso subsecuente.

Estándares para el aseguramiento de calidad relacionados con el plan de calidad, según la norma ISO 9000-3 .

Planeación de la calidad.
Como parte de la planeación del desarrollo, el desarrollador deberá preparar un plan de calidad. El plan de calidad deberá ser actualizado a lo largo del avance del desarrollo y las partes concernientes a cada fase deberán ser completamente definidas al inicio de cada fase. El plan de calidad debe ser formalmente revisado y aceptado por todos los involucrados dentro de la organización.
Contenido del plan de calidad.
El plan de calidad deberá especificar o hacer referencia a los siguientes componentes:
Objetivos de calidad, expresados en términos medibles siempre que sea posible; Definir criterios de entrada y de salida para cada fase de desarrollo; Identificación de tipos de pruebas, verificación y actividades de validación a ser generadas; Planeación detallada de pruebas, verificaciones y actividades de validación a ser generadas, incluyendo planes, recursos y aprobación de autoridades; Especificar responsabilidades para actividades de calidad, tales como revisiones y pruebas, gestión de configuración y control de cambios, control de defectos y acciones correctivas. .

Estándares para el aseguramiento de calidad relacionados con el diseño y la implementación, según la norma ISO 9000-3

Diseño e implementación.

Las actividades de diseño e implementación son aquellas que transforman la especificación de requerimientos del cliente en un producto de software, las cuales deben ser generadas en una forma disciplinada, con el fin de producir un producto que este acorde a las especificaciones más que depender en actividades de prueba y validación para asegurar la calidad. El nivel de presentación de la información que será entregada al cliente requiere ser mutuamente acordada por ambas partes, ya que los procesos de diseño e implementación frecuentemente son propiedad del desarrollador.

Adicionalmente a los requerimientos comunes para todas las fases de desarrollo, los siguientes aspectos inherentes a las actividades de diseño deberán ser tomadas en cuenta:

Diseño:

a).- Identificación de consideraciones de diseño: adicionalmente a las especificaciones de entrada y salida, aspectos tales como reglas de diseño y definiciones de interfaces internas deberán ser examinadas;

b).-Metodología de diseño: deberá ser usada una metodología de diseño sistemática apropiada para el tipo de producto de software que está siendo desarrollado;

c).- Uso de experiencias de diseño, pasadas: utilizando lecciones aprendidas de experiencias de diseño pasadas, el desarrollador deberá evitar le recurrencia de los mismos o similares problemas;

d).- Proceso subsecuente: el producto deberá ser diseñado para facilitar las pruebas, mantenimientos y usos

Implementación.

Adicionalmente a todos los requerimientos comunes a las actividades de desarrollo, los siguientes aspectos deberán ser considerados en cada actividad de implementación.

a).- Reglas: reglas como de programación, lenguajes de programación, convenciones para la asignación consistente de nombres, codificación y reglas adecuadas para los comentarios deberán ser especificadas y observadas;

b).- Metodologías de implementación: el desarrollador deberá usar métodos apropiados de implementación y herramientas para satisfacer los requerimientos del cliente.

Revisiones:

El desarrollador deberá llevar a cabo revisiones para asegurar que los requerimientos son alcanzados y los métodos antes mencionados son correctamente aplicados. Ni el diseño ni el proceso de implementación deberán continuar hasta que todas las deficiencias conocidas sean satisfactoriamente resueltas o bien sea conocido el riesgo de continuar. Se deberán mantener registros de tales revisiones.

Estándares para el aseguramiento de calidad relacionados con las pruebas y validaciones, según la norma ISO 9000-3.

Pruebas y validaciones.

Las pruebas pueden ser requeridas en diferentes niveles desde elementos individuales de software hasta el producto completo de software. Existen varios caminos diferentes para realizar las pruebas y la integración. En algunos casos, la validación, la prueba de campo y las pruebas de aceptación pueden ser una misma actividad. El documento que describe el plan de pruebas puede ser un documento independiente o una parte de otro documento o puede estar compuesto de diferentes documentos.

Planeación de las pruebas.

El desarrollador debe establecer y revisar los planes de pruebas, especificaciones y procedimientos antes de comenzar las actividades de pruebas. Se debe dar consideración a:

a).- Planes para partes del software, integración, pruebas del sistema y pruebas de aceptación;

b).- Casos de prueba, datos de prueba y resultados esperados;

c).- Tipos de pruebas a ser ejecutadas, por ejemplo pruebas funcionales, pruebas de frontera, pruebas de rendimiento, pruebas de usabilidad;

d).- Medio ambiente de pruebas, herramientas y software de pruebas;

e).- Los criterios por medio de los cuales será juzgada la terminación de las pruebas;

f).- La documentación del usuario;
g).- El personal requerido y los requerimientos de entrenamiento asociados;
Pruebas.
Se deberá poner especial atención en los siguientes aspectos de las pruebas:
a).- Los resultados de las pruebas deberán ser registrados como se define en la especificación;
b).- Los problemas descubiertos y sus posibles impactos en otras partes del software deberán ser anotadas y notificadas a los responsables y así los problemas puedan ser rastreados hasta que queden resueltos;
c).- Las áreas impactadas por cualquier modificación deberán ser identificadas y reaprobadas;
d).- Se deberán evaluar las pruebas de adecuación y relevancia;
e).- La configuración del hardware y del software deberán ser documentadas.
Validación.
Antes de poner el producto listo para su liberación y aceptación del cliente, el desarrollador deberá validar su operación como un producto completo, cuando sea posible bajo condiciones similares al medio ambiente de la aplicación como se especifico en el contrato.
Pruebas de campo.
Donde se requieran pruebas bajo condiciones de campo, se deberán tomar en cuenta los siguientes aspectos:
a).- Las características a ser probadas en el medio ambiente de campo;
b).- Las responsabilidades específicas del desarrollador y del cliente para realizar y evaluar la prueba;
c).- La restauración del medio ambiente del usuario, después de la prueba.

Estándares para el aseguramiento de calidad relacionados con la aceptación, según la norma ISO 9000-3.

Aceptación.
Cuando el desarrollador está listo para liberar el producto validado, el cliente deberá juzgar si el producto es aceptable o no de acuerdo a los criterios aprobados con anterioridad en la forma especificada dentro del contrato. El método de manejar los problemas detectados durante el procedimiento de aceptación y su disposición deberá ser acordado entre el cliente y el desarrollador y deberá estar documentado.
Planeación de la prueba de aceptación.
Antes de realizar las actividades de aceptación, el desarrollador deberá asistir al cliente para identificar los siguiente:
a).- Tiempo programado;
b).- Procedimiento de evaluación;
c).- Recursos y medio ambiente de hardware y de software;
d).- Criterio de aceptación.

Estándares para el aseguramiento de calidad relacionados con la replicación, liberación e instalación, según la norma ISO 9000-3.

Replicación, liberación e instalación.
Replicación.
La replicación es un paso que deberá ser realizado antes de la liberación. Al proveer la replicación, se deberá considerar lo siguiente:
a).- El número de copias de cada artículo de software que será liberado;
b).- El tipo de medio para cada artículo de software, incluyendo formato, versión y forma de lectura;
c).- Derechos de copia y licencias acordadas;
e).- Custodia de copias de respaldo y copias maestras, donde sea aplicable, incluyendo planes de recuperación de desastres;
f).- El período de obligación del desarrollador para suministrar las copias.
Liberación.
Se deberán hacer provisiones para verificar que las copias del producto de software liberado estén correctas y completas.
Instalación.
Los roles, responsabilidades y obligaciones del desarrollador y el cliente deberán de ser claramente establecidas, tomando en cuenta los siguiente:

a).- Horario, incluyendo horas de trabajo fuera de lo normal y fines de semana;
b).- Accesos del cliente, facilitando señales de seguridad, claves de acceso, resguardos;
c).- Disponibilidad de personal calificado;
d).- Disponibilidad y acceso al equipo y los sistemas del cliente.
e).- La necesidad de validar como parte de cada instalación debiendo estar determinada contractualmente;
f).- Un procedimiento formal de aprobación para cada instalación realizada.

Estándares para el aseguramiento de calidad relacionados con el mantenimiento, según la norma ISO 9000-3

Mantenimiento.

