MEJORES PRÁCTICAS PARA EL ESTABLECIMIENTO Y ASEGURAMIENTO DE LA CALIDAD DE SOFTWARE

Por:

Vega Lebrún Carlos
Rivera Prieto Laura Susana
García Santillán Arturo

Serie

Libros y Manuales: Informática
Unidad Multidisciplinaria: CIET

Libros de Texto/02/2008

UNIVERSIDAD CRISTOBAL COLON
Campus Calasanz y Campus Torrente
Carr. Veracruz-Medellín s/n Col. Puente Moreno, Boca del Río, Ver.,
Tel. (01 229) 9230170 al 76 Ext. 2060 y 2069
http://dgip.ver.ucc.mx
http://dgip.ver.ucc.mx/centros/ciet/ciet.htm

Diseño de Portada
Julio César Hernández Rivera
Centro de Cómputo Académico UCC, Campus Calasanz

Vega, Rivera y García.: (2008) "*MEJORES PRÁCTICAS PARA EL ESTABLECIMIENTO Y ASEGURAMIENTO DE CALIDAD DE SOFTWARE*" Edición electrónica. Texto completo en www.eumed.net/

ISBN: En trámite
Registro Biblioteca Nacional de España No. En trámite

Índice del contenido

Pág.

Capítulo I: El Contexto

Capítulo II: Argumentación Teórica

RESUMEN

"No hay viento favorable para el que no sabe a donde va"
Séneca.

Hoy en día, en las empresas generadoras de software se hace necesario que la alta gerencia reconozca la importancia de llevar a cabo una reingeniería con respecto a los sistemas de calidad, y que a su vez se pueda manifestar en el desempeño de los empleados. Si la gerencia observa a la norma como algo requerido por los clientes y no como algo beneficioso, lo mismo ocurrirá con el personal. La gerencia es responsable de proporcionar los recursos necesarios para poder implementar un sistema de calidad así como el compromiso en darle seguimiento y avance a dicho sistema.

Por estas situaciones, se emprende el siguiente estudio exploratorio, en el cual se estará proporcionando una guía de mejores prácticas para el aseguramiento de calidad de software (SQA), fundamentada en la recopilación bibliográfica bajo los conceptos de Sistemas de Información, Ingeniería de Software, Aseguramiento de Calidad, Mejores Prácticas, Métricas de Calidad.

Para validar las mejores prácticas que se propondrán, se estarán desarrollando casos de estudio y a su vez estos serán apoyados por las empresas participantes de la Cd. de Veracruz, que serán tomadas como muestra para llevar a cabo esta investigación. La verificación de esta propuesta se realiza a partir de los instrumentos utilizados en la obtención de información acerca del SQA y las formas de trabajo actual que lleven las organizaciones. Los resultados apoyan la propuesta para la generación de las mejores prácticas, como parte integral del éxito de los sistemas de información de software.

Los autores

Capítulo I: El Contexto

1.1 Antecedente

El aseguramiento de calidad en el software, no es sólo una moda hoy en nuestros días, es una necesidad. Así como la ingeniería de software (que sugiere la integración de actividades de ingeniería en el desarrollo de software), también es necesario el uso de prácticas que nos lleven hacia productos terminados con calidad. Así mismo, se requiere de procesos que nos ayuden a verificar, validar y controlar que dichas prácticas sean realizadas correctamente. El aseguramiento y mejora de la calidad del software es reconocido cada vez más como un problema fundamental, o quizá como el problema fundamental en cuanto a la ingeniería de software en los años de 1990 [21].

Los principales problemas son dos y son de naturaleza distinta; por un lado las empresas de hoy en día no dedican recursos ni atención necesaria para que la actividad tenga éxito [21] y por otro que los esfuerzos que se hacen en su mayoría son reactivos, es decir, cuando ya detectaron un problema en el producto final [28]. Para muchas organizaciones, la información y la tecnología que la soporta, representan los activos más valiosos de la empresa; es más, en nuestro competitivo, rápido y cambiante ambiente actual, la gerencia ha incrementado sus expectativas relacionadas con la entrega del servicio y la calidad en cada uno de los sistemas de información.

Al igual que el ser humano, la empresa necesita de un mecanismo de comunicación interna, "un sistema nervioso" que coordine sus acciones. Un sistema nervioso digital que le permita realizar operaciones a la velocidad del pensamiento: condición clave del éxito en el siglo XXI. Ahora bien Data Mining, EIS, DSS, e-business, ERP, CRM, Knowledge Management, GroupWare, Intranets, Sistemas Dinámicos, COBIT, RUP, CMM, son sólo algunos de los conceptos que prometen llevar a las organizaciones a un desempeño de clase mundial, a la ruta de la excelencia y a la

calidad en sus productos y al control de mejores prácticas sobre los niveles que garanticen la confianza y seguridad de los clientes, donde ellos son quienes deciden, donde además, ahora más que nunca el futuro es impredecible. La calidad de software debe de interpretarse, como la concordancia con los requisitos funcionales y de rendimientos explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente [22].

Esto es muy cierto y, aunado a esta reflexión, la norma de Calidad ISO 8402, UNE 66-01-92 TQM [32], menciona a la Calidad de software como el conjunto de características de una entidad que le confiere su aptitud para satisfacer las necesidades expresadas y las implícitas. Es importante mencionar que dentro de las Normatividades de Calidad como ISO 9000-9004, ISO QS9000 [14], SEI [26], CMM [3], IEEE [11], por mencionar las más importantes, la administración correcta en el control de calidad debe aportar a las organizaciones "el conjunto de actividades de la función general de la dirección que determine la calidad como objetivo de éxito".

Este trabajo tiene la finalidad de proponer y dar a conocer una serie de mejores prácticas para el establecimiento de Aseguramiento de Calidad de Software en pequeñas, medianas y grandes empresas en donde el desconocimiento de llevar una serie de pasos a seguir para garantizar la calidad de sus productos no ayuda a dar una respuesta más confiable a las necesidades de los usuarios.

1.2 Situación Actual

Con la globalización, la llegada de Internet, y la tecnología, se hace cada vez más esencial estar al mando de la información. Los mercados requieren flexibilidad y dinamismo de las empresas. El que tiene la mejor información y la tiene a tiempo, será el ganador. Los sistemas de información y la tecnología usada, determinarán si las organizaciones puedan sobrevivir al proveer la información requerida para tomar decisiones acertadas [9].

Estudios realizados demuestran que un alto porcentaje del éxito o fracaso del proyecto no solamente está en la tecnología disponible y en el conocimiento que se tenga de ella, sino también en la forma en que el proyecto no lleva un control de calidad durante su desarrollo [32].

El desempeño de los proyectos de sistemas actualmente es: 26% de ellos son exitosos, un 46% son proyectos cuestionables y un 28% son proyectos fallidos, arrojando una cifra de 97 Miles de Millones de USD de desperdicio, (Standish Group International). Casi el 25% de los proyectos de software son cancelados por atraso o por salirse del presupuesto, o por tener una baja calidad, o por experimentar alguna combinación de ellos [5].

En los últimos quince años la industria mundial de desarrollo de software se ha preocupado por mejorar sus capacidades en el desarrollo de software de calidad. Las empresas están invirtiendo en la mejora de los procesos de desarrollo para poder distinguirse en el mercado [9]. Se han definido varios modelos basados en las experiencias exitosas de la Ingeniería de Software que sirven de guía para las mejoras y unifican los criterios de evaluación de las empresas. Las normas ISO de serie 9000, el modelo estadounidense conocido como CMM (Capability Maturity Model), el BOOTSTRAP (Estándar Europeo para Evaluación y Mejoras de Procesos de Desarrollo de Software) y la norma ISO 15504, conocida como SPICE, (Software Process Improvement and Capability determination) son los ejemplos más reconocidos de estos modelos.

Las empresas hacen esfuerzos para implantar estos modelos y lograr la certificación o evaluación en alguno de ellos con el objetivo de obtener ventajas competitivas [9]. Uno de los modelos de procesos de software de referencia, que adquiere mayor interés en México, es el CMM [3]. Podemos mencionar por lo menos dos razones de lo anterior. Los procesos deben ayudarnos a lograr un objetivo de la organización más no son ellos mismos el objetivo. La burocratización es el resultado de ver al proceso como objetivo [10].

El proceso es la unión que mantiene juntas las capas de tecnología y que permite un desarrollo racional y oportuno de la ingeniería del software. Los métodos indican cómo construir técnicamente el software y con la calidad con la que éste debe de contar. Las herramientas proporcionan un soporte automático o semiautomático para el proceso y para los métodos [23].

“Hablando de modelos y herramientas para guiar el mejoramiento del proceso de calidad software, ¿es posible aplicarlos en forma directa a una empresa u organización informática altamente inmadura de América Latina?” [2].

1.3 Justificación

La mayoría de las empresas, al menos en Latinoamérica, no utilizan procedimientos, mejores prácticas o siguen algún lineamiento para la evaluación de procesos de calidad en los productos de software, es más, pocas son las que están certificadas o han identificado sus niveles de madurez y se han dado cuenta de la importancia que tiene el hecho de considerar este seguimiento y mejoras continuas en sus procesos de desarrollo. Para un director de TI y todo su equipo de trabajo, los sistemas de información deben estar en orden y bien documentados para futuras revisiones, y operaciones, que implica la creación, manejo y seguimiento de un sistema de información.

Hoy día en las organizaciones es un problema el no cuantificar la cantidad de mantenimiento, retrabajos (reprogramaciones) que se realizan producto del desarrollo de mala calidad, o no se mide el impacto de estos malos proyectos, muchas veces por querer dar una respuesta rápida a los usuarios nos enfrentamos más adelante a situaciones críticas como aspectos financieros (errores de facturación, etc.), costos (hora/hombre invertidas en reparar lo hecho, etc.) o de seguridad, sólo por mencionar algunos.

1.4 Propósito

Para este estudio se ha propuesto definir una serie de pasos que guíen de una forma estructurada, las mejores prácticas en el establecimiento y Aseguramiento de Calidad de Software en los departamentos de Sistemas de Información en las empresas que sean generadoras de productos a fines. De tal manera que el nivel de certeza sobre las mejores prácticas a seguir ayude a evaluar los procesos de desarrollo y que estas prácticas a su vez estén fundamentadas en parámetros objetivos, y aporte un panorama mejor estructurado en su actual forma de trabajo.

1.5 Alcance

"La calidad del software es tan importante como la calidad de la productividad con la cual es generada. Esto es y será la llave del éxito junto con el servicio a los usuarios" [36]. La calidad requiere de control y a su vez, se hace necesario un método universalmente aceptado que permita diagnosticar en forma ordenada aquello que deseamos resolver o mejorar. A lo largo de estas líneas, se enumera una vasta cantidad de factores de calidad. No son factores nuevos, por el contrario, se ha optado por usar, aquellos que ya han sido definidos, probados y aceptados con anterioridad.

A partir de ello, en los siguientes apartados de este documento se estará describiendo, una serie de mejores prácticas que puedan favorecer el establecimiento y aseguramiento de calidad de software y que a su vez se evalué en las empresas la forma de cultura de trabajo en los desarrollos de software, y con ello establezca las pautas y lineamientos a seguir en la búsqueda del desarrollo de software con calidad y éxito.

Capítulo II: Argumentación Teórica

2.1 Sistemas de Información

Definición

Se considera que por mucho tiempo, el desarrollo de hardware ha superado al de software, por ello en las últimas décadas se ha puesto mayor atención al componente lógico de los sistemas de información. Es decir, en el nivel más abstracto se puede presentar a los sistemas de información como5 una "caja negra". Para transformar las entradas en salidas existen ciertos elementos que integran los sistemas de información, los cuales se ilustran en la Figura 2.1.

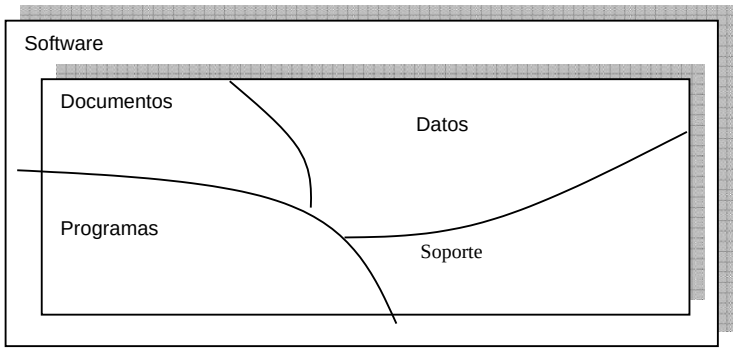


Figura 2.1 Elementos que integran los Sistemas de Información

2.2. Elementos

Los elementos que integran los sistemas de información pueden definirse de la siguiente manera [29]:

- Documento. Información escrita, necesaria para desarrollar implantar y utilizar los programas.
- Programas. Conjunto de líneas de código fuente, asociados con alguna aplicación o producto.
- Datos. Grupo de elementos que tienen forma y contenido similares en estructura de implantación y en la composición (datos elementales y datos complejos que además pueden dividirse).
- Soporte. Todas aquellas actividades y recursos encaminados a auxiliar el logro.
- Ya con la integración de estos componentes [22] los sistemas de información se conducen a las características siguientes:
- Es Ingenieril. Requiere alto grado de conocimiento y capacidad por quienes desarrollan sistemas de información.
- Se deteriora. Esto ocurre debido a los cambios o actualizaciones.
- Es complejo. Es difícil de encontrar una pieza de repuesto, si el sistema de información se deteriora.
- Es una oportunidad de trabajo. Buenas prácticas de sistemas de información o de ingeniería de software, pueden conducir al éxito de los desarrolladores.

2.3 Conclusión

Se concluye al respecto, que los sistemas de información, como subconjunto de los sistemas computacionales, requieren de integración efectiva y mejores prácticas de sus elementos (documentos, programas, datos y soporte) para poder cumplir con el objetivo propuesto.

2.4 Ingeniería de Software

Definición

Los componentes de los sistemas de información, son integrados por la ingeniería de software; aunado a ello, se encontraron las siguientes referencias:

- La aplicación práctica de las ciencias computacionales y otras disciplinas, al análisis, diseño, construcción y

mantenimiento de software y a la documentación asociada [22].

- Disciplina tecnológica y administrativa orientada a la producción sistemática de productos de programación, que son desarrollados y modificados a tiempo, dentro de un presupuesto definido [6].

Los métodos, procedimientos y herramientas son conjuntados en modelos conocidos como paradigmas. Existen varios paradigmas de la ingeniería de software, algunos de ellos son: cascada, prototipo, costo, espiral, entre otros. [22]. Se han identificado diversas definiciones acerca de la ingeniería de software. El ciclo de vida es el periodo de tiempo que empieza cuando un producto de software es concebido y termina cuando el producto ya no se encuentra disponible para su uso [7]. Este ciclo de vida se compone de 5 etapas principales: (a) Planeación, (b) Análisis y Diseño, (c) Codificación, (d) Pruebas, (e) Operación y Mantenimiento.

2.4.1 Objetivos básicos de la ingeniería de software:

1. Satisfacer los requerimientos del cliente,
2. Mejorar la calidad de los productos de software,
3. Mejorar la productividad de los desarrolladores,
4. Satisfacción profesional de la gente dedicada al desarrollo,
5. Incluir nuevas tecnologías que integran los objetivos anteriores.

Por otra parte, la Figura 2.2 muestra los elementos que integran la ingeniería de software, los cuales permiten el logro de los objetivos.

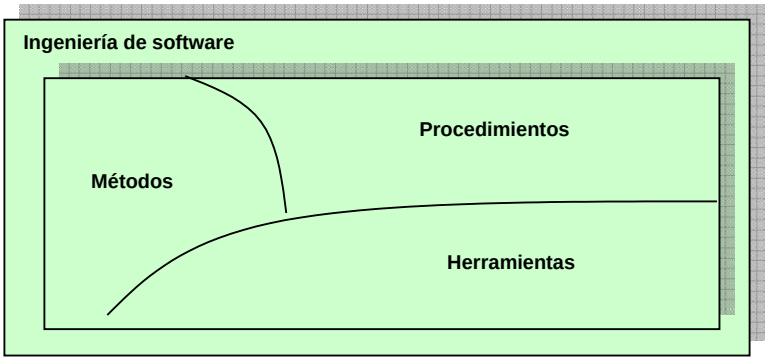


Figura 2.2 Elementos que integran la Ingeniería de Software.

El ciclo de vida que integran los sistemas de información incluye diversas etapas o fases. El número o clasificación de estás,

depende del enfoque de cada persona o autor. Independientemente del número de fases analizadas, se concluyó que todas ellas integran un paradigma generalizado [22], en la Figura 2.3 se muestra este paradigma.

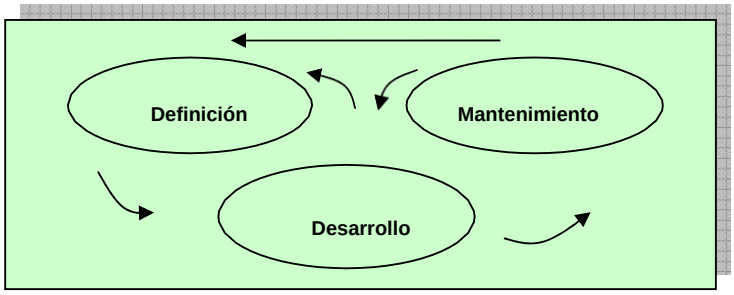


Figura 2.3 Paradigma generalizado de la ingeniería de software.

Durante la fase de definición, el desarrollo del sistema de información identifica la información a procesar, su función, rendimiento, interfaz y los criterios de validación necesarios para definir un sistema correcto. Así mismo, la fase de desarrollo se enfoca al "cómo". Durante esta fase el desarrollador, intenta descubrir cómo han de implantarse los detalles de los procedimientos, cómo ha de trasladarse el diseño a un lenguaje de programación y cómo ha de realizarse la prueba del producto.

Por otra parte la fase de mantenimiento se enfoca a la integridad de errores, adaptaciones requeridas por la evolución del sistema de información y modificaciones de los requerimientos del cliente o usuario para reforzar o aumentar el sistema. Pressman define cada una de estas fases de la siguiente manera:

a) Subsistemas de la fase de definición, los sistemas de la fase de definición se muestran en la Figura 2.4 y se describen a continuación:

Análisis: Define las razones y justificaciones de los sistemas de información.
Planeación: Estimación de recursos, definición de responsabilidades y actividades así como su secuencia de ejecución.
Análisis de Requerimientos: Se define "el qué" de los sistemas de información.

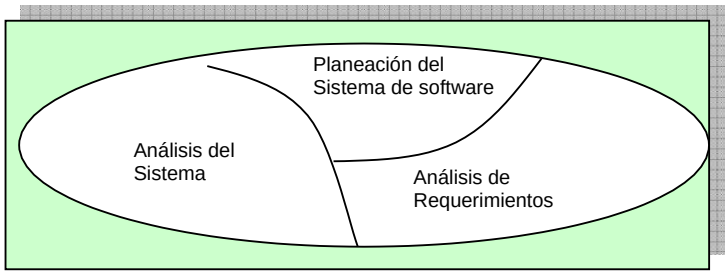


Figura 2.4 Elementos de la fase de Definición.

b) Subsistemas de la fase de desarrollo, se muestran en la Figura 2.5 y se describen a continuación:

Diseño: Convierte los requerimientos de los sistemas de información a representaciones, como tablas, gráficas, basadas en lenguajes. Son estructuras de datos.
Codificación: Conversión del diseño de sistemas de información a instrucciones ejecutables por computadora.
Prueba: Procedimientos para verificar que no existen errores.

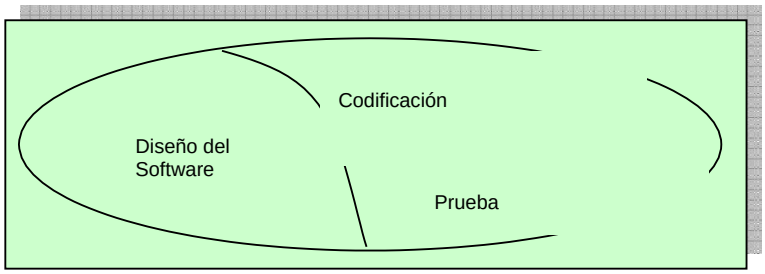


Figura 2.5 Elementos de la fase de Desarrollo.

c) Subsistemas de la fase de Mantenimientos, se muestran en la figura 2.6 y se describen a continuación:

Integridad: Eliminar errores que surgen en la etapa de prueba.
Adaptación: Adecuar el sistema de información al entorno externo.
Funcionalidad nueva: Otorga funciones adicionales a los requerimientos originales del sistema de información.

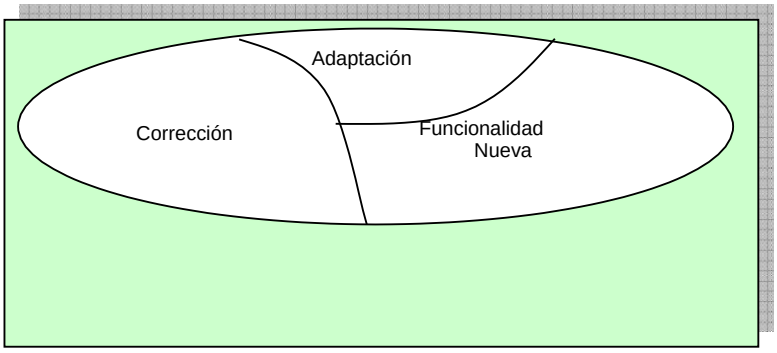


Figura 2.6 Elementos de la fase de Mantenimiento.

La fase de planeación y definición, particularmente el subsistema de requerimientos de los sistemas de información. Figura 2.7

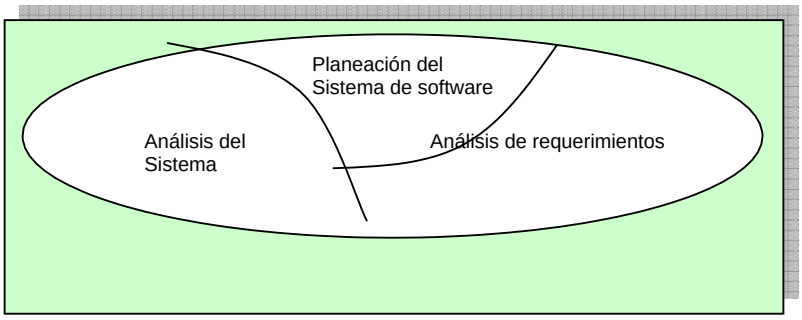


Figura 2.7 Análisis de requerimientos de la fase de definición.

El estudio de las fases de desarrollo y mantenimiento quedan fuera del alcance y propósito de este trabajo.

2.4.2. Conclusión

Se concluye al respecto que, con el fin de formalizar más el proceso de desarrollo de productos de software, se creó el concepto de ingeniería de software. De una manera simple, la ingeniería de software puede ser definida como *"el establecimiento y uso de principios de ingeniería robustos, orientados a obtener económicamente software que sea confiable y funciones eficientes sobre máquinas reales"* [23].

2.5 Calidad de Software

Definición

El término puede ser ambiguo e incluso subjetivo porque, como la belleza, la calidad depende de quien la observa. Es necesario definir el concepto con claridad, ya que si la calidad no puede ser definida, no puede ser medida; y donde la calidad no puede ser medida entonces no puede ser controlada [30]. Para trabajar sobre un esquema consistente y evitar ambigüedades, a continuación se ofrece la definición de calidad ofrecida por la organización ISO [14], siendo esta "La totalidad de características de un producto, proceso o servicio que cuenta con la habilidad de satisfacer necesidades explícitas o implícitas".

Para complementar la definición, dado que el concepto calidad puede ser subjetiva y debido a que las necesidades explícitas o implícitas varían de organización en organización o de usuario en usuario [20], es esencial identificar dichas necesidades para el usuario o para la organización [23].

2.5.1. Calidad en el Software

Dentro del contexto de Ingeniería de Software, se tomará la definición de calidad en el software propuesta por la organización internacional de estándares (ISO/IEC DEC 9126): La totalidad de características de un producto de software que tienen como habilidad, satisfacer necesidades explícitas o implícitas. Otra definición bastante completa de calidad en el software es la que se presenta más adelante [35]: Se puede decir que el software tiene calidad si cumple o excede las expectativas del usuario en cuanto a:

1. Funcionalidad (que sirva un propósito),
2. Ejecución (que sea práctico),
3. Confiabilidad (que haga lo que debe),
4. Disponibilidad (que funcione bajo cualquier circunstancia) y
5. Apoyo, a un costo menor o igual al que el usuario está dispuesto a pagar.

Resumiendo podemos decir, que la calidad de software se refiere a: *"Los factores de un producto de software que contribuyen a la satisfacción completa y total de las necesidades de un usuario u organización".*

2.5.2. Aseguramiento de Calidad de Software (SQA)

La función de aseguramiento de la calidad tiene como finalidad primaria el determinar si las necesidades de los usuarios están siendo satisfechas adecuadamente. Otra de sus funciones, aunque no se tocará mucho en la presente investigación, es la de determinar los costos que puede causar el añadir ciertas características al producto, ya que tarde o temprano, la economía resulta ser un factor decisivo para obtener un producto de calidad. Para determinar si las necesidades de los usuarios están siendo satisfechas, se deben de evaluar tres áreas:

Objetivos: Los objetivos de la organización son primero, luego vienen los requerimientos del usuario. Los objetivos de cualquier usuario deben de estar en armonía con los objetivos de la organización,
Métodos: Deben de utilizarse métodos que contengan u observen las políticas, procedimientos y estándares de la organización,
Ejecución: Optimización del uso de hardware y software al implementar los productos de software [21].

Para evaluar las áreas expuestas con anterioridad, es necesario que se cuente con un programa de aseguramiento de calidad que sea efectivo y que tenga un impacto dentro del desarrollo y prueba del producto de software final.

2.5.3. Necesidad de la Calidad y de sus Procesos de Aseguramiento.

La calidad en el software es imprescindible. Las organizaciones de la actualidad se encuentran en una situación donde deben idear estrategias que las pongan en ventaja con sus competidores y las tecnologías de información son herramientas usualmente escogidas con este propósito.