Cuando el mantenimiento del producto sea requerido por el cliente, después de la instalación y liberación inicial. Esto deberá estar estipulado en el contrato. El desarrollador deberá establecer y mantener los procedimientos para ejecutar actividades de mantenimiento y verificación y verificar que tales actividades cumplan con los requerimientos especificados para el mantenimiento.

Las actividades de mantenimiento para productos de software típicamente están clasificados dentro de los siguientes:

a).- Resolución de problemas;
b).- Modificación de interfaces;
c).- Expansión funcional o mejoras al rendimiento.

Los artículos a ser mantenidos, así como el periodo de tiempo por el cual deberán ser mantenidos, deberá estar especificado en el contrato. Los siguientes son ejemplos de tales artículos:

a).- Programas;
b).- Los datos y sus estructuras;
c).- Las especificaciones;
d).- La documentación para el desarrollador y/o usuario;
e).- La documentación para el uso del desarrollador.

Plan de mantenimiento.

Todas las actividades de mantenimiento deberán ser realizadas y gestionadas de acuerdo al plan de mantenimiento definido y acordado previamente por el desarrollador y el cliente. El plan deberá incluir lo siguiente:

a).- El alcance del mantenimiento;
b).- Identificación del estado inicial del producto;
c).- Organización del soporte;
d).- Actividades de mantenimiento;
e).- Mantenimiento de registros y reportes.

Identificación del estado inicial del producto.

El estado inicial del producto a ser mantenido deberá estar definido, documentado y acordado tanto por el cliente como por el desarrollador.

Organización del soporte.

Puede ser necesario establecer una organización con representantes tanto del cliente como del desarrollador, para soportar las actividades de mantenimiento. Puesto que las actividades en la etapa de mantenimiento no siempre pueden ser efectuadas de forma programada, esta organización deberá ser lo suficientemente flexibles para arreglárselas con la ocurrencia inesperada de problemas. Puede ser necesario identificar facilidades y recursos para ser usados para actividades de mantenimiento.

Tipos de actividades de mantenimiento.

Todos los cambios al software (por razones de resolución de problemas, modificaciones a las interfaces, expansiones funcionales o mejoras al rendimiento) realizadas durante el mantenimiento deberán ser hechas de acuerdo con los mismos procedimientos, tanto como sea posible, usados para el desarrollo de productos de software. Todos los cambios deberán ser documentados de acuerdo con los procedimientos para el control de documentos y gestión de la configuración.

a).- Resolución de problemas: La resolución de problemas envuelve la detección, análisis y corrección de incorformidades del software que causan problemas operacionales. Cuando se resuelven los problemas, se pueden hacer reparaciones temporales para minimizar interrupciones al sistema y efectuar modificaciones permanentes posteriormente.

b).- Modificaciones a interfaces: Modificaciones a las interfaces pueden ser requeridas cuando sean hechos cambios o adiciones al sistema de hardware o componentes controlados por software.

c).- Expansión funcional o mejoras del rendimiento: La expansión funcional o mejoras del rendimiento de funciones existentes pueden ser requeridas por el cliente en la etapa de mantenimiento.
Registros de mantenimiento y reportes.
Todas las actividades de mantenimiento deberán ser registradas y conservadas en formatos predefinidos. Las reglas para presentar los reportes de mantenimiento deberán ser establecidas y acordadas por el cliente y el desarrollador.
Los registros de mantenimiento deberán incluir los siguientes apartados de cada elemento de software que es mantenido:
a).- Lista de peticiones para asistencia o reportes de problemas que han sido recibidos y el estado actual de cada uno de ellos;
b).- La organización responsable de responder a las peticiones de asistencia o implementar las acciones correctivas apropiadas;
c).- Prioridades que han sido asignadas a acciones correctivas;
d).- Resultados de las acciones correctivas;
e).- Datos estadísticos acerca de la ocurrencia de fallas y actividades de mantenimiento.
El registro de las actividades de mantenimiento puede ser utilizado para la evaluación y mejora del producto de software y para mejorar la calidad del sistema en si misma..
Procedimiento de liberación.
El desarrollador y el cliente deberán acordar y documentar procedimientos para incorporar cambios en un producto de software como resultado de la necesidad de mantener el rendimiento.
Estos procedimientos deberán incluir lo siguiente:
a).- Reglas para determinar dónde pueden ser incorporados los "parches" localizados o liberar una copia actualizada completa del producto de software si esto es necesario;
b).- Descripciones de los tipos o clases de liberaciones dependiendo de su frecuencia y/o impacto en las operaciones del cliente y la habilidad de implementar cambios en cualquier instante de tiempo;
c).- Métodos por los cuales el cliente debe ser avisado de cambios presentes o de futuros cambios planeados;
d).- Métodos para confirmar que los cambios implementados no introducirán otros problemas;
e).- Requerimientos para registros, indicando que los cambios han sido implementados y en qué ubicación, para múltiples productos e instalaciones.

A continuación se presentan otra serie de estándares relacionadas con actividades de soporte, las cuales no son dependientes de las fases del ciclo de desarrollo.

Estándares para el aseguramiento de calidad relacionados con la gestión de la configuración, según la norma ISO 9000-3

Gestión de la configuración.
La gestión de la configuración proporciona un mecanismo para identificar, controlar y dar seguimiento a las versiones de cada elemento de software. En muchos casos versiones iniciales aún en uso deben ser mantenidas y controladas.
La gestión de configuración deberá:
a).- Identificar únicamente las versiones de cada elemento de software;
b).- Identificar las versiones de cada elemento de software el cual constituya en conjunto una versión específica de un producto completo;
c).- Identificar el estado de construcción de los productos de software en desarrollo o liberados e instalados;
d).- Controlar simultáneamente la actualización de un elemento de software dado por más de una persona;
e).- Proporcionar la coordinación para la actualización de múltiples productos en uno o más lugares conforme sea requerido;
f).- Identificar y dar seguimiento a todas las acciones y cambios resultantes de una solicitud de cambio, desde la iniciación hasta su liberación..
Plan de la gestión de la configuración.
El grupo desarrollador debe desarrollar e implantar un plan de gestión de la configuración el cual incluya lo siguiente:
a).- Organizaciones envueltas en la gestión de la configuración y responsabilidades asignadas a cada una de ellas;
b).- Actividades de gestión de la configuración por ser realizadas;
c).- Herramientas para la gestión de la configuración, técnicas y metodologías a ser usadas
d).- La etapa en la cual los elementos deben ser tomados bajo el control de la configuración.
Actividades de la gestión de la configuración.

Identificación de la configuración y seguimiento.
El desarrollador deberá establecer y mantener procedimientos para identificar los elementos de software durante todas las fases, comenzando desde la especificación hasta el desarrollo, replicación y liberación. Donde sea requerido por el contrato, estos procedimientos pueden también aplicar después de la liberación de los productos. Cada elemento individual del software deberá de tener una identificación única.
Los procedimientos deberán ser aplicados para asegurar que lo siguiente puede ser identificado para cada versión de un elemento de software:
a).- Las especificaciones funcionales y técnicas;
b).- Todas las herramientas de desarrollo que afectan las especificaciones técnicas y funcionales.
c).- Todos los documentos y archivos de computadora relacionados con los elementos del software.
La identificación de los elementos de software deben ser manejados de tal forma que las relaciones entre los elementos y los requerimientos del contrato puedan ser demostrados.
Para productos liberados, deben existir procedimientos que faciliten el seguimiento de los elementos de software o el producto.
Control de cambios.
El desarrollador debe establecer y mantener procedimientos para identificar, documentar, revisar y actualizar cualquier cambio a los elementos de software bajo la gestión de la configuración. Todos los cambios de los elementos del software deberán ser realizados en conformidad con estos procedimientos.
Antes de que sea aceptado un cambio, su validez deberá ser confirmada y los efectos en otros elementos deberán ser identificados y examinados.
Se deben proporcionar los métodos para notificar los cambios a todos aquellos involucrados y mostrarles los seguimientos entre los cambios y las partes modificadas de los elementos del software.
Reporte de estado de la configuración.
El desarrollador deberá establecer y mantener los procedimientos para registrar, administrar y reportar el estado de los elementos de software, solicitudes de cambio y de la implementación de los cambios aprobados.

Estándares para el aseguramiento de calidad relacionados con el control de documentos, según la norma ISO 9000-3

Control de documentos.
El desarrollador deberá establecer y mantener procedimientos para controlar todos los documentos que se relacionen con esta parte de la ISO 9000, esto cubre:
a).- La determinación de aquellos documentos que deberán ser sujetos a los procedimientos de control de documentos;
b).- La aprobación y emisión de procedimientos.
c).- El cambio de procedimientos incluyendo su retiro y , cuando sea apropiado, su liberación.
Tipos de documentos.
Los procedimientos para el control de procedimientos deberán ser aplicados con los documentos relevantes, incluyendo los siguientes:
a).- Documentos procedurales, que describan el sistema de calidad que será aplicado en el ciclo de vida del software;
b).- Documentos de planeación, que describan la planeación y el progreso de todas las actividades del proveedor y su interacción con el cliente;
b).- Documentos de productos, que describan un producto de software en particular, incluyendo:
- Entradas a la fase de desarrollo.
- Salidas de la fase de desarrollo
- Planes de verificación y validación y resultados.
- Documentación para el cliente y el usuario.
- Documentación de mantenimiento.
Emisión y aprobación de documentos.
Todos loa documentos deberán ser revisados por el personal autorizado antes de su emisión. Deben existir procedimientos que aseguren que:
a).- La emisión pertinente de los documentos apropiados está disponible en ubicaciones apropiadas donde son ejecutadas operaciones elementales para el funcionamiento efectivo del sistema de calidad.
b).- Documentos obsoletos sean oportunamente removidos de los lugares apropiados de emisión o de uso.

Donde se haga uso de archivos de computadora, se tienen que poner especial atención en los procedimientos para la aprobación correcta, acceso, distribución y archivamiento.
Cambios a los documentos.
Los cambios en los documentos deberán ser revisados y aprobados por la misma gente dentro de la organización que ejecutó la revisión original y aprobación a menos que de otra forma se designe específicamente. La gente designada deberá ser identificada en el documento o anexos apropiados.
Una lista maestra o un procedimiento para el control de documentos deberá ser establecido para identificar la versión actual de los documentos con el fin de excluir el uso de documentos no aplicables.
Los documentos deberán ser remitidos después de un número práctico de cambios que se les hayan hecho.

Estándares para el aseguramiento de calidad relacionados con el registro de calidad, según la norma ISO 9000-3

Registros de calidad.
El desarrollador deberá establecer y mantener procedimientos de identificación, recolección, indexamiento, llenado, almacenamiento, mantenimiento y disposición de registros de calidad.
Los registros de calidad deberán ser mantenidos para demostrar la realización de la calidad requerida y la efectiva operación del sistema de calidad. Deberán ser un elemento de estos datos los registros de calidad de los subcontratistas pertinentes.
Todos los registros de calidad deberán ser legibles e identificables con el producto envuelto. Los cuales deberán ser almacenados y mantenidos en tal forma que sean oportunamente recuperables en condiciones que proporcionen un medio ambiente apropiado para minimizar su deterioro o daño y prevenir pérdidas. Se deberán establecer y registrar tiempos de retención de los registros de calidad. Cuando se acuerde contractualmente, los registros de calidad deberán estar a disposición del cliente o su representante para su evaluación por un periodo convenido.

Estándares para el aseguramiento de calidad relacionados con la medición, según la norma ISO 9000-3

Medición.
Medición de productos.
Se deben reportar y usar mediciones para gestionar el desarrollo y liberación de procesos y deben ser relevantes para un producto de software en particular.
No existen en la actualidad medidas universales aceptadas de la calidad del software. Sin embargo, por lo menos algunas medidas deben ser usadas que representen fallas de campo reportadas y/o defectos desde el punto de vista del cliente. Se deben describir medidas seleccionadas de tal forma que los resultados sean comparables.
El desarrollador de los productos de software deberá recoger y actuar sobre las medidas cuantitativas de la calidad de estos productos de software. Estas medidas deben ser utilizadas para los siguientes propósitos:
a).- Para recolectar datos y reportar valores medidos sobre una base regular;
b).- Para identificar el nivel actual de rendimiento sobre cada medida;
c).- Para tomar alguna acción de remedio si el nivel medido se pone peor o excede los niveles mínimos establecidos.
d).- Para establecer metas de mejora específica en términos de sus medidas.
Medición de procesos.
El desarrollador deberá tener medidas cuantitativas para la calidad del desarrollo de los procesos liberados. Estas medidas deberán reflejar:
a).- Que tan bien el proceso de desarrollo se esta realizando en términos de que los objetivos del programa sean cumplidos.
b).- Que tan efectivo es el proceso de desarrollo al reducir la probabilidad de fallas internas o que cualquier falla interna esté sin detectarse.
Aquí, como en la medición de productos, la cosa importante es que los niveles asean conocidos y usados para el control de procesos y mejoración y no que métrica especial es usada. La selección de la métrica deberá encajar con el proceso que está siendo usado y, si es posible, tener un impacto directo en la calidad del software liberado. Pueden ser apropiadas diferentes métricas para diferentes productos de software producidos por el mismo desarrollador.

Estándares para el aseguramiento de calidad relacionados con las reglas, prácticas y convenciones, según la norma ISO 9000-3

Reglas, prácticas y convenciones.
El desarrollador deberá proporcionar reglas, prácticas y convenciones con el fin de hacer el sistema de calidad especificado.
El desarrollador deberá revisar estas reglas, prácticas y convenciones como se requiera.

Estándares para el aseguramiento de calidad relacionados con las técnicas y herramientas, según la norma ISO 9000-3

Técnicas y herramientas.
El desarrollador deberá usar herramientas, facilidades y técnicas con el fin de hacer efectiva esta guía del sistema de calidad. Éstas pueden ser efectivas para propósitos de gestión, así como para el desarrollo de productos.
El desarrollador deberá mejorar estas herramientas y técnicas como se vayan requiriendo.

Estándares para el aseguramiento de calidad relacionados con una adquisición, según la norma ISO 9000-3

Adquisición.
El desarrollador debe asegurar que un producto adquirido o servicio cumple con los requerimientos especificados.
Los documentos de adquisición deberán contener claramente los datos que describan el producto o servicio ordenado. El desarrollador debe revisar y aprobar los documentos de la adquisición para la adecuación de los requerimientos especificados antes de la liberación.
Un producto adquirido puede ser un elemento de software o hardware destinado para ser incluido en el producto final requerido o una herramienta destinada a asistir en el desarrollo de un producto de software.
Involucración de subcontratistas.
Un desarrollador deberá seleccionar subcontratistas sobre la base de su habilidad para cumplir con los requerimientos del subcontrato, incluyendo los requerimientos de calidad. El desarrollador deberá establecer y mantener registros de los subcontratistas aceptables.
La selección del subcontratistas y el tipo y la excepción del control ejercido por el desarrollador deberá ser dependiente del tipo de producto y, donde sea apropiado, en registros de subcontratistas previamente demostrada su capacidad y rendimiento.
El desarrollador deberá asegurarse de que sean efectivos los controles del sistema de calidad.
Validación del producto adquirido.
El desarrollador es responsable de la validación del trabajo subcontratado. Esto puede requerir que el desarrollador conduzca el diseño y otras revisiones en línea con el propio sistema de calidad del desarrollador y así tales requerimientos deberán estar incluidos en el subcontrato. Cualesquier requerimiento que requiera pruebas de aceptación, deberá ser similarmente incluido.
Cuando sea especificado en el contrato, el cliente o su representante deberán ser provistos del derecho a determinar en el inicio o al recibir, que el producto adquirido cumple con los requerimientos especificados. La validación del cliente puede no absolver al desarrollador de la responsabilidad de proporcionar un producto aceptable o no puede excluir un rechazo subsecuente.
Cuando el cliente o su representante elige realizar validación en las instalaciones del subcontratista, dicha validación no deberá ser por el desarrollador como evidencia de un control efectivo de calidad.

Estándares para el aseguramiento de calidad relacionados con la inclusión de productos de software, según la norma ISO 9000-3

Inclusión de productos de software.
Se le puede requerir al proveedor incluir o usar un producto de software provisto por el cliente o por una tercera persona. El desarrollador deberá establecer y mantener procedimientos de validación, almacenamiento, protección y mantenimiento de dicho producto. Se deberá dar consideración al soporte de tal producto de software en cualquier acuerdo de mantenimiento relacionado con el producto que será liberado.
El producto provisto por el cliente que se encuentre no apropiado para usarse deberá ser registrado y reportado al cliente. La validación del desarrollador no absuelve al cliente de la responsabilidad de la responsabilidad de proporcionar un producto aceptable.