Estas razones, que sustentan lo anteriormente escrito, son:

La naturaleza crítica de algunas tareas realizadas por las computadoras. En la actualidad se está dando una creciente dependencia de los sistemas computacionales, donde alguna falla puede resultar en catástrofes personales (sistemas de control aéreo, en los aeropuertos) y económicas (sistemas transaccionales en los bancos).

El crecimiento de los costos de desarrollo de productos de software. Los costos causados por mantenimiento de software son cada vez mayores, por lo que se vuelve indispensable evitar errores desde la definición de requerimientos [17].

La competencia entre los desarrolladores de productos de software para producir software de alta calidad, como un medio para ganar mercado.

Sin embargo, pese a que se conoce la necesidad de producir software de calidad, la cultura actual de la calidad enseña que en las organizaciones los administradores empiezan a involucrarse en los procesos de desarrollo de tecnología de información una vez que se han incurrido en costos de mantenimiento, ya sean ocasionados por un mal diseño, o por no satisfacer los requerimientos correctamente [21].

2.5.4. Beneficios de los procesos de Aseguramiento de la Calidad en el Software.

Los beneficios que se pueden obtener como resultado de aplicar los procesos de aseguramiento de calidad son muchos y variados, algunos que se pueden citar con brevedad son:

1. Se detectan problemas rápidamente, Es posible identificar problemas en tempranas etapas del desarrollo de productos de software, ayudando al desarrollador a corregirlos inmediatamente y poder avanzar con más rapidez.

.

2. Se crean y se siguen estándares de trabajo, Con apoyo del proceso de aseguramiento de calidad, se pueden establecer estándares tan diversos como son los de codificación o de documentación, los cuales apoyan a uniformizar y consolidar el proceso de desarrollo.

.

3. Se verifica que los objetivos individuales vayan acordes con los objetivos de la organización, Se busca y se recomienda que los requerimientos expuestos por usuarios finales estén alineados con los objetivos globales de la empresa, facilitando así el logro de los mismos y la integración total de los usuarios a la organización.

.

4. Se recomiendan métodos para realizar el trabajo, Las prácticas de aseguramiento de calidad, como son muy robustas ya que aplican técnicas muy completas de medición, pueden proponer en un momento dado qué métodos se ajustan más a la naturaleza del producto a ser desarrollado, teniendo como efecto final que el producto tenga más posibilidades de ser un producto con calidad.

.

5. **Se evita incurrir en costos innecesarios,** Como un efecto generalizado de algunos de los puntos mencionados con anterioridad, la práctica de procesos de aseguramiento de calidad lleva a las organizaciones a evitar costos no deseados como pueden ser todos aquellos ocasionados por mantenimiento correctivo.

.

6. **Se planea la calidad,** Está claro que el concepto de calidad no es algo que se da de una manera automática e impredeciblemente. Es algo que se busca. Por lo mismo, se debe de planear, construir e implantar en el producto [28].

2.5.5. Problemas y costos del Aseguramiento de la Calidad en el Software

Uno de los principales problemas con los que se encuentra la actividad de aseguramiento de la calidad en el software es la falta de apoyo por parte de la alta dirección de las organizaciones. Este apoyo es esencial para que la función de aseguramiento de calidad tenga éxito. Los costos económicos de la función de aseguramiento de la calidad en el software se han estimado que varía entre un 2.5 y 5 por ciento del costo total de un proyecto de desarrollo de un producto de software.

El costo se localiza en las actividades (como son revisiones periódicas y constantes de las aplicaciones) que tienen que realizar algunos desarrolladores de software, mismas que se deben de integrar a sus actividades ordinarias [21].

2.5.6. Control de la Calidad

Son las Técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales:

1. Mantener bajo control los Productos de Desarrollo de software,
2. Eliminar las causas de los defectos en las diferentes fases del ciclo de vida, que puedan presentarse en los desarrollos de los productos de software.

2.5.7. Aseguramiento de Calidad vs. Control de la Calidad.

Es de suma importancia entender las diferencias que existen entre el control de la calidad y el aseguramiento de la calidad. El aseguramiento de la calidad aprovecha los resultados del control de calidad para evaluar y mejorar los procesos con los que se desarrolla el producto. Esto dicho, el control de calidad se enfoca en productos, mientras que el aseguramiento de la calidad lo hace en los procesos [22].

2.5.8. Gestión de la Calidad.

Gestión de la Calidad [ISO QS/900], conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implantan por medios tales como la planificación de la calidad, el control de la calidad, el aseguramiento (garantía) de la calidad y la mejora de la calidad, en el marco del sistema de calidad.

2.5.9. Sistema de Calidad

Los sistemas de calidad representan la estructura organizativa, procedimientos, procesos y recursos necesarios para implantar la gestión de calidad, en donde esta debe de adecuarse a los objetivos de calidad de la empresa. La dirección de la empresa es la responsable de fijar las políticas de calidad y las decisiones relativas a iniciar, desarrollar, implantar y actualizar el sistema de calidad. Un sistema de calidad consta de varias partes:

Documentación: Manual de Calidad, es el documento principal para establecer e implantar un sistema de calidad. Puede haber manuales a nivel de empresa, departamento, producto, específicos (compras, proyectos).

Parte Física: Locales, herramientas, ordenadores, etc.

Aspectos Humanos: Formación de personal, Creación y coordinación de equipos.

Algunas Normativas de Calidad en los sistemas de información que ayudan a la realización, al aplicar mejores prácticas en las organizaciones son:

- ISO 9000, gestión y aseguramiento de calidad (conceptos y directrices generales).
- Recomendaciones externas para aseguramiento de la calidad (ISO 9001, ISO 9002, ISO 9003).
- Recomendaciones externas internas para aseguramiento de la calidad (ISO 9004).
- MALCOM BALDRIGE NATIONAL QUALITY AWARD
- Software Engineering Institute (SEI).
- Capability Maturity Model (CMM).

Por mencionar algunas de ellas.

2.5.10. Certificación de la Calidad

Un sistema de certificación de calidad permite una valoración independiente y que intenta demostrar que la organización es capaz de desarrollar productos y servicios de calidad. Los pilares básicos de la certificación de calidad son tres:

a) Una metodología adecuada,
b) Un medio de valoración de la metodología,
c) La metodología utilizada como el medio de valoración de la metodología que deben estar reconocidos ampliamente por la industria.

2.5.11. Factores que determinan la Calidad de Software.

La calidad, para poder ser entendida de una mejor manera y posteriormente ser medida con eficacia, debe ser expresada por medio de otros términos que tengan más sentido para el usuario, o en este caso, para el desarrollador. Estos factores son el medio por el cual se traduce el término “calidad” al lenguaje de las personas que manejan la tecnología [16].

2.5.12. Clasificación de Factores de Calidad

Éstos se clasifican en tres grupos:

1. Operaciones del producto, Características operativas
Corrección (¿Hace lo que se le pide?)
El grado en que una aplicación satisface sus especificaciones y consigue los objetivos encomendados por el cliente.
Fiabilidad (¿Lo hace de forma fiable, todo el tiempo?)
El grado que se puede esperar de una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.
Eficiencia (¿Qué recursos hardware y software necesito?)
La cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados.
Integridad (¿Puedo controlar su uso?)
El grado con que puede controlarse el acceso al software o a los datos a personal, no autorizado.
Facilidad de uso (¿Es fácil y cómodo de manejar?
El esfuerzo requerido para aprender el manejo de una aplicación, trabajar con ella, introducir y conseguir resultados.

2. Revisión del Producto, Capacidad para soportar cambios
Facilidad de Mantenimiento (¿Puedo localizar los fallos?)
El esfuerzo requerido para localizar y reparar errores
Flexibilidad (¿Puedo añadir nuevas opciones?)
El esfuerzo requerido para modificar una aplicación en funcionamientos
Facilidad de prueba (¿Puedo probar todas las opciones?)
El esfuerzo requerido para probar una aplicación de forma que cumpla con lo especificado en los requisitos.

3. Transición del Producto, Adaptabilidad a nuevos entornos
Portabilidad (¿Podré usarlo en otra máquina?)
El esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.
Reusabilidad (¿Podré utilizar alguna parte del software en otra aplicación?)
Grado que define que partes de una aplicación pueden utilizarse en otras aplicaciones.
Interoperabilidad (¿Podrá comunicarse con otras aplicaciones o sistemas informáticos?)
El esfuerzo necesario para comunicar la aplicación con otras aplicaciones o sistemas informáticos.

La calidad en el producto final puede ser determinada como una función parametrizada por los factores de calidad, donde el peso de cada uno de los factores depende de los requerimientos del usuario [30]. La importancia de los factores de calidad radica en que dichos factores hacen posible medir objetiva y absolutamente el cumplimiento de los requerimientos [8].

Una forma más en la que los factores de calidad pueden definirse es como lo manifiestan Pressman y Perry, en el esquema que se presenta a continuación en la Figura 2.8

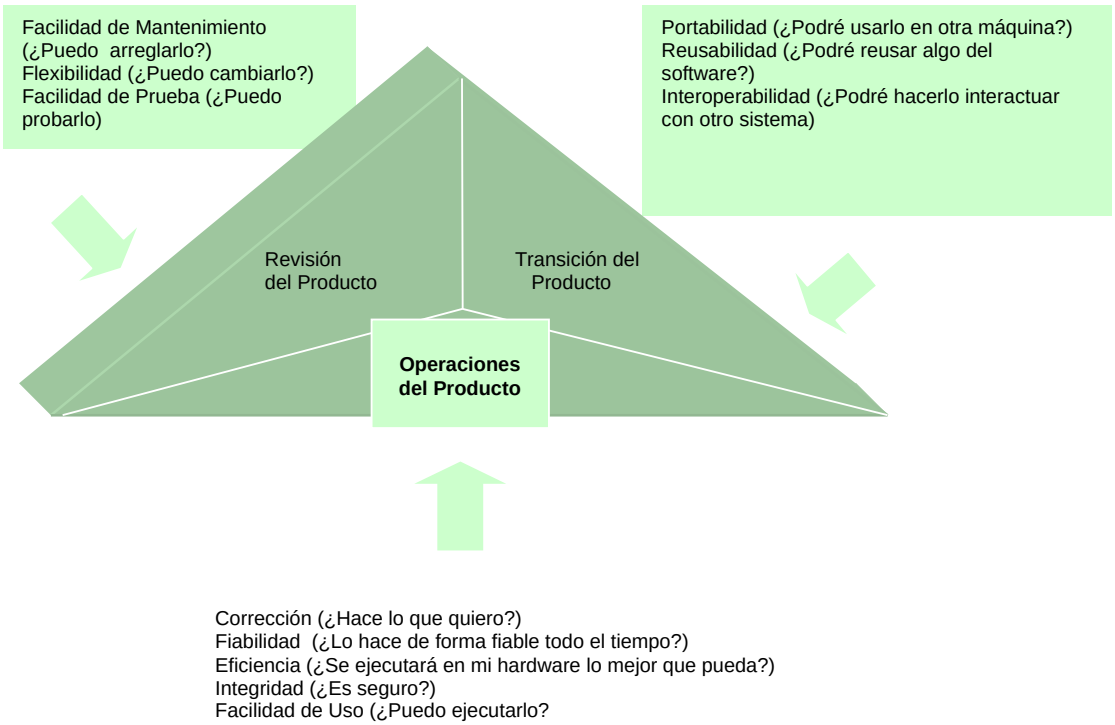


Figura 2.8 Factores de Calidad según Pressman y Perry.

2.5.13. Factores de Calidad a utilizar en esta propuesta.

Para este trabajo, se utilizan aquellos factores que son comúnmente más aceptados en el área de Sistemas de Información, tomando como referencia las fuentes citadas anteriormente y a partir de ello, se adecuan al propósito del estudio.

Los factores de calidad utilizados para el estudio son:

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

Confiabilidad
El grado en que se puede esperar que un producto de software lleve a cabo sus funciones esperadas con la precisión requerida.

Eficiencia
La cantidad de recursos computacionales y de código requeridos por un producto de software para llevar a cabo las funciones encomendadas.

Integridad
El grado en que puede controlarse (facilitar y restringir) el uso y acceso al software y a los datos, tanto al personal autorizado como al no autorizado.

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

Facilidad de mantenimiento
El esfuerzo necesario para localizar y corregir los errores en un producto de software.

Flexibilidad
El esfuerzo requerido para modificar un producto de software una vez que se encuentra ya liberado o en producción, esto es, una vez que el usuario esté haciendo uso de él.

Facilidad de prueba
El esfuerzo requerido para probar un producto de software, de tal forma que se asegure que realiza las funciones especificadas por el usuario.

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

Reusabilidad
El grado en que un producto de software (o alguna de sus partes) pueda volver a ser utilizado en otras aplicaciones, aún cuando la funcionalidad de la misma cambie.

Facilidad de interoperación
El esfuerzo requerido para lograr que un producto de software trabaje con otro, compartiendo recursos.

2.5.14. Entorno de los Productos de Software

Para poder identificar los factores de calidad que son más relevantes en un producto de software, es importante tomar en cuenta las características básicas del mismo.

> Con intención de presentar una guía sólida, se muestra en la siguiente sección, el conjunto de características que afectan directamente la calidad del software. ***(2.5.14.1)***

Con respecto a las características del Entorno que afectan el aseguramiento de la calidad, es importante tomar en cuenta que puede existir un sin fin de elementos cuyas características puedan afectar el aseguramiento de calidad de software. También es necesario que discriminemos los factores de calidad que son importantes de los que no lo son tanto.

A continuación se exponen las características más importantes y que se pueden resumir en tres tópicos principales:

1. Características que se derivan del uso del software,
2. Características que se asocian con el desarrollo,
3. Consideración del software como parte de un sistema.

2.5.14.1.-Características del software utilizado.

Aplicación
La naturaleza/finalidad de la aplicación para la que fue desarrollado el software puede determinar algunos factores de calidad. La mayoría de las aplicaciones de cierta área ya cuentan con un método de desarrollo, estándares, lenguajes de programación y plataformas de hardware.

****

Ambiente de uso
El ambiente de uso de los productos de software puede ser extremadamente difícil de definir. La palabra ambiente implica desde el lugar donde se encuentra la computadora que corre la aplicación, hasta las circunstancias con las que se puede encontrar.

Riesgos y consecuencias de fallas
Las consecuencias varían ampliamente, desde contratiempos triviales hasta catástrofes donde se ven en peligro vidas humanas o la ecología.

****

Computadora Anfitriona (Host computer)

La computadora donde corra el software puede llegar a tener delimitaciones que atenúen o acentúen ciertos factores de calidad. Los productos de software deben de ejecutarse en más de una computadora con más de un sistema operativo.

****

Madurez

La experiencia que puedan haber adquirido los desarrolladores en el diseño de aplicaciones similares puede facilitarles la identificación de factores de calidad.

****

Experiencia del usuario

La mayoría de los productos de software serán usados por usuarios cuyo nivel de experiencia variará desde los usuarios novatos hasta los expertos, por lo que las dificultades percibidas y actuales serán diferentes para cada producto de software. Los desarrolladores deben de contemplar esto para reforzar ciertos factores de calidad en el producto final (Ej. Si el usuario es novato, el factor facilidad de uso se vuelve de suma relevancia).

****

Apoyo de los productores

El nivel de apoyo que sea disponible por parte del productor/desarrollador (soporte técnico) puede tener una gran influencia en la integridad del producto de software (Ej. si no se contara con mucho apoyo por parte de los desarrolladores posterior a la entrega del producto, el factor facilidad de mantenimiento adquiere mucha importancia).

****

2.5.14.2. Características asociadas al desarrollo de software.

Las características asociadas al desarrollo del software, son las siguientes.

Experiencia del productor
La experiencia del desarrollador de software debe ser también considerada. La madurez de la organización, la habilidad del recurso humano, los métodos de desarrollo empleados, familiaridad con las aplicaciones son todos factores que han mostrado influencia en la calidad de un producto de software.

****

Interacción con el usuario final
Además del apoyo que debe darle el desarrollador al usuario final una vez entregado el producto, el nivel de interacción con el usuario para formular los requerimientos es un factor importante.

****

Restricciones comerciales
La calidad del producto final se puede ver influenciada por algunas restricciones comerciales. Algunas de estas restricciones pueden ser el presupuesto, tiempos de entrega y las consecuencias de los riesgos, las cuales pueden influir en el método de desarrollo, diseño y el nivel de mantenimiento y apoyo que se le puede dar al producto de software.

****

Metodología en uso
Hoy en día, existen muchas metodologías en uso, algunas muy maduras con estándares sólidos, otras más nuevas que vienen emergiendo. La madurez de estas metodologías (y los logros y -experiencia que se tengan) pueden influenciar la calidad requerida.

****

Lenguaje de programación
El lenguaje de programación que sea seleccionado para la implementación de un producto de software puede traer consigo restricciones propias, como puede ser la dificultad de migrar de una plataforma a otra.

2.5.14.3. Características del software como parte de un sistema.

Sistema mínimo

En ocasiones, un programa debe ser asociado a un procesador antes de ser usado, lo cual forma un sistema mínimo. Tal es el caso, aunque no muy común, de los programas que son parte integral de un procesador (chip en una computadora) y que están "microprogramados".

Sistema típico

Con más frecuencia, la computadora y los productos de software son administrados por un sistema operativo, el cual interactúa a través de equipo periférico (hardware) con un operador y otros equipos como los dispositivos de almacenamiento, monitores, impresoras, etc.

Sistema complejo

Los sistemas más difíciles de asegurar la calidad son aquellos donde se involucra el control -a través de la computadora- de sistemas electrónicos, dispositivos mecánicos o hidráulicos, procesos o plantas.

2.5.14.4. Características del entorno de los productos de software a utilizar en esta propuesta.

Para facilitar su conceptualización, se proponen definiciones sencillas y prácticas a continuación:

Aplicación
Representación del giro sobre el cual el software va dirigido (negocios, ingeniería, medicina, etc.).

****

Ambiente de Uso
Espacio físico y condiciones en que se utiliza el software.

****

Riesgos y consecuencias de fallas
Lo que puede implicar que el software falle.

****

Computadora anfitriona
La computadora donde correrá el software.

****

Madurez del desarrollador
Experiencia en el desarrollo de software similar.

****

Experiencia del usuario
La familiaridad de los usuarios con computadoras o software similares

****

Experiencia del usuario
La familiaridad de los usuarios con computadoras o software similares

****

Apoyo de los desarrolladores
Asesoría o ayuda por parte de los desarrolladores.

****

Experiencia de los desarrolladores
Tiempo trabajado en desarrollos de software similares.

****

Interacción con el usuario final
Comunicación ente usuarios y desarrolladores.

****

Restricciones comerciales
Falta de presupuesto o de tiempo.

****

Metodología de desarrollo
El uso de algún método establecido para la creación del software.

****

Lenguajes de programación
El lenguaje o paquete computacional seleccionado para desarrollar el software

****

Complejidad del software
Grado en que se tenga que involucrar a muchos elementos físicos (periféricos), que de alguna manera ayudan a la ejecución del software.

****

2.5.14.5. Identificación de los factores de calidad relevantes para un producto de software.

Los requerimientos de calidad de software para cada producto de software son únicos y son influenciados por características dependientes de la aplicación. Existen características que afectan los requerimientos de calidad, y cada software debe ser evaluado por sus características básicas [21].

Algunos ejemplos claros de cómo algunas características organizacionales o del entorno pueden afectar los requerimientos de calidad, son los siguientes:

- Si el ciclo de vida de un sistema se espera que sea muy largo es muy probable que se requiera hacer modificaciones elementales al sistema sin que éste deje de operar por lo que entonces el factor facilidad de mantenimiento debe tomar mucha importancia.

- Si la aplicación es un sistema experimental donde las especificaciones finales de los usuarios varían con una razón acelerada, los cambios y añadiduras de última hora empiezan a fluir por lo que la flexibilidad del producto se vuelve una prioridad.

- Si las funciones de un sistema se espera que sean requeridas por mucho tiempo mientras que el sistema cambia y se desarrollan otros sistemas donde se requieran funciones iguales o similares a las ya existentes, la Reusabilidad es un factor que debe estar presente en los módulos que implementan las funciones principales del sistema.

- Si el uso de una aplicación puede poner en peligro la vida de seres humanos, dicha aplicación se espera que funcione siempre y que además lo haga siempre bien por lo que los factores de confiabilidad correctitud y facilidad de prueba se vuelven indispensables.

- En un complejo de redes computacionales donde la facilidad de comunicación es prioridad los sistemas que operan dichas redes deben tener una fuerte interfaz con otros sistemas por lo que el concepto de interoperabilidad es extremadamente importante [21].

Se necesitan requerimientos de calidad que establezcan desde un principio lo que se espera del producto final. Es absolutamente necesario que se trabaje en identificar los factores de calidad relevantes para cada tipo de producto a desarrollar [30].

Los requerimientos de calidad son un rubro más que hay que cubrir junto con los requerimientos funcionales, de ejecución, de costos y de calendarización que normalmente son especificados para el desarrollo de software [21].

2.5.14.6. Identificación de los requerimientos de calidad de un Producto de Software

William Perry propone una manera de identificar los requerimientos de calidad de un producto de software. Para ilustrar este argumento, se muestra la Figura 2.9 donde se aprecia en el nivel más alto un conjunto de factores que son definidos con términos de los usuarios. En el siguiente nivel, para cada factor se muestra un conjunto de criterios que en realidad son atributos que, si se encuentran presentes, proveen características representadas por los factores de calidad. Por último, en el nivel más bajo del modelo nos encontramos con métricas, las cuales son medidas cuantitativas de los atributos del software definidos por los criterios.

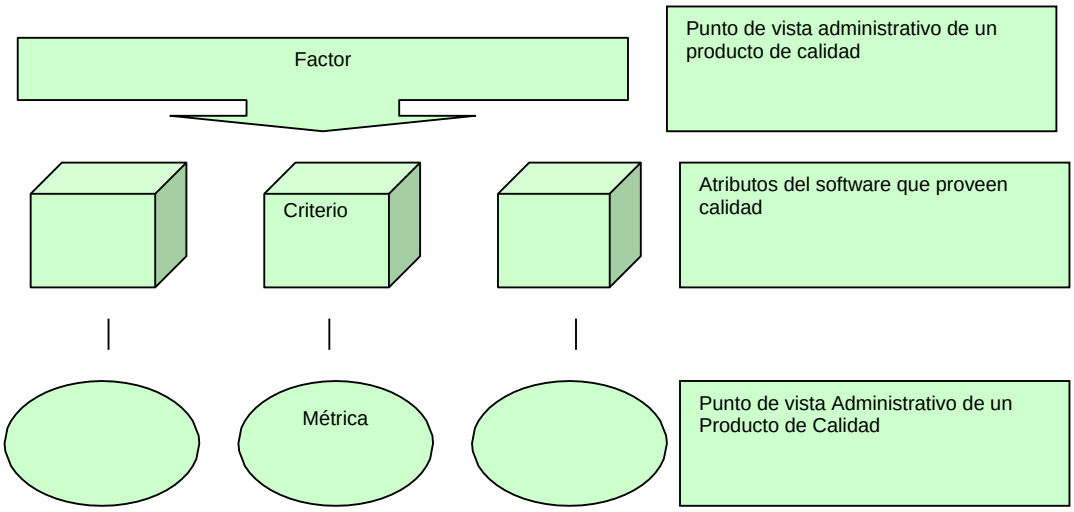


Figura 2.9 Modelo jerárquico de calidad del software

El modelo de Perry es un modelo de tres niveles, de los cuales en esta sección se tratarán los dos primeros, que corresponden a la determinación de los factores de calidad. La herramienta básica para identificar los factores de calidad más importantes es la encuesta de requerimientos de calidad de software (Software Quality Requirements Survey) que se muestra en la Figura 2.10 En ocasiones, es necesario mostrar información adicional a los encuestados para que tomen las decisiones correctas, tal y como se muestra en la figura 2.11

1. Los 11 factores de calidad listados abajo han sido tomados de la literatura actual. No es una lista exhaustiva, pero si representativa de lo que se cree es importante. Por favor indica si consideras cada factor Muy Importante (MI), Importante (I), Poco Importante (PI) o No Importante (NI) en el diseño de objetivos del sistema en el cual trabajas actualmente.

Respuesta Factores	Definiciones
Corrección:	Grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el usuario final.
Confiabilidad:	Grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida.
Eficiencia:	Cantidad de recursos de una computadora y de código requeridos por un programa para llevar a cabo sus funciones.
Integridad:	Grado en que puede controlarse el acceso al software o a los datos por personal no autorizado.
Facilidad de uso:	Esfuerzo requerido para aprender, trabajar, preparar las entradas e interpretar las salidas de un programa.
Fac. de mantenimiento:	El esfuerzo requerido para localizar y arreglar un error en un programa.
Flexibilidad:	Esfuerzo requerido para modificar un programa que se encuentre en operación.
Fac. de prueba:	El esfuerzo requerido para probar un programa de manera que se asegure que realiza su función requerida.
Portabilidad:	El esfuerzo requerido para transferir un programa desde una plataforma de hardware y/o entorno de software a otro.
Reusabilidad:	Grado en que un programa (o partes de él) se pueden volver a utilizar en otras aplicaciones. Esto se relaciona con el alcance de las funciones que realiza el programa.
Interoperabilidad:	El esfuerzo requerido para acoplar un sistema a otro.

2. En qué tipos de aplicaciones estás involucrado en estos momentos?

_____

3. Estás involucrado en:
1. Fase de desarrollo:

_____

2. Fase de Operación/Mantenimiento:

_____

4. Por favor indica el puesto que más se asemeja al tuyo:
__ Administrador de programas
__ Consultor técnico
__ Analista de sistemas
__ Otro (por favor especifica) _____

Figura 2.10 Encuesta de Requerimientos de calidad de software

Característica	Factor de Calidad
Si vidas Humanas son afectadas	Reusabilidad Corrección Facilidad de Prueba
Ciclo de Vida Largo	Facilidad de mantenimiento Flexibilidad Portabilidad
Aplicación de Tiempo Real	Eficiencia Mantenimiento Confiabilidad Corrección
Procesos de Información clasificada	Integridad
Sistema Iterrelacionado	Interoperabilidad

Figura 2.11 Ejemplo de Información adicional a la encuesta de Requerimientos de Software

Para completar la encuesta se necesita seguir los siguientes procedimientos:

- Considerar las características básicas de la aplicación (o del entorno, como fueron anteriormente descritas), los requerimientos de calidad de todas las aplicaciones son únicos y éstos son influenciados por características básicas de la aplicación.

- Considerar las implicaciones del ciclo de vida. Los once factores de calidad identificados en la encuesta pueden ser agrupados de acuerdo a las tres actividades del ciclo de vida asociadas a la entrega de producto de software. Estas actividades son: operación, revisión y transición del producto. La relación de los factores de calidad con estas actividades es mostrada en la Figura 2.12.

Esta figura muestra dónde se deben hacer las mediciones de los factores de calidad, dónde se puede sufrir el impacto de una calidad deficiente y la relación del ahorro por prevenir errores (causados por calidad deficiente) versus el costo por proveer calidad. El impacto de una calidad deficiente es el que determina el ahorro en los costos, si se define como “costo” lo que se dejaría de perder si el producto de software tuviera una calidad superior. Al mismo tiempo, el ahorro en los costos pudiera ser insuficiente, debido a que en algunas ocasiones los costos en los que se tienen que incurrir para aplicar métricas y por desarrollar productos de software de alta calidad pueden ser mayores. Esto debe ser siempre puesto en comparación (ver gráfica 2.13).

	Desarrollo			Eval.	Postdesarrollo			
FASES / FACTORES	Req. de Análisis	Diseño	Codf.	Prueba de Sistema	Operac.	Revisión	Transición	Ahorro esperado en Costo por Proveedor
Corrección	△	△	△	X	X	X		ALTO
Confiabilidad	△	△	△	X	X	X		ALTO
Eficiencia	△	△	△		X			BAJO
Integridad	△	△	△		X			BAJO
Facilidad de Uso	△	△		X		X		MEDIANO
Facilidad de Mannto.		△	△			X	X	ALTO
Facilidad de Prueba		△	△	X		X	X	ALTO
Flexibilidad		△	△			X	X	MEDIANO
Portabilidad		△	△				X	MEDIANO
Reusabilidad		△	△				X	MEDIANOV
Interoperabilidad		△		X			X	BAJO

△ Donde los Factores deben ser Medidos

X Donde se detecta el impacto de una Calidad Pobre

Figura 2.12 Relación de los factores de calidad con las actividades del ciclo de vida de los productos de productos de software.

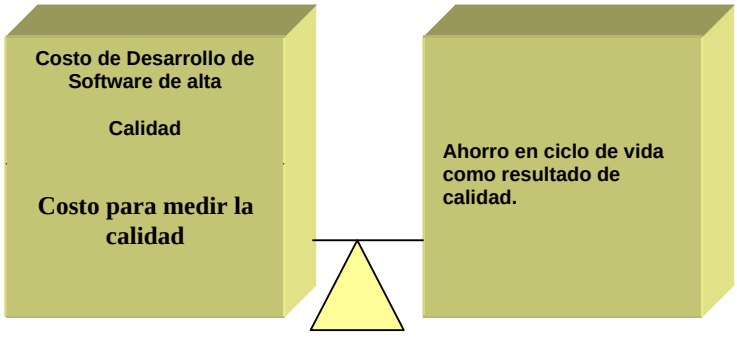


Figura 2.13 Consideración de costos de desarrollo y ahorros en el ciclo de vida de un producto de software.

FACTORES	CORRECCION	CONFIABILIDAD	EFICIENCIA	INTEGRIDAD	FACILIDAD DE USO	FACILIDAD DE MANTTO	FACILIDAD DE PRUEBA	FLEXIBILIDAD	PORTABILIDAD	REUSABILIDAD	INTEROPERABILIDAD
CORRECCION											
CONFIABILIDAD	○										
EFICIENCIA											
INTEGRIDAD			●								
FACILIDAD DE USO	○	○	●	○							
FACILIDAD DE MANTENIMIENTO	○	○	●		○						
FACILIDAD DE PRUEBA	○	○	●		○	○					
FLEXIBILIDAD	○	○	●	●	○	○	○				
PORTABILIDAD			●			○	○				
REUSABILIDAD			●	●		○	○	○	○		
INTEROPERABILIDAD			●	●			○				

● Alto ○ Bajo

LEYENDA

En Blanco = No hay relación o dependencia de la aplicación.

NOTA: Si un alto grado de calidad es presentar por factor qué grado de calidad es esperado para el otro.

Figura 2.14 Relación entre los factores de calidad.

- Realizar "trueques" entre la lista tentativa de factores de calidad. Como resultado de los pasos anteriores, se debe obtener una lista tentativa de factores de calidad importantes. El siguiente paso es considerar las relaciones entre los factores seleccionados. La Figura 2.14 puede ser una buena guía para determinar las relaciones entre los factores de la lista. Algunos factores funcionan en sinergia con los demás, mientras que otros producen conflicto. El resultado de que un factor cause conflicto se refleja en el incremento en los costos de implementación. Esto a su vez, provoca que la razón beneficio versus costo-por-proveer disminuya.

- Identificar los factores de calidad más importantes. La lista de factores de calidad considerados como importantes para un producto de software obtenidos después de seguir los tres procedimientos anteriores- deben ser organizados por orden de importancia. Las definiciones de los factores elegidos deben ser proporcionadas en la lista.

- Proveer explicaciones para cada elección. Es importante documentar todas las decisiones hechas durante los primeros tres pasos. Como se puede observar, la propuesta del Señor Perry para identificar los requerimientos de calidad de un producto de software parte de un modelo jerárquico de tres niveles. Para la presente investigación sólo se considerarán los primeros dos niveles (factores y criterios de calidad), ya que el tercer nivel involucra el uso de métricas.

2.5.14.7. Control Interno de la Calidad de Software

Tradicionalmente en materia de control interno se adoptaba un enfoque bastante restringido limitado a los controles contables internos. En tanto se relacionaba con la información financiera, el control interno era un tema que interesaba principalmente al personal financiero de la organización y, por supuesto, al auditor externo. Las tendencias externas que influyen en las empresas son, entre otras, las siguientes:

1. La globalización, 2. La diversificación de actividades, 3. La eliminación de ramas de negocios no rentables o antiguas, 4. La introducción de nuevos productos como respuesta a la competencia, 5. Las fusiones y la formación de alianzas estratégicas.

Ante la rapidez de los cambios los directores toman conciencia de que para evitar fallos de control significativos deben reevaluarse y reestructurarse sus sistemas de control interno.

En muchas organizaciones, el auditor ha dejado de centrarse en la evaluación y la comprobación de los recursos de procesos, desplazando su atención a la evaluación de riesgos y la comprobación de controles. Muchos de los controles se incorporan en programas informáticos o se realizan por parte de la función informática de la organización, representado por el Control Interno Informático.

2.5.14.8. Las funciones de control interno y Auditoría Informática

El control interno informático controla diariamente que todas las actividades de sistemas de información sean realizadas cumpliendo los procedimientos, estándares y normas fijados por la Dirección de la Organización y/o la Dirección de Informática así como los requerimientos legales.

Las actividades son:

1. El cumplimiento de procedimientos, normas y controles dictados, merece resaltarse la vigilancia sobre el control de cambios y versiones del software, 2. Controles sobre la producción diaria, 3. Controles sobre la calidad y eficiencia del desarrollo y mantenimiento del software y del servicio informático, 4. Controles en las redes de comunicaciones, 5. Controles sobre el software de base, 6. Controles en los sistemas microinformáticos.

2.5.14.9. Auditoría Informática

Es el proceso de recoger, agrupar y evaluar evidencias para determinar si un sistema informatizado salvaguarda los activos, manteniendo la integridad de los datos, lleva acabo eficazmente los fines de la organización, administra eficientemente los recursos.

De este modo la auditoría informática sustenta y confirma la consecución de los objetivos tradicionales de la auditoría, siendo éstos: a) Objetivos de protección de activos e integridad de datos, b) Objetivos de gestión que abarca, no solamente los de protección de los activos sino también los de eficacia y eficiencia.

El auditor evalúa y comprueba en determinados momentos del tiempo los controles y procedimientos informáticos más complejos, desarrollando y aplicando técnicas mecanizadas de auditoria, incluyendo el uso del software.

Se pueden establecer tres grupos de funciones a realizar por un auditor informático:

1. Participa en las revisiones durante y después del diseño, realización, implantación y explotación de aplicaciones informáticas, así como en las fases análogas de realización de cambios importantes,

2. Revisar y juzgar los controles implantados en los sistemas informáticos para verificar su adecuación a las órdenes e instrucciones de la Dirección, requisitos legales, protección de confidencialidad y cobertura ante errores y fraudes,

3. Revisa y juzga el nivel de eficacia, utilidad, fiabilidad y seguridad de los equipos e información.

En la siguiente Tabla, se muestra la diferencia y similitudes entre control interno informático y auditor informático:

Tabla 2.1 Diferencia y similitudes entre control interno informático y auditor informático:

Evento	**Control Interno Informático**	**Auditor Informático**
Similitudes	Personal interno Conocimientos especializados en Tecnología de la información Verificación del cumplimiento de controles internos normativa y procedimientos establecidos por la Dirección de Informática y la Dirección General para los sistemas de información.	
Diferencias	Análisis de los controles en el día a día Información a la Dirección del Departamento de Informática Sólo personal interno El alcance de sus funciones es únicamente sobre el Departamento de informática	Análisis de un momento informático determinado Informa ala Dirección General de la Organización Personal interno o externo Tiene cobertura sobre todos los componentes de los sistemas de información de la organización

2.5.14.10. Definición y tipos de controles internos

Se puede definir el control interno como cualquier actividad o acción realizada manualmente y/o automáticamente para prevenir, corregir errores o irregularidades que puedan afectar al funcionamiento de un sistema para conseguir sus objetivos.

Clasificación de los controles informáticos:

Controles preventivos: para tratar de evitar el hecho, como un software de seguridad que impida los accesos no autorizados al sistema,

Controles detectivos: cuando fallan los preventivos para tratar de conocer cuanto antes el evento,

Controles correctivos: facilitan la vuelta a la normalidad cuándo se han producido incidencias.

La auditoria en informática debe ser respaldada por un proceso formal que asegure su previo entendimiento por cada uno de los responsables de llevar a la práctica dicho proceso en la empresa. El uso de un proceso de trabajo metodológico y estándar en la función de auditoria informática genera las siguientes ventajas:

1. Los recursos orientan sus esfuerzos a la obtención de productos de calidad, con características y requisitos comunes para todos los responsables,

2. Las tareas y productos terminados de los proyectos se encuentran definidos y formalizados en un documento al alcance de todos los auditores en informática,

3. Se facilita en alto grado la administración y seguimiento de los proyectos pues la metodología obliga a la planeación detallada de cada proyecto bajo criterios estándares,

4. Facilita la superación profesional y humana de los individuos, ya que orienta los esfuerzos hacia la especialización, responsabilidad, estructuración y depuración de funciones del auditor en informática,

5. Es un complemento clave en el desarrollo de cada individuo, ya que su formal seguimiento, aunado a las habilidades, normas y criterios personales coadyuva al cumplimiento exitoso de los proyectos de auditoria en informática,

6. El proceso de capacitación o actualización en el uso de un proceso metodológico es más ágil y eficiente, dado que se trabaja sobre tareas y productos terminados perfectamente definidos.

Contar con un proceso metodológico formalmente documentado no es garantía de que los proyectos de auditoría en informática tendrán éxito; empero, no cumplir con las siguientes condiciones llevará a la función de auditoría en informática a que sus proyectos no cumplan con los tiempos, costos o resultados esperados, pero también es importante el mencionar que se puede presentar la situación en la que se justifique la revisión o evaluación de las áreas o funciones críticas relacionadas con informática.

Los productos terminados más importantes de la etapa son tres: Matriz de riesgo, Plan general de auditoría en informática, Compromiso ejecutivo. Cada uno forma parte esencial del proceso metodológico.

El primero porque define las áreas que serán auditadas; el segundo porque establece las tareas, tiempos, responsables, etc., del proyecto, y el tercero debido a que da el visto bueno al líder de proyecto para continuar con las siguientes etapas contempladas en el plan general.

2.5.14.11. Áreas de oportunidad para la función de informática

El diagnóstico actual es el que conduce al auditor en informática a vislumbrar ciertas áreas de oportunidad para el mejoramiento de alguna función específica del negocio a través de informática y en ocasiones, el beneficio es directo para el área de informática. Aquí se comienza a conformar una parte de auditoría en informática debido a que las áreas de oportunidad que ven los entrevistados en la etapa preliminar no han sido implantadas y urge aprovecharlas.

2.5.14.12. Matriz de riesgo

La siguiente tarea del auditor en informática en la presente etapa es elaborar la matriz de riesgo, cuyo objetivo principal es detectar las áreas de mayor riesgo en relación con informática y que requieren una revisión formal y oportuna. En la siguiente tabla, se representa lo relacionado a lo expuesto anteriormente.

Tabla 2.2 Matriz de Riesgos

Áreas susceptibles de auditar	**Aspectos o componentes por evaluar del área**	**Área por auditar**
Administración de informática	Misión y objetivos, Organización, Servicios, Parámetros de medición	Secuencia sugerida para auditar cada componente y área según el nivel de riesgo estimado
Dirección y niveles ejecutivos	Seguimiento a la función de información or la dirección, Comunicación e integración, Apoyo a toma de decisiones	Secuencia sugerida para auditar cada componente y área según el nivel de riesgo estimado
Usuarios de informática	Comunicación e integración, Proyectos conjuntos, Admón. de recursos, Grado de satisfacción	Secuencia sugerida para auditar cada componente y área según el nivel de riesgo estimado
Control interno	Políticas y procedimientos	Secuencia sugerida para auditar cada componente y área según el nivel de riesgo estimado
Ciclo de desarrollo e mplantación de sistemas	Metodología, Técnica, Herramienta, Capacitación/ actualización	Secuencia sugerida para auditar cada componente y área según el nivel de riesgo estimado

2.5.14.13. Conclusión

La calidad de un producto de software está sujeta a una serie de factores internos y externos llamados características del entorno, las cuales deben ser tomadas en cuenta. Si no se les da la importancia requerida, los costos por correcciones y modificaciones pueden ser muy altos o las consecuencias por resultados erróneos cuantiosas.

En resumen podemos mencionar que en la calidad de software debe de contribuir con la satisfacción completa y total de las necesidades de un usuario u organización.

2.6 Métricas de Software

Definición

Se refiere a aquella aplicación continua de técnicas basadas en la medida de los procesos de desarrollo del software, para producir una información de gestión significativa al mismo tiempo que se mejoran aquellos procesos y sus productos, se denominan métrica de software. “Un Método y una escala cuantitativos que pueden ser usados para determinar el valor que toma cierta característica en un producto de software concreto”[14].

"*Una función que toma como entrada cierta información del software que se está midiendo, y que devuelve como salida un valor numérico sencillo, el cual es interpretada, como el grado en que el producto de software posee un atributo dado que afecta a su calidad"* [11].

2.6.1. Importancia

Hay varias razones que justifican el uso de las métricas en el proceso de desarrollo de software. Por un lado se dice que cuando se puede medir aquello de lo cual se está hablando y se puede expresar en números, se sabe realmente acerca de ello; pero cuando no puede medirse, y no puede expresarse en números, el conocimiento que se tiene de ello es escaso e insatisfactorio [12].

El contar con datos estadísticos de métricas de software, da un panorama de situaciones reales que ayudan aplicar y dar seguimiento a las diferentes formas de evaluar y determinar métricas de calidad para un mejor desempeño en la calidad de software. Información reciente, de Bancomext (2001), EUA tiene un Déficit de 600,000 expertos en informática, creciendo al 10% anual; una demanda infinita para México.

Hace 15 años la India facturaba US $20 M. Hoy facturan US $6,500 M, para 2008 proyectan exportar US$50,000M. México tiene como ventaja sobre otros países una ubicación privilegiada, costos competitivos, cultura empresarial similar a la de EUA. Según la UNCTAD [33] de la ONU, el desarrollo de la industria del software en los países emergentes se puede llevar a cabo a través de dos alternativas de políticas estratégicas.

Una opción alternativa para el desarrollo de la industria del software consiste en integrar ambas estrategias bajo un orden secuencial; el uso de software libre como herramienta para el fortalecimiento de la infraestructura económica sería reemplazada gradualmente por la estrategia de desarrollo de software para exportación.

Parte de las recomendaciones de Bancomext (2001), para fomentar el desarrollo de la industria del software, se requiere de prácticas y de programas de ayuda para facilitar su crecimiento sin lastres tributarios, en especial en dos rubros: los impuestos por ingreso y facilidades para la compra e importación de tecnología.

Hay que tener en cuenta que un paso importante para incursionar en el mercado internacional es el de contar con la certificación de calidad en CMM (Capability Maturity Model CMM) [3], expedido por el SEI (Software Engineering Institute) [26].

Otras de las necesidades, según Bancomext (2001), es la de establecer una industria nacional competitiva en el desarrollo de software para competir en el mercado internacional y así introducir las mejores soluciones al Sistema e-México.

El Plan Nacional de Desarrollo 2001–2006 (PND) de la Secretaría de Economía (2001) plantea el objetivo de elevar y extender la competitividad del país, mediante la estrategia de promover el uso y aprovechamiento de la tecnología y de la información.

La producción de software es una actividad económica que se caracteriza por generar un alto valor agregado y aportar a la economía productos y servicios esenciales para su modernización.

De igual forma la AMITI [1] dice que la industria norteamericana de software, tomada como tendencia del mercado mundial de software, hace patente que esta industria representa uno de los segmentos más vibrantes de la economía cuya contribución supera a cualquier otra rama de la industria manufacturera de ese país.

En el caso de México, dicha contribución es totalmente insignificante, lo cual indica que la industria Mexicana de software no ha respondido y está dejando de aprovechar esta oportunidad económica que actualmente representa el mercado mundial de software.

Sin embargo, otros países han detectado oportunamente este 'nicho' de mercado y han respondido con la eficacia y agresividad que demanda el reto de la globalización. En México, la Industria de Software debería ser considerada como estratégica y prioritaria para todo tipo de apoyos gubernamentales.

2.6.2. Características

Se afirma que para que las métricas sean útiles estás deben tener las siguientes cuatro características:

1. Cuantificables, deben basarse en hechos, no en opiniones.
2. Independientes, los recursos no deben poder ser alterados por los miembros que las apliquen o utilicen.
3. Explicable, debe documentarse información acerca de la métrica y de su uso.
4. Precisas, debe de conocerse un nivel de tolerancia permitido cuando se mide [12].

2.6.3. Beneficios de Medir Software

Los beneficios los podemos enumerar de la siguiente forma:

- El proceso del software (para mejorarlo)
- El proyecto del software (para ayudar a estimar, control de calidad, evaluación de productividad, control de proyectos)
- Calidad del producto (para ayudarse en la toma de decisiones tácticas a medida que el proyecto evoluciona)

2.6.4. Tipos

Se afirma existen dos tipos de métricas: las métricas de producto y las métricas de proceso. Las primeras son un valor numérico extraído de algunos documentos o de una pieza de código; las segundas son un valor numérico que describe un proceso de software, por ejemplo. El tiempo que se toma para depurar un módulo o la cantidad de errores que permanecen en un sistema después de las pruebas finales.

2.6.5. Métricas de Calidad

Métricas del producto

Para justificar la existencia de este tipo de métrica, se argumenta que éstas deben ser enunciadas y utilizadas para administrar el proceso de desarrollo y debe ser conforme al producto de software particular [4]. El proveedor de productos de software debe de recopilar y actuar sobre las medidas cuantitativas de la calidad de esos productos de software.

Estas medidas deben ser utilizadas para los propósitos siguientes:

1. Para recopilar información y reportar valores de métricas sobre bases regulares,
2. Para identificar el actual nivel de desempeño por cada métrica,
3. Para tomar la acción remedial si los niveles de las métricas crecen peor o exceden los niveles objetivos establecidos,
4. Para establecer metas de mejoras especificas en términos de las mismas métricas.

2.6.6. Métricas de proceso

El proveedor debe tener métricas cuantitativas de la calidad del proceso de desarrollo y de liberación. Estas métricas deben de reflejar:

a) Qué tan bien el proceso de desarrollo está siendo llevado a cabo en términos de puntos de revisión y en objetivos de calidad en el proceso, siendo cumplidos en tiempo de calendario,

b) Qué tan efectivo es el proceso de desarrollo, al reducir la probabilidad que se introduzcan fallas o que cualquier falla introducida sea detectada.

Aquí como las partes de las métricas del producto, lo importante es que los niveles sean conocidos y utilizados para el control del proceso y de las mejoras y no sean utilizadas métricas fijas. Las métricas seleccionadas deben de ajustarse al proceso utilizado y si es posible, tener un impacto directo sobre la calidad de software liberado.

Las métricas también pueden ser categorizadas como métricas de resultado o métricas de predicción [35]. Una medición de predicción es normalmente una métrica de producto que puede ser utilizada para predecir el valor de otra métrica. La métrica es predicha, una métrica de proceso, es conocida como una métrica de resultado.

2.6.7. Uso

Conjunto de Métricas del Software, para los cuales la información debe ser recolectada y analizada, basada en el marco de referencia de madurez desarrollado por la SEI [26]. Las métricas deben ser implantadas paso a paso en cinco niveles, correspondientes al nivel de madurez del proceso de desarrollo.

2.6.7.1. Proceso Inicial (Nivel 1)

Su objetivo es formar una base de comparación con la forma en que las mejoras se realicen y se incremente la madurez, estos incluyen:

a) El tamaño del producto,
b) El esfuerzo del personal (Utilidades para determinar una tasa de productividad)

2.6.7.2. Proceso Repetible (Nivel 2)

Las métricas a este segundo nivel incluyen como objetivos de medición:

1. La cantidad de esfuerzo necesaria para desarrollar un sistema,
2. La duración del proyecto,
3. El tamaño y la volatilidad de los requerimientos,
4. El costo global del proyecto (Por lo que el tipo de métrica que se recomiendan incluye a las siguientes):

a) Tamaño del software (Líneas de codillo fuente no comentadas)
b) Puntos de Función,
c) Cuenta de objetos y métodos,

5. Esfuerzo del trabajo de personal:
   a) Esfuerzo real medido en unidades persona/mes,
   b) Esfuerzo reportado en unidades persona/mes,
6. Volatilidad de los requerimientos (Cambios de los requerimientos).

Éstas como métricas principales, además las siguientes pueden añadirse a consideración de la administración del proyecto de software:

7. Experiencia (del dominio o aplicación, de la arquitectura de desarrollo utilizada, de las herramientas y métodos empleados, años globales de experiencia en el desarrollo),
8. Rotación de personal

2.6.7.3 Proceso definido (Nivel 3)

En este nivel de madurez, se recomienda evaluar la complejidad de los requerimientos, el diseño, el código y los planes de prueba, y evaluar la calidad de los requerimientos del diseño del código y de las pruebas. En términos de complejidad, se sugiere que los siguientes puntos se midan a este nivel:

1. Complejidad de los requerimientos (Número de distintos objetos y acciones llevadas a cabo en los requerimientos).
2. Complejidad del Diseño (Número de módulos de diseño, Complejidad Ciclomática, Complejidad de Diseño de McCabe.
3. Complejidad del Código (Números de Módulos de Código, Complejidad Ciclomática.
4. Complejidad de las pruebas (Número de Caminos a probar, Si el desarrollo es orientado a objetos, debe de considerarse el número de interfaces de objetos a probar.

Se puede evaluar la minuciosidad de las pruebas. Así, por mencionar algunas métricas recomendadas de calidad, podemos decir las siguientes:

a) Defectos descubiertos,
b) Defectos descubiertos por unidad de tamaño (densidad de defectos), c) Fallas de requerimientos descubiertos,
d) Fallas de diseño descubiertas,
e) Fallas de Código descubiertas,
f) Densidad de fallas por cada producto. Se enfatiza que este conjunto no es representativo del espectro completo de medidas que pueden ser empleadas. Aspectos tales como

facilidad de mantenimientos, grado de utilización facilidad de uso y otros atributos de calidad de software que no son considerados por la cuenta de defectos.

2.6.7.4 Proceso Administrado (Nivel 4)

En este nivel la retroalimentación determina cómo son asignados los recursos pues las actividades básicas por si mismas no cambian. Las métricas recolectadas son utilizadas para encontrar y estabilizar el proceso, así la productividad y la calidad coinciden con las expectativas. Se recomienda por lo tanto recolectar información al respecto de los siguientes aspectos:

Tipo de proceso, se refiere a qué tipo de modelo se utiliza para el desarrollo de software.

****

Cantidad de reuso del productor, este aspecto se relaciona con que tanto se diseña el software para su reuso.

****

Cantidad de reuso del Consumidor, Este aspecto es establecido en consideración a cuanto reusa un proyecto componentes de otros aspectos.

****

Identificación de defectos, se relaciona con cómo y cuándo se descubren los defectos.

****

Uso de la densidad de defectos para las pruebas, se relaciona con la extensión del número de defectos que determina cuándo están completas las pruebas.

****

Uso de la administración de la configuración, cuestiona acerca de que si la configuración de la administración es un esquema impuesto sobre el proceso de desarrollo.

****

Terminación de módulos sobre tiempo, considera en qué porcentaje los módulos están siendo debidamente terminados.

2.6.7.5. Optimización del Proceso (Nivel 5)

A este nivel las métricas de las actividades son utilizadas para mejorar el proceso. Como ejemplo a ello, y como parte del desarrollo de esta investigación, se constato que de las cuatro empresas que se han considerado como entidades a entrevistar (Icave, Tamsa, C.F.E) de todas ellas solo dos de ellas (C.F.E., Tamsa) se encuentran en los niveles 4 o 5 niveles del modelo de madurez, por lo que las métricas recomendadas sólo incluyen los primeros cuatro niveles, es decir, estas entidades aún no cumplen con ciertas métricas y prácticas que los lleven al 100% al máximo nivel de madurez.

2.6.8 Conclusión

La realización de registros y obtención de métricas son una de las prácticas sugeridas por la Ingeniería de software y son parte de las actividades que propone el Aseguramiento de Calidad de Software para el desarrollo del software de calidad. Su difusión y aplicación no ha sido la ideal. Se requiere impulsar este campo y se ha considerado incluir este capítulo como uno de los esfuerzos para destacar su importancia.