Estándares para el aseguramiento de calidad relacionados con el entrenamiento, según la norma ISO 9000-3

Entrenamiento.
El desarrollador deberá establecer y mantener procedimientos para identificar las necesidades de entrenamiento y proporcionar el entrenamiento a todo el personal que realiza actividades que afecten la calidad. El personal que realiza tareas asignadas específicas deberá estar calificado en base a la apropiada educación, entrenamiento y/o experiencia, como se requiera.
Se deberán determinar la calificación de los sujetos considerando las herramientas específicas, técnicas, metodologías y recursos de cómputo que será usados en el desarrollo y gestión de productos de software. Puede también requerirse incluir el entrenamiento de habilidades y conocimientos de campos específicos en lo que el software trata.
Se deberán mantener registros apropiados en lo referente a entrenamiento/experiencia.

ANEXO 5
Ejemplos de formatos

Un ejemplo de un formato de reporte de una revisión:

FRR _____

FORMATO DE REPORTE DE UNA REVISIÓN

IDENTIFICACIÓN DE LA REVISIÓN:

Nombre del proyecto: _____
Número de control: _____
Fecha de realización: _____
Lugar y hora: _____

IDENTIFICACIÓN DEL PRODUCTO QUE SE REVISA:

Nombre del material: _____
Descripción: _____
_____
Nombre del productor: _____
Nombre del elemento revisado: _____

INTEGRANTES DEL GRUPO REVISOR:

	Nombre	Firma
Jefe de la revisión:		
Secretario de la revisión:		
Miembros participantes:		

RESULTADO DE LA REVISIÓN:

Aprobado totalmente: ☐ Aprobado parcialmente: ☐
No aprobado: ☐
Revisión no concluida: ☐
Comentarios: _____
_____
_____
Documentación que se anexa:

Lista de sucesos: ☐
Otra: (especificar) _____

Un formato de una lista de sucesos sería por ejemplo:

FLS _____

FORMATO DE UNA LISTA DE SUCESOS

IDENTIFICACIÓN DE LA LISTA DE SUCESOS:

Número de control de la revisión: _____
Fecha de la revisión: _____
Nombre del jefe de la revisión: _____
Nombre del secretario de la revisión: _____

LISTA DE SUCESOS:

Descripción del suceso 1: _____

Descripción del suceso 2: _____

Descripción del suceso 3: _____

Firma del Jefe de la revisión. _____

Firma del Secretario de la revisión. _____

Un ejemplo de un reporte de falla o de error se muestra a continuación:

FRF

FORMATO DE REPORTE DE FALLA

IDENTIFICACIÓN DEL REPORTE:

Descripción del reporte: _____

Fecha y hora: _____

Nombre de la prueba: _____

Nombre del elemento que fallo: _____

DESCRIPCIÓN DEL PROBLEMA ENCONTRADO:

_____

_____

_____

_____

_____

ACCIONES DE SOLUCIÓN RECOMENDADAS:

_____

_____

_____

_____

_____

INFORMACIÓN DE CONTROL:

Nombre de la persona que levanta el reporte y fecha: _____

_____

Nombre de la persona a la que se le asigna y fecha: _____

_____

Prioridad asignada: _____

Número de control de la autorización del cambio: _____

Resultado de la solicitud de cambio:

Aprobada: ☐ No aprobada: ☐ Diferida: ☐

Nombres y firmas de las personas que examinaron y dictaminaron;

_____

_____

_____

_____

Fecha en que se dictaminó el resultado: _____

Disposición y acciones a realizar: _____

_____

_____

INFORMACIÓN ADICIONAL:

Documentos de solicitud de cambio: _____

Sujeto a la orden de cambio del software con número: _____

Relacionada con las órdenes de cambio números: _____

Un formato de una orden de cambio sería por ejemplo:

FOC _____

FORMATO DE UNA ORDEN DE CAMBIO

INFORMACIÓN DE LA IDENTIFICACIÓN DEL CAMBIO:

Nombre de la persona que lo solicita: _____
Número del reporte de falla de referencia: _____
Nombre del software afectado: _____
Nombres de los elementos del software: _____
Números de versiones: _____
Números de identificación asignados: _____

RESUMEN DEL CAMBIO Y RAZÓN PARA HACERLO: _____
_____
_____

ACCIONES REQUERIDAS EN:

Elementos de software: _____
Documentación: _____
Pruebas: _____

RELACIÓN CON OTROS CAMBIOS:

Impacto en: _____
Plan: _____
Especificaciones del cliente: _____
Subcontrataciones: _____
Pruebas del software: _____
Herramientas del software, etc.: _____
_____

INFORMACIÓN SOBRE EL RESULTADO DE LA ORDEN DEL CAMBIO DEL SOFTWARE:

Aprobada: ☐ No aprobada: ☐ Diferida: ☐
Nombres y firmas de las personas que participaron: _____
_____
_____
_____
Fecha: _____

RESUMEN DEL CAMBIO E IMPACTO: _____
_____
_____
_____
_____

ANEXO 6
Ejemplos de procedimientos.

Ejemplo de un procedimiento para llevar a cabo una reunión:

- El individuo que ha desarrollado el producto informa al jefe del proyecto de que el producto está terminado y que se requiere una revisión.
- El jefe del proyecto contacta con un jefe de revisión, que es el que evalúa la disponibilidad del producto, genera copias del material del producto y las distribuye a los revisores para que se preparen por adelantado.
- Cada revisor deberá tomar el tiempo necesario para revisar el producto, tomando notas y también familiarizándose con el trabajo. De forma concurrente, también el jefe de revisión revisa el producto y establece una agenda para la reunión de revisión y la convoca.
- La reunión de revisión es llevada a cabo por el jefe de revisión, los revisores y el productor.
    - Uno de los revisores toma el papel de secretario, el cual será el encargado de ir anotando en forma escrita todos aquellos problemas que vayan surgiendo, así como los sucesos importantes que se produzcan durante la revisión.
    - La reunión comienza con una explicación de la agenda y una breve introducción a cargo del productor. Entonces el productor procede con el recorrido de inspección del producto, mientras que los revisores exponen sus preguntas basándose en su preparación previa.
    - Cuando se descubren problemas o errores válidos, el secretario los va anotando.
- Al final de la revisión, el secretario prepara un informe sumario de la misma, el cual debe comprender: ¿Qué fue revisado?, ¿Quién lo revisó?, ¿Qué se descubrió y cuáles son las conclusiones?, resumiendo todos los problemas y generando una lista de sucesos de revisión. Además todos los participantes en la reunión deben decidir si: aceptan el producto tal y como está; rechazan el producto debido a serios errores encontrados o aceptan el producto provisionalmente (los errores encontrados deberán ser corregidos).
- Una vez que se toma la decisión todos los participantes terminan firmando, indicando así que han participado en la revisión y que están de acuerdo con las conclusiones del equipo de revisión.

Durante la revisión, el secretario Al final de la reunión de revisión, resume todos los problemas y genera una lista de sucesos de revisión.

Ejemplo de un Manual de Procedimientos de Análisis y Diseño de Sistemas[20].

Objetivo del procedimiento:

Realizar el análisis, diseño, desarrollo, implantación y evaluación del software que tenga como finalidad automatizar todas aquellas actividades que sean susceptibles de realizarse a través de equipo de cómputo, dentro de la Dirección General de Asuntos del Personal Académico (DGAPA) de la UNAM.

Normas de operación:

- Los programas fuentes son propiedad de la Dirección General de Asuntos del Personal Académico y su custodia queda bajo responsabilidad del Departamento de Estadística y Sistemas.

- Es responsabilidad del Departamento de Estadística y Sistemas realizar los programas necesarios para hacer más ágiles las labores administrativas de la dependencia.

- Es responsabilidad del departamento solicitante definir claramente por escrito los requerimientos del sistema.

- El periodo de tiempo necesario para el desarrollo de los sistemas de información, será determinado por el Jefe del Departamento de Estadística y Sistemas de común acuerdo con el Jefe del Departamento solicitante.

- Para el buen desarrollo de sistemas se llevarán a cabo reuniones de trabajo con el Subdirector de Diagnóstico e Información Académica y Jefes de Departamento solicitantes.

- El requerimiento de la elaboración y diseño de sistemas deberá solicitarse por escrito al Subdirector de Diagnóstico e Información Académica.