Las métricas son por lo tanto, una faceta de la ingeniería de software, parte del aseguramiento de calidad y las implantaciones de mejores prácticas para toda disciplina que busca el desarrollo de software con calidad.

2.7 Mejores Prácticas

Introducción

La creación sólida de software de calidad requiere el conocimiento específico de las tareas que deben llevarse a cabo en cada entorno. Ahí radica la importancia de aplicar un proceso de desarrollo flexible y adaptado a cada objetivo de desarrollo. Es por ello que el combinar el conjunto básico de mejores prácticas, para el desarrollo de software de calidad, ayudará en aportar los complementos opcionales del proceso a fin de dar cabida y soporte a proyectos de cualquier envergadura o alcance.

Cualquier tipo de proyecto (incluidos los pequeños, los basados en Web, aquéllos fundamentales para un proyecto y los proyectos integrados) etc., permiten obtener unos resultados más acordes con las previsiones, gracias al control de calidad y en los procesos de desarrollo. Hacer aplicaciones de alta calidad, en corto tiempo y a un bajo costo, no es una necesidad… ¡es Imprescindible!

La industria del software ha atravesado un importante periodo de transición. Esta transición presenta un aumento en la demanda de calidad, así como la disminución de errores, retrasos en entregas y las cancelaciones de proyectos. Es por esto que se hace necesario la aplicación de prácticas que permiten una visión global y aplicación de prácticas y disciplinas para minimizar los errores y garantizar el éxito.

2.7.1. Prácticas de Desarrollo de Software

2.7.1.1 Desarrollo iterativo

Cada una de las fases del proceso de desarrollo de una aplicación, Requerimientos, Análisis, Diseño, Implementación, Pruebas, Evaluación es repetida y refinada hasta que finalmente se cumplen los requerimientos del sistema para que la aplicación se ponga en marcha.

2.7.1.2. Administración de requerimientos

Los requerimientos representan una condición y/o capacidad la que el sistema debe cumplir. Todos los requerimientos y sus correspondientes atributos son identificados y almacenados en una base de datos para poder identificar el impacto de los cambios que forman parte del proyecto.

2.7.1.3 Desarrollo de arquitecturas de n-capas basado en componentes

Para la creación de sistemas de misión crítica que sean robustos, escalables se exige el desarrollo de sistemas basados en componentes distribuidos en 3 o más capas. Así mismo, se recomienda que los componentes diseñados para esta arquitectura sean reutilizables.

2.7.1.4 Modelar visualmente

El lenguaje de Modelación UML (Unified Modeling Language) [34] está diseñado para ilustrar tanto la organización de datos como el comportamiento del sistema a desarrollar. Así mismo, el lenguaje UML está también diseñado para simplificar la comunicación entre técnicos y usuarios. Esta práctica es vital para manejar la complejidad del sistema a desarrollar, y para facilitar la identificación de componentes reutilizables.

2.7.1.5 Verificar constantemente la calidad

Control de Calidad es responsabilidad de todos los involucrados en el proyecto de desarrollo y no un grupo singular de Control de Calidad. Asimismo, los criterios para verificar y evaluar la calidad de los componentes desarrollados deben ser directamente relacionados a los requerimientos definidos para el proyecto.

La ejecución de pruebas debe ocurrir durante el desarrollo de componentes y previo a la integración del sistema a usuarios finales.

2.7.1.6 Administración de cambios y defectos

Coordinación de grupos para organizar las actividades relacionadas a evaluación y asignación de cambios y defectos. Esta organización facilita la centralización de peticiones de cambios y defectos mediante un flujo de trabajo organizado y automatizado. Asimismo, el uso de métricas y estadísticas facilita la distribución de trabajo y la toma de decisiones claves durante la gestión del proyecto.

2.7.2 Calidad vs. Costo vs. Tiempo

¿Cómo puede asegurar que está desarrollando una aplicación de alta calidad y libre de errores?

La respuesta es: contando con un proceso de pruebas efectivo. Lamentablemente no se ha creado una cultura en donde las pruebas se tomen como un proceso de gran importancia, lo que significa que las empresas no cuentan con una buena metodología y/o con las herramientas adecuadas para manejar un excelente proceso de pruebas. Por otro lado, con las nuevas oportunidades y desafíos que presenta la Internet y el comercio electrónico, dramáticamente aumenta la presión por liberar aplicaciones en corto tiempo, lo cual aumenta el riesgo de obtener software de baja calidad poniendo en riesgo la imagen de las personas que participaron en el proyecto junto con la de su empresa.

****

¿Cuál es la solución?

La respuesta no ha cambiado, ¡teniendo un proceso de pruebas efectivo! Esto es algo que no se debe tomar a la ligera, si realmente se quiere continuar en la lucha por mantener y conseguir nuevos clientes.

****

¿Realmente estamos listos para liberar?

No es ningún secreto que en un proyecto de desarrollo sea necesario incluir un proceso de pruebas. Es cierto que probar una aplicación no es nada fácil, puesto que existe la dificultad de abarcar todos los caminos posibles, sin embargo, un proceso efectivo de pruebas debe iniciar junto con el ciclo de desarrollo de la aplicación, lo que llevará finalmente a una disminución en los costos de desarrollo y mantenimiento del software, así como una reducción en los riesgos de funcionalidad incrementando la calidad de nuestras aplicaciones.

Tabla 2.3 Dimensiones de Calidad

Tres dimensiones de calidad para lograr procesos de pruebas efectivo: 🕓 CONFIABILIDAD La aplicación opera sin causar errores inesperados en tiempo de ejecución y maneja los recursos de manera adecuada. 🕓 FUNCIONABILIDAD La aplicación cumple con los requerimientos del negocio. 🕓 DESEMPEÑO El comportamiento y los tiempos de respuesta del sistema con la cantidad de usuarios que se estiman son aceptables.

2.7.3 Modelo de Madurez de Capacidad (CMM)

Éste corresponde a uno de los modelos más conocidos creado por el SEI (Software Engineering Institute) de la Carnegie Mellon University. El CMM pretende conseguir mejorar la calidad del software mejorando la calidad de los procesos utilizados en su desarrollo. Por eso es importante mencionar que: "Las herramientas y las plataformas cambian de forma continua. Pero siempre podemos usar el mismo proceso si éste está bien definido y se sabe utilizar de forma adecuada."

2.7.3.1 Niveles de Madurez

EL CMM puntúa los procesos de desarrollo del software en una escala de cinco niveles. En donde se tienen muy en cuenta aspectos muy variados de los procesos de desarrollo, como el grado de ambigüedad de las especificaciones, la verificación independiente de la fiabilidad de los programas etc.; estos cinco niveles de madurez están definidos por la presencia de áreas claves del proceso.

También se definen listas de mejores prácticas para conseguir alcanzar cada una de estás áreas clave.

Los cinco niveles que describen avances en el proceso de ingeniería del Software se describen a continuación:

1. El primer nivel (Caos) se produce cuando en la empresa no existe ningún modelo y que todo se hace sobre la marcha es decir no se emplea ningún proceso definido.

2. En el segundo nivel (Repetible) se encuentran las empresas en las que existe planificación y seguimiento de proyectos y está implementada la gestión de los mismos.

3. El tercer nivel (Definido) documenta y normaliza los procesos a nivel organizativo. Las claves de este nivel son la gestión de los requisitos, planificación de proyectos y su seguimiento a través de toda la organización

4. El cuarto nivel (Medible) pone énfasis en la calidad del proceso y del producto. Lo tienen las empresas capaces de medir el estado de un proyecto y utilizar esta información para que los jefes introduzcan los cambios y correcciones necesarias. Una vez adquirido este nivel en la gestión de los proyectos se pueden establecer objetivos.

5. El quinto nivel (Mejora continua) se conoce como proceso continuo de mejora. Las áreas clave del proceso incluyen prevención de defectos, administración de cambios tecnológicos y gestión de cambios en los procesos.

2.7.3.2. Métodos de Evaluación

Para conseguir la certificación CMM, es necesario contactar con algún evaluador acreditado por el SEI. Éstos utilizan distintos métodos para determinar en las organizaciones el nivel de madurez en el que se encuentra el proceso utilizado en el desarrollo de software.

Entre estos métodos destaca el SCE y el CBA-IPI. El primero consiste fundamentalmente en una auditoría mientras que el segundo utiliza entrevistas y otros procedimientos encaminados a ayudar a la mejora de los procesos seguidos en la organización.

2.7.4. Metodología de Desarrollo Estándar de la Industria

Al respecto, la Rational Unified Process (RUP) [24], es la metodología estándar de la industria para la construcción completa del ciclo de ingeniería de software, tanto para sistemas tradicionales como, para sistemas Web. Esta metodología le permite mayor productividad en equipo y la realización de mejores prácticas de software a través de plantillas y herramientas que lo guían en todas las actividades de desarrollo crítico del software.

RUP unifica las disciplinas en lo que a desarrollo de software se refiere, incluyendo modelado de negocio, manejo de requerimientos, componentes de desarrollo, ingeniería de datos, manejo y configuración de cambios, y pruebas, cubriendo todo el ciclo de vida de los proyectos basado en la construcción de componentes y maximizando el uso del UML (Unified Modeling Language) [34].

2.7.5. Lenguaje de Modelación Estándar de la Industria

La UML (Unified Modeling Language) es el diseño de aplicaciones basadas en componentes. El UML, del cual fue pionero Rational y fue adoptado oficialmente como una norma por el OMG (Object Management Group) [19], es el lenguaje estándar de la industria para especificar, visualizar, construir, y documentar los artefactos de un sistema de software.

El OMG, [19] aceptó el proyecto de Rational de lanzar a UML como el estándar oficial de notación para el desarrollo de software bajo el esquema Object Oriented. Rational toma la iniciativa y tiene el liderazgo.

El OMG, [19] aceptó el proyecto de Rational de lanzar a UML como el estándar oficial de notación para el desarrollo de software bajo el esquema Object Oriented. Rational toma la iniciativa y tiene el liderazgo.

UML no es simplemente un lenguaje para modelación orientado al objeto de tercera generación, su alcance extiende su uso más allá de sus predecesores. UML, es un lenguaje de modelación para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. UML no es:

a) Un lenguaje de programación visual, sino un lenguaje de modelación visual,
b) Una herramienta o depósito de especificación, sino un lenguaje de modelación de especificación,
c) Un proceso, sino que habilita procesos,
d) Principalmente, UML está relacionado con la captura, comunicación y nivelación (disgregación en niveles) de conocimientos.
e) Un lenguaje de programación visual, sino un lenguaje de modelación visual,
f) Una herramienta o depósito de especificación, sino un lenguaje de modelación de especificación,
g) Un proceso, sino que habilita procesos,
h) Principalmente, UML está relacionado con la captura, comunicación y nivelación (disgregación en niveles) de conocimientos.
i) Un lenguaje de programación visual, sino un lenguaje de modelación visual,
j) Una herramienta o depósito de especificación, sino un lenguaje de modelación de especificación,
k) Un proceso, sino que habilita procesos,
l) Principalmente, UML está relacionado con la captura, comunicación y nivelación (disgregación en niveles) de conocimientos.

A continuación se muestra en la Tabla 2.4 las utilidades del UML
Tabla 2.4 Utilidades de UML:

Como lenguaje de modelación, **con** el propósito general evolutivo, ampliamente aplicado, debe de ser soportado por herramientas e industrialmente estandarizado. Se aplica a una multitud de diferentes tipos de sistemas, dominios, métodos o procesos.

Como lenguaje de propósito general, se enfoca en el corazón de un conjunto de conceptos para adquirir, compartir y utilizar conocimientos emparejados con mecanismos de extensión.

Como un lenguaje de modelación ampliamente aplicable, puede ser aplicado a diferentes tipos de sistemas (software y no-software), dominios (negocios versus software) y métodos o procesos.

Como un lenguaje para la modelación soportable por herramientas, las herramientas ya están disponibles para soportar la aplicación del lenguaje para especificar, visualizar, construir y documentar sistemas.

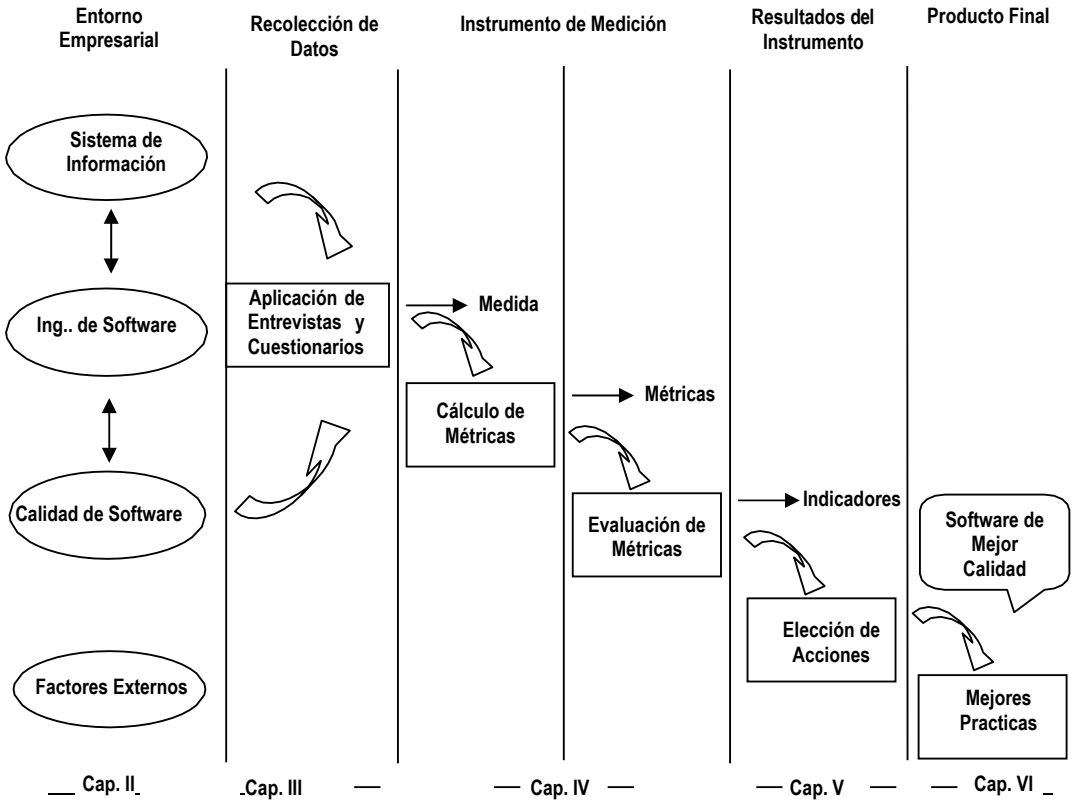
Como un lenguaje de modelación industrialmente estandarizado, no es un lenguaje cerrado, propiedad de alguien sino más bien un lenguaje abierto y totalmente extensible reconocido por la industria.

2.7.6 Conclusión

Lo que se plantea en este capítulo es muy sencillo: las mejores prácticas que se manejen y apliquen en las organizaciones, deben de aportar a éstas, las mejores opciones de estructuras de trabajo, así como las herramientas de apoyo para encontrar el equilibrio entre las organizaciones que prestan servicios de calidad y la respuesta satisfactoria del cliente.

Capítulo III: Método y técnica

Continuando con la metodología que nos lleve a generar elementos sólidos para la propuesta sobre "MEJORES PRÁCTICAS PARA EL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE", a continuación se describe el modelo a seguir en dicho desarrollo.



3.1. Propósito del modelo:

Demostrar que con la información adecuada y utilizando los elementos correctos, se puede definir el método que lleve a obtener las métricas y mejores prácticas para el aseguramiento de calidad de software.

3.2 Explicación del Modelo Particular

El Modelo anterior esquematiza el propósito del estudio, y muestra el panorama general de la propuesta. En el modelo se presentan las secuencias que guían hacia el camino de las prácticas para el establecimiento de calidad de software.

Este modelo inicia con la fase de entorno empresarial en el que se toma como base las referencias bibliográficas mencionadas en la argumentación teórica, en donde además, los elementos dan la pauta para ubicarnos en la siguiente fase que es la recolección de datos, medición y evaluación, para obtener la evidencia que permita establecer la propuesta para el establecimiento de calidad de software.

El estudio se enfoco a cuatro empresas del Puerto de Veracruz, cuyos departamentos de desarrollo de software son los más demandados en numerosos diseños de sistemas de información. Estas empresas son las más representativas y de mayor impacto dentro de las tecnologías de información en la localidad en cuestión. Las empresas estudiadas son: dos particulares y dos oficinas gubernamentales.

3.3. Variables

Las variables a considerar en este modelo son de dos tipos:

Independientes / Causa:

- Métricas de Calidad de Software
- Control de Calidad de Software.
- Objetivo en el Desarrollo de Proyectos

Estas se resumen en: "El conocimiento del grado de calidad de un software obtenido por medio de la medición de sus factores de calidad".

Dependientes / Efecto:

- Éxito del proyecto
- Identificación de Mejores Prácticas para el establecimiento de Calidad de Software.

Estas se resumen en: "Aplicar mejores prácticas para el aseguramiento de calidad de software".

3.4. La obtención de la información y aplicación del instrumento

La recolección de datos se basa en entrevistas, cuestionarios, encuestas a administradores y líderes de proyectos, que tengan experiencia con la implantación de metodología de calidad, administración de proyectos, desarrollo de software y personal involucrado en los procesos de desarrollo de software.

3.5. Dispersión Geográfica de Empresas Visitadas.

Durante la realización de las encuestas que se aplicaron a las áreas generadoras de productos de software, se visitaron las siguientes empresas:

Tabla 3.5 Dispersión geográfica de las empresas visitadas.

Empresa	**Ubicación**	**Tipo**
C.F.E Dos Bocas	CFE / Está en todo el país. El departamento de SC tiene presencia directa en la CLV, Farallón y Dos Bocas, es definida como la Subgerencia de Ingeniería de la Gerencia de Centrales Nucleoeléctricas. Ubicada en la Carretera Veracruz-Medellín KM 4.5, Dos Bocas, Veracruz.	Empresa de Gobierno
C.F.E. Laguna Verde	CFE/Laguna Verde, forma parte importante de las Plantas Nucleoeléctricas del País, es definida como la Central Nucleoeléctrica. Ubicada en la Carretera Veracruz – Laguna Verde, Veracruz, Ver.	Empresa de Gobierno
ICAVE	Terminal Especializada de contenedores, se encuentra estratégicamente localizada en la parte media del Golfo de México, que es el principal centro de transferencia de mercancías contenerizadas por vía marítima. Es la Empresa Internacional de Contenedores Asociados más importante en la Ciudad de Veracruz. Ubicada en la Zona Portuaria del Estado.	***Empresa Privada***
TAMSA	Representada en México como la Industria Productora de Creación de Tubos de Acero más importante en el País. Ubicada en la Carretera México - Veracruz, Km. 433.5 Veracruz, Ver. México	Empresa Privada

Se consideraron estas entidades, principalmente por tres elementos claves

- Por su ubicación dentro de la Ciudad, Puerto de Veracruz, Ver.
- Por la importancia y experiencia que estas empresas tienen en el desarrollo de productos de software.
- Por el Tipo de empresas privadas y de gobierno a las cuales representan.

3.6 Tamaño del Departamento de Desarrollo de Software de las Empresas Visitadas.

Para la aplicación de las encuestas, es importante entrevistar a profesionales involucrados en el desarrollo de productos de software, en las respectivas empresas a las cuales prestan sus servicios. Se toma como referencia a un grupo representativo de cada uno de los departamentos de las áreas afines, y en su momento este grupo, dará como resultado una serie de elementos que refleja la eficiencia y deficiencia del conocimiento y manejo de controles de calidad de software.

Tabla 3.6 Tamaño de los departamentos de desarrollo de software encuestados.

Empresa	Departamento	Num. de Encuestados
C.F.E Dos Bocas	En el departamento de Informática, actualmente se cuenta con diez personas en el grupo de desarrollo, considerándose un total de 25 personas involucradas en el área.	9
	6	
C.F.E Laguna Verde	En el departamento de Ingeniería de Software se cuenta con 22 personas en el grupo de desarrollo de software.	9
ICAVE	En el departamento de Informática, se cuenta con la participación de 10 personas involucradas en el área de desarrollo, considerándose un total de 15 personas, involucradas en el área de informática.	9
TAMSA	En el departamento de Informática, se cuenta con la participación de 45 personas involucradas en el área de desarrollo de software considerándose un total de 77 personas, involucradas en el área de informática.	9

3.7 **Posicionamiento del Mercado:** Algunas características de cada una de las empresas, que la hacen líder en el mercado son las siguientes:

C.F.E. - Dos Bocas, C.F.E como tal es líder del mercado en México y quinta empresa a nivel mundial y una de sus secciones como compañía es la de Dos Bocas, teniendo esta mis a una gran responsabilidad ante sus servicios a las zonas a las cuales respalda esta entidad.

ICAVE, Es la empresa líder nacional en manejo de contenedores y servicios a la carga que opera la Terminal especializada del puerto de Veracruz.

CLV - Central Laguna Verde, De igual forma que Dos Bocas, Laguna Verde es otra entidad generadora de servicios provinente de C.F.E cuya función es tan importante en el mercado nucleoeléctrico.

TAMSA, Empresa que fabrica anualmente 670 000 toneladas de tubería de acero sin costura en la planta de Veracruz y 6000 toneladas de conexiones sin costura en la planta de Monterrey. Además Proporciona servicios integrales para las diversas ramas industriales entre las que destacan las industrias petrolera y automotriz así como proyectos de ingeniería y construcción.

3.8 **Giro del Negocio.** El giro con el que operan estas emprestases:

C.F.E Dos Bocas, empresa pública, dedicada a la generación, distribución y comercialización de energía eléctrica.

ICAVE, empresa privada desde sus inicios, han realizado importantes inversiones en tecnología de punta y sistemas de vanguardia, además de avanzados esquemas de organización, todo esto para satisfacer las expectativas de nuestros clientes y garantizar nuestra permanencia en el mercado, contribuyendo además al fortalecimiento del Puerto de Veracruz y al engrandecimiento del Comercio Internacional de nuestro país, teniendo una participación muy importante en el manejo de contenedores en México.

CLV, Central Laguna Verde, empresa pública, dedicada a la generación, distribución y comercialización de energía eléctrica.

TAMSA, empresa privada mexicana que forma parte del grupo Tenaris, la alianza compuesta por ocho plantas productoras de tubos de acero con centros de atención en más de 25 países que provee servicios a clientes en todo el mundo. Además de atender a la industria nacional, Tamsa produce gran parte de las soluciones tubulares que Tenaris comercializa en el mercado automotriz norteamericano (México, E. U. y Canadá).

Conclusión al capítulo:

La obtención de resultados con la selección de la muestra no probabilística empleada para el desarrollo de este estudio, no limita el utilizar este ejercicio de investigación, en futuros levantamientos estadísticos de conocimiento del grado de calidad de un producto de software, también no limita a un grupo muy pequeño, ya que toda unidad de análisis es válida, siempre y cuando ayude a determinar las métricas de evaluación de calidad de software.

El recurrir a técnicas de recolección de datos como la aplicación de encuestas y entrevistas con los representantes de las áreas desarrolladoras de software, da la oportunidad de llevar a cabo un modelo particular que lleve a proponer y aplicar un estudio completo de la forma más adecuada para llevar un control de calidad sobre los productos de software que se van generando.

Capítulo IV: Medición y diseño de datos

4.1 Introducción

El proceso llevado a cabo respecto al cálculo y evaluación de las métricas, da los elementos para el establecimiento de mejores prácticas para el control de calidad en los productos de software.

La idea central del estudio, ha estado centrado en la propuesta de un modelo de mejores prácticas para el establecimiento de un control de calidad en un producto de software dado. Tomando como referencia un conjunto de características del entorno que influyen en el proceso de desarrollo de un producto de software, es que se hace necesario definir, qué factores de calidad requieren estar presentes para que se pueda cumplir, en su totalidad, con las especificaciones de los usuarios.

La investigación de campo ha tomado como base, parte del procedimiento de identificación de factores de calidad de Perry (21).

Es por ello que se ha planteado como hipótesis central: *"El conocimiento del grado de calidad de un software obtenido por medio de la medición de sus factores de calidad, permitirá aplicar mejores prácticas para el aseguramiento de calidad de software".*

El planteamiento matemático de la hipótesis toma la siguiente expresión matemática que propone Perry. (21)

$$F(c) = \sum_{i=1}^{n} (aiCi) = a1C1+a2C2+a3C3+\ldots anCn$$

El grado y los factores de calidad, dan el soporte para la propuesta sobre mejores prácticas. A saber, el peso de cada factor depende de la importancia de las características del entorno de los productos de software y de la ponderación de la característica por factor.

Así, el modelo matemático planteado, sugiere que:

$$F(c) = \sum_{i=1}^{n} (aiCi) = a1C1+a2C2+a3C3+\ldots anCn$$

En donde:

F(c) ➔ Representa el valor final del grado de importancia de un factor de calidad.

ai➔ Representa un cociente de importancia (ponderación de la característica) por cada factor.

Ci➔Representa el valor del grado de importancia de una característica del entorno de la aplicación

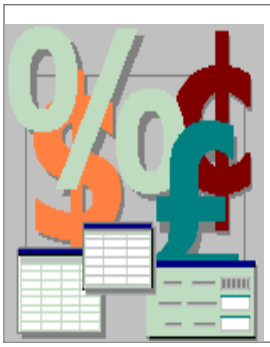
n➔ Representa el número de características del entorno de la aplicación ser tomadas en cuenta.

4.2 Instrumento de Medición y Cálculo de Métricas:

Para la aplicación del instrumento de medición en las empresas seleccionadas para la generación de mejores prácticas sobre el control de calidad de software, se desarrollaron y aplicaron durante la investigación cuatro formatos que a continuación se describen a detalle, dando lugar con ello al cálculo de métricas que se efectuaron durante el análisis y medición de datos:

1. *El desarrollo de la Encuesta - Formato_1,* proporciona un panorama general del giro de la Empresa, es decir, conoceremos su forma actual de llevar a cabo su trabajo, obteniendo con ello en sus respuestas porcentajes, que reflejen las características de funcionalidad y desempeño de las organizaciones que se estarán visitando.

2. *El desarrollo de la encuesta - Formato_2,* refleja la importancia que las organizaciones a entrevistar le dan a los Factores de Calidad de Software, si los conocen y si los aplican, para ello se deberá especificar Intervalos en sus respuestas, como: MI (Muy Importante) ➔10, I (Importante) ➔ 7, PI (Poco Importante) ➔ 4, NI (Nada Importante) ➔1.



Nota: Para el caso de la obtención del cociente ai (Factores de calidad), se estarán sumando todas las posibles respuestas en los valores definidos y se dividirán entre el número total de personas entrevistadas obteniendo con ello un panorama y valores en % de los factores de calidad y así de una manera más clara conocer la situación de la empresa y su actual cultura de trabajo en sus desarrollos de software.

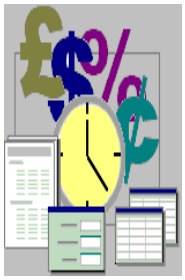
3. *El desarrollo de la encuesta - Formato_3,* refleja el grado de importancia que consideren las empresas a entrevistar a las Características de Entorno de Desarrollo que afecten la calidad final del producto de software, y a su vez si conocen y aplican estos elementos, para ello se deberá especificar Intervalos en sus respuestas, como: MI (Muy Importante) ➔10, I (Importante) , ➔7, PI (Poco Importante) ➔ 4, NI (Nada Importante) ➔1.

Nota: Para el caso de la obtención del cociente Ci (Características de entorno), se estarán sumando todas las posibles respuestas en los valores definidos y se dividirán entre el número total de personas entrevistadas obteniendo valores en %, estoa se toman como base para llevar a cabo las operaciones que más adelante dan apoyo al Formato_4, y de esta manera se puede determinar la situación de la empresa y su actual cultura de trabajo en sus desarrollos de software.

Para el caso de la obtención del grado de importancia de las características del entorno de desarrollo de software, las posibles respuestas se ubican en una matriz de relación (Formato_4) entre las características y los factores de calidad, en donde esto debe de hacerse teniendo en cuenta los requerimientos del usuario obtenidos en la etapa de análisis del producto de software y la naturaleza de la aplicación (se debe de determinar que dicha naturaleza caiga en las características de la aplicación: programa, sistema, modelo, prototipo, base de datos, etc.).

4. *El desarrollo de la encuesta (Formato_4),* refleja la Matriz la relación entre los factores de calidad y las características de desarrollo, es decir se cuestionan a las personas del área de sistemas, si creen que de alguna manera afectan moderadamente como fuertemente una característica de entorno dada, a un factor de calidad dado.

Ejemplo: La "experiencia del usuario" (característica) afecta de gran manera la "Facilidad de Uso" (factor de calidad) del producto de software.



Nota: La fórmula antes mencionada, propone obtener los cocientes de importancia (ai en la fórmula) de las características del entorno por cada factor de calidad. Estos cocientes de importancia serán obtenidos directamente de la investigación de campo que se estará realizando, donde se establece la relación existente entre los factores de calidad y las características de calidad y para ello se propondrán manejar valores numéricos a las evaluaciones de "afecta moderadamente = 5" o "afecta fuertemente = 10".

4.3 Evaluación de las Métricas

Las ponderaciones antes mencionadas son determinadas considerando el hecho de que una característica deL entorno que afecta "fuertemente" a un factor de calidad tiene el doble de efecto que una que lo afecta "moderadamente".

Posterior a ello se le dará un valor numérico a los % obtenidos, será 1 por 100% de menciones; esto se hará con el propósito de tener un valor que pueda ser utilizado para cálculos aritméticos, de esta manera se calculan los cocientes (ai) para todas las características de entorno de cada uno de los factores de calidad.

Ejemplo:-

Factor de Calidad	Caract. / Entorno	Núm. / Personas	Calc. / Aplicar	Res. Final
Corrección	Aplicación	5 Moderadamente 10 Fuertemente	**M = 5 5*10 = 50** **F =10 10*10 = 100** **50 + 100 = 150 /100%**	1.5

El valor de 1.5 será sólo una parte del cálculo que se efectuará para la obtención final del cociente **ai;** este valor 1.5 posteriormente se multiplicará por los valores estándar que arroja el formato_3. Para obtener F(c) que representa este al valor final del grado de importancia de un factor de calidad, debe llevarse a cabo la multiplicación de **ai** (formato_4) * **ci** (formato_3).

Después de llegar a estos valores se efectúa la sumatoria de todos y cada uno de los factores de calidad y se aplica la fórmula

antes mencionada. Al obtener este valor final en % dará la pauta para ubicar los factores de calidad que están manejando y considerando en las empresas, para ello se maneja una escala de 50 a 100 en donde va de menor impacto a mayor impacto.

Ejemplo: Corrección: 86 %, Fiabilidad: 71%. Considerando que ya fue aplicado todo lo anterior (fórmula, etc…) y se han encontrado los % antes mencionados en estos dos factores de calidad; el punto medular en esto, es que se estarán desarrollando ciertas prácticas, así como procedimientos que dan apoyo de manera general, a los Factores de Calidad que se recomiendan para llevar a cabo en las empresas entrevistadas. De esta manera se puede contribuir con las organizaciones, para que, siguiendo ciertos pasos, pueda aplicar un control de calidad en sus productos de software, derivando además con ello, todo lo referente a un orden en sus documentaciones, auditorias, entre otras cosas.

Por otra parte se parte de la premisa, que probablemente estos procedimientos no apliquen en todos sus puntos, a los factores en los que se hayan encontrado algunas deficiencias, ya que estos bien pueden aplicarse o no a todas las recomendaciones (Mejores Prácticas). En posteriores párrafos se estarán mostrando las estadísticas en % y sus respectivas gráficas de los valores obtenidos en las entidades entrevistadas.

Las encuestas que se aplicaron buscan, entre otras cosas, el grado en que afecta cada característica del entorno de una aplicación a un factor dado. Ejemplo: la metodología de programación (que es una característica de entorno) afecta directamente el factor de flexibilidad; si la metodología no sigue lineamientos rigurosos de documentación, cuando el producto esté terminado será prácticamente imposible modificarlo exitosamente.

Las encuestas tuvieron como universo, programadores, líderes de proyecto, jefes de sección, es decir todo aquel personal involucrado en el desarrollo de software.

A continuación se muestra la descripción de los formatos de las encuestas aplicadas:

Tabla 4.1 Descripción del Formato_1

Encuesta que lista una serie de preguntas formuladas de manera breve, puede ser que se encuentre más de una opción como respuesta, marque con una 'X' en los espacios propuestos (dentro de los recuadros) la respuesta (s) correcta (s), así como dar respuesta a aquellas preguntas abiertas.

1. El software que desarrollan consiste de:
Sistemas Modelos Prototipos Otros:_____

2. El software que desarrollan es para uso:
Interno Externo Otros:_____

3. En el software que se desarrolla, el papel que juega la calidad del mismo se puede decir que es:
Sí es Importante No es importante Es poco importante
Es muy importante Juega un papel moderado

4. La calidad del software desarrollado es buscada:
Desde la planeación Durante la Implementación Durante la etapa de pruebas

5. Para sus desarrollos de software, siguen lineamientos o procedimientos metodológicos de apoyo en la calidad de software de tipo:
Modelado de Objetos (UML) Administración de proyectos Metodologías de Apoyo en la calidad de software (COBIT, CMM, RUP)
Otros:_____

6. En qué consiste dicho procedimiento, metodología:
_____

7. Ha dado resultados:
Si No _____ % Cubierto

8. Cómo han medido los resultados:
_____

9. Qué tiempo tiene actualmente este procedimiento, metodología:
< 1 año 1 a dos años > 3 de años

10. En qué tipo de aplicaciones está involucrado en estos momentos y qué rol desempeña:
_____

11. El departamento o área de sistemas cuenta con alguna certificación de calidad como:
ISO IEEE Ninguno Otros:_____

12. En qué fase de desarrollo se encuentran los proyectos en los que esta involucrado actualmente (considere el proyecto de mayor prioridad):
Fase de Análisis Fase de desarrollo Fase de Operación/Mantenimiento Otros:_____

13. Con qué frecuencia recibe capacitación por parte de sus Jefes de Sección/ Gerencia:
dos veces al año cinco veces al año Ninguna
Otros:_____

14. Con qué frecuencia tienen auditorías de sistemas:
Una vez al año dos veces al año Otros:_____

15. Qué tipos de auditorías manejan:
Interna Externas Otros:_____

Tabla 4.2 Descripción del Formato_2

Indique la importancia que se le da a cada factor de calidad entre los valores de 10 muy importante, 7 importante, 4 Poco importante, 1 Nada importante.	
Corrección:	Grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el usuario final.
Confiabilidad:	Grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida.
Eficiencia:	Cantidad de recursos de una computadora y de código requeridos por un programa para llevar a cabo sus funciones.
Integridad:	Grado en que puede controlarse el acceso al software o a los datos por personal no autorizado.
Facilidad de uso:	Esfuerzo requerido para aprender, trabajar, preparar las entradas e interpretar las salidas de un programa.
Fac. de mantenimiento:	El esfuerzo requerido para localizar y arreglar un error en un programa.
Flexibilidad:	Esfuerzo requerido para modificar un programa que se encuentre en operación.
Fac. de prueba:	El esfuerzo requerido para probar un programa de manera que se asegure que realiza su función requerida.
Portabilidad:	El esfuerzo requerido para transferir un programa desde una plataforma de hardware y/o entorno de software a otro.
Reusabilidad:	Grado en que un programa (o partes de él) se pueden volver a utilizar en otras aplicaciones. Esto se relaciona con el alcance de las funciones que realiza el programa.
Interoperabilidad:	El esfuerzo requerido para acoplar un sistema a otro.

Tabla 4.3 Descripción del Formato_3

Indique la importancia que se le da a las Características de Entorno entre los valores de 10 muy importante, 7 importante, 4 Poco importante, 1 Nada importante.	
Aplicació	Aplicación sobre la cual el software va dirigido (negocios, ingeniería, medicina, etc.)
Por qué se le da tal rubro:	_____
Ambiente de Usu	Espacio físico y condiciones en que se utiliza el software.
Por qué se le da tal rubro:	_____
Riesgos y consecuencias de fallas:	Lo que puede implicar que el software falle.
Por qué se le da tal rubro:	_____
Computadora Anfitriona:	La computadora donde correrá el software.
Por qué se le da tal rubro:	_____
Madurez del Desarrollador:	Experiencia en el desarrollo de proyectos de software similar.
Por qué se le da tal rubro:	_____
Experiencia del Usuario:	Experiencia en el manejo de aplicaciones de software.
Por qué se le da tal rubro:	_____
Apoyo de los desarrolladores:	Asesoría o ayuda por parte de los desarrolladores.
Por qué se le da tal rubro:	_____

Experiencia de los Desarrolladores:	Experiencia en el manejo de herramientas de software y desarrollo de sistemas de información.
Por qué se le da tal rubro:	_____
Interacción con el usuario final:	Comunicación entre usuarios y desarrolladores.
Por qué se le da tal rubro:	_____
Restricciones Comerciales:	Falta de presupuesto o tiempo.
Por qué se le da tal rubro:	_____
Metodología de desarrollo:	Guía en estándares, normas y procedimientos para el desarrollo de software.
Por qué se le da tal rubro:	_____
Lenguajes de Programación:	Los diferentes lenguajes de programación que se lleven en el área.
Por qué se le da tal rubro:	_____
Complejidad del Software:	Grado en que tenga que involucrar muchos elementos físicos (periféricos).
Por qué se le da tal rubro:	_____
La salud:	Salud de los desarrolladores de software (que no presenten, ni padezcan enfermedades.
Por qué se le da tal rubro:	_____

Tabla 4.4 Descripción del Formato_4

Indique con el símbolo ✓ en los espacios propuestos sobre el rubro 'M' si usted cree que afecta moderadamente una característica de entorno dada, a un factor de calidad dado. Si cree que afecta fuertemente, indíquelo con el rubro F, si no cree que afecte, deje el espacio en blanco.

Corrección: Grado en que el software consigue sus objetivos.

Característica / Factor	Corrección
Aplicación	**( )F** **( )M**
Ambiente de Uso	**( )F** **(M**
Riesgo	**( )F** **( )M**
Computadora Anfitriona	**( )F** **( )M**
Madures del Desarrollador	**( )F** **( )M**
Experiencia del Usuario	**( )F** **( )M**
Apoyo de los Desarrolladores	**( )F** **( )M**
Experiencia de los Desarrolladores	**( )F** **( )M**
Interacción con el Usuario	**( )F** **( )M**
Restricciones Comerciales	**( )F** **( )M**
Metodología de Desarrollo	**( )F** **( )M**
Lenguajes de Programación	**( )F** **( )M**
Complejidad de Software	**( )F** **( )M**
Salud de los Desarrolladores	**( )F** **( )M**

4.4 Conclusión

Para el análisis y medición de los datos es muy importante aplicar un instrumento de medición que nos arroje valores que nos lleve a las métricas reales de las áreas de desarrollo de software que se están evaluando.

Capítulo V: Análisis de los Resultados

5.1. Introducción

En este apartado, se presentan los resultados obtenidos de la investigación de campo, los cuales dan las bases para la propuesta. La aplicación de cada formato permitió conocer cómo se desenvuelven las áreas de desarrollo de software dentro de las empresas estudiadas, lo que derivo en indicadores cualitativos y cuantitativos que apoyan en la identificación de esos indicadores de calidad que hoy en día prevalecen, al interior de las empresas estudiadas.

Consideraciones por tipo de análisis (cualitativos y cuantitativos)

5.1.1.- Análisis Cualitativo

Las encuestas (Formatos_1,2,3,4) desarrolladas para la obtención de resultados cualitativos, tienen como objetivo recaudar información de la entidad y del personal encuestado, en el área de Desarrollo de Sistemas de Información, es decir, busca el contacto con las personas involucradas en la creación de Software y que a su vez lleven acabo mantenimientos, reingeniería y soporte de aplicaciones a proyectos generadores de algún servicio de software. Estas encuestas están dirigidas a los Jefes de sección, coordinadores/líderes de proyecto analistas/diseñadores y desarrolladores de software.

La información obtenida nos dará como resultado, un panorama de la situación actual de la empresa. Así, con ello, se

estará evaluando su forma actual de trabajo y se analizará qué tan importante es para cada una de estas empresas, considerar a la calidad de software, como el factor clave para el éxito en sus productos finales.

La siguiente serie de encuestas está formada por 4 formatos, teniendo cada unos de ellos su propia importancia y finalidad, ya que conforme se presentaron, se fue entendiendo la importancia del aseguramiento de calidad de software como una cultura de trabajo que debe estar presente desde las primeras fases del desarrollo del software.

Tabla 5.1 Definición de cada uno de los formatos aplicados en los departamentos de desarrollo de software.

FORMATO_1 **Panorama General de la Empresa**	Objetivo de la Encuesta: Adquirir un panorama general de la Empresa/Área de Sistemas encuestada.
FORMATO_2 **Conocimiento de los Factores de Calidad de Software.**	Objetivo de la encuesta: Analizar qué tanto conocen y toman en cuenta las personas involucradas en el área de sistemas de información a los factores de calidad en el desarrollo de sus productos finales de software.
FORMATO_3 **Conocimiento de las Características del entorno de desarrollo.**	Objetivo de la encuesta: Analizar qué tanto conocen y toman en cuenta las personas involucradas en el área de sistemas de información las Características del entorno de desarrollo de software en sus productos finales.
FORMATO_4 **Matriz de Relación entre los Factores de Calidad de Software y de las Características del entorno de desarrollo.**	Objetivo de la encuesta: Analizar qué tanto afectan una característica de entorno dada, a un factor de calidad dado en el desarrollo de sus productos finales de software.

5.1.2.- Análisis Cuantitativo

Basados en los formatos (1,2,3,4) que se aplicaron en las áreas desarrolladoras de software, se presentan los resultados de las encuestas realizadas al personal de desarrollo de software, tomando como muestra en su totalidad a 36 profesionistas de las empresas objeto de estudio, obteniendo con ello indicadores que reflejan los valores finales encontrados como resultado cuantitativo del estudio.

Con estas consideraciones, ahora se muestran los resultados obtenidos por formato.

FORMATO_1 (Las respuestas están en porcentajes.)

1. ¿El Software que desarrolla consiste de?
( ) Sistemas ( ) Modelos ( ) Prototipos ( ) Otros: _____

Encuestados que desarrollan sistemas: 78%
Encuestados que desarrollan modelos: 6%
Encuestados que desarrollan prototipos: 11%
Profesionales encuestados que desarrollan otros tipos de software: 5%

2. ¿El software que desarrolla, es para uso?
( ) Interno ( ) Externo ( ) Otros: _____

Encuestados que desarrollan software interno: 86%
Encuestados que desarrollan software externo: 11%
Encuestados que desarrollan software para otro tipo de uso: 3%

3. ¿En el software que se desarrolla, el papel que juega la calidad del mismo se puede decir que es?
( ) Sí es importante ( ) No es importante ( ) Es poco importante ( ) Es muy importante ( ) Juega un papel Moderado

Encuestados que consideran que la calidad sí es importante: 27%
Encuestados que consideran que la calidad es poco importante: 17%
Encuestados que consideran que la calidad es muy importante: 42%
Encuestados que consideran que la calidad juega un papel moderado: 14%

4. ¿La calidad del software desarrollado es buscada?
( ) Desde la planeación ( ) Durante la implementación ( ) Durante la etapa de pruebas

Encuestados que buscan la calidad desde la planeación: 53%
Encuestados que buscan la calidad durante la implementación: 36%
Encuestados que buscan la calidad durante la etapa de pruebas: 11%

5. ¿Para sus desarrollos de software, siguen lineamientos o procedimientos metodológicos de apoyo en la calidad de software de tipo?
( ) Modelado de objetos/UML ( ) Administración de proyectos
( ) Metodologías de apoyo en la calidad de software /Cobit, CMM, RUP
( ) Otros: _____

Encuestados que siguen lineamientos de Modelado de objetos/UML: 14%
Encuestados que siguen lineamientos de admón. de proyectos: 67%
Encuestados que siguen lineamientos de Metodologías de apoyo en la calidad de software /Cobit, CMM, RUP: 8%
Encuestados que siguen otros lineamientos metodológicos de apoyo en la calidad de software: 11%

6. ¿En qué consiste dicho procedimiento, metodología?
Las respuestas están en porcentajes y como es una pregunta abierta, se recibieron diferentes tipos de respuestas. Las más frecuentes son:

Ciclo de Vida de un producto de software: 41%
Tener información de requerimientos completa: 27%
Sólo el 17% de los encuestados no tenían idea de los procedimientos que se llevaban a cabo en las empresas a las cuales laboraban.
Sólo un 15% mencionó tener metodologías establecidas para asegurar la calidad.

7. ¿Ha dado resultado?
Las respuestas están en porcentajes y como es una pregunta abierta, se agrupan las respuestas y se le indican porcentaje de incidencia.

Encuestados que comentan resultados favorables: 42%
Encuestados que comentan resultados no favorables: 36%
No Contestaron: 22%

8. ¿Cómo han medido los resultados?
Las respuestas están en porcentajes y como es una pregunta abierta, las respuestas más indicadas son:

En base al cumplimiento de objetivos de requerimientos: 69%
Por comentarios de los departamentos usuarios: 11%
Algunos comentan no contar con métodos de medición: 20%

9. ¿Qué tiempo tiene actualmente este procedimiento, metodología?
Las respuestas están en porcentajes.

Encuestados que comentan un tiempo menor de un año: 22%
Encuestados que comentan un tiempo de 1 a 2 años: 17%
Encuestados que comentan un tiempo mayor de tres años: 61 %

10. ¿En qué tipo de aplicaciones está involucrado en estos momentos y qué rol desempeña?
Las respuestas están en porcentajes y como es una pregunta abierta, aquí tratamos de agrupar las respuestas y darles un porcentaje de incidencia.

Aplicaciones Técnicas: 36%
Sistemas Administrativos: 53%
No Contestaron: 11%

11. ¿El departamento o área de sistemas cuenta con alguna certificación de calidad como ISO, IEEE, ninguno, otros?
Las respuestas están en porcentajes. Encuestados que tienen conocimiento que en sus empresas llevan certificaciones de calidad en:

ISO: 23%
IEEE: 19%
Ninguno: 47%
Otros: 11%

12. ¿En qué fase de desarrollo se encuentran los proyectos en los que está involucrado actualmente (considere el proyecto de mayor prioridad)?

Fase de Análisis: 19%
Fase de desarrollo: 19%
Fase de Operación/Manto.: 44%
Solo el 18% esta involucrado en las demás fases del ciclo de vida de los productos de software.

13. ¿Con qué frecuenta recibe capacitación por parte de sus Jefes de Sección/Gerencia?

Encuestados que comentan un tiempo de dos veces al año: 17%
Encuestados que comentan un tiempo de cinco veces al año: 3%
Encuestados que comentan no recibir capacitación: 47%
Sólo un 33% mencionó otros intervalos de tiempo en los que reciben capacitación.

14. ¿Con qué frecuencias tienen auditorias de sistemas?

Encuestados que comentan un tiempo de una vez al año: 50%
Encuestados que comentan un tiempo de dos veces al año: 22%
Encuestados que comentan un tiempo de dos veces al año: 28%
Encuestados que comentan no tener conocimiento al respecto: 14%

15. ¿Qué tipos de auditorías manejan?

Encuestados que comentan manejar auditorías internas: 50%
Encuestados que comentan manejar auditorías externas: 33%
Encuestados que comentan no tener conocimiento al respecto: 17%

Ver el Anexo 1, donde se presenta la plantilla que se utilizó para la obtención de los resultados finales mostrados anteriormente.

FORMATO_2

Los números aquí presentes son el promedio obtenido, que nos da un panorama acerca de qué tanto conocen y toman en cuenta las personas involucradas en los departamentos desarrolladores de software a los Factores de Calidad.

Tabla 5.2 Promedios obtenidos, después de aplicar el formato_2.

Fact. de Calidad	Tamsa	Icave	Dos Bocas	Laguna Verde	Valor Final
Corrección	9.0	5.2	9.6	9.0	**8.2**
Confiabilidad	9.6	5.5	10.0	9.3	**8.6**
Eficiencia	7.3	4.5	8.0	8.0	**6.9**
Integridad	9.3	5.5	9.3	9.3	**8.3**
Fac. de Uso	6.6	4.8	7.6	8.0	**6.7**
Fac. de Mannto.	7.4	4.8	8.3	7.6	**7.0**
Flexibilidad	8.6	4.8	7.6	8.0	**7.2**
Fac. de Prueba	8.6	4.5	8.3	8.0	**7.3**
Portabilidad	6.6	3.5	5.6	5.6	**5.3**
Reusabilidad	7.0	4.2	5.6	5.5	**5.5**
Fac. de Interop.	8.3	4.5	8.6	7.0	**7.1**

Ver el Anexo 2, donde se presenta la plantilla que se utilizó para llevar a cabo los cálculos, obteniéndose con ellos los resultados finales.

FORMATO_3

Los números aquí presentes son el promedio obtenido, que da un panorama acerca de qué tanto conocen y toman en cuenta las personas involucradas en los departamentos desarrolladores de software a las Características de Entorno en la creación de sus productos finales. Es importante mencionar que los valores promedios obtenidos en este formato, arrojan los valores estándares que forman parte del cálculo que se efectuó para la obtención del cociente ***ai***.

Utilizar los valores promedio de las características de entorno para la obtención del cociente ***ai*** y no los valores promedio de los factores de calidad, es obvio que dan la pauta para evaluar más aún aquellos elementos básicos e importante que siempre afectan o no a los desarrollos de software.

Tabla 5.3 Promedios obtenidos, después de aplicar el formato_3.

Características de Entorno	Tamsa	Icave	Dos Bocas	Laguna Verde	Valor Final
Aplicación	7.0	4.5	9.3	8.6	**7.3**
Ambiente de Uso	7.0	3.2	4.3	5.3	**4.9**
Riesgo	7.4	5.2	8.6	8.6	**7.4**
Comp. Anfitriona	7.1	4.8	7.6	7.0	**6.6**
Mad. desarrollador	7.3	4.5	8.3	7.0	**6.7**
Exp. del Usuario	5.0	2.5	7.0	5.3	**4.9**
Apoyo Desarrollador	7.0	3.5	8.0	6.3	**6.2**
Exp. Desarrollador	7.6	4.5	8.3	7.6	**7.0**
Iterac. Usuario	8.3	5.5	8.0	9.3	**7.7**
Rest. Comercial	8.0	4.8	7.3	8.0	**7.0**
Método. Desarrollo	8.0	4.5	7.3	8.3	**7.0**
Leng. Programación	6.6	3.8	6.0	8.4	**6.2**
Complej. Software	6.3	3.2	7.6	7.6	**6.1**
Salud Desarrollador	6.0	3.2	7.3	7.3	**5.9**

Ver el Anexo 3, donde se presenta la plantilla la cual se utilizó para llevar a cabo los cálculos, obteniéndose con ellos los resultados finales.

FORMATO_4

Los números que se describen en la siguiente tabla (5.4), son el promedio final obtenido, después de haber llevado a cabo los cálculos pertinentes en la matriz de relación entre los factores de calidad de software y las características de entorno, lo que permitió conocer de las empresas encuestadas, qué tanto afecta una característica de entorno, a un factor de calidad, considerando que con ello, podremos definir si conocen, toman en cuenta y aplican ambos elementos claves de calidad de software en sus productos finales.

Tabla 5.4 Promedios obtenidos, después de aplicar el formato_4

Fact. de Calidad	Tamsa	Icave	Dos Bocas	Laguna Verde	% Final
Corrección	72.4 %	20.8 %	80.3 %	73.1%	**79.7 %**
Confiabilidad	62.6 %	29.6 %	86.6 %	72.4 %	**62.8 %**
Eficiencia	72.0 %	24.4 %	70-9 %	77.3 %	**61.1 %**
Integridad	71.2 %	22.7 %	80.1 %	64.3 %	**59.5 %**
Fac. de Uso	71.4 %	21.8 %	86.1 %	72.8 %	**63.0 %**
Fac. de Mannto.	68.6 %	22.4 %	70.7 %	64.8 %	**56.6 %**
Flexibilidad	69.6 %	25.6 %	70.3 %	68.8 %	**58.5 %**
Fac. de Prueba	72.0 %	17.3 %	74.7 %	68.1 %	**58.0 %**
Portabilidad	73.3 %	21.4 %	84.2 %	65.6 %	**61.1 %**
Reusabilidad	66.0 %	23.4 %	86.7 %	69.8 %	**61.4 %**
Fac. de Interop.	66.7 %	22.4 %	92.4 %	69.7 %	**62.8 %**

Ver el Anexo 4, donde se presenta la Matriz de Relación la cual se utilizó para llevar a cabo la obtención de los resultados finales mostrados anteriormente.