Descripción narrativa del procedimiento de análisis y diseño de sistemas[20], de la DGAPA, UNAM.

Responsable	Actividad
Subdirector solicitante	1.- Solicita al Subdirector de Diagnóstico e Información Académica la construcción del sistema.
Subdirector de Diagnóstico e Información Académica	2.- Realiza entrevistas con jefes de departamento solicitantes y jefe del Departamento de Estadística y Sistemas.
Jefe del Departamento de Estadística y Sistemas	3.- Realiza evaluación del anteproyecto con el personal de sistemas 4.- Designa un técnico responsable del proyecto. 5.- Solicita reunión de trabajo con el subdirector de Diagnóstico e Información Académica con el fin de precisar detalles.
Subdirector de Diagnóstico e Información Académica	6.- Realiza reunión de trabajo con subdirectores involucrados, jefes de departamento y técnico responsable para definir lineamientos.

Técnico responsable del proyecto	7.- Realiza análisis previo y diseño conceptual del sistema, el cual turna al Jefe del Departamento de Estadística y Sistemas.
Jefe del Departamento de Estadística y Sistemas	8.- Revisa el análisis y diseño conceptual del sistema realizando correcciones a éste.
Subdirector solicitante	9.- Da seguimiento al proyecto.
Técnico responsable del proyecto	10.- Realiza el sistema y lo somete a pruebas. 11.- Turna el sistema al Jefe del Departamento de Estadística y Sistemas, para su revisión.
Jefe del Departamento de Estadística y Sistemas	12.- Revisa el sistema con el técnico responsable del proyecto, realizando correcciones y cambios a éste. 13.- Turna el sistema al Jefe del Departamento solicitante para sus primeras pruebas.
Jefe del Departamento solicitante	14.- Realiza las primeras pruebas al sistema y propone cambios y/o correcciones. 15.- Turna el sistema con las sugerencias de cambios y correcciones propuestos al Jefe del Departamento de Estadística y Sistemas.
Jefe del Departamento de Estadística y Sistemas	16.- Analiza con el técnico responsable del proyecto las correcciones propuestas.
Técnico responsable del proyecto	17.- Realiza los cambios y correcciones finales del sistema. 18.- Turna el sistema en su versión final al Jefe del Departamento de Estadística y Sistemas.
Jefe del Departamento de Estadística y Sistemas	19.- Entrega el sistema al Subdirector de Diagnóstico e Información Académica.
Subdirector de Diagnóstico e Información Académica	20.- Entrega el sistema al Subdirector solicitante.

ANEXO 7

Ejemplos de listas de preguntas de comprobación para las fases del ciclo de vida clásico, según Pressman.

Ingeniería del Sistema.	Planificación del proyecto de software.	Análisis de los requisitos del software.	Diseño del software.	Codificación.	Prueba del software.	Mantenimiento.
La especificación del sistema asigna la función y el rendimiento de muchos elementos del sistema, tendrá por ejemplo la siguiente lista de comprobaciones: ¿Se han definido las funciones principales de forma delimitada y sin ambigüedad? ¿Se ha definido la interfaz entre los elementos del sistema? ¿Se han establecido límites para el sistema como un todo y para cada elemento? ¿Se han establecido restricciones en el diseño de cada elemento? ¿Se ha elegido la mejor alternativa? ¿Es la solución técnicamente posible? ¿Se ha establecido un mecanismo de validación y verificación? ¿Existe consistencia entre todos los elementos del sistema?	Las estimaciones de recursos, costos y tiempos para el desarrollo, llevadas a cabo en la planeación del proyecto de software, se basan en la asignación del software establecida dentro de la actividad de la ingeniería del sistema. Una posible lista de comprobaciones sería: ¿Se ha definido el alcance del software de forma limitada y sin ambigüedad? ¿Es clara la terminología? ¿Son adecuados los recursos para ese alcance? ¿Están fácilmente los recursos? ¿Se han definido los riesgos en todas las categorías importantes? ¿ Existe un plan de	Las revisiones del análisis de requisitos del software se centran en el seguimiento de los requisitos del sistema y de la consistencia y corrección de la representación. Se pueden consideran los siguientes aspectos: ¿Es completo, consistente y exacto el análisis del campo de información? ¿Es completa la partición del problema? ¿Están definidas adecuadamente las interfaces internas y externas? ¿Refleja el modelo de datos correctamente los datos, sus atributos y sus relaciones? ¿Se pueden seguir todos los requisitos a nivel del sistema? ¿Se ha realizado un prototipo para el usuario?	Las revisiones del diseño del software se centran en el diseño de datos, el diseño arquitectónico y el diseño procedimental, en general existen dos tipos de revisiones del diseño. La revisión del diseño preliminar, la cual confirma la traducción de los requisitos al diseño y se centra en la arquitectura del software y la revisión o inspección del diseño, centra su atención en la corrección procedimental de los algoritmos tal y como están implementados en los módulos del programa. Una lista de comprobaciones para la revisión del diseño preliminar sería: ¿Están reflejados los requisitos del software en la arquitectura del mismo? ¿Se ha conseguido una modularidad efectiva? ¿Son funcionalmente independientes los módulos? ¿Depende de algunos factores la arquitectura del programa? ¿Se han definido las interfaces para los módulos y los elementos externos del sistema? ¿Es consistente la estructura de datos con el ámbito de información? ¿Es consistente la estructura de datos con el de los requisitos del software? ¿Se ha considerado la facilidad de mantenimiento?	Aunque la codificación es un resultado mecánico del diseño procedimental, se pueden introducir errores al traducir el diseño a un lenguaje de programación. Una posible lista de comprobaciones sería: ¿Se ha introducido adecuadamente el diseño al código? ¿Hay errores mecanográficos? ¿Se ha hecho un uso adecuado de las convenciones del lenguaje? ¿Se han seguido los estándares de codificación para el estilo del lenguaje, los comentarios y los prólogos de los módulos?	La prueba del software es una actividad de aseguramiento de calidad por derecho propio. Una posible lista de comprobaciones para el plan de pruebas sería: ¿Se han identificado y secuenciado adecuadamente las principales fases de prueba? ¿ Se ha establecido un seguimiento de los criterios - requisitos de validación como parte del análisis de requisitos del software? ¿ Se han comprobado todas las funciones importantes? ¿ Es consistente el plan de prueba con el plan global del proyecto? ¿ Se ha definido explícitamente un plan de tiempos para la prueba? ¿ Se han identificado y están disponibles los recursos y las herramientas para la prueba? ¿ Se ha establecido un mecanismo para registrar los resultados de las pruebas? ¿ Se han identificado los conductores y los resguardos y se ha planificado el trabajo para desarrollarlos?	Para el mantenimiento se deberá tomar en cuenta por ejemplo la siguiente lista de comprobaciones: ¿ Se han considerado los efectos laterales asociados con el cambio? ¿ Se ha documentado, evaluado y aprobado la petición de cambio? ¿ Se ha documentado el cambio, una vez hecho, e informado a las partes interesadas? ¿ Se han hecho las reuniones técnicas formales adecuadas? ¿ Se ha hecho una revisión de aceptación final para garantizar que todo el software ha sido actualizado, probado y reemplazado adecuadamente?

	gestión de riesgos? ¿Se han definido las tareas y su secuencia adecuadamente? ¿Es razonable el paralelismo en función de los recursos disponibles? ¿Es razonable la base de la estimación de costos? ¿Se han utilizado datos históricos de productividad y de calidad? ¿Se han reconciliado las diferencias entre estimaciones? ¿Son realistas el presupuesto y la fecha tope preestablecidos? ¿Es consistente la agenda?	¿Son alcanzables las prestaciones con las restricciones impuestas por otros elementos del sistema? ¿Son consistentes los requisitos con la planificación, los recursos y el presupuesto? ¿Son completos los criterios de validación?	¿Se han evaluado explícitamente los factores de calidad? Una lista de comprobaciones para la inspección del diseño ¿Realiza el algoritmo la función deseada? ¿Es el algoritmo lógicamente correcto? ¿Es consistente la interfaz con el diseño arquitectónico? ¿Es razonable la complejidad lógica? ¿Se ha especificado el tratamiento de errores y la tolerancia a errores? ¿Se han definido adecuadamente las estructuras de datos locales? ¿Se han utilizado ampliamente las construcciones de la programación estructurada? ¿Es adecuado el nivel de detalle del diseño para el lenguaje de implementación? ¿Se han utilizado características dependientes del sistema operativo o del lenguaje? ¿Se usa lógica inversa? ¿Se ha tenido en cuenta la facilidad de mantenimiento?	¿Hay comentarios incorrectos o ambiguos? ¿Son apropiadas las declaraciones de tipos y de datos? ¿Son correctas las constancias físicas? ¿Se han vuelto a aplicar todos los puntos de la lista de comprobaciones de la inspección del diseño?	¿ Se ha especificado la prueba de resistencia para el software? Una posible lista de comprobaciones para el procedimiento de prueba sería: ¿ Se han especificado tanto las pruebas de la caja negra como las de la caja blanca? ¿ Se han probado todos los caminos lógicos independientes? ¿ Se han identificado y listado los casos de prueba junto con los resultados esperados? ¿ Se va a probar el manejo de errores? ¿ Se van a probar los valores límites? ¿ Se va a probar el rendimiento y las limitaciones temporales? ¿ Se ha especificado la variación aceptable respecto a los resultados esperados?	