A continuación se muestran las gráficas de los resultados obtenidos, de cada empresa estudiada.

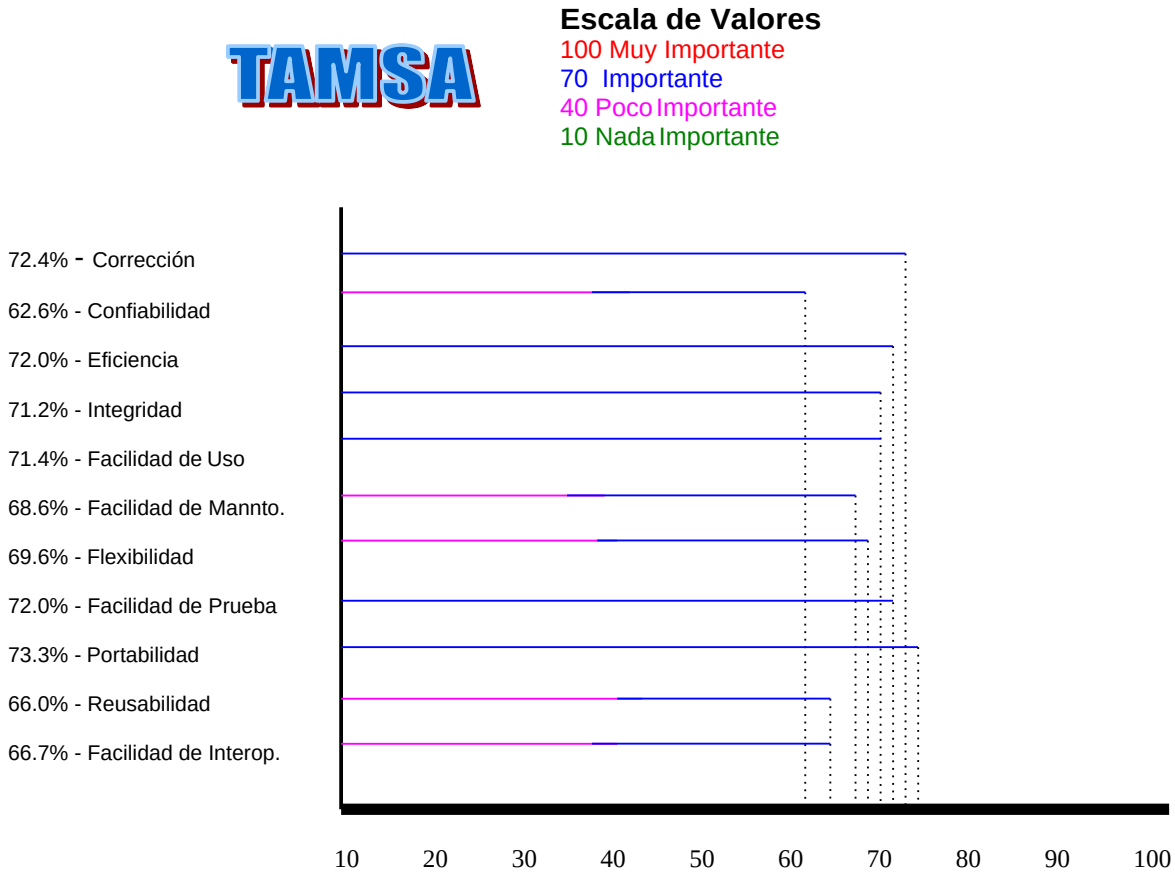


Figura 5.1 Gráfica Estadística de valores promedios obtenidos en el Formato_4, representando la Empresa TAMSA.

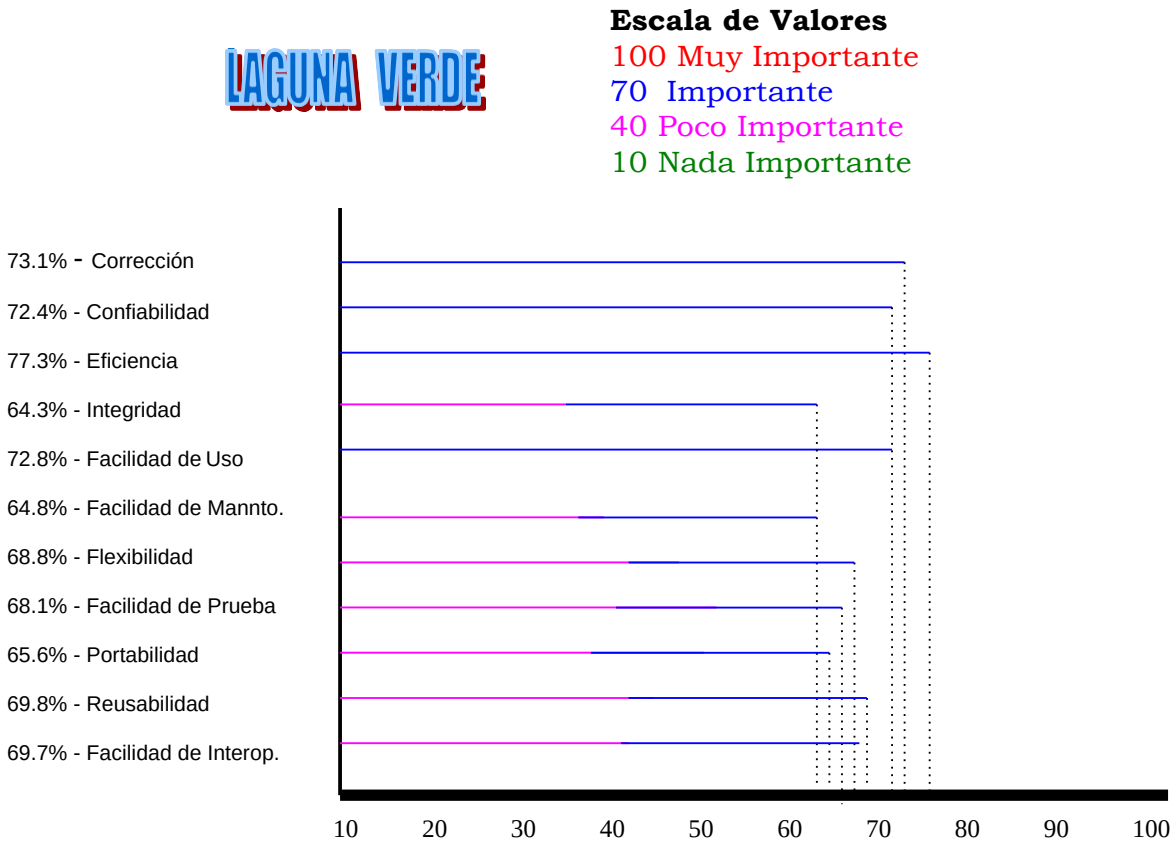


Figura 5.2 Gráfica Estadística de valores promedios obtenidos en el Formato_4, representando la Empresa C.F.E - Laguna Verde.

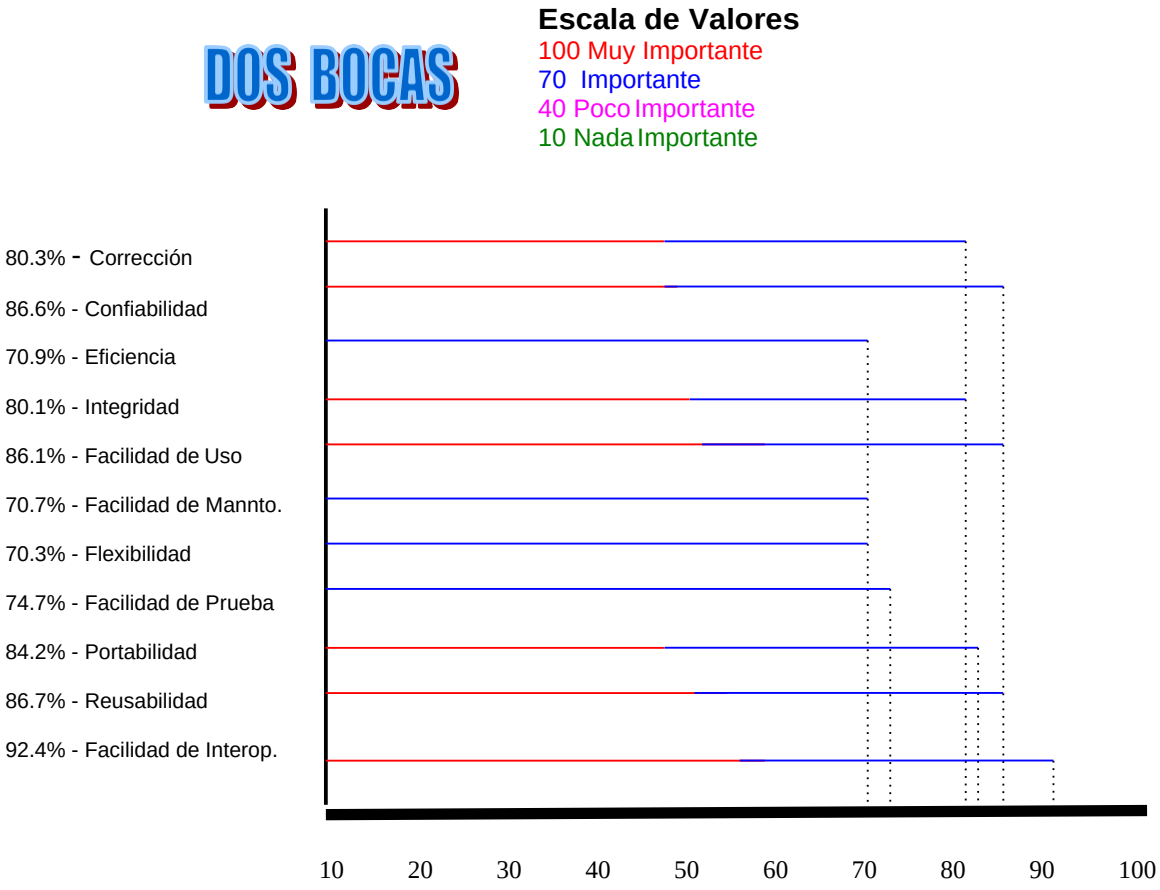


Figura 5.3 Gráfica Estadística de valores promedios obtenidos en el Formato_4, representando la Empresa C.F.E - Dos Bocas.

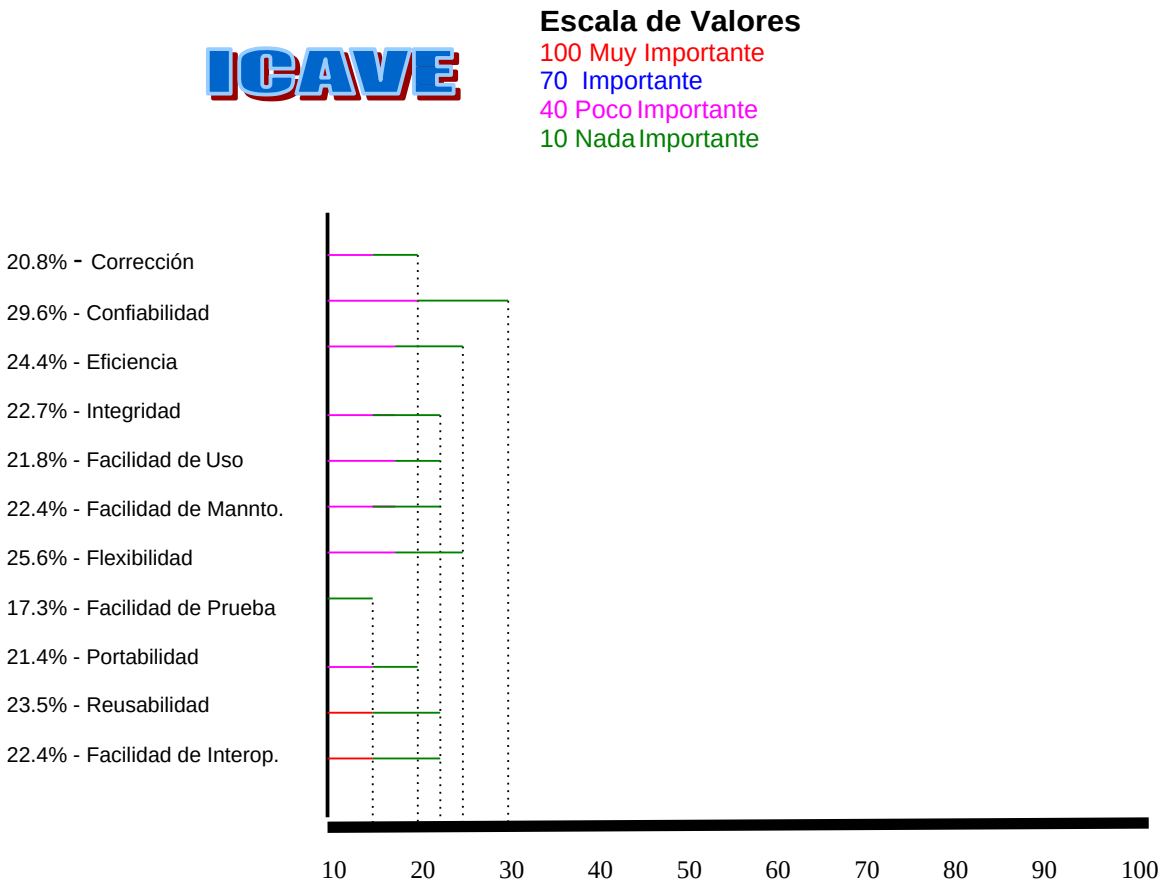


Figura 5.4 Gráfica Estadística de valores promedios obtenidos en el Formato_4, representando la Empresa ICAVE.

En la siguiente gráfica se puede visualizar, los promedios finales obtenidos de las cuatro empresas encuestadas, estos valores finales se ubican dentro de la escala que se ha planteado desde un inicio de la propuesta (10 - 100), es decir, la calificación final indicará los valores que nos han arrojado los cálculos efectuados, y la posición en la que estas empresas están.

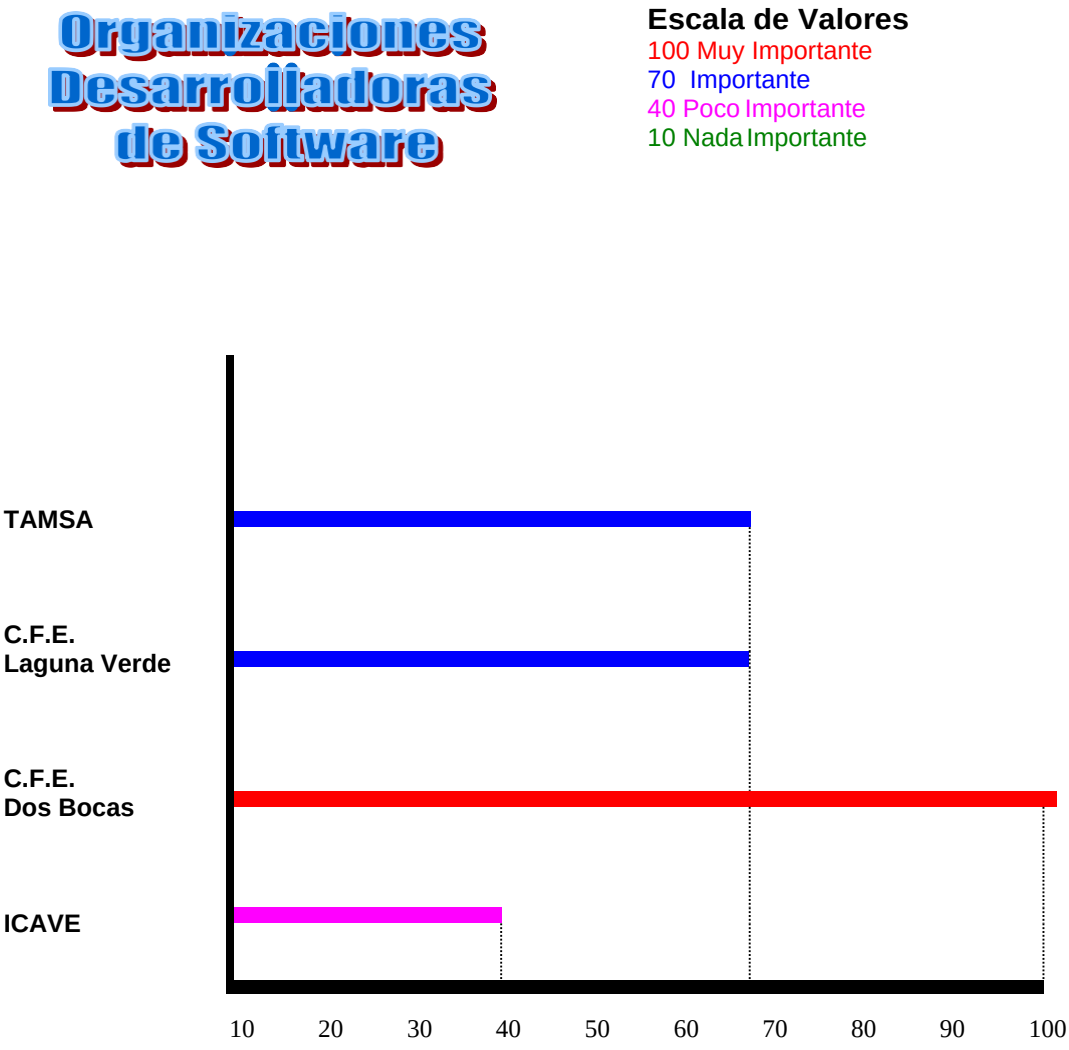


Figura 5.5 Gráfica Estadística de valores promedios de cada una de las empresas entrevistadas.

5.2 Definición de Indicadores de Calidad de Software

5.2.1 Elección de Acciones para mejorar la Calidad de Software

Los resultados que se presentan en esta sección, arrojan una serie de datos de gran relevancia a las personas que se preocupan por el aseguramiento de calidad de software.

Es importante que, aunque se trata de ser objetivo en la presentación de resultados, se incluirán algunas apreciaciones con la intención de ampliar la información de toda la investigación que se llevó a cabo. Los puntos más sobresalientes son los siguientes:

- El perfil de los encuestados muestra un porcentaje bajo (6%) de desarrolladores que utilizan modelos de información para implementar sistemas.

- Aunque la mayoría de los encuestados (42%) manifiesta que el aseguramiento de calidad de sus productos es muy importante, sólo el 8% cuenta con algún procedimiento específico para llevar a cabo la tarea.

- El porcentaje de profesionales encuestados que buscan el aseguramiento de calidad en sus productos desde la planeación es alto (53%) lo cual demuestra que existe una sensibilidad de hacer un esfuerzo mayor de prevención para evitar dentro de lo posible, los esfuerzos de corrección.

- Los desarrolladores que procuran no dejar el aseguramiento de calidad de software hasta la fase de prueba (haciendo de esto un procedimiento de control de calidad más que de aseguramiento de calidad) es bastante baja 11%.

- Aunque un 67% de los encuestados dijeron contar con algún procedimiento de aseguramiento de calidad sólo un 8% cuenta con una metodología específica para la tarea. Esto se pudiera entender como que el porcentaje restante hacen esfuerzos para lograr la calidad de sus productos con regularidad, aunque no de una manera disciplinada y rigurosa.

- El porcentaje de encuestados que contestaron de manera no muy favorable a la pregunta de si les ha dado resultados sus esfuerzos de aseguramiento de la calidad es de 36%. Esto pudiera sugerir que los resultados no son muy favorables para ellos, aunque sólo un 42% lo admite de manera favorable abiertamente.

- Una buena cantidad de los comentarios recibidos sobre cómo se han medido los resultados de tener un proceso de aseguramiento de calidad de software, ha sido sobre el hecho de que se han comprometido en una buena programación y que ésta sea a su vez más eficiente. Esto sugiere que deben aplicarse a un buen levantamiento de requerimientos y que la comunicación con el usuario es una actividad clave.

- Los factores de calidad que se consideraron más importantes (confiabilidad y facilidad de uso) son aquéllos que están directamente relacionados con la operación/iteración que tiene el usuario con el producto final. Es decir la principal preocupación de los desarrolladores está relacionada con primeras impresiones o asuntos a flor de piel que tiene un usuario cuando empieza a trabajar con un software. Esto se presenta, cuando los resultados que esperaban no son los esperados por el usuario.

- Los factores de calidad considerados como menos importantes (portabilidad y facilidad de interoperación) están ambos relacionados con que el software desarrollado pueda interactuar o ser transportado a otros entornos. Esto es quizás un reflejo de la madurez tecnológica en que viven inversos las empresas desarrolladoras de software, que normalmente su trabajo está enfocado a una plataforma específica y que éste se desecha o se rehace cuando tienen que cambiar la plataforma.

- Las características del entorno de desarrollo determinadas como más importantes son aquéllas que tienen que ver con la habilidad de los desarrolladores (madurez del desarrollador y experiencia del desarrollador) y la comunicación con el usuario (interacción con el usuario final). De manera particular, aquéllas relacionadas con los desarrolladores son de notarse, ya que la literatura normalmente no las considera dentro de las más relevantes.

- Las características del entorno del desarrollo que se mencionan como menos importantes son la aplicación que se dará al software y el lenguaje de programación a escoger para el mismo. En el caso del lenguaje de programación, puede ser que la mayoría de los encuestados den por hecho que ellos escogerán las herramientas que crean necesarias para la tarea y no que les serán impuestas.

- La salud de los desarrolladores es una característica que no está considerada en la literatura utilizada en el marco teórico. Originalmente fue introducida como un elemento de control para ver cómo se comportaban los encuestados ante una pregunta que al parecer estaba -fuera de lugar-. Como era de esperarse fue la peor evaluada, auque el porcentaje de la calificación promedio no es tan bajo tomando en cuenta lo antes expuesto.

- El análisis de los resultados de la matriz de relación entre los factores de calidad con las características del entorno serán expuestos de manera más clara en el ejemplo que se presenta en el capítulo siguiente, llegando de esta manera a la generación de mejores prácticas para el aseguramiento de calidad que se proponen en esta investigación para un mejor control de calidad en los desarrollos de productos finales de software.

5.3 Consideración

La obtención de resultados cualitativos y cuantitativos, permite identificar las posibles ventajas y desventajas, que las áreas de desarrollo de software tienen, en relación con los controles de calidad que conocen y además llevan a cabo. Es por ello que al proporcionarles nuevos elementos sobre nuevas prácticas, les puede aportar un mejor control y seguimiento en su desarrollo de software.

5.4.- Producto Final

En este espacio, se retoma el análisis de la relación que existe entre las características del entorno en el desarrollo de un producto y los factores de calidad del mismo. En base a estos resultados y usando la fórmula planteada para probar la hipótesis, se definen las mejores prácticas para el control de aseguramiento de calidad de software.

5.4.1.- Ponderación de Factores de Calidad

Para poder llegar a las prácticas que a continuación se muestran, la investigación tomó como base, fundamentos sólidos y válidos para que la investigación de campo pudiera ser medida cuantitativamente. Además se buscó un sistema métrico que representase lo que en primera instancia, parece cualitativo o de apreciación. Así, lo primero que se propone, es obtener los cocientes de importancia (***ai*** en la formula) de las características del entorno por cada factor de calidad.

Estos cocientes de importancia serán obtenidos directamente de la investigación de campo realizada, donde se establece la relación existente entre factores de calidad y características de calidad. Para esto se definieron valores numéricos a las evaluaciones de “afecta moderadamente” o “afecta fuertemente” de las tablas de evaluación presentadas ya con anterioridad, las ponderaciones quedaron como:

Afecta Moderadamente (M) ➔ 5
Afecta Fuertemente (F) ➔ 10

Estas ponderaciones son determinadas considerando el hecho de que una característica de entorno, afecta fuertemente a un factor de calidad, y tiene el doble de efecto que uno que lo afecta moderadamente. Posteriormente se le da un valor numérico a los porcentajes obtenidos. Será 1 por cada 100% de menciones, esto se hace con el propósito de tener un valor que pueda ser utilizado para cálculos aritméticos. De esta manera se calcularon los respectivos cocientes (***ai***) para todas las características de entorno de cada uno de los factores de calidad: Estos cocientes tienen que ser calculados cada vez que se requiera establecer los cálculos respectivos en la matriz de relación para poder generar las prácticas de calidad de software.

Los cálculos que se efectúan en los formatos ya mencionados (Formatos_2, Formato_3 y Formato_4) deben ser utilizados como tablas de referencia cuando sea necesario obtener un cociente específico. Una vez obtenidos todos los valores y cocientes, se cuenta ya con la materia prima para determinar y generar las prácticas a evaluar y aplicar en las entidades respectivas donde se lleve a acabo este ejercicio de calidad de software.

Al término de esta sección, lo que se tendrá será un número asociado a cada factor de calidad, estos valores llevan el nombre de grado de importancia del factor de calidad. Al total de factores con su grado de importancia ayudan a generar las mejores prácticas para el aseguramiento de calidad de software.

Los factores con el grado de importancia más alto son aquellos que debe asegurarse se encuentren presentes al tener el producto de software terminado. Los pasos a seguir para obtener las prácticas de un producto de software en una entidad generadora de productos de software son:

i. Determinar la naturaleza de la aplicación, formato_1 (Sistemas, Modelos, Prototipos, etc.)

ii. Asignar el grado de importancia de cada una de las características del entorno (C) así como los factores de calidad dado un valor entre 1 y 10 (1 nada importante, 10 muy importante) que se verán reflejadas en el formato_2 y formato_3. Es importante mencionar que cuando se realice la suma total de todos los cocientes correspondientes de cada uno de los factores de calidad se tomará en cuenta la escala de 1 a 100 para con ello definir los valores numéricos representativos de esta escala.

    Tanto las características de entorno como los factores de calidad de software se han definido a detalle en el capítulo 2. Esto debe hacerse teniendo en cuenta los requerimientos del usuario obtenidos en la etapa de análisis del producto de software y la naturaleza de la aplicación.

iii. Obtener el grado de importancia (F(c)) para cada unos de los factores de calidad (formato_4), utilizando la siguiente formula:

$$F(c) = \sum_{i=1}^{N} (aiCi) = a1C1 + a2C2 + a3C3 + \ldots anCn$$

Donde:

F(c): Representa el valor final del grado de importancia de un factor de calidad.

ai: Representa un cociente de importancia (ponderación de la característica) por cada factor. Estos valores deberán ser obtenidos de las tablas expuestas en el anexo_2.

Ci: Representa el valor del grado de importancia de una característica del entorno de la aplicación. Estos valores deberán ser obtenidos de las tablas expuestas en el anexo_3.

n: Representa el número de características del entorno de la aplicación ser tomadas en cuenta.

iv. Enlistar los factores con sus grados de importancia en orden descendente y esta lista nos ayudará a definir los valores numéricos que nos darán la pauta para conocer en qué situación actual se encuentra la entidad evaluada y así poder definir las prácticas a seguir en el aseguramiento de calidad de software.

Las prácticas para el control de calidad de software, deben de contemplarse y tomarse en cuenta desde la etapa de especificación del requerimiento, análisis, porque como se ha explicado en los capítulos anteriores la calidad debe ser planeada y aplicada. Para tener una manera de visualizar limpiamente el proceso antes expuesto, se propone tomar en cuenta la plantilla que se muestra en el Anexo_5, esto es para el caso de aplicar el ejercicio a un sistema de información particular.

5.4.2.- Mejores Prácticas a seguir para el Aseguramiento de Calidad de Software.

La siguiente lista de prácticas que a continuación se muestran se ha definido a través de todo el ejercicio que hemos propuesto anteriormente y cuyo objetivo es presentar los lineamientos a seguir para el control de calidad de los productos de software que se desarrollen en empresas de este giro.

Cada una de estas recomendaciones, se presentan por las necesidades y falta de conocimiento de elementos claves que son de gran importancia y que deben de contemplarse en cada unos de sus productos finales. Estas prácticas se presentan a continuación:

- Durante la fase de especificación de requerimientos del software, definir de manera clara y precisa todas y cada una de las funcionalidades que el sistema en cuestión debe cubrir.

- Considerar que el software a corregir dentro de las especificaciones iniciales no debe de perder la misión encomendada por el usuario, a menos que esté justifique un cambio radical y sea valido alterar las especificaciones originales.

- Llevar control en las versiones generadas de las funcionalidades, que éstas ameriten efectuar eventos tales como; llevar a cabo operaciones del producto, revisiones del producto y transición del mismo.

- Llevar la documentación adecuada del historial generado por el producto de software, en donde en este se reflejen las acciones y eventos, para con ello verificar si realmente el producto está cumpliendo con lo que el usuario/cliente solicita.

- Llevar el control de las fallas de software, dentro de un entorno determinado y dentro de periodos específicos que ayuden más adelante a prevenir y evitar más fallas llegando a la confiabilidad de los productos finales.

- Generar una base de conocimiento en donde se estén registrando estas mismas acciones, así como definir perfiles de acceso por sección.

- Buscar en cada una de las funcionalidades del software la eficiencia, tomando en cuenta todos los recursos de hardware y software que son requeridas por éste para llevar a cabo todas las funciones encomendadas.

- Definir algún documento (Formato) en el que se estipule lo antes mencionado, esto con el objetivo de que queden todas las acciones a desarrollar bajo conocimiento del usuario/cliente, así como también de los mismos miembros de equipo de desarrolladores de software.

- Llevar a cabo el control en el seguimiento de los diferentes accesos en cuanto la integridad del uso y acceso al software y a los datos por parte del personal autorizado y no autorizado.

- Hacer conciencia en los usuarios que todas y cada una de las acciones y eventos a emprender se analizarán en tiempo y recurso (recursos económicos y humanos) y éstas mismas se evaluarán con respecto a las especificaciones originales.

- Interactuar con los usuarios y apoyarlos con manuales y procedimientos para darles una mejor interpretación en cuanto al uso de cada unos de los productos de software que se les entrega, y que les permita con el tiempo poder ejecutarlo sin ningún inconveniente.

- Tomar en cuenta a una herramienta de apoyo en la generación de flujo de procesos que se vayan generando como parte del historial documental del producto de software.

- Tener la actitud de servicio y colaboración hacia los usuarios para facilitarles los mantenimientos y errores que pudieran presentarse en los productos de software los cuales ellos manejan, y de esta manera, ellos en algún momento tengan la confianza y el conocimiento de poder dar solución a los problemas que se les presenten.

- Llevar comunicación amigable con los usuarios e involucrarlos en todas las actividades que originen nuevos eventos claves a llevar a cabo.

- Hacer conciencia en el usuario de cada ambiente de prueba por los que pasan los desarrollos así como las revisiones que estos presentan y brindarles la flexibilidad para poder modificar alguna opción del software, ya sea que este producto esté liberado o en producción, un vez que el usuario este haciendo uso de él.

- Definir a una persona o grupo de personas denominadas, Ingenieros de procesos que den seguimiento y certifiquen todas y cada una de las acciones a llevar a cabo con el producto de software.

- Darle la confianza al usuario y la credibilidad de que puede probar el producto de software, de tal manera de que se asegure que éste está realizando todo lo planeado.

- Generar escenarios de prueba para la validación y certificación de los diferentes eventos a llevar a cabo.

- Desarrollar procedimientos y planes de contingencia si surge la necesidad de transferir un producto de software de una plataforma a otra, sea ésta de hardware o software.

- Contar con un mapa oficial de modelado de todos y cada uno de los flujos de trabajo, en el que se manifiesten los procesos que dan vida a los productos de software que se manejen dentro y fuera de la empresa. Esto servirá para el personal involucrado en diferentes desarrollos de proyectos de software, como a su vez para el personal que se va incorporando en la empresa.

- Contemplar la reingeniería de procesos en los desarrollos actuales de software para otros desarrollos, aún cuando la funcionalidad de la misma cambie.

- Tener al personal de desarrollo de software, motivado, capacitado y actualizado en procesos y estrategias de administración de proyectos, especificación de requerimientos, y calidad de vida, por mencionar algunos temas en particular.

- Hacer conciencia en los usuarios de los recursos actuales de desarrollo con que cuenta el área para poder hacerlos interactuar con otros sistemas.

- Realizar auditorias constantes de software y hardware dentro de la empresa, para ir generando con ello una cultura de trabajo bajo control y en las mejores condiciones de emprender cualquier producto que se solicite.

- Realizar secciones constantes con todos los miembros del departamento, con fines de llevar un mejor ambiente de trabajo.

5.4.3.- Descripción de los resultados de los casos estudiados.

Ejemplo del procedimiento aplicado en el estudio

Con la intención de clarificar el procedimiento que se ha desarrollado en los estudios de caso llevado a cabo en las entidades participantes durante este proceso de investigación, a continuación se describe un ejemplo del ejercicio en mención, para con ello, poder encaminar al lector interesado en este tema, a la aplicación de los procedimientos y mejores prácticas de calidad que deban considerar, a efecto de emprender sus productos de software en las entidades a las cuales representan.

El ejemplo muestra la forma actual de trabajo de la empresa en cuestión, mismo ejercicio en el que se denotan las necesidades y falta de conocimiento de los elementos claves en el desarrollo del software, que sus empresas generan

5.4.3.1.- Caso de Estudio “Empresa de Gobierno”

La empresa que se maneja como ejemplo es C.F.E Laguna Verde, esta entidad forma parte importante de las Plantas Nucleoeléctricas del País. En el departamento de Ingeniería de Software el cual se encuestó, se cuenta con 22 personas en el grupo de desarrollo de software.

Cabe mencionar que esta organización es de tipo pública, dedicada a la generación, distribución y comercialización de energía eléctrica, cuyos desarrollos de software se llevan a cabo principalmente bajo estas necesidades. Al entrevistar a las personas que en su momento se tomaron como grupo representativo de profesionales desarrolladores de software, se percato el interés por la investigación que se estaba emprendiendo en su ambiente de trabajo

Algunos de los encuestados no tenían conocimiento o no les era familiar algunos de los elementos o términos que manejábamos en las encuestas a aplicar, pero fue de mucha utilidad las explicaciones manejadas en cada uno de los formatos que mostrábamos y así de esta manera se fueron encaminando las entrevistas y no se presentaron más problemas.

Actividades:

i. Al ser entrevistado el grupo de personas bajo el rol de analistas y programadores encontramos que principalmente los productos a realizar para esta empresa consisten en desarrollos de Sistemas de Información, más que desarrollos de modelos y prototipos.

ii. Enseguida se hará la ponderación a cada uno de los factores de calidad como también a las características de entorno dando algunas explicaciones en las ocasiones que se consideran pertinentes.

iii. Formato_2

Num. de Encuestados	FACTORES DE CALIDAD DE SOFTWARE Corrección Confiabilidad Eficiencia Integridad Fac. de Uso Fac. Mannto. Flexibilidad Fac. de Prueba Portabilidad Reusabilid. Fac. Inter.										
√	I = 7	M = 10	M = 10	M = 10	M = 10	I = 7	M = 10	M = 10	I = 7	N = 1	M = 10
√	M = 10	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7
√	M = 10	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7	I = 7
√	M = 10	M = 10	I = 7	M = 10	I = 7	I = 7	I = 7	I = 7	P = 4	P = 4	I = 7
√	M = 10	M = 10	I = 7	M = 10	M = 10	M = 10	I = 7	M = 10	P = 4	I = 7	I = 7
√	I = 7	M = 10	M = 10	M =10	M = 10	I = 7	I = 7	M = 10	I = 7	I = 7	M = 10
√	M = 10	M = 10	I = 7	M = 10	I = 7	I = 7	M = 10	P = 4	N = 1	N = 1	P = 1
√	I = 7	M = 10	I = 7	M =10	I = 7	M = 10	M = 10	M= 10	I = 7	I = 7	I = 7
√	M = 10	M = 10	M = 10	M = 10	I = 7	I = 7	I = 7	I = 7	I = 7	M = 10	I = 7
	81/9 9 %	84/9 9.3 %	72/9 8 %	84/9 9.3 %	72/9 8 %	69/9 7.6 %	72/9 8 %	72/9 8 %	51/9 5.6 %	51/9 5.6 %	63/9 7 %

Escala a Manejar de 1 a 10 %
10 % Muy Importante
7 % Importante
4 % Poco Importante
1 % Nada Importante

Formato_3

Num. de Encuestados	CARACTERISTICAS DE ENTORNO DE CALIDAD DE SOFTWARE													
	Aplicac.	Amb./Uso	Riesgo	Comp.Anf	Mad./Desa	Exp./Usu	Apo./Desa	Exp./Desa	Inter/Usu	Res./Com	Met./Desa	Leng./Prog	Comp./Soft	Salud
√	I = 7	I = 7	M=10	I = 7	M=10	P = 4	I = 7	I = 7	M=10	M=10	M =	I = 7	I = 7	I = 7
√	M=10	I = 7	M=10	M=10	I = 7	I = 7	P = 4	M=10	M=10	M=10	I = 7	M=10	M=10	I = 7
√	M=10	N = 1	M=10	M=10	N = 1	I = 7	P = 4	M=10	M=10	M=10	M=10	M=10	M=10	I = 7
√	M=10	N = 1	N = 1	N = 1	N = 1	N = 1	N = 1	N = 1	M=10	N = 1	P = 1	N = 1	I = 7	I = 7
√	M=10	M=10	M=10	I = 7	M=10	I = 7	I = 7	M=10	P = 4	P = 4	M=10	I = 7	M=10	I = 7
√	I = 7	I = 7	M=10	I =7	M=10	P = 4	I = 7	I = 7	M=10	M=10	M=10	I = 7	I = 7	M=10
√	M=10	N = 1	I = 7	I = 7	I = 7	P = 4	I = 7	I = 7	M=10	M=10	M=10	I = 7	I = 7	I = 7
√	I = 7	I = 7	M=10	I =7	I = 7	I = 7	M=10	I = 7	M= 10	M=10	I = 7	I = 7	P = 4	M=10
√	I = 7	I = 7	M=10	I = 7	M=10	I = 7	M=10	M=10	M=10	I = 7	M=10	M=10	I = 7	P = 4
	78/9 **8.6 %**	48/9 **5.3 %**	63/9 **8.6 %**	63/9 **7 %**	48/9 **7 %**	57/9 **5.3 %**	69/9 **6.3 %**	84/9 **7.6 %**	84/9 **9.3 %**	72/9 **8 %**	75/9 **8.3 %**	76/9 **8.4 %**	69/9 **7.6 %**	69/9 **7.3 %**

Escala a Manejar de 1 a 10 %
10 % Muy Importante
7 % Importante
4 % Poco Importante
1 % Nada Importante

iv. Posteriormente se obtiene el grado de importancia (F(c)) para cada unos de los factores de calidad, utilizando la siguiente formula:

$$F(c) = \sum_{i=1}^{N} (aiCi) = a1C1 + a2C2 + a3C3 + \ldots anCn$$

Corrección:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.8	8.6	6.8	
Ambiente de Uso	0.65	5.3	3.4	
Riesgo	0.6	8.6	5.1	
Computadora Anfitriona	0.45	7	3.1	
Madures del Desarrollador	0.6	7	4.2	
Experiencia del Usuario	0.8	5.3	4.2	73.1
Apoyo de los Desarrolladores	0.6	6.3	3.7	
Exp. de los Desarrolladores	0.85	7.6	6.4	
Interacción con el Usuario	O.85	9.3	7.9	
Restricciones Comerciales	0.75	8	6	
Metodología de Desarrollo	0.75	8.3	6.2	
Lenguajes de Programación	0.65	8.4	5.4	
Complejidad de Software	0.65	7.6	4.9	
Salud de los Desarrolladores	0.8	7.3	5.8	

Confiabilidad:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.9	8.6	7.7	
Ambiente de Uso	0.65	5.3	3.4	
Riesgo	0.85	8.6	7.3	
Computadora Anfitriona	0.6	7	4.2	
Madures del Desarrollador	0.7	7	4.9	
Experiencia del Usuario	0.65	5.3	3.4	72.4
Apoyo de los Desarrolladores	0.7	6.3	4.4	
Exp. de los Desarrolladores	0.75	7.6	5.7	
Interacción con el Usuario	0.7	9.3	6.5	
Restricciones Comerciales	0.65	8	5.2	
Metodología de Desarrollo	0.7	8.3	5.8	
Lenguajes de Programación	0.6	8.4	5	
Complejidad de Software	0.65	7.6	4.9	
Salud de los Desarrolladores	0.55	7.3	4	

Eficiencia:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.8	0.65	6.8	
Ambiente de Uso	0.65	5.3	3.4	
Riesgo	0.75	8.6	6.4	
Computadora Anfitriona	0.75	7	5.2	
Madures del Desarrollador	0.65	7	4.5	
Experiencia del Usuario	0.65	5.3	3.4	77.3
Apoyo de los Desarrolladores	0.6	6.3	3.7	
Exp. de los Desarrolladores	0.7	7.6	5.3	
Interacción con el Usuario	0.65	9.3	6	
Restricciones Comerciales	0.7	8	5.6	
Metodología de Desarrollo	0.45	8.3	3.7	
Lenguajes de Programación	0.6	8.4	5	
Complejidad de Software	0.65	7.6	4.9	
Salud de los Desarrolladores	0.45	7.3	3.2	

Integridad:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.8	8.6	6.8	
Ambiente de Uso	0.75	5.3	3.9	
Riesgo	0.8	8.6	6.8	
Computadora Anfitriona	0.75	7	5.2	
Madures del Desarrollador	0.6	7	4.2	
Experiencia del Usuario	0.55	5.3	2.9	64.3
Apoyo de los Desarrolladores	0.6	6.3	3.7	
Exp. de los Desarrolladores	0.7	7.6	5.3	
Interacción con el Usuario	0.6	9.3	5.5	
Restricciones Comerciales	0.45	8	3.6	
Metodología de Desarrollo	0.4	8.3	3.3	
Lenguajes de Programación	0.6	8.4	5	
Complejidad de Software	0.6	7.6	4.5	
Salud de los Desarrolladores	0.5	7.3	3.6	

Facilidad de Uso:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.85	8.6	7.3	
Ambiente de Uso	0.8	5.3	4.2	
Riesgo	0.55	8.6	4.7	
Computadora Anfitriona	0.75	7	3.8	
Madures del Desarrollador	0.75	7	5.2	
Experiencia del Usuario	0.75	5.3	3.9	72.8
Apoyo de los Desarrolladores	0.65	6.3	4.0	
Exp. de los Desarrolladores	0.9	7.6	6.8	
Interacción con el Usuario	0.85	9.3	7.9	
Restricciones Comerciales	0.5	8	4	
Metodología de Desarrollo	0.65	8.3	5.3	
Lenguajes de Programación	0.65	8.4	5.4	
Complejidad de Software	0.8	7.6	6	
Salud de los Desarrolladores	0.6	7.3	4.3	

Facilidad de Mantenimiento:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.85	8.6	7.3	
Ambiente de Uso	0.8	5.3	4.2	
Riesgo	0.55	8.6	4.7	
Computadora Anfitriona	0.55	7	3.8	
Madures del Desarrollador	0.75	7	5.2	
Experiencia del Usuario	0.75	5.3	3.9	72.8
Apoyo de los Desarrolladores	0.65	6.3	4.0	
Exp. de los Desarrolladores	0.9	7.6	6.8	
Interacción con el Usuario	0.85	9.3	7.9	
Restricciones Comerciales	0.5	8	4	
Metodología de Desarrollo	0.65	8.3	5.3	
Lenguajes de Programación	0.65	8.4	5.4	
Complejidad de Software	0.8	7.6	6	
Salud de los Desarrolladores	0.6	7.3	4.3	

Flexibilidad:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.7	8.6	6.0	
Ambiente de Uso	0.55	5.3	2.9	
Riesgo	0.7	8.6	6.0	
Computadora Anfitriona	0.55	7	3.8	
Madures del Desarrollador	0.9	7	6.3	
Experiencia del Usuario	0.55	5.3	2.9	68.8
Apoyo de los Desarrolladores	0.85	6.3	5.3	
Exp. de los Desarrolladores	0.85	7.6	6.4	
Interacción con el Usuario	0.7	9.3	6.5	
Restricciones Comerciales	0.45	8	3.6	
Metodología de Desarrollo	0.7	8.3	5.8	
Lenguajes de Programación	0.75	8.4	6.3	
Complejidad de Software	0.6	7.6	2.7	
Salud de los Desarrolladores	0.6	7.3	4.3	

Facilidad de Prueba:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.65	8.6	5.5	
Ambiente de Uso	0.7	5.3	3.7	
Riesgo	0.7	8.6	6.0	
Computadora Anfitriona	0.55	7	3.8	
Madures del Desarrollador	0.8	7	5.6	
Experiencia del Usuario	0.7	5.3	3.7	68.1
Apoyo de los Desarrolladores	0.65	6.3	4.0	
Exp. de los Desarrolladores	0.8	7.6	6.0	
Interacción con el Usuario	0.8	9.3	7.9	
Restricciones Comerciales	0.4	8	3.2	
Metodología de Desarrollo	0.6	8.3	4.9	
Lenguajes de Programación	0.55	8.4	4.6	
Complejidad de Software	0.55	7.6	4.1	
Salud de los Desarrolladores	0.7	7.3	5.1	

Portabilidad:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.75	8.6	6.4	
Ambiente de Uso	0.65	5.3	3.4	
Riesgo	0.6	8.6	4.9	
Computadora Anfitriona	0.75	7	5.2	
Madures del Desarrollador	0.8	7	5.6	
Experiencia del Usuario	0.5	5.3	2.6	65.6
Apoyo de los Desarrolladores	0.75	6.3	5.7	
Exp. de los Desarrolladores	0.8	7.6	5.0	
Interacción con el Usuario	0.45	9.3	4.1	
Restricciones Comerciales	0.6	8	4.8	
Metodología de Desarrollo	0.6	8.3	4.9	
Lenguajes de Programación	0.7	8.4	4.1	
Complejidad de Software	0.65	7.6	4.9	
Salud de los Desarrolladores	0.55	7.3	4.0	

Reusabilidad:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.9	8.6	7.7	
Ambiente de Uso	0.65	5.3	3.4	
Riesgo	0.7	8.6	6.0	
Computadora Anfitriona	0.45	7	3.1	
Madures del Desarrollador	0.75	7	5.2	
Experiencia del Usuario	0.5	5.3	2.6	69.8
Apoyo de los Desarrolladores	0.75	6.3	4.7	
Exp. de los Desarrolladores	0.85	7.6	6.4	
Interacción con el Usuario	0.5	9.3	4.6	
Restricciones Comerciales	0.6	8	4.8	
Metodología de Desarrollo	5.8	8.3	5.8	
Lenguajes de Programación	0.8	8.4	6.7	
Complejidad de Software	0.6	7.6	4.5	
Salud de los Desarrolladores	0.6	7.3	4.3	

Facilidad de Interoperación:

Característica./Factor	a	c	a*c	∑ a*c =F
Aplicación	0.9	8.6	7.7	
Ambiente de Uso	0.65	5.3	3.4	
Riesgo	0.75	8.6	6.4	
Computadora Anfitriona	0.75	7	5.2	
Madures del Desarrollador	0.45	7	3.1	
Experiencia del Usuario	0.5	5.3	2.6	69.7
Apoyo de los Desarrolladores	0.7	6.3	4.4	
Exp. de los Desarrolladores	0.85	7.6	6.4	
Interacción con el Usuario	0.45	9.3	4.1	
Restricciones Comerciales	0.5	8	4	
Metodología de Desarrollo	0.7	8.3	5.8	
Lenguajes de Programación	0.75	8.4	6.3	
Complejidad de Software	0.8	7.6	6.0	
Salud de los Desarrolladores	0.6	7.3	4.3	

v. Enlistar los factores con sus grados de importancia en orden descendente.

Fact. de Calidad	**Laguna Verde**
Eficiencia	77.3 %
Corrección	73.1 %
Facilidad de Uso	72.8 %
Confiabilidad	72.4 %
Fac. de Interop.	69.7 %
Reusabilidad	69.8 %
Flexibilidad	68.8 %
Facilidad de Prueba	68.1 %
Portabilidad	65.8 %
Facilidad de Mannto.	64.8 %
Integridad	64.3 %

En este ejemplo se refleja que la empresa en mención ha denotado valores numéricos no muy altos y aceptables respecto a los factores de calidad de software, es decir, que existen algunas deficiencias en la formas de llevar a cabo controles de calidad o simplemente aquellos elementos claves que encaminen de mejor forma sus productos de software, respecto a la calidad de mismo. Las gráficas 5.2 y 5.5 presentadas anteriormente, muestra resultados similares.

Con estos valores obtenidos no quiere decir que la empresa sea muy mala, o sus productos estén mal desarrollados; ése no ha sido el objetivo de este trabajo, pero sí es su objetivo el poder contribuir a las entidades bajo este giro, algunas prácticas de apoyo y seguimiento en el aseguramiento de calidad en sus productos de software, y dar inicio con ello, a la adquisición de nuevo aprendizaje y que este documento pueda aportar elementos a tomar en cuenta para una mejor forma de trabajo.

En el capítulo anterior se redactaron las prácticas obtenidas como resultado de este ejercicio y de todo el análisis realizado durante esta investigación; estos lineamientos deberán ser aplicados según lo consideren las organizaciones, ya que estas prácticas están dirigidas a las entidades cuya necesidad sea la de establecer controles y seguimientos en sus desarrollos de software.

Conclusiones

El producto final propuesto en este capítulo es en su totalidad cuantitativo, ya que puede ser calculado con una fórmula matemática. Sin embargo, es también fácil hacer notar como algunos factores son identificables como importantes desde un principio.

a) Sobre la Hipótesis:

El aseguramiento de calidad de los productos de software es una actividad necesaria, pero que debe planearse correctamente. Por un lado el contar con una metodología de aseguramiento de calidad cuando se desarrolla software puede terminar con aquellas situaciones en la que la producción de software se generé con muchos defectos, así con altos costos que hay que solventar por motivo de estas correcciones, y lo más importante, la insatisfacción del usuario.

Por otro lado contar con estas metodologías pudiera significar incurrir en costos de implementación y la realización de estudios de análisis costo-beneficio. Una buena planeación de calidad puede ayudar a determinar con tiempo y prever todo este tipo de detalles.

La investigación aquí presentada intenta hacer una aportación empírica con respecto a la planeacion de la calidad de un producto de software, advirtiendo además los costos en que se podrá incurrir para cumplir con esta tarea. Además se planteó una hipótesis bajo la cual se había indicado: e*l conocimiento del grado de calidad de un software obtenido por medio de la medición de sus factores de calidad, permitirá aplicar mejores prácticas para el aseguramiento de calidad de software".*

Una vez obtenidos los elementos que favorecen la prueba de la hipótesis, se llevó a cabo la medición de los factores de calidad, mismos que fueron presentado durante la investigación de campo; estos valores numéricos dan la pauta para reflexionar y aplicar prácticas de aseguramiento de calidad en los productos de software.

b) Sobre el Aprendizaje Adquirido

El aprendizaje adquirido a través de este estudio, se resume en los siguientes puntos:

- Es requerido un sustento teórico para el desarrollo de la investigación y su posterior prueba de hipótesis. Sin la investigación bibliográfica que avalara la importancia del aseguramiento de la calidad en el software, habría sido irrelevante proponer una manera óptima de planear su calidad.

- La calidad de un producto de software, está sujeta a una serie de factores internos y externos aquí llamados características de entorno, las cuales deben ser tomadas en cuenta. Si no se les da la importancia requerida, los costos por correcciones y modificaciones pueden ser muy altos o las consecuencias por resultados erróneos, serían cuantiosas.

- Es posible encontrar métodos cuantitativos para medir conceptos tan cualitativos como la calidad.

- El primer paso para buscar la calidad es la planeación de la misma.