ANEXO 8

Fases, actividades y listas de preguntas de comprobación para una auditoría

Fases de una auditoría, según la NASA.

Planeación y preparación.	Visita al sitio.	Reporte de la auditoría.	Seguimiento.
Durante la fase de planeación y preparación, se debe entender y quedar claro el objetivo del desarrollo del proyecto, basándose en el alcance de la auditoría, el auditor determina que cuestiones específicas se necesitarán ser contestadas, así como las personas que intervendrán en las auditorías, los productos que serán examinados y los registros de las respuestas a las preguntas. El auditor interviene en la conducción y registro de los productos examinados durante la visita.	La fase de visita al sitio tiene como propósito que se recolecten los datos necesarios para asegurar que los productos solicitados están bien producidos, de acuerdo a los estándares aplicados, así como a los procedimientos empleados y al proceso de seguimiento utilizado y además que el estado de los reportes correspondan al estado actual. Se usan dos técnicas básicas: La primera, las entrevistas con el personal del grupo del proyecto y la segunda, la examinación de los registros de información y la documentación, como pueden ser los manuales de diseño, de usuario, de código, los folders de desarrollo, etc., así como todos los registros de información que se tengan, como por ejemplo: los memorándum, los formatos y documentos de los eventos en la vida de los productos, etc. El auditor debe empezar por examinar las partes tangibles del proyecto, en este caso, los registros y los productos. * La examinación de registros dependerá de los procesos principales que se examinen, y comprenden tres tipos: la de control de cambios (CM), la del ciclo completo de vida (NRCA) y la de la matriz de verificación y validación (V&V). La auditoría CM., debe comenzar con la solicitud inicial de cambio, continuando con el análisis de impacto y disponibilidad; diseño, codificación y pruebas; la actualización de la documentación; la submisión de modificar la librería de los productos modificados y cierre de la solicitud de cambio. Los registros deben mostrar la autorización de cada cambio, el producto a ser cambiado y el número de la versión de cambio. El auditor debe checar que las librerías del producto estén al día con el código de cambios, para asegurar que la documentación este actualizada. Para esto, el auditor debe de checar que el número de versión, los esquemas de identificación y los documentos de control sean los últimos. Además que todos los códigos y documentaciones en las librerías sean correctamente recibidas. La auditoría NRCA. En esta auditoría el auditor debe observar el ciclo completo, debiendo revisar: - La concordancia de los reportes archivados, para asegurar que estén completos y correctamente archivados. - La disposición de procesos, así como las listas y las formas de acciones deben ser revisadas de igual forma. Se debe de mostrar en una reunión que los cambios fueron hechos, probados y revisados, aprobados, publicados y reconocidos. El auditor debe poner particular atención en los productos corregidos para asegurar que ellos cumplieron satisfactoriamente los requerimientos y estándares. La auditoría V & V., debe incluir un chequeo de la matriz de verificación y validación o equivalente, para asegurar que todos los requerimientos han sido probados y que todos los requerimientos de los planes de chequeos de prueba, sean adecuados, especificando los resultados para cada prueba. Los procedimientos de prueba deben ser claros y detallados. Los planes y procedimientos de prueba deben ser revisados y aprobados. * La examinación de productos. Los productos se pueden examinar de dos formas: Ver si los estándares se han sido seguidos; Ver si el estado del producto es realmente el reportado. El auditor debe de tener la habilidad para determinar que documentos deben ser mostrados en el diseño, que en realidad contengan información de diseño. El código también debe ser examinado para determinar si cumplen con las métricas estándares. Los códigos estándares incluyen las reglas para la documentación interna, tamaño de módulos, estilos de formato y otras cosas que el auditor pueda verificar. Los productos también deben ser examinados para comparar el estado actual con el reportado. Los documentos reportados deben estar completos, deben de mostrar el contenido total de las secciones que tiene el índice, además deben estar firmados por las autoridades apropiadas. Durante el proceso de chequeo de los registros y productos, el auditor usualmente no puede examinar cada uno de los artículos, sin embargo, alguna forma de muestreo debe de ser usado.	La fase de reportes consiste en la preparación para obtener los reportes escritos de la auditoría, clasificando detalles y proveyendo la información relacionada que se requiera. El auditor debe preparar el reporte final: El reporte debe estar organizado, mostrando los resultados de los más importantes a los menos, deben comentarse los problemas, debiéndose incluir una narrativa general de la auditoría. El objetivo del reporte de auditoría es presentar una fiel imagen del estado de la actividad de desarrollo o una faceta de la actividad de la gestión del proyecto. El reporte debe ser claro y preciso, basado en los hechos.	Los cambios deben tener un seguimiento para asegurar que ellos ocurran y sean efectivos llegándose a su conclusión, debiéndose documentar ampliamente. La fase de seguimiento, en donde los problemas y deficiencias encontrados en la auditoría deben ser remediados. La parte de seguimiento incluye reauditorías para asegurar o valorar los remedios. En muchos casos, la mejor manera de determinar que los problemas fueron resueltos, es a través de auditorías de seguimiento.

Auditorías de aseguramiento de la calidad durante el ciclo de vida del software, según la NASA.

Fase de iniciación y conceptualización del software.	Fase de definición de requerimientos del software.	Fase del diseño preliminar del software.	Fase del diseño detallado del software.	Fase de programación del software.	Fase de pruebas e integración del software.	Fase de aceptación y de entrega.	Fase de operación y soporte.
La intención de las auditorías en esta fase sería revisar lo que la corporación provee sobre los estándares y procedimientos de proyectos pasados. Se debe examinar sobre el contexto del proyecto propuesto y juzgar sobre su eficacia. Se requiere de un auditor con experiencia. Los procedimientos y estándares para el proyecto deben ser formulados en esta fase, debiendo ser apropiados para el proyecto y para auditarlo. El contrato debe de especificar si habrá auditorías internas y/o externas.	Las auditorías internas deben concentrarse sobre los procesos de desarrollo, documentación y control de los requerimientos. Debe de verificarse que los documentos de los requerimientos siguen el formato especificado del documento estándar. Una auditoría externa se puede permitir y sería en los mismos términos que la interna.	Las auditorías internas y externas en esta fase deben incluir la documentación del diseño, verificando los formatos estándar y las métricas. Es importante los mecanismos de control para asegurar que no haya cambios no autorizados.	Las auditorías deben dirigirse al progreso y documentación del diseño detallado.	Los auditores deben checar los resultados del diseño y la codificación. Auditorías internas durante esta fase deben ser frecuentes. El grupo de desarrolladores deben ser los mejores y debe ir creciendo el número de actividades simultáneas. Las auditorías SQA es una de las más importantes formas para el gestor mantenga el control del proceso, asegurando la calidad de los productos al iniciar el desarrollo y que el estado actual es realmente el reportado. Los productos terminados están listos para ser enviados a la etapa de pruebas, así los productos y sus procesos de control deben ser auditados. Una auditoría SQA externa asegura que se esta siguiendo el camino deseado y que el estado reportado es el correcto. Si se identifica algún problema, en esta etapa, es un momento temprano y los cambios y acciones correctivas serán fáciles de realizar.	Auditorías internas deben concentrarse para asegurar que los cambios hechos a los productos son los correctos y no tener errores descubiertos posteriores a la fase de pruebas.	Los pasos de auditorías son similares a las auditorías CM y NRCA.	