- Existe diversa literatura sobre el aseguramiento de la calidad de software. Varios conceptos como en el caso de las definiciones de los factores de calidad y las características del entorno, algunas veces tienen significados encontrados, por lo cual es valioso hacer la investigación bibliográfica a fondo e ir comparando toda la información recaudada, así como estudiar e identificar los conceptos afines a la investigación y proponer elementos comunes.

- No existe una herramienta única para medir la importancia de los factores de calidad. Existen otros medios cuantitativos para hacerlo, lo que se presenta aquí es sólo una de entre tantas herramientas.

- No importa qué tan extensa sea la literatura acerca de las características de entorno de desarrollo, siempre habrá características nuevas propuestas por desarrolladores durante las investigaciones de campo. Es importante y necesario delimitar claramente desde un principio aquellas que no serán utilizadas.

c) Sobre el Producto Final

Las prácticas para el aseguramiento de calidad de software que se han generado bajo esta investigación de campo, en conjunto originan una herramienta cuantitativa, que puede ayudar a

planear la calidad de un producto dado. Desde luego no es la única herramienta, por el contrario en la literatura revisada se vienen proponiendo otros instrumentos que al igual al que se propone, pudieran favorecer para cumplir y dar apoyo al control de calidad.

Dentro de los problemas actuales del aseguramiento de calidad, contar con prácticas de calidad para un producto de software, puede ser el primer paso para resolver el problema de reacción a la calidad (*actuar una vez ya detectada la falla*) ya que permite conocer las características necesarias que debe tener desde la etapa de planeacion.

Una vez planeada la calidad, puede escogerse una selección de métricas para verificar que cada uno de los factores que se estén llevando a cabo, se realice correctamente.

Sugerencias y Trabajos Futuros

a) Sobre las Áreas de Oportunidad

Diseñar instrumentos de medición, para una investigación de campo es una tarea compleja y desde luego muy importante. Si en un futuro se volviera a aplicar la encuesta, sería recomendable incluir espacios para comentarios adicionales, en las preguntas que buscan la relación entre los factores de calidad y las características de entorno. Ello nos daría la oportunidad de ver si pueden aportar más elementos o criterios particulares.

En cuanto al análisis de resultados, sería recomendable validar nuevas formas estadísticas de medición, digamos, alternas a lo que en este estudio, se desarrollo. Analizar la mezcla de características de factores que afectan a un factor de calidad, entendiéndose como mezcla, a la interpretación de los porcentajes de las respuestas que involucran a más de una característica por factor.

Ejemplo: un 85 % de los encuestados opinan que la experiencia de los desarrolladores y la metodología de desarrollo afectan el factor de corrección. Un área interesante de oportunidad es aplicar la encuesta a una muestra más amplia de empresas, aprovechando la proliferación de las empresas de tipos Web y de las dedicadas al comercio electrónico.

Recientemente con la ayuda del Internet se han creado grupos de organizaciones que se dedican al aseguramiento de calidad en la industria del software como el Quality Assurance Institute (http:/www.qaiusa.com/) y el Software Quality Engineering (http:/www.sqe.com/) así que valdría la pena investigar más a fondo acerca de estos temas de interés.

b) Sobre las investigaciones Futuras

La presente investigación deja una amplia gama de temas que pueden ser utilizados para trabajos futuros. En este caso se mencionan sólo algunos de los siguientes temas de interés.

- Planeación estratégica en la Evaluación de Metodologías de Calidad de Software en Latinoamérica.

- Modelo de reducción de riesgos sobre las fallas de software ante un mal QA.

- Cómo los Procesos de COBIT y RUP ayudan a las empresas de Latinoamérica en el QA.

- Modelo de Auditoría en la administración del desarrollo de sistemas de información.

- Análisis del impacto sociológico del QA de software en las estructuras organizacionales.

- Contemplar los lineamientos expuestos con la guía ISO-9000-3, que es aquélla descrita específicamente para la industria del desarrollo de software.

- Proponer más características de entorno y factores de calidad que no se hayan mencionado en esta investigación y a su vez definir lineamientos para llevarlos a cabo.

GLOSARIO:

Adaptación: Adecuar el sistema de información al entorno externo.

Administración de Requerimientos: Se define como el control en todos aquellos requerimientos y sus correspondientes atributos, que son identificados y almacenados en una base de conocimiento para poder identificar el impacto de los cambios que forman parte del proyecto.

Ambiente de Uso: Espacio físico y condiciones en que se utiliza el software.

Análisis: Fase en la que se definen las razones y justificaciones de los sistemas de información.

Análisis de Requerimientos: Define el momento del "qué de los sistemas de información".

Aplicación: Aplicación sobre la cual el software va dirigido.

Apoyo de los Desarrolladores: Asesoría o ayuda por parte de los desarrolladores.

Aseguramiento de Calidad (SQA): Determina si las necesidades de los usuarios están siendo satisfechas adecuadamente.

Aspectos Humanos: Formación de personal, creación y coordinación de equipos.

Auditoría Informática: Proceso de recoger, agrupar y evaluar evidencias para determinar si un sistema informatizado salvaguarda los activos, manteniendo la integridad de los datos. Lleva a cabo eficazmente los fines de la organización, administra eficientemente los recursos.

ai: Representa un cociente de importancia (ponderación de la característica) por cada factor. **Ciclo de vida:** Periodo de tiempo que empieza cuando un producto de software es concebido y termina cuando el producto ya no se encuentra disponible para su uso.

Codificación: Conversión del diseño de sistemas de información a instrucciones ejecutables.

Calidad de Software: 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 desarrollados profesionalmente.

Calidad en el Software: Totalidad de características de un producto, proceso o servicio que cuenta con la habilidad de satisfacer necesidades explícitas o implícitas.

Control de la Calidad: Técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad.

Certificación de la Calidad: Validez, que demuestra que la organización es capaz de desarrollar productos y servicios de calidad, bajo estándares certificados.

CMM: Capability Maturity Model.

Corrección: Grado en que un programa al corregirse satisface las especificaciones y consigue los objetivos de la misión encomendada por el usuario final.

Característica de entorno: Elementos claves para el desarrollo de software, cuyas características pueden afectar el aseguramiento de calidad.

Computadora anfitriona: Computadora donde correrá el software.

Complejidad del Software: Grado en el que se van involucrando muchos elementos físicos (periféricos), que de alguna forma contribuyen con ejecución del software.

Control Interno: Actividades operativas claves destinadas a prevenir los riesgos efectivos y potenciales a los que se enfrentan las organizaciones.

Controles Preventivos: Controles que evitan el hecho, como un software de seguridad que impida los accesos no autorizados al sistema.

Controles Detectivos: Controles que manifiestan cuando fallan los controles preventivos para tratar de conocer cuanto antes el evento.

Controles Correctivos: Controles que facilitan la vuelta a la normalidad cuando se han producido incidencias.

Ci: Representa el valor del grado de importancia de una característica del entorno de la aplicación.

Documento: Información escrita necesaria para desarrollar, implantar y utilizar los programas.

Datos: Grupo de elementos que tienen forma y contenido similares en estructura de implantación y en la composición.

Documentación: Documentos, principales para establecer e implantar un sistema de calidad; puede haber manuales a nivel empresa, departamento, producto especifico.

Desarrollo Iterativo: Define a cada una de las fases del proceso de desarrollo de una aplicación, requerimientos, análisis, diseño, implementación, pruebas, evaluación; es repetida y refinada hasta que finalmente se cumplen los requerimientos del sistema.

Diseño: Cambio que sufren los sistemas de información a representaciones, como tablas, gráficas, basadas en lenguajes y estructuras de datos.

Eficiencia: Cantidad de recursos hardware y software que necesita una aplicación para realizar las operaciones con los tiempos de respuesta adecuados.

Experiencia del Usuario: La familiaridad de los usuarios con computadoras o software similares.

Ejecución: Optimización del uso de hardware y software al implementar los productos de software.

Factores de Calidad: Constituyen los bloques fundamentales de construcción sobre los que se edifica la calidad. Estos factores son el medio por el cual se traduce el término de calidad al lenguaje de las personas que manejan la tecnología.

Fiabilidad: Grado en que se puede esperar que una aplicación lleve a cabo las operaciones especificadas y con la precisión requerida.

Facilidad de Uso: El esfuerzo para aprender el manejo de una aplicación, trabajar con ella, introducir y conseguir resultados.

Facilidad de Mantenimiento: El esfuerzo requerido para localizar y reparar errores.

Flexibilidad: Esfuerzo requerido para modificar una aplicación en funcionamiento.

Facilidad de Prueba: Esfuerzo requerido para probar una aplicación de forma que cumpla con lo especificado en los requisitos.

F(c): Representa el valor final del grado de importancia de un factor de calidad.

Funcionalidad nueva: Funciones adicionales a los requerimientos originales del sistema de información.

Gestión de la Calidad: Conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implantan por medios tales la planificación, el control, el aseguramiento y la mejora de la calidad en el marco del sistema de calidad.

Ingeniería de Software: Disciplina tecnológica y administrativa orientada a la producción sistemática de productos de programación, que son desarrollados y modificados a tiempo dentro de un presupuesto definido.

ISO 9000: Normativa de calidad en la gestión y aseguramiento de calidad de software. Define los conceptos y directrices de la calidad de software.

Integridad: Grado con que puede controlarse el acceso al software o a los datos a personal no autorizado. Proceso que permite eliminar errores que se presenten en la etapa de prueba.

Interoperabilidad: Esfuerzo necesario para comunicar la aplicación con otras aplicaciones o sistemas informáticos.

Interacción: Interacción con el usuario final, donde se establece la comunicación entre el usuario y los desarrolladores.

Lenguaje de Programación: Lenguaje o paquete computacional seleccionado para desarrollar el software.

Madurez del Desarrollador: Experiencia en el desarrollo de software similar.

Metodología de Desarrollo: Uso de algún método establecido para la creación de software.

Métricas de Software: Aquella aplicación continúa de técnicas basadas en la medida de los procesos de desarrollo del software, para producir una información de gestión significativa.

Métricas de Calidad: Métricas, que se argumentan que éstas deben ser enunciadas y utilizadas para administrar el proceso de desarrollo y debe ser conforme al producto de software particular.

Métricas del Producto: Medidas que deben ser utilizadas para distintos propósitos y un solo objetivo, medir la calidad del producto a evaluar.

Métricas de Proceso: Son aquellas métricas cuantitativas de la calidad de los procesos de desarrollo y de liberación. También llamada métrica de resultado.

Métrica de Predicción: Definida como una métrica de producto, que puede ser utilizada para predecir el valor de otra métrica.

Mejores Prácticas: Conjunto de acciones cuya principal disciplina en los equipos de desarrollo de software es garantizar la calidad mediante la reducción de fallas al liberar el sistema.

Niveles de Madurez: procesos de desarrollo de software en una escala de cierta cantidad de niveles en donde se tiene en cuenta aspectos muy variados de los procesos de desarrollo, como el grado de ambigüedad de las especificaciones, la verificación independiente de la fiabilidad de los programas, etc.

Oportunidad de Trabajo: Buenas prácticas de sistemas de información o de ingeniería de software, pueden conducir el éxito de los desarrolladores.

Operaciones del Producto: Define las características operativas del software.

OMG: Object Management Group.

Programas: Conjunto de líneas de código fuente, asociados con alguna aplicación o producto.

Paradigma: Conjunto de métodos, procedimientos y herramientas que conforman un modelo en particular y dan apoyo a eventos específicos.

Planeación: Estimación de recursos, definición de responsabilidades y actividades así como su secuencia de ejecución.

Prueba: Procedimientos para verificar que no existan errores.

Parte Física: Locales, ordenadores, herramientas, etc.

Portabilidad: Esfuerzo requerido para transferir la aplicación a otro hardware o sistema operativo.

Producto de Software: Todo aquel producto final que se obtiene como resultado de la aplicación de los procesos de la ingeniería de software.

PND: Plan Nacional de Desarrollo de la Secretaría de Economía.

Revisión del Producto: Define la capacidad para soportar cambios en el software.

Reusabilidad: Grado en que partes de una aplicación pueden utilizarse en otras aplicaciones.

Riesgos: elementos que se presenten en cualquier situación y que estos puede implicar que el software falle.

Sistemas de Información: Son sistemas que se desarrollan con diferentes propósitos para procesamientos de datos, administración para apoyo en la toma de decisiones.

Sistema de Calidad: Representación de la estructura organizativa, procedimientos, procesos y recursos necesarios para implantar la gestión de calidad, adecuándose a los objetivos de calidad de la empresa.

SEI: Software Engineering Institute.

Sistema Mínimo: Característica del software como parte de un sistema, un programa debe ser asociado a un procesador antes de ser usado.

Sistema Típico: Define aquellos elementos que en algunas ocasiones será necesario asegurarse que dichos componentes formen parte integral de lo que se desea evaluar.

Sistema Complejo: Es aquel en que resulta más difícil de asegurar la calidad. En estos casos la funcionalidad de los dispositivos no puede ser separada de la del software.

Transición del Producto: Define la adaptabilidad a nuevos entornos del software.

UML: Unified Modeling Language. Es el diseño de aplicaciones basada en componentes.

BIBLIOGRAFÍA

[1] AMITI, Asociación Mexicana de la Industria de Tecnologías de Información, Articulo Año 2002, Web Site http://www.amiti.org.mx/

[2] BENDENI, ALEJANDRO G. 1999, Lo malo, y lo feo de los Marcos de Trabajo 1999 (Elementos a considerar antes de su aplicación en forma directa a una unidad informática inmadura de América Latina)

[3] CMM, The Capability Maturity Model, Guidlines for Improving the Software Process, SEI Series in Software Engineering, Addison Wesley 1995.

[4] DAVIS, CAROL J. "Industrial acceptance of software quality assurance standards" En IEEE Journal; 1993

[5] Delivering Software Project Success, 2000 Web Site http://construx.com

[6] FAIRLEY R. ingeniería de Software Mc Graw – Hill México D.F. 1989

[7] GLICK, BUD. "An SQA quality tracking methodology". En: IEEE Journal; 1990.

[8] HAMBLING, B.F. "Verification, validation and the achievement of quality: a holistic approach". En: IEEE Journal, 1991.

[9] HANNA OKTABA, Facultad de ciencias, UNAM, 2000. Web Site http://redii.iimas.unam.mx/redii/Ciclo2000/REDII2000/OktabaHanna/Oktaba.htm

[10] HUMPHREY, WATTS HUMPHREY, The Process Bureaucracy, November 2000

[11] IEEE the Institute of Electrical and Electronics Engineers Web Site http://www.ieee.org/organizations/history_center/

[12] INCE, DARREL. “Introduction to software project management and quality assurance”. En: London; New York : McGraw-Hill Book Co, 1993.

[13] IUNAV, Instituto Universitario de Venezuela, 2002. Web Site http://www.iunav.org/~mayr/materia/ingsoftware

[14] ISO INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, ISO 9000 Standards on Quality Management and Quality Systems, ISO 9001-4 Quality Management and Quality Assurance Estándar, Web Site http://www.iso.org/iso/en/ISOOnline.frontpage.

[15] JOHNSON, PHILIP M. “An instrumented approach to improving software quality through formal technical review”. En: IEEE Journal; 1994.

[16] JURAN, J. M. “Juran on quality by design: the new steps for planning quality into goods and services”. En: New York Free Press ;Toronto: Maxwell Macmillan Canada; New York: Maxwell Macmillan International, 1992.

[17] LITTLEWOOD B and N. FENTON. “Software reliability and metrics”. En: London; New York; Elsevier Applied Science, 1991

[18] LOWE, JOHN E. “SQA-A customer service approach”. En: IEEE Journal; Febrero 1991; Volumen 8; Número 2, pp. 1276-1281.

[19] OMG, Unified Modeling Languaje Specification Object Management Group, Framingham, MA, 1998. Web site http://www.omg,org

[20] PERRY, WILLIAM E. “Effective methods of EDP quality assurance”. En: Englewood Cliffs, N. J.: Prentice-Hall, 1983.

[21] PERRY, WILLIAM E. “Quality assurance for information systems”. En: QED technical publishing group; Primera edición; Wellesley, MA, U.S.A.; 1991.

[22] PRESSMAN, ROGER.S.; ingeniería de Software. Un enfoque práctico. Cuarta Edición. Mc Graw Hill. 1998

[23] PRESSMAN, ROGER.S, Software Engineering, A Practitioner's Approach, Fifth Edition Mc Graw Hill, 2001 Excellent and complete Resources at R.S. Pressman and Associate Web site http://www.rspa.com/spi/index.html

[24] RUP, IVAR JACOBSON, GRADY BOOCH, JAMES RUMBAUGH. El Proceso Unificado de Desarrollo de Software Addison Wesley, 2000.

[25] SALAZAR L., SANDRA E. Tesis: "Lineamientos para Implantar la administración de la configuración del Software en el área de Informática". ITESM, 1992.

[26] SEI, Software Engineering Institute, Web Site http://www.sei.cmu.edu/

[27] SHERI LAWRENCE PFLEEGER, Software Engineering, Theory and Practice 1er edition, Prentic Hall, 1998

[28] SIMMONS, RONALD A. "Software quality assurance (SQA) early in the acquisition process". En: IEEE Journal; 1990.

[29] SOMMERVILLE, LAN, Software Engineering 6th Edition. 2000

[30] STOCKMAN, GUILLAUME SINCLAIR. "A Framework for software quality Measurement". En: IEEE Journal; Febrero 1990; Volumen 8; Número 2, pp. 224-233.

[31] TERRY, GEORGE R. "Principios de Administración". Ed. Continental. Mexico. 1997.

[32] TOTAL QUALITY MANAGEMENT (TQM) Stauss/ Friege: Zehn Lektionen in TQM; in: Havard Business manager, 2/96; Seite 20-33 Oess: Total Quality Management; Wiesbaden 1993

[33] UNCTAD, (United Nations Conference on Trade and Development, 2001 Web Site http://www.unctad.org/Templates/Startpage.asp

[34] UML, Ivar Jacobson, Grady Booch, James Rumbaugh. El Lenguaje Unificado de Modelado, Addison Wesley, 2000.

[35] WESENBERG, DAVID P. “A system approach for software quality assurance”. En: IEEE Journal; 1991.

[36] YOURDON EDWARD, The decline and Fall of the American Programmer, USA. 1995

[37] SAMPIERI, COLLADO, BAPTISTA, “Metodología de la Investigación” Segunda Edición, Mc Graw Hill, 2003.

ANEXOS

Anexo_1

FORMATO_1 Panorama General de la Empresa	Objetivo de la Encuesta: Adquirir un panorama general de la Empresa/Área de Sistemas encuestada Panorama General de la Empresa

Nombre de la Empresa: _____

Nombre de la Persona Encuestada: _____

Puesto/Función: _____

Experiencia: _____ Fecha: _____

A continuación se listara una serie de preguntas formuladas de manera breve, puede ser que se encuentre mas de una opción como respuesta, marque con una 'X' en los espacios propuestos (dentro de los recuadros) la respuesta (s) correcta (s), así como dar respuesta a aquellas preguntas abiertas.

1. El software que desarrollan consiste de:
☐Sistemas ☐Modelos ☐Prototipos ☐Otros:_____
2. El software que desarrollan es para uso:
☐Interno ☐Externo ☐Otros:_____
3. En el software que se desarrolla, el papel que juega la calidad del mismo se puede decir que es:
☐Si es Importante ☐No es importante ☐Es poco importante
☐Es muy importante ☐Juega un papel moderado
4. La calidad del software desarrollado es buscada:
☐Desde la planeación ☐Durante la Implementaron ☐Durante la etapa de pruebas
5. Para sus desarrollos de software, siguen lineamientos o procedimientos metodológicos de apoyo en la calidad de software de tipo:
☐Modelado de Objetos (UML) ☐Administración de proyectos ☐Metodologías de Apoyo en la calidad de software (COBIT, CMM, RUP)
☐Otros:_____
6. En que consiste dicho procedimiento, metodología:
_____
7. Ha dado resultados:
☐Si ☐No___% Cubierto
8. Como han medido los resultados:
_____
9. Que tiempo tiene actualmente este procedimiento, metodología:
☐ < 1 año ☐1 a dos años ☐ > 3 de años
10. En que tipo de aplicaciones esta involucrado en estos momentos y que rol desempeña:
_____
11. El departamento o área de sistemas cuenta con alguna certificación de calidad como:
☐ISO ☐IEEE ☐Ninguno ☐Otros:_____

12. En que fase de desarrollo se encuentran los proyectos en los que esta involucrado actualmente (considere el proyecto de mayor prioridad):
Fase de Análisis Fase de desarrollo Fase de Operación/Mantenimiento
Otros:_____
13. Conque frecuencia recibe capacitación por parte de sus Jefes de Sección/ Gerencia:
dos veces al año cinco veces al año Ninguna
Otros:_____
14. Conque frecuencia tienen auditorias de sistemas:
Una vez al año dos veces al año Otros:_____
15. Que tipos de auditorias manejan:
Interna Externas Otros:_____

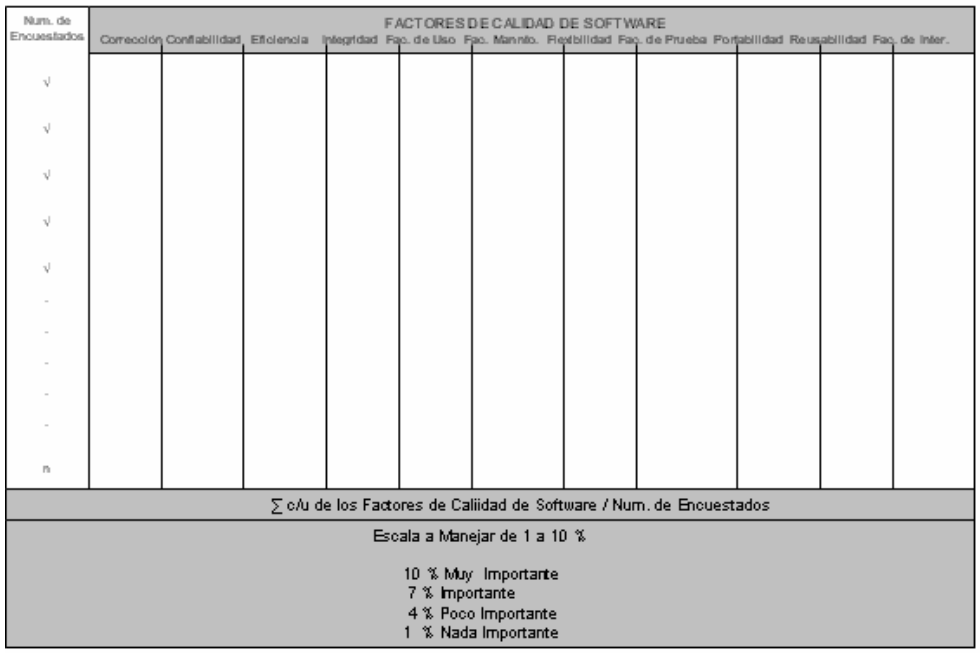
Anexo_2

Plantilla de apoyo para determinar los valores numéricos de las características de calidad de un producto de software.

Num. de Encuestados	CARACTERISTICAS DE ENTORNO DE CALIDAD DE SOFTWARE													
	Aplicación	Amb./Uso	Riesgo	Comp.Anf	Mad./Desa	Exp./Usu	Apo./Desa	Exp./Desa	Inter/Usu	Res./Com.	Met./Desa	Leng./Prog	Comp./Soft.	Salud
√														
√														
√														
√														
√														
-														
-														
-														
-														
-														
n														
∑ c/u de las Caracerísticas de Entorno de Calidad de Software // Num. de Encuestados														
Escala a Manejar de 1 a 10 % 10 % Muy Importante 7 % Importante 4 % Poco Importante 1 Nada Importante														

Anexo_3

Plantilla de apoyo para determinar los valores numéricos de los factores de calidad de un producto de software.



Anexo_4

Plantilla de apoyo que nos representa la matriz de relación para determinar los valores numéricos entre las características de calidad y los factores de calidad de un producto de software.

Fac/Car	Aplica Ción	Amb. Uso	Riesgo	Comp. Anf.	Mad. Desarr.	Exp. Usuario .	Apoyo Desarr.	Exp. Desarr.	Itera. C/Usu	Restric. Comer.	Metod. desarr.	Leng. Prog.	Comp. Soft.	Salud
Corrección	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Confiab.	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Eficiencia	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Integridad	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Fac. Uso	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Fac.Mannto	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Flexibilidad	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Fac. Prueba	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Portabilidad	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Reusab.	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M
Fac. Inter.	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M	()F ()M

Anexo_5

Plantilla de apoyo para determinar la importancia de las características de entorno de calidad de un producto de software.

i: Aplicación:

_____

ii: Asignar valores numéricos que consideren caen en los rubros propuestos (1 No importante, 4 Poco Importante, 7 importante, 10 Muy importante) a cada una de las características de entorno las cuales se vean involucradas en su desarrollo actual. Explicar por que se le da ese rubro.

Aplicación: _____
Ambiente de Uso: _____
Riesgos y consecuencias de fallas: _____
Computadora Anfitriona: _____
Madurez del Desarrollador: _____
Experiencia del Usuario: _____
Apoyo de los desarrolladores: _____
Experiencia de los Desarrolladores: _____
Interacción con el usuario final: _____
Restricciones Comerciales: _____
Metodología de desarrollo: _____
Lenguajes de Programación: _____
Complejidad del Software: _____
Salud de los desarrolladores:: _____

iii: Obtener el grado de importancia para cada factor de calidad (usar formula y tablas de cocientes de importancia (a) para realizar los cálculos pertinentes.):

Corrección: _____ Confiabilidad: _____
Eficiencia _____ Integridad: _____
Facilidad de Uso: _____ Facilidad de Mantenimiento _____
Flexibilidad _____ Facilidad de Prueba: _____
Portabilidad _____ Reusabilidad _____
Facilidad de Interoperación _____

iv: Enlistar factores de calidad de manera descendente.

1) _____ ____
2) _____ ____
3) _____ ____
4) _____ ____
5) _____ ____
6) _____ ____
7) _____ ____
8) _____ ____
9) _____ ____
10) _____ ____
11) _____ ____

NOTA: Estos valores nos darán un panorama de que tanto conocen y aplican las características y factores de calidad de software. Todo este ejercicio nos ayudara a tomar en cuenta las mejores practicas de control de calidad en la generación de los productos de software que se generen.