Ejemplos de listas de preguntas de comprobación para una auditoría de SQA.

Aseguramiento del software.	¿Se cuenta con un plan SQA preparado? ¿Esta el plan actualizado de acuerdo con los requerimientos actuales del programa? ¿El plan SQA está por escrito y aprobado? El plan incluye o define: ¿Requerimientos de aseguramiento de calidad del software y actividades a realizar? ¿Un plan que muestre cuando cada una de las actividades que deberán ser llevadas a cabo? ¿Existen presupuestos para las actividades a realizar? ¿Se cuenta con una organización con tareas específicas asignadas? ¿Existe interacción entre el grupo SQA y el grupo desarrollador? ¿Participa el grupo SQA en el proceso gestor de cambios? ¿Existe evidencia de que las actividades SQA están implementadas de acuerdo con el ciclo de vida del software?
Desarrollo de la documentación	¿Existen estándares establecidos para la entrega de documentación? ¿La documentación cumple con los estándares? ¿Los procedimientos establecidos y la documentación asegura que los estándares sean seguidos? ¿Los procedimientos dirigen los cambios de la documentación del software bajo el control de la gestión de la configuración? ¿Están los cambios revisados como lo indica el documento base? ¿Existen métodos establecidos para el manejo de la documentación, incluyendo cambios? ¿Los contenidos de los documentos entregados están limpios, concisos, completos y entendibles? ¿Está el grupo revisor familiarizado con el material que revisará para detectar inconsistencias? ¿Existe la autoridad para aprobar fácilmente la documentación presentada? ¿Es proporcionada la documentación requerida a tiempo y de manera responsiva? ¿Hay suficientes copias a disposición? ¿Hay procedimientos establecidos que sigan la producción de documentos? ¿Los documentos de los folders de desarrollo, concuerdan con la fase del ciclo de vida? ¿El nivel de detalle en la documentación se ve razonable?
Código.	¿El código, el prólogo y el lenguaje programado observa todos los estándares y convenciones? ¿Están los elementos necesarios del prólogo completos?, ¿están descritos todos los elementos datos y las subroutinas? ¿La documentación del código interno presentada reúne los requerimientos de los estándares? ¿Es el código consistente con el diseño, cómo se presenta en el prólogo y en el lenguaje de diseño? ¿El código que se presenta es el correcto para los casos de pruebas, puede ser verificado por una inspección visual? ¿Todo el código es fácilmente identificable? ¿Los casos de prueba que se presentan están adecuadamente basados en el lenguaje de programación?
Gestión de la configuración del software.	¿Se tiene un plan de gestión de configuración de software desarrollado? ¿El plan está basado en líneas bases? ¿Existen instrucciones para identificar elementos de líneas bases y revisiones subsecuentes o siguientes versiones? ¿Existen procedimientos de gestión de configuración en los cuales se requiere aprobación para agregar o modificar elementos en los programas de librerías existentes? ¿Esta la organización de la gestión de la configuración adecuadamente proveída de personal calificado y responsable? ¿Los documentos de las líneas bases cumplen con los requerimientos contractuales? ¿Están las responsabilidades claramente entendidas? ¿Las especificaciones aprobadas sirven como una línea base para el control de cambios? ¿Esta actualizada la lista de especificaciones aprobadas, está al corriente, cambios posteriores? ¿Están establecidos los procedimientos para la producción de la documentación adecuada de software? ¿Existen procedimientos para el manejo adecuado y eficiente de reportes? ¿Existe una tabla de control de la configuración bien establecida? ¿Quienes son sus miembros? ¿Está el grupo SQA representado? ¿Todos sus miembros asisten regularmente? ¿Se tienen formalmente auditorías de configuración conducidas o planeadas?
Librerías de programas.	¿Se tiene una librería de programas establecida? ¿Se tiene aprobado un programa de librerías? ¿Se tienen procedimientos de identificación adecuados para: control de librerías, control elementos de configuración, manejo de reportes de problemas, etc.? ¿El programa de librería cumple con los procedimientos establecidos?

	¿Los reportes de problemas realizados están apropiadamente llenados y guardados en sus folders? ¿Las versiones de los programas están adecuadamente identificados, controlados y documentados a lo largo del ciclo de vida? ¿Se usa un sistema de control de programas fuentes? ¿Está adecuadamente actualizado? ¿Cómo es controlado? (error reportado, solicitud de cambio, etc.) ¿Qué medidas se toman para asegurar que todas las modificaciones estén aprobadas, integradas y que el software sea sometido a pruebas y que sea la versión correcta? ¿Está monitoreado y controlado el software que no ha sido liberado? ¿Los folders de desarrollo están regularmente sometidos a los programas de librerías? ¿Existe un índice de la documentación de librería, está actualizado? ¿Existe un logo que muestre que material ha sido checado y se encuentra fuera de la librería? ¿Todo el código sometido a una auditoría incluye la información de transito necesaria? ¿Está disponible para la auditoría? ¿Todos los elementos de la librería de programas tienen un número de identificación asignado con su número de versión? ¿Este número está relacionado con información asociada? ¿El flujo a través del ciclo de vida esta claro, preciso, documentado y correcto?
Reportes de errores o inconformidades y acciones correctivas.	¿Se tienen procedimientos establecidos que aseguren la detección y corrección de deficiencias encontradas? ¿Son analizados y examinados los problemas y reportes para determinar sus causas? ¿Se cuenta con acciones correctivas bien documentadas sobre los reportes de problemas? ¿Se cuentan con acciones revisadas y monitoreadas para determinar adecuadamente los requerimientos? ¿Están todos los reportes de acciones correctivas en archivos? ¿Existe un soporte gestor para el sistema de acciones correctivas? ¿El programa de librerías sigue los procedimientos para mantenimiento, control y estado de los problemas reportados? ¿Los problemas reportados pertenecientes a una unidad, están dentro del folder de desarrollo de esa unidad? ¿Los desarrolladores de software cumplen con los requerimientos para hacer los reportes de problemas? ¿Existe documentación aprobada para los cambios? ¿Todas las formas requieren autorizaciones?
Verificación y validación.	¿Se tienen los requerimientos de software que han sido analizados para determinar las pruebas? ¿Son los objetivos de las pruebas adecuados, factibles y suficientes para demostrar la eficiencia del software? ¿Las pruebas y metodologías están basadas en suposiciones que son aceptadas por el grupo SQA? ¿Existen procedimiento para monitorear y dar seguimiento a los rechazos? ¿Existen planes de prueba y procedimientos completos basados en estándares específicos? ¿Son los planes y procedimientos de pruebas aprobados por el usuario cuando es necesario? ¿Todas las herramientas y equipos están identificados, definidos, calibrados y controlados para las pruebas? ¿Existen líneas bases de software antes de ser aprobadas? ¿Se verifica que sea la versión correcta del software y la documentación asociada certificada antes de probarlo? ¿Están las pruebas de aceptación monitoreadas por un representante del grupo SQA, por el adquiridor cuando se requiere, si no, entonces quien monitorea dichas pruebas? ¿Están todas las pruebas de hardware certificadas? ¿Existen resultados de pruebas que han sido certificadas por miembros participantes? ¿Se tienen reportes de pruebas que han sido revisados y certificados,?, ¿Por quién? ¿Existen deficiencias en la documentación de los reportes de problemas? ¿Se tiene documentación relacionada con pruebas para permitir la repetividad de las mismas en otros casos? ¿Existe una matriz de verificación de pruebas para asegurar que todos los requerimientos han sido probados? ¿Existen procedimientos de pruebas limpios y completos?
Estado del proyecto.	¿Concuerdan las fechas de los folders de desarrollo con el reporte del gestor? ¿Si no, que tan grande es la diferencia? ¿Según el plan de desarrollo? ¿Dónde debe de estar el proyecto? ¿Qué actividades deben estar ejecutándose? ¿Qué proyectos intermedios deben de haberse liberado? ¿Qué revisiones deben de haberse realizado?, etc.

Glosario de términos.

A

Acción correctiva: Acción tomada para eliminar las causas de una no-conformidad, defectos u otra situación indeseable a fin de prevenir su recurrencia.

Acción preventiva: Acción tomada para eliminar las causas potenciales de no-conformidades, defectos u otra situación a fin de prevenir su ocurrencia.

Análisis: Es la identificación de cada elemento en un sistema, obteniéndose las restricciones y necesidades de funcionamiento.

Auditor: Persona capacitada para realizar auditorías.

Auditoría de calidad: Contrastar el ser contra el deber ser, es decir, verificar si se está cumpliendo con los procedimientos o no.

Aseguramiento de calidad: Conjunto de actividades planeadas y sistemáticas que lleva a cabo una empresa, con el objeto de brindar la confianza apropiada, de que un producto y/o servicio cumple con los requisitos de calidad especificada.

Aseguramiento de calidad del software: Es una actividad de protección que se aplica a lo largo de todo el proceso de Ingeniería de software.

B

Base de datos: Una colección de registros almacenados electrónicamente.

C

Calidad: Conjunto de características de un elemento que le confieren la aptitud para satisfacer necesidades explícitas e implícitas.

Calidad de software: Es la concordancia de los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo.

Cliente: Se emplea para denotar a un individuo o a una organización que solicita un producto de software o bien su modificación. Puede ser o no el usuario final del producto. El receptor de un producto suministrado por el proveedor.

Codificación: Es el proceso de transformar el diseño en instrucciones máquina.

Conformidad: Cumplimiento de los requisitos especificados.

Control de calidad: Conjunto de métodos y actividades de carácter operativo, que se utilizan para satisfacer el cumplimiento de los requisitos de calidad establecidos.

Computadora: Dispositivo electromecánico que recibe como entrada programas y datos, dando como salida resultados que dependen de los anteriores.
Desarrollo del software:

Configuración del software: Significa la función y/o características físicas del software, así como la documentación técnica realizada en un producto de software.

Corrección: Grado en el que un producto de programación está libre de defectos de diseño y de codificación, esto es, libre de fallas. Grado en que un producto de programación cumple los requisitos especificados. Grado en que un producto de programación cumple con las expectativas del usuario.

D
Diseño: Es el proceso de planificación de cómo se va a construir el sistema para encontrar su solución.

E
Eficiencia: Grado con el que un producto de programación efectúa sus funciones, mediante un mínimo de recursos computacionales.

Error: Discrepancia entre una condición o valor calculado y la condición real, especificada, o valor correcto teórico.

Especificación: Un documento que establece requisitos.

Exactitud: Especificación cualitativa de ausencia de error. Medida cuantitativa de la magnitud del error, de preferencia expresada como una función del error relativo.

Estándares: Normas, reglas, criterios establecidos de cómo se debe hacer algo

F
Factores de calidad: Elementos o causas que determinan o afectan a la calidad

G
Gestión: Acción y efecto de administrar. Conducción

Gestión de calidad: Aspecto de la función de administración o gestión general que determina e implanta la política de calidad, que incluye la planeación

P

Paradigmas de la Ingeniería del software: Serie de pasos que abarcan los métodos, técnicas y herramientas de la Ingeniería del software, para llevar a cabo el proceso de desarrollo del software.

Plan: Conjunto de disposiciones adoptadas para la ejecución de un proyecto.

Plan de calidad: Un documento que establece las prácticas relevantes específicas de calidad, los recursos y secuencia de actividades pertenecientes a un producto, proyecto o contrato particular.

Plan de aseguramiento de calidad: Es un plan de inspección sistemático que se requiere para asegurar la calidad del software que se desarrolle.

Política de calidad: Conjunto de directrices y objetivos generales de una empresa relativos a la calidad y que son formalmente expresados, establecidos y firmados por la alta dirección.

Procedimiento: Forma especificada de desarrollar una actividad.

Proceso de desarrollo: Son los pasos o etapas que comprende la elaboración de un programa de computadora.

Producto: Es el resultado de actividades o de procesos y puede ser tangible o intangible o bien una combinación de los dos.

Programa: Conjunto de instrucciones que permiten adaptar a la computadora a distintos objetivos.

Proveedor: Organización que suministra un producto al cliente.

Pruebas: Es el proceso de demostrar que un software satisface los requerimientos del usuario y funciona correctamente para todos los posibles datos de entrada.

Puesta en marcha: Una vez que se han realizado todas las pruebas al sistema y se han corregido los errores encontrados, se procede a la instalación del sistema en el lugar donde estará funcionando normalmente el sistema, incluye capacitación y asesoría a los usuarios, manual de instalación, etc..

R

Registro: Un documento que provee evidencia objetiva de las actividades ejecutadas o resultados obtenidos.

Requerimientos: Son las necesidades o requisitos que debe cubrir.

S

Sistema: Conjunto de elementos, componentes, funciones o etapas que interrelacionados entre si, trabajan para lograr un objetivo.

Sistema de calidad: Estructura organizacional, conjunto de recursos, responsabilidades y procedimientos establecidos para asegurar que los productos, procesos y servicios cumplan satisfactoriamente con el fin a que están destinados; los cuales están dirigidos hacia la implantación de la gestión de calidad.

Sistema de información: Es un sistema de cómputo el cual esta integrado por hardware, software, base de datos, un administrador y los usuarios. En otras palabras es una aplicación de cómputo que requiere de los anteriores componentes.

Software: Es el producto de programación que incluye el código fuente y todos los manuales asociados, así como la documentación propia del producto. Una creación intelectual que consiste en información expresada a través de medios de soporte. El software puede encontrarse en forma de conceptos, transacciones o procedimientos.

Solidez: Grado con el que un producto de programación puede continuar operando correctamente, a pesar de la introducción de datos inválidos.

V

Validación: Confirmación del cumplimiento de los requisitos particulares para un uso intencionado propuesto, por medio del examen y aporte de evidencia objetiva.

Verificación: Confirmación del cumplimiento de los requisitos especificados por medio del examen y aporte de evidencia objetiva.

Bibliografía

[1] NMX-CC-001:1995 IMNC, *Administración de la Calidad y Aseguramiento de la Calidad - Vocabulario* (ISO 8402:1994).

[2] NMX-CC-002/1:1995 IMNC, *Administración de la Calidad y Aseguramiento de la Calidad. Parte1: Directrices para selección y uso.* (ISO 9000-1: 1994).

[3] Pressman, R., *Ingeniería del Software, un enfoque práctico*, McGraw-Hill,1995.

[4] *Outline,* http://www.cs.bsu.edu/edu/homepages/00c0lin/680/outline.html.

[5] Sommerville, I.., *Ingeniería de Software*, Addison-wesley Iberoamericana, 1988.

[6] Fairley, R., *Ingeniería de Software*, McGraw-Hill, 1988.

[7] Yourdon E., *Análisis Estructurado Moderno,* Prentice Hall, 1993.

[8] Kendall, K., y Kendall, J., *Systems Analysis and Design,* Prentice Hall, 1995.

[9] McClure, C., CASE la automatización del software, Addisson-Wesley Iberoamericana, 1993.

[10] NASA, *Software Configuration Management Guidebook*, August, 1995., http://satc.gsfc.nasa. gov/ GuideBoobs/cmpub.html

[11] SYBASE, *Introduction to SYBASE Client/Server Architecture*, Student Guide, 1995.

[12] Chow, T., *TUTORIAL software quality assurance a practical approach*, IEEE Computer Society Press, 1985.

[13] Acis, *Software Quality Assurance Plan*, http://acis.mit.edu/acis/sqap/#4.

[14] Butler, D., *Software Quality Assurance Plan*, http://www.csc.calpoly.edu/~dbutler/s1fall95/Baselines/ SQA-sqaplan.html.

[15] Dobbins, J., *Software Quality Assurance and Evaluation,* Quality Press, American Society for Quality Control, 1990.

[16] Stamm, S, *Assuring Quality Quality assurance*, Datamation, March 1981 Technical Publishing Co., Dun and Bradstreet company.

[17] ISO 9000 - 3, *Quality management and quality assurance standards - Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software.* First edition 1991.

[18] Cavano, J.P. and McCall, J.A.,*A Framework for the Measurement of Software Quality*,The proceeding of the ACM Software Quality Assurance Workshop, November 1978.

[19] NASA, *Software Quality Assurance Audits Guidebook*, Nov. 1990., http://satc.gsfc.nasa.gov/audit/audgb. txt.

[20] *Manual del procedimiento de análisis y diseño de sistemas del Departamento de Estadística y Sistemas*, Dirección General dc Asuntos del Personal Académico, Secretaría General, UNAM. Oct. 1995